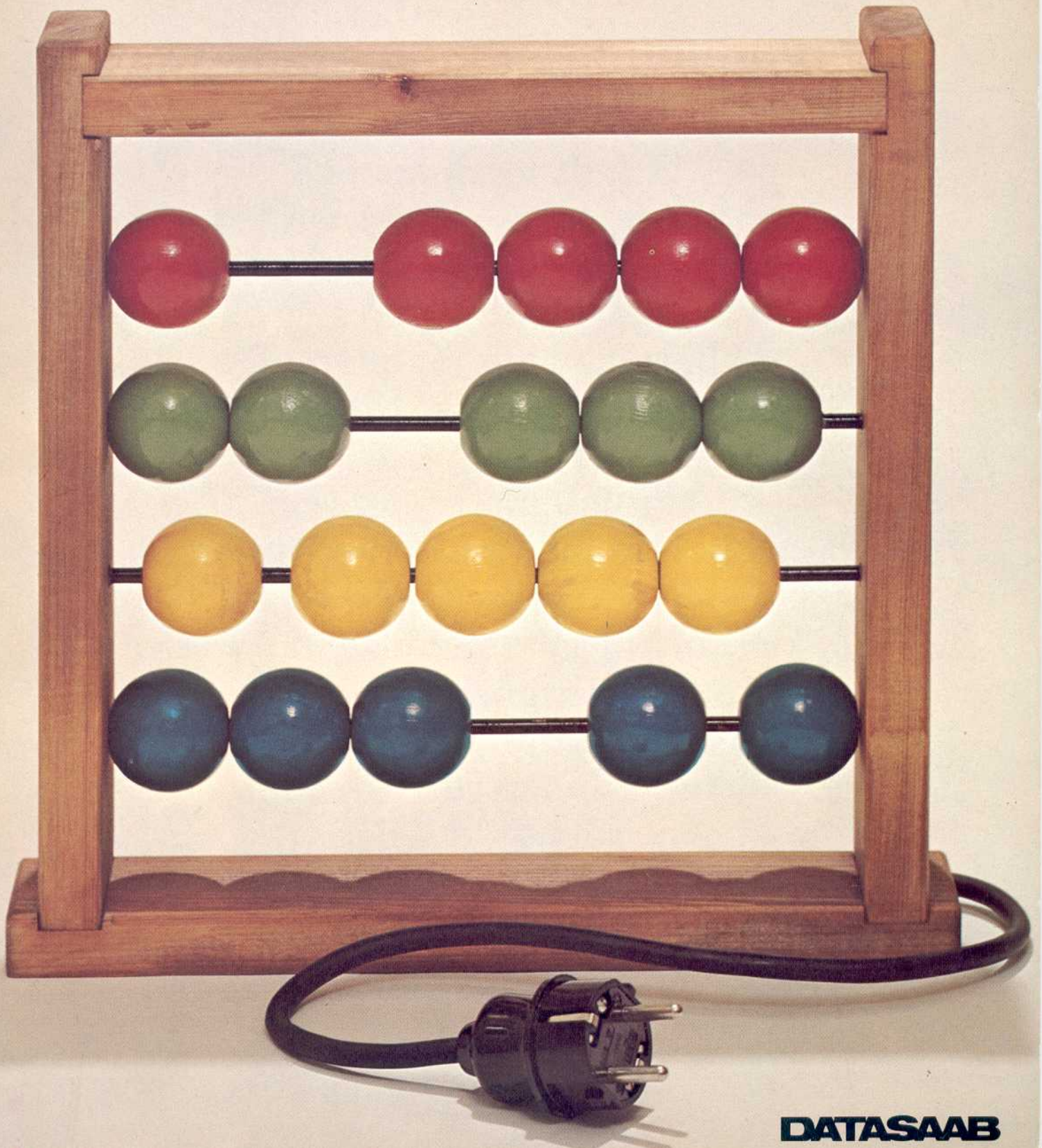
FRÅN KULRAM TILL DATASKÄRM

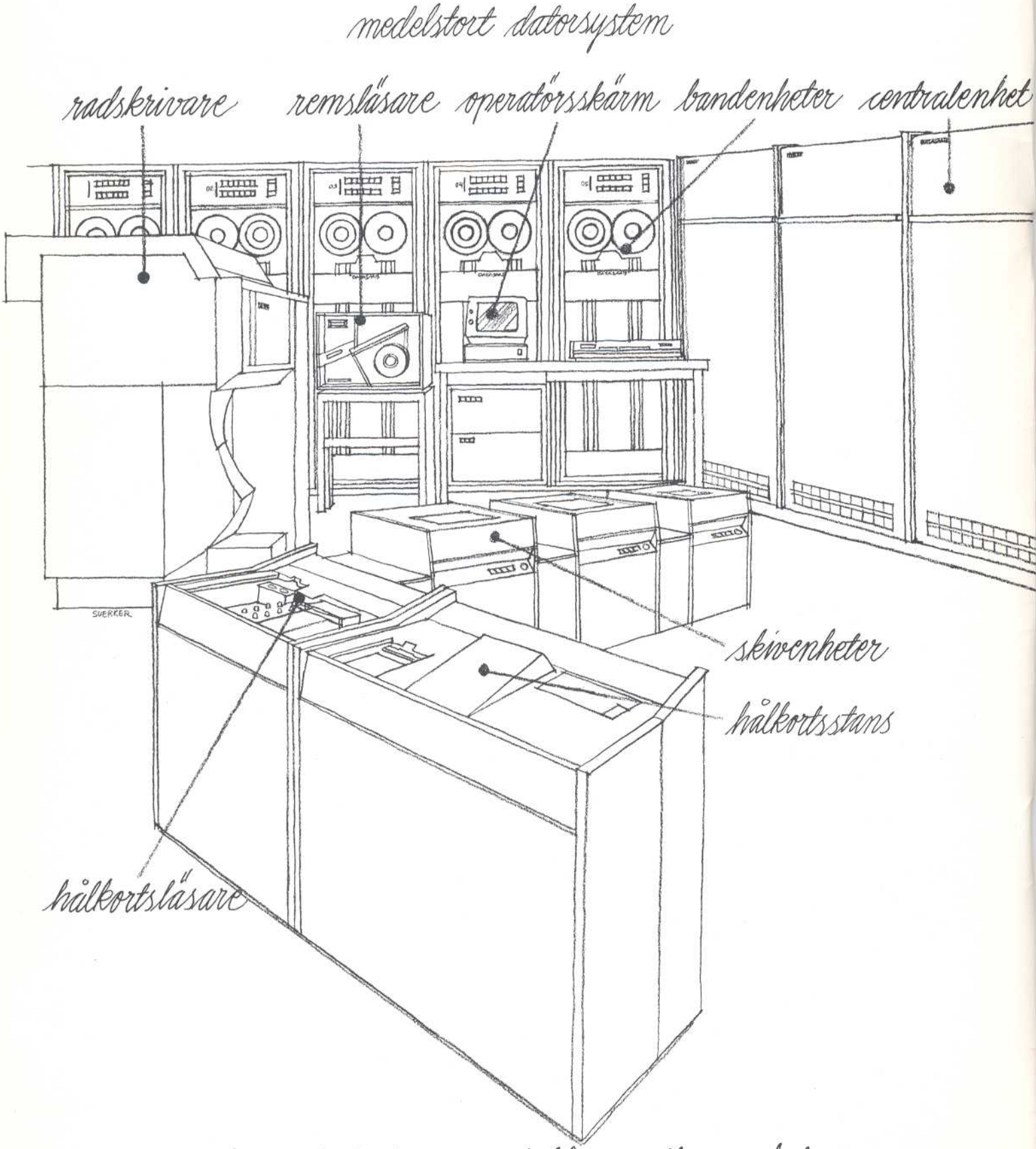


DATASAAB

Vad menas egentligen med data?
Vad är ADB?
Kan en dator tänka?
Det har kanske hänt nån gång
att Du har ställt Dig såna frågor.
Vi vill försöka ge svar på dom i den här broschyren.
Du behöver inga speciella
förkunskaper för att förstå innehållet.
I tio avsnitt
berättar vi om datorer,
programmering, systemarbete och tillämpningar.
Vi vänder oss till Dig
som går på högstadiet eller gymnasieskolan
och till Dig som är vuxen
och som vill ha ett hum om vad det här
med datorer handlar om.
Håll till godo!

Text: Leif Ingevaldsson
Teckningar och layout: Sverker Lundgren

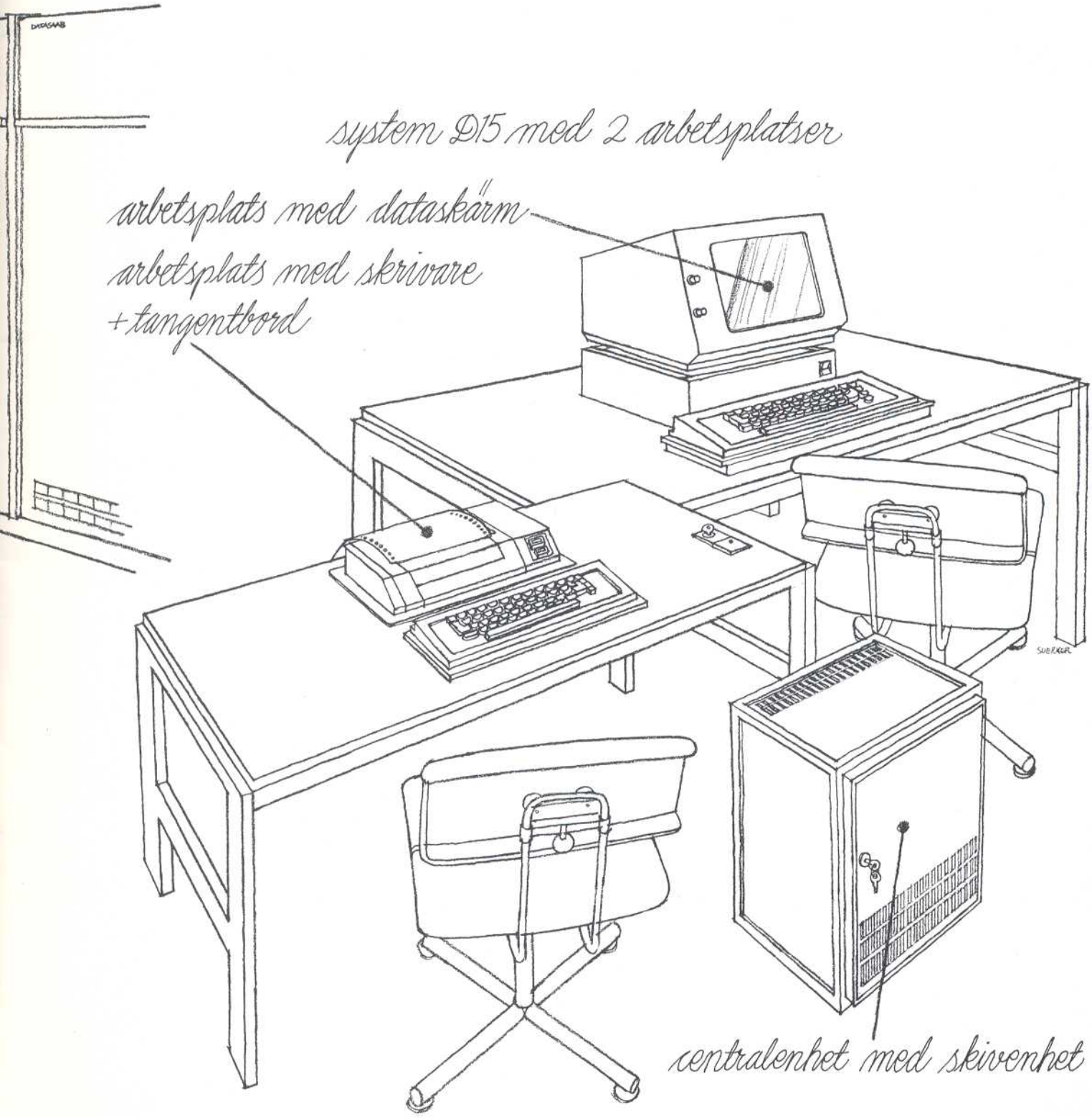
Vad det handlar om



Här bredvid har vi ett exempel på ett medelstort datorsystem och ett exempel på en minidatoranläggning i system D15 med arbetsplatser. Vi ska i de kommande avsnitten beskriva bl.a. vad en centralenhet gör, hur band- och skivenheter används, var en dataskärm kommer in i bilden och hur de olika delarna samverkar. Vi ska också sätta in datorn i ett lite större systemsammanhang.

Det handlar som synes om ett uppbåd av apparater, både elektroniska och elektromekaniska.

En dator kan läsa, skriva och räkna men är på inget sätt intelligent. Benämningen elektronhjärna, som datorn fick i sin barndom, är missvisande.



1. Hur det hela började

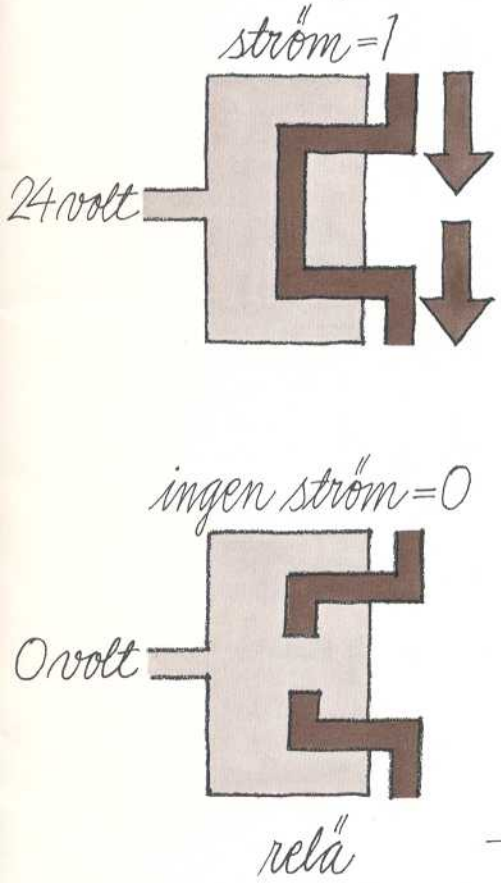
Kulramen, datorns anfader

Datorns ursprung kan spåras ca 5000 år tillbaka. Den äldsta kulramen daterar sig från den tiden. Det hände ingenting med räknehjälpmedlen förrän på 1600-talet, då Pascal konstruerade den första adderingsmaskinen. Omkring år 1800 uppfann Jacquard en hålkortsstyrd vävstol och under de följande decennierna såg idén till en matematisk analysmaskin dagens ljus. Det var Babbage som stod för den idén. Den är så hållbar att den ligger till grund för dagens datorer. Tyvärr fick Babbage aldrig se sin idé förverkligad — den tidens teknik var inte tillräckligt utvecklad.

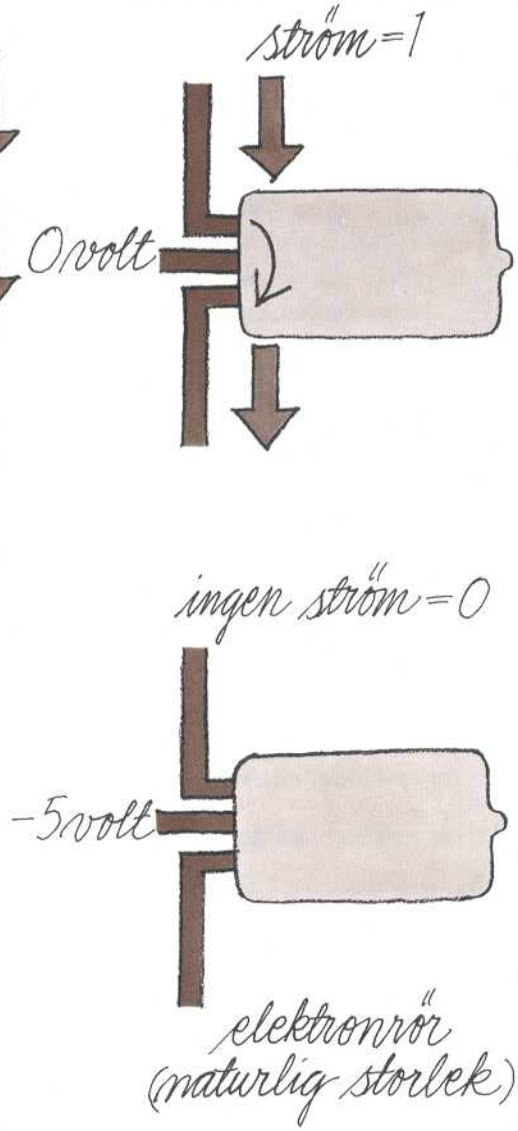
På 1880-talet konstruerade Hollerith i USA den första hålkortsmaskinen för databearbetning. IBM köpte patentet och sålde hålkortsmaskiner över hela världen. När datorn på 1950-talet började bli kommersiellt presentabel, kopplade IBM ihop datorn med hålkortssystem och lyckades därvid grundlägga sin dominans i datorvärlden.

Ettor, nollor och elektricitet

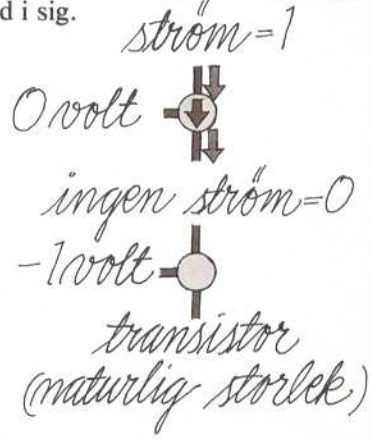
De första datorerna vid slutet av 30-talet arbetade med reläer. Man kan släppa fram ström eller stänga av ström med ett relä. Låter man ström betyda 1 och ingen ström 0, kan man räkna. Hur det går till ska vi berätta om längre fram.



Ett relä är långsamt. Det kan ta flera tusendels sekunder (millisekunder, ms) för ett relä att reagera på en styrspänning. Elektronröret är snabbare. Det kan reagera på miljondels sekunder (mikrosekunder, µs). Senare delen av 50-talets datorer hade därför elektronrör och kunde utföra 20 000 additioner per sekund.



Transistorn, en liten knopp med tre trådar och som finns i varje transistorradio, har många fördelar jämfört med elektronröret. Den är mycket mindre och blir inte varm som elektronröret, som har en glödtråd i sig.



Man kan också tränga ihop hundratals eller tusentals transistorfunktioner i en enda liten kapsel. Den är ett par cm lång och kallas IC-kapsel (IC=integrated circuit, integrerad krets). Numera används IC-kapslar i alla datorer.

ic-kapsel
(naturlig storlek)

=hundratals eller tusentals transistorer

Typisk räknehastighet är för dagens datorer några hundratusen till några millioner additioner per sekund.

2. Databehandling - vad är det egentligen?

Datorn bearbetar data (ett datum, flera data). Vad är då data? Här nedan har vi en samling data, bestående av symboler, dvs. bokstäver, siffror, specialtecken (som t.ex. ett minustecken). Organiseras data som i figuren, får dessa mening. Data måste organiseras med hänsyn till mottagaren. Då kan de lätt tolkas rätt och informationen nå fram.

Data bearbetas av datorn som en samling symboler och talvärden. För datorn har data ingen innebörd.

Gamle kamrern och datorn

En populär modell för datorns arbetsprincip är gamle kamrerns metoder, t.ex. när han ska räkna ut löner. Hans arbete är ett exempel på MDB, manuell databehandling, till skillnad från datorn, som utför ADB, automatisk databehandling.

Så här räknar kamrern ut löner i stora drag:

1. Han tar en arbetssedel från IN-korgen ...
2. tar ut ur kartoteket ett personalkort med samma anställningsnummer som på arbetssedeln ...
3. multiplicerar på räknemaskinen antalet timmar på arbetssedeln med den timlön, som finns angiven på personalkortet ...
4. skriver resultatet på lönebeskedet ...
5. skriver resultatet på personalkortet ...
6. sätter tillbaka personalkortet i kartoteket och
7. lägger lönebeskedet i UT-korgen.

Dessa moment upprepar han till dess arbetssedlarna i IN-korgen är slut.

När gamle kamrern var ny på sin post, fick han lära sig rutinen genom att arbeta efter en *arbetsinstruktion*. Den kunde han snart utantill. Därmed kunde man säga att kamrern var *programmerad* för uppgiften.

Datorn — världens snabbaste korkskalle

På motsvarande sätt kan man få en dator att utföra vad man vill genom att ge den en arbetsinstruktion eller ett *program*. Om man skulle lära en dator att äta skulle programmet se ut så här.

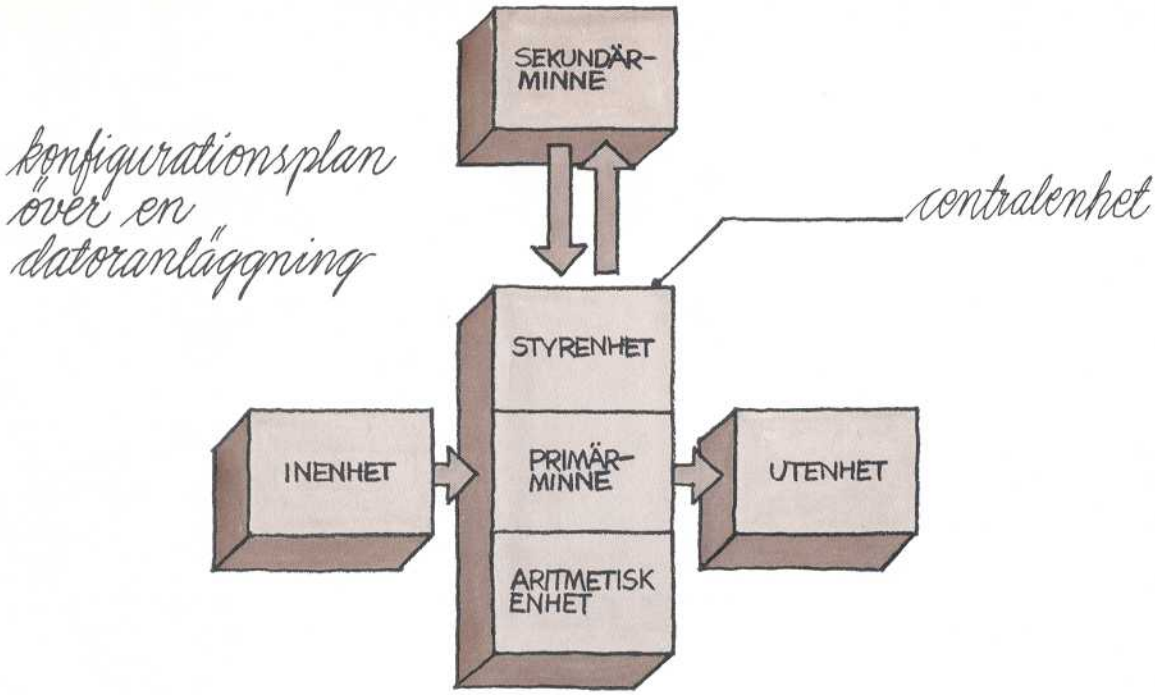
```
1. LÄGG UPP MAT PÅ TALLRIKEN
2. TAG MED KNIV OCH GAFFEL
   EN BIT PÅ TALLRIKEN
3. FÖRPASSA TILL MUNNEN
4. BIT AV
5. TUGGA 24 GÅNGER
6. SVÄLJ
7. OM MATEN EJ SLUT
   GÅ TILL 2
8. SLUT
```

Instruktion nr 7 är speciellt intressant. Den är ett exempel på villkorsinstruktion med hopp (till instruktion nr 2). Om maten är slut, tittar datorn på nästa instruktion, nr 8. Där står det SLUT, dvs. programgenomloppet ska sluta.

Eftersom programmeraren skrivit TUGGA 24 GÅNGER kommer datorn att tugga exakt 24 gånger varje gång instruktion nr 5 genomlöps, aldrig 23, aldrig 25. Skulle av misstag nr 6 skrivits före instruktion nr 5, skulle datorn svälja maten hel med risk för att kvävas och därefter tugga 24 gånger i tomgång.

Datorn har ingen egen intelligens, den gör bara vad den blir tillsagd, varken mer eller mindre.

3. Gamle kamrern elektrifierad



Datorns viktigaste delar kan ritas som en konfigurationsplan — en plan som i mycket liknar gamle kamrern och hans arbetsplats. (Inga jämförelser i övrigt: gamle kamrern har själv hittat på listiga rutiner och är väldigt trevlig att prata med. Han har mycket att förtälja om det glada 30-talet, vilket inte datorn har). In- och utenheterna motsvarar IN- och UT-korgarna. Centralenheten (CPU= Central Processing Unit) motsvarar både gamle kamrern själv och räknemaskinen (aritmetisk enhet=sifferräkningsenhet). Sekundärminnet, som kan vara ett magnetband, rymmer mycket mera än primärminnet.

Ett personalregister kan lagras i sekundärminnet och motsvarar då kamrerns personalregister i kartoteket. Sekundärminnet kallas också yttre minne och primärminnet får ibland heta inre minne.

In- och utenheter kallas även perifera enheter eller kringutrustning.

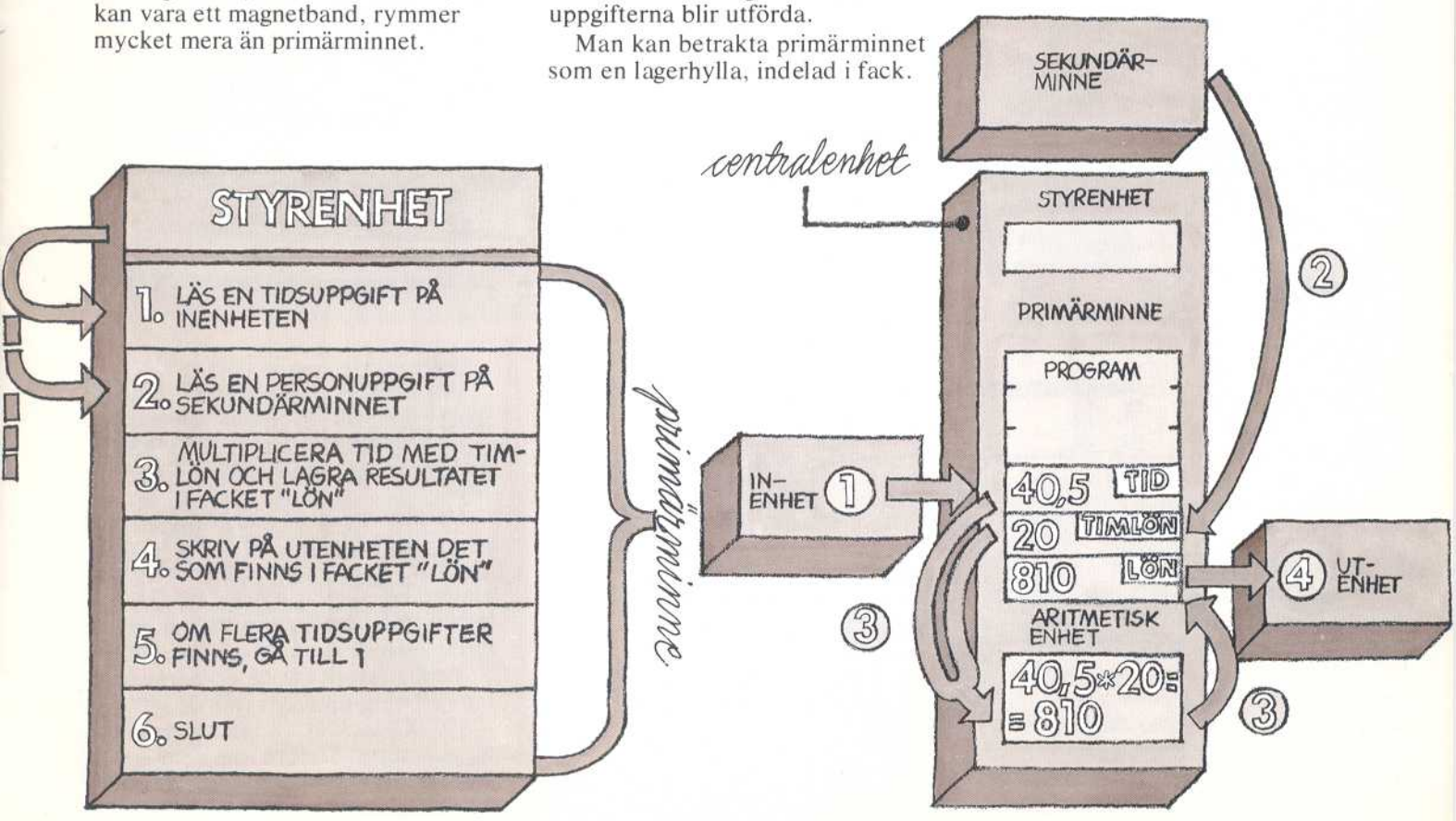
När datorn arbetar med en lönerutin måste först ett program lagras i primärminnet som i figuren. (Vi förenklar något och avstår här från att lagra den beräknade lönen på sekundärminnet). Styrenheten tittar på en instruktion i taget och ser till att uppgifterna blir utförda.

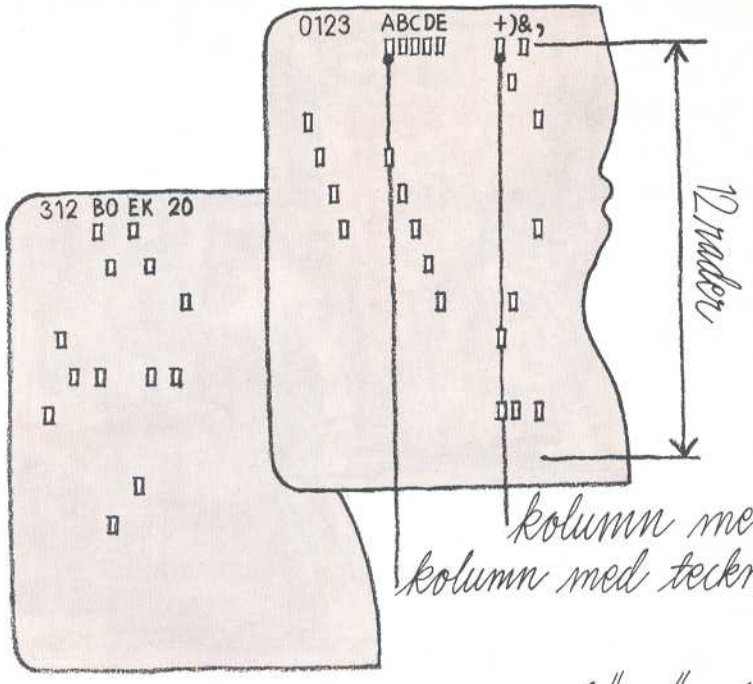
Man kan betrakta primärminnet som en lagerhylla, indelad i fack. Varje instruktion ligger i var sitt fack.

Pilarna i figuren har samma nummer som instruktionerna i programmet och visar vad som händer så länge tiduppgifter finns att hämta i inenheten. Vi konstaterar att datorn måste kunna

- läsa
- lagra
- beräkna
- skriva

Låt oss titta på principerna för de här fyra funktionerna i tur och ordning.





Läsa

Olika typer av utrustning kan fungera som inenhet. Hålkortsläsaren är en vanlig sådan.

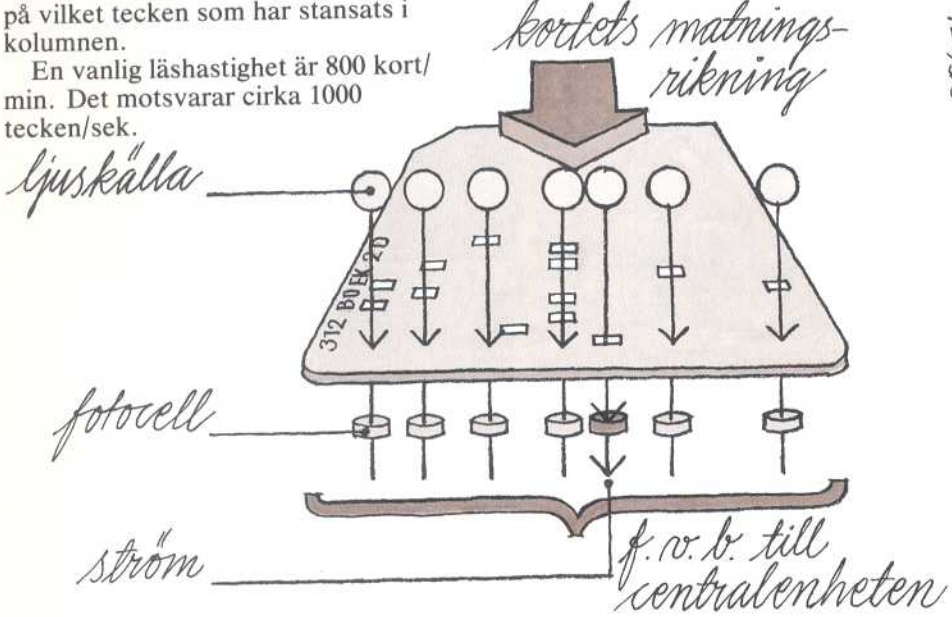
Ett hålkort indelas i 80 kolumner och 12 rader. Ett hål eller en kombination av hål i en kolumn motsvarar ett tecken. Ovanför varje kolumn kan man, om så önskas, få tecken även i klarskrift.

Nu gäller det att ''översätta'' hålen till strömpulser. Då har vi användning för fotoceller. En fotocell finns snart sagt i varje kamera och är en liten transistorliknande manick. När den får ljus på sig, kan den skicka ström genom en strömkrets. När den inte får ljus på sig, skickar den ingen ström. Olika trådar från hålkortsläsaren till centralenheten blir strömförande beroende på vilket tecken som har stansats i kolumnen.

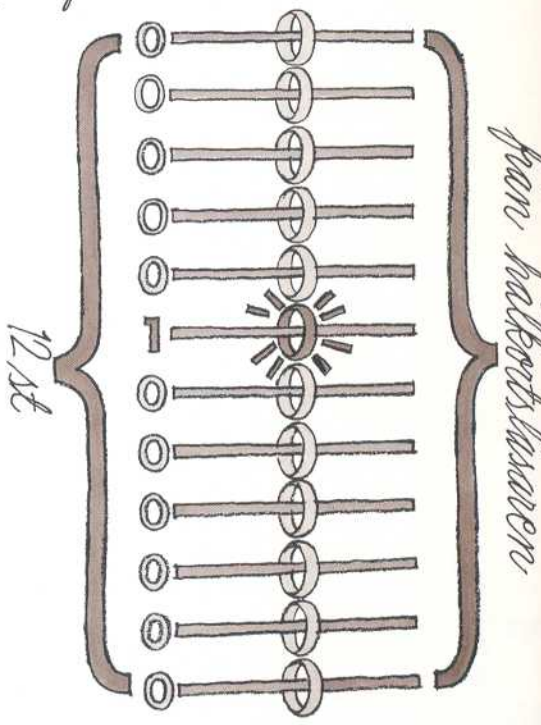
En vanlig läshastighet är 800 kort/min. Det motsvarar cirka 1000 tecken/sek.

För varje hålkortstecken som ska lagras, går det åt en ''stapel'' med 12 kärnor enligt figuren. En sådan stapel kallas minnescell och motsvarar ett av de hyllfack, som vi tidigare talat om.

Men datorn måste ju ''veta'' var data finns lagrade för att kunna hämta dem vid behov. Därför arrangeras minnescellerna i ett rutnät. 312 BO EK kan nu återfinnas t.ex. i cellerna Q2 t.o.m. Q10. Observera att mellanrumstecken också är ett tecken, lika viktigt som övriga.



här är tecknet 3
lagrat i en minnescell



Iskriftspråktalarmellanrumstecknet omattettordslutarochettannatbörjar.

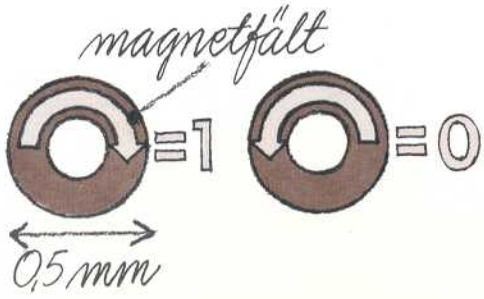
En minnescell i D20-serien rymmer 24 st ettor eller nollor, i D5-serien 16 st.

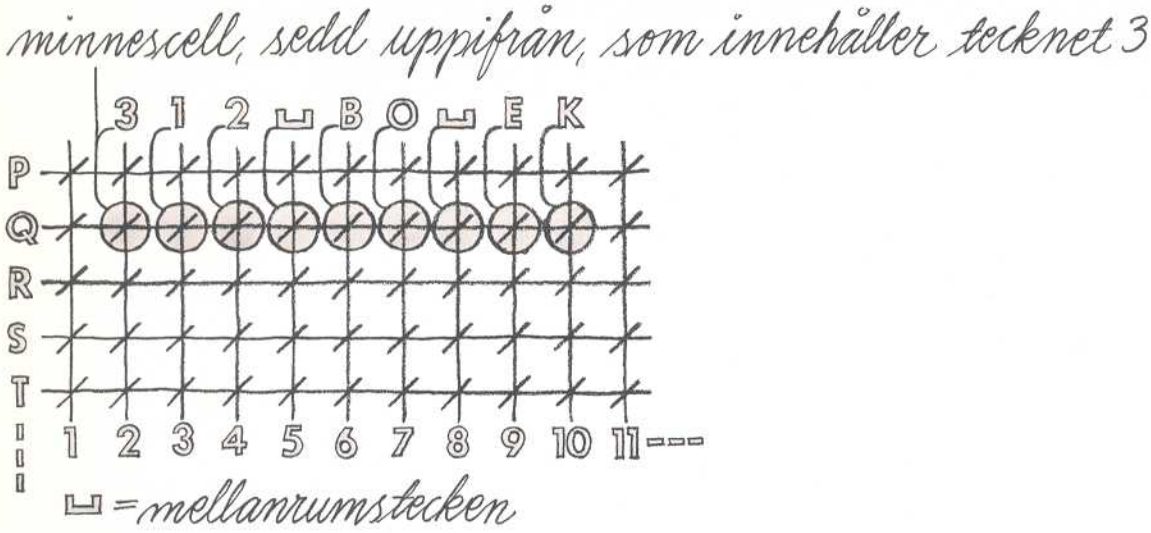
Halvledarminnet innehåller transistorer i IC-kapslar i stället för kärnor. Det har fått sitt namn av transistormaterialet, som är halvledare. Halvledare är ett mellanting mellan elektriskt ledande material och isolerande.

Vanliga storlekar på primärminnet i medelstora datorer rymmer 180 000—320 000 tecken, i smådatorer ungefär tiondelen. Datorn kan hämta eller lagra tecken i primärminnet på mindre än en miljondels sekund.

Lagra i primärminnet

Man måste kunna lagra ettor och nollor i primärminnet. Två minnestyper är vanliga: kärnminnet och halvledarminnet. Kärnminnet består av ett antal små magnetiserbara kärnor (ringar). De sitter uppträdda på ledningstrådar och kan magnetiseras med en strömstöt genom tråden i ena eller andra riktningen.





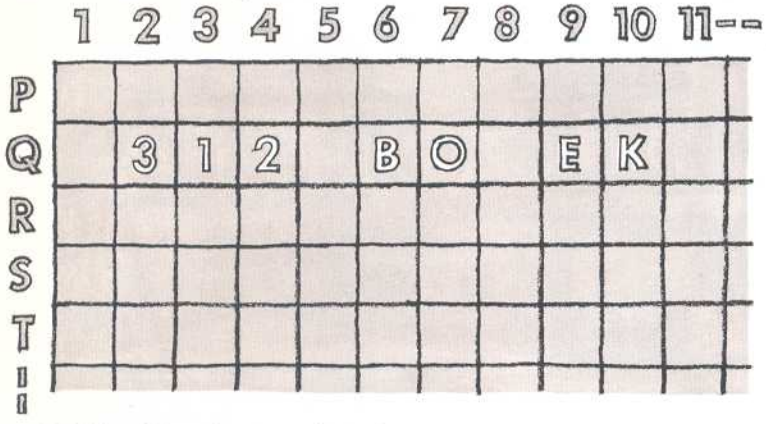
Beräkna

Datorn kan utföra mycket komplicerade beräkningar. Men den enda matematik, som är ''inbyggd'', är de fyra enkla räknesätten. När mera komplicerade beräkningar ska utföras, måste man ibland skriva program, som innehåller matematiska knep.

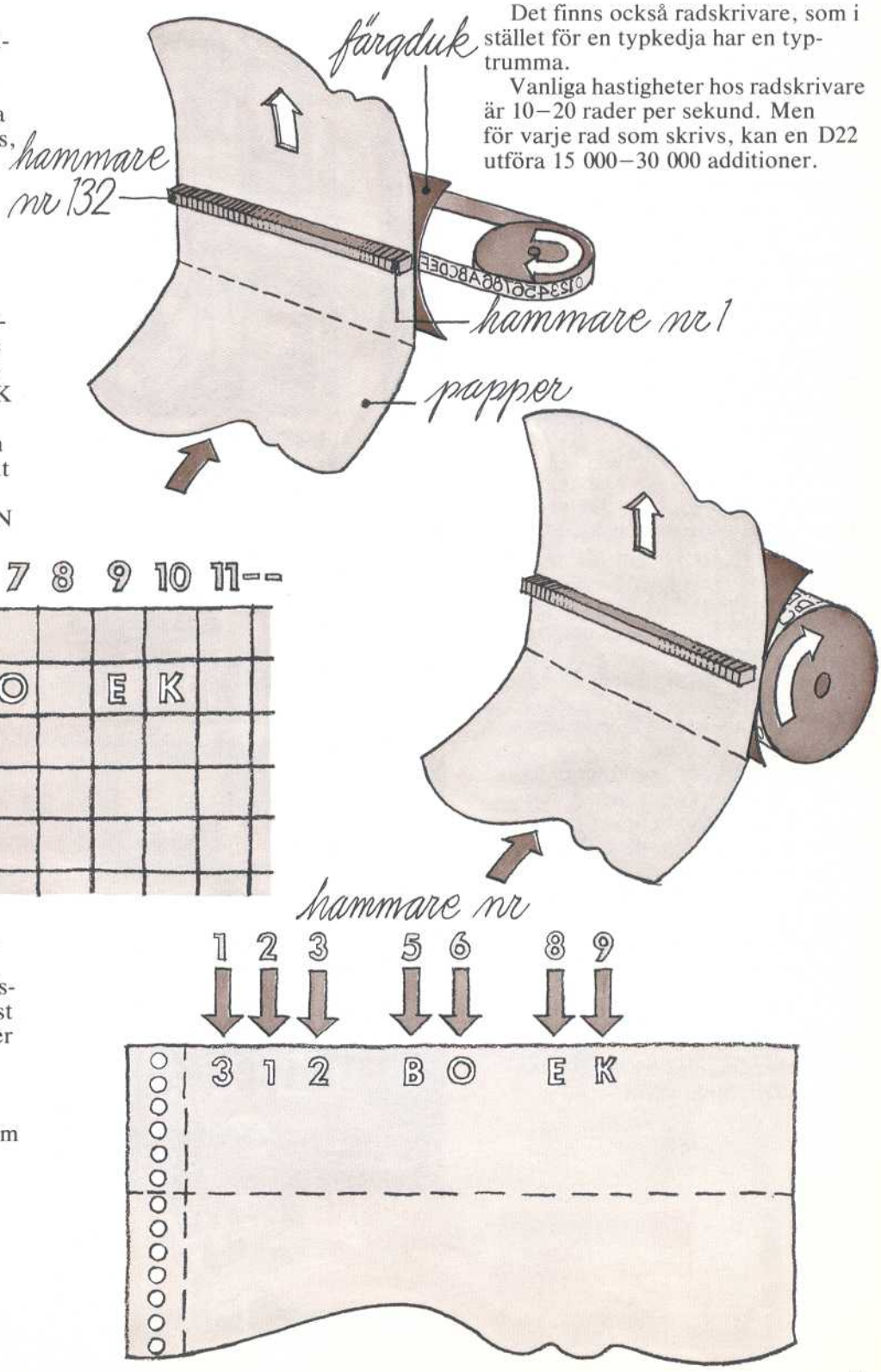
Skriva

Ett exempel på utenhet är den så kallade radskrivaren. Den har en kedja av typer, som defilerar förbi papperet från morgon till kväll. Bakom papperet finns en liten hammare för varje skrivposition, vanligen 132 stycken. Vi tänker oss att 312 BO EK ligger lagrat i primärminnet. Vill vi ha detta utskrivet på radskrivaren kan instruktionen i programmet se ut ungefär så här:

SKRIV EN RAD MED 9 TECKEN
MED BÖRJAN I Q2.



Då förs från minnet strömpulser, som svarar mot ettor, till radskrivarens s.k. buffert (en samling transistorer). Så snart E (om det låg närmast i tur) passerar hammare nr 8 klämmer denna till, dito när B passerar nr 5 osv. Hammare nr 4 och 7 påverkas inte (mellanrumstecken). Det hela går mycket snabbt. Det ser ut som om hela rader skrivs på en gång.



Det finns också radskrivare, som i stället för en typkedja har en typtrumma.

Vanliga hastigheter hos radskrivare är 10–20 rader per sekund. Men för varje rad som skrivs, kan en D22 utföra 15 000–30 000 additioner.

4. Kulramen, tvåsystemet och teckenkoderna

En enkel kulram kan ibland vara till hjälp för att visa sambandet mellan ettor, nollor och olika tecken.

Vi börjar med en fyrkulig ram. Kulornas *värden* är från vänster till höger: 8, 4, 2, 1. En kulas värde ska räknas endast när den är i 1-läge.

Observera att figuren också har en kolumn för tvåsystemet och en kolumn för tiosystemet!

Här några kommentarer:

Första ramen
alla kulor i 0-läge
alltså 0+0+0+0=0

Andra ramen
kulan med *värdet* 1 i 1-läge
0+0+0+1=1

Tredje ramen
kulan med *värdet* 2 i 1-läge
0+0+2+0=2

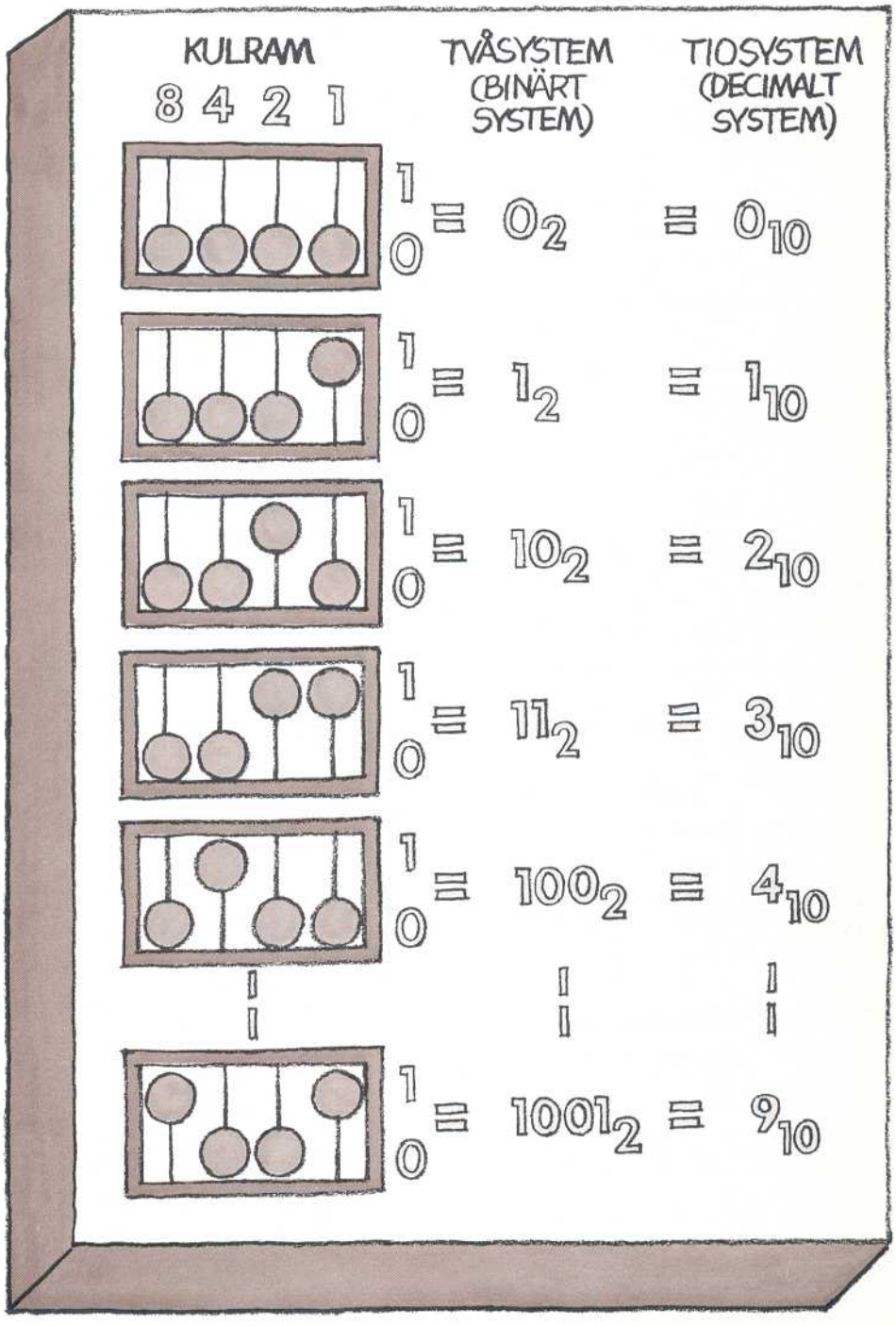
Fjärde ramen
kulorna med *värdena* 2 och 1 i 1-läge
0+0+2+1=3

Femte ramen
kulan med *värdet* 4 i 1-läge
0+4+0+0=4

Nedersta ramen
kulorna med *värdena* 8 och 1 i 1-läge
8+0+0+1=9

Du kan själv knåpa ihop kulramsbilderna för 5, 6, 7 och 8.

I figurens två-systemkolumn har kula i 1-läge markerats med 1 och 0-läge med 0. Man behöver inte skriva ut inledande nollor: 0010_2. Dessa är insignifikanta (utan betydelse). Den lilla tvåan nere till höger behöver inte heller skrivas ut om inget missförstånd kan uppstå. Skriv gärna 5, 6, 7 och 8 i tvåsystemet. En etta eller en nolla i tvåsystemet eller binära systemet (binär=tvåsiffrig) kallas *bit*. Detta är en sammandragning av engelskans *bi*nary digi*t*, binär siffra.

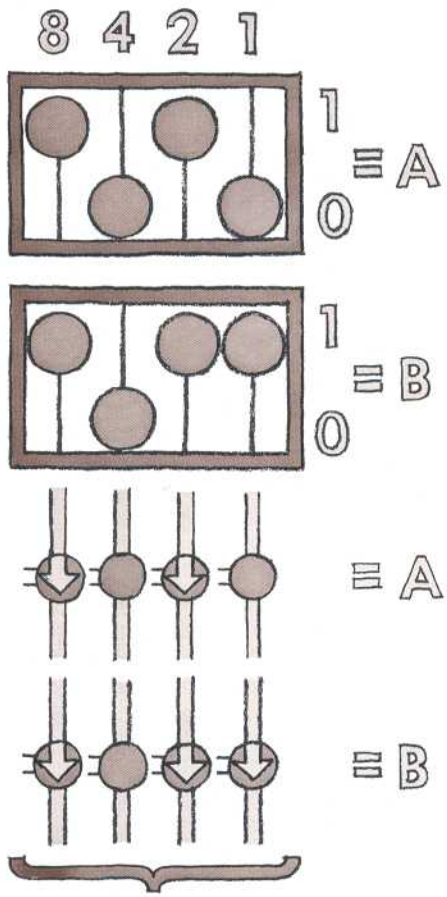


Vi har lärt oss att tolka exempelvis 101 som etthundraett. Men detta gäller bara i det decimala systemet där vi har tio siffersymboler. I det binära systemet betyder 101 (ett-noll-ett) fem. Strängt taget är det ju bara skolfröken, som påstått att en krumelur som ser ut som en tillknycklad metkrok (5) betyder fem!

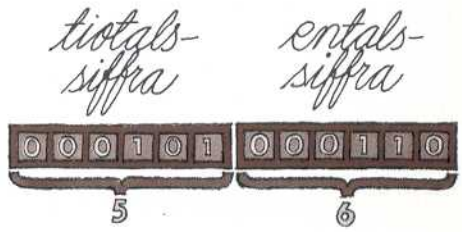
Vi kan fortsätta med flera kombinationer av ettor och nollor och låta dessa betyda bokstäver, siffror och specialtecken som + − , : * ( ). Ett A och ett B kan tänkas lagrade som figuren visar, dels i kulram, dels i transistorregister.

Om vi vill ha med hela svenska alfabetet inklusive W, går det år 29 kombinationer förutom de tio för siffrorna. Nu uppstår frågan: hur många kombinationer kan vi få med fyra bitar? $1111_2=8+4+2+1=15_{10}$ betyder att vi kan få totalt 16 olika kombinationer, eftersom 0000_2 också är en kombination. Ökar vi till fem bitar och låter den femte biten från höger ha värdet 16, får vi 32 kombinationer (kolla!). Detta räcker inte till både siffror och bokstäver. Alltså får vi ta till ytterligare en bit till vänster och ge denna värdet 32. Detta ger 64 kombinationer, vilket är tillräckligt. Vi kan nu binärt koda siffror, bokstäver och specialtecken.

Den 6-bitskod som vi diskuterat oss fram till är bara *ett* exempel på binär kod och kallas BCD-kod (Binary Coded Decimal). Det finns också bl.a. 7-bits och 8-bitskoder av olika slag. En bitgrupp kallas ofta för byte (engelskt uttal). Omfattar bitgruppen 8 bitar heter den på engelska eight-bit byte. Oktad är ett svenskt fackuttryck för samma sak (oktav= åttonde tonen).



Man kan lagra ett tal på flera sätt i datorn. T.ex. 56 kan lagras i en BCD-kod så här:



Men 56 kan också lagras rent binärt: 111000. (Kolla med hjälp av en sexkulig ram!) Detta tar mindre plats. Det är ingen risk att de olika lagringssätten förväxlas. Vilket som gäller anges i datorns program.

Man kan fråga sig varför just det binära talsystemet används i datorer. Det går ju åt många fler siffror än i det decimala talsystemet. Kunde man inte låta olika starka strömmar genom en transistor betyda 0, 1, 2 ... 9? Jo, det kan man, men det går inte att räkna snabbt och exakt på det sättet. Ström=1 och ingen ström =0 kan hanteras både snabbt och exakt med elektronik.

Många tycker, med rätta, att det här med ettor, nollor och det binära talsystemet är svårt. Tycker du att du inte fått grepp om detta behöver du inte ge upp läsningen av fortsättningen. Vi tror inte att resten är så svårsmält.

5. In- och utmatning - ett nyckelproblem

Det finns många sätt att umgås med datorn när det gäller in- och utmatning. I regel använder man något slag av databärare.

Till att börja med ska vi berätta lite om några databärare med tillhörande enheter. De vanligaste finns upptagna i tabellen.

Hålkort, hålkortsläsare och radskrivare har vi berättat om i tidigare avsnitt.

Databärare	Inenhet	Utenhet
Hålkort	Hålkortsläsare	Hålkortsstans
Hålremsa	Hålremsläsare	Hålremsstans
Band och kassettband	Bandenhet	Bandenhet
Skiva	Skivenhet	Skivenhet
Datalista		Radskrivare, skrivmaskin

Hålremsa

Tecken på hålremsan lagras och läses i princip på samma sätt som tecknen på hålkort: hål=1, icke hål =0. Olika spårantal finns (5 till 8) liksom olika koder. Hålremsans fördel: tar mindre plats än en kortbunt, nackdel: besvärligare att ändra i, jämfört med en kortbunt.

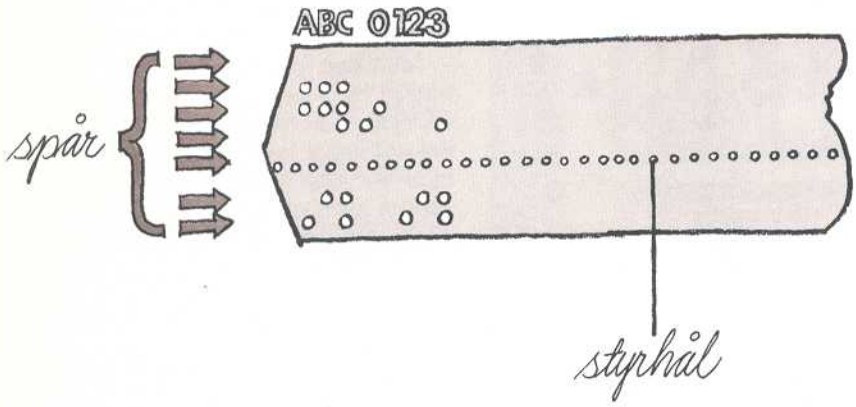
Band, kassettband

Data kan lagras på band (magnetband), ibland kallat bandminne. Det används ofta som sekundärminne.

Bandet är belagt med ett magnetiserbart skikt på samma sätt som bandet för en vanlig hemmabandspelare. Det kan ligga på bandrulle och är då vanligen 1/2 tum (12,7 mm) brett. Bandkassetter av gängse typ förekommer också.

Något förenklat kan man säga att magnetiserade fläckar motsvarar ettor och omagnetiserade motsvarar nollor. Teckenkoden ligger tvärs över bandet, jämför hålremsan. Figuren visar ett så kallat 9-spårsband. Varje tecken representeras av 9 bitar.

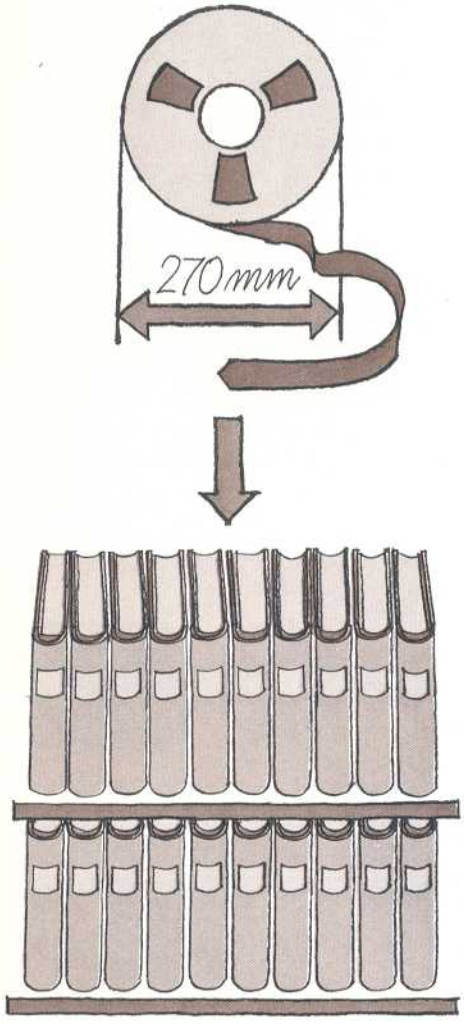
9spårs magnetband (naturlig storlek)



hålremsa (naturlig storlek)

När data skrivs eller läses på band, dras bandet förbi så kallade skriv- och läshuvuden, ett av vardera per spår. Huvudena består av en metallbit med en spole av ledningstråd.

En bandrulle med 320 tecken per cm kan rymma upp till ca 20 miljoner tecken. En A4 helryggspärm, full med normalt tätt skrivna blad, rymmer cirka 1 miljon tecken.



Datorn kan läsa och skriva band med en hastighet av 60 000–120 000 tecken per sekund. Just nu läser du med en hastighet av 15–30 tecken per sekund.

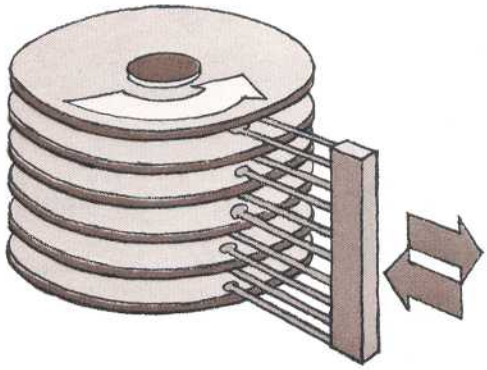
Ett kassettband rymmer cirka 75 000 tecken och läses/skrivs med cirka 3 000 tecken per sekund.

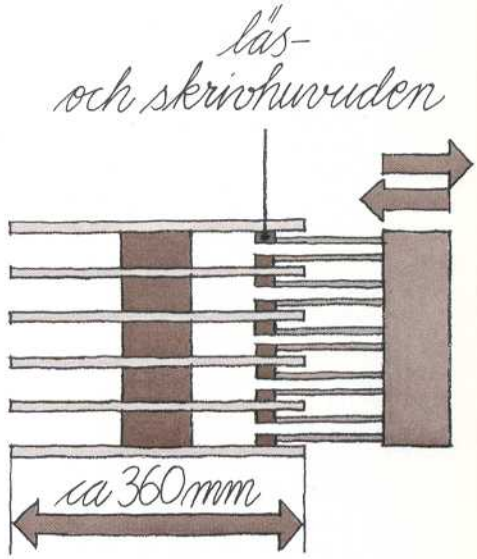
Data på band måste bearbetas i *sekvens*, dvs. i den turordning de ligger lagrade. Det tar för lång tid att spola bandet fram och tillbaka för att söka upp data. Band är ett exempel på *sekvensminne*.

Skivor

Även skivor används som sekundärminnen. Data som ligger lagrade på skivor kan man nå direkt, var de än finns på skivan. Skivminnet är därför ett exempel på *direktminne*.

I större datorer används skivor i packar om 6–12 st. Skivpacken roterar oavbrutet från morgon till kväll med några tusen varv per minut. Läs- och skrivhuvuden sitter i ändarna på så kallade sökarmar. Dessa kan snabbt flyttas in och ut i skivpacken. Liksom banden har skivorna magnetiserbar beläggning. Magnetfläckarna ligger efter varandra i cirkulära spår som ringarna på en måltavla. 200–400 spår per skivsida är vanligt.





De större skivpackarna i D20 rymmer 85 miljoner tecken. Man kan nå data var som helst i skivpacken på fyra hundradels sekunder (medelvärde, så kallad medelåtkomsttid eller medelaccesstid).

Skivenheten i D15-systemet har en fast och en utbytbar skiva och rymmer upp till 10 miljoner tecken.

Läs- och skrivhastigheten på skivor rör sig om några hundratusen tecken upp till en miljon tecken per sekund.

Skrivmaskin

Det finns skrivmaskiner som arbetar enligt samma principer som radskrivaren. De skriver med cirka 150 tecken per sekund eller knappt två rader per sekund. En radskrivare är alltså tio till femton gånger snabbare.

Skrivmaskiner med vanliga typarmar används också. De är de långsammaste elektromekaniska apparaterna. Skrivhastigheten är 15 tecken per sekund – utan felslag. En snabb maskinskrivare klarar drygt tre tecken per sekund – med ett och annat felslag.

6. In- och utmatning på andra sätt

Klarskriftläsning

Det finns maskiner som kan läsa vanliga tecken, maskinskrivna eller handskrivna enligt enkla regler. Denna metod kallas optisk teckenidentifiering eller OCR (Optical Character Recognition, optisk tecken-igenkänning). Tecknen läses av maskinen och lagras på magnetband i någon binär teckenkod. Bandet bearbetas sedan av datorn.

Postgiroblanketterna läses genom optisk teckenidentifiering med cirka 20 blanketter per sek.

Det finns också magnetisk teckenidentifiering. Tecknen är då tryckta med magnetiserbar färg och har speciell utformning. De läses med läshuvuden av samma slag som i en bandenhet och läggs som binärkod på band. Checkblanketter har ofta magnetisk skrift längst ned.

Dataskärm — en korsning mellan TV och skrivmaskin

Dataskärmen är en bekväm in- och utmatningsenhet. Tangentbordet liknar skrivmaskinens. Tecknen kan skrivas på skärmen från tangentbordet. När dessa ska matas in till datorn, trycker man på en sändknapp. Datorn kan också skriva på skärmen. Ofta används dataskärmen som terminal (betyder ändpunkt), ansluten till datorn via telenätet.

dataskärm

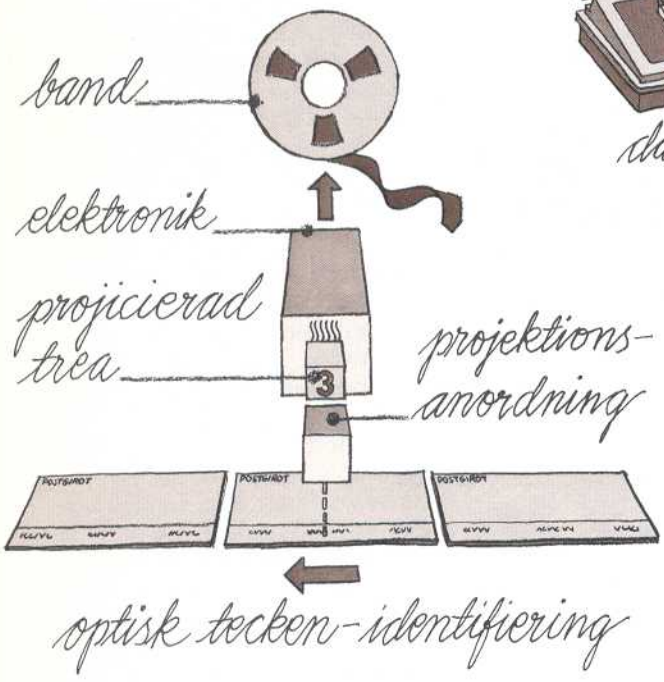
COM eller hur man bantar ned 10 kg till 30 gram utan att egentligen anstränga sig.

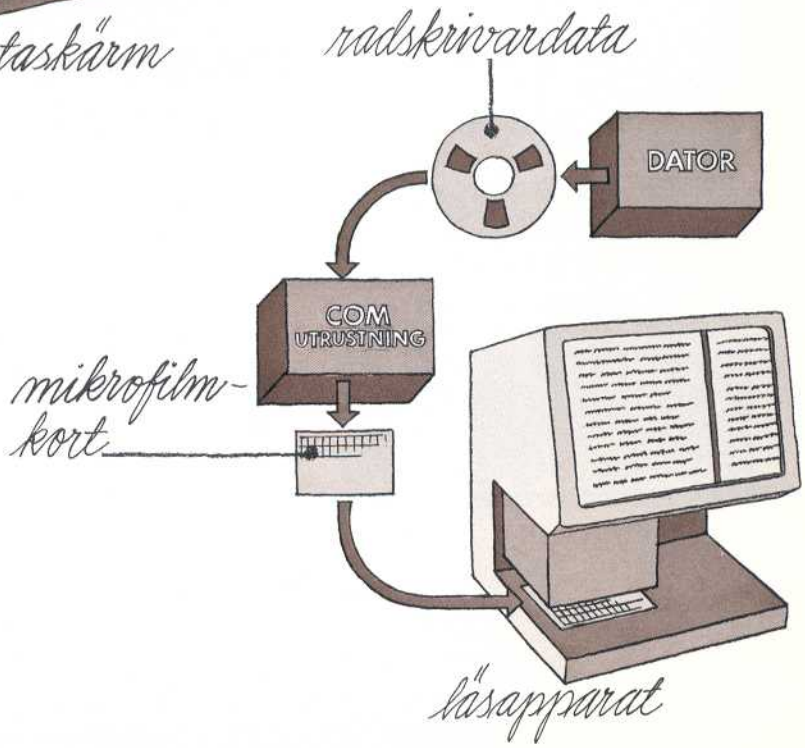
COM betyder Computer Output Microfilm.

I stället för att skriva ut data på radskrivarlistor kan man lagra dessa data på band.

Bandet skickar man sedan till en servicebyrå, som har COM-utrustning (om man inte har en egen förstås, vilket inte är vanligt för närvarande). Där använder man bandet som indatabärare till COM-utrustningen. Utdata kommer i form av radskrivarlistor i mikroformat. Dessa är fotograferade på ett mikrofilmkort. Kortet är stort som ett vykort och rymmer över 200 A4-sidor. Ett sådant kort kallas även mikrofiche eller bara fiche. Mikrofilm i rullar förekommer också.

Mikrofilmen måste läsas i speciella läsapparater. I gengäld slipper man hanteringen (och lagringen) av klumpiga pappersbuntar. 10 kg radskrivarlistor får plats på 30 gram mikrofilmkort!



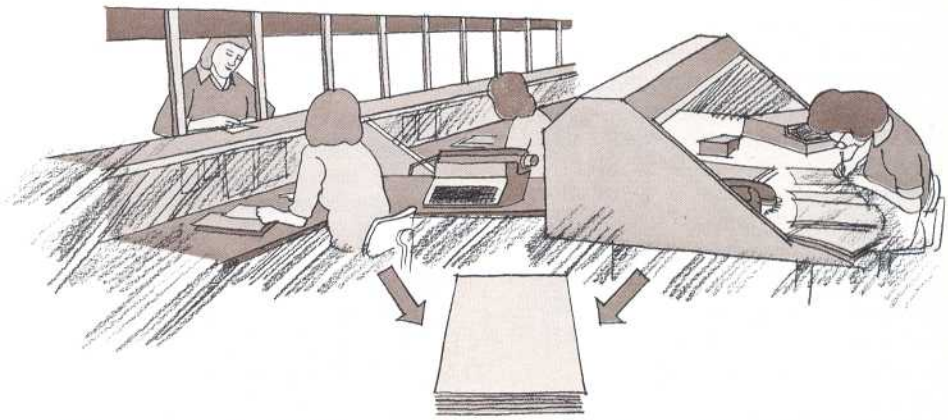


Datainsamling

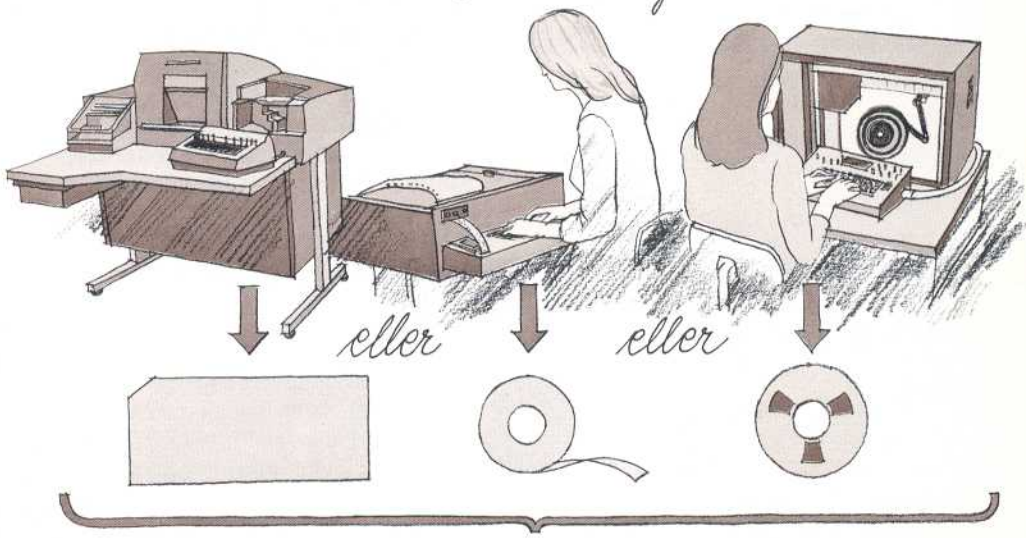
Data kan inte uppstå av sig själva. Data föds i samband med manuell verksamhet — den primära och fundamentala verksamheten.

Ofta skriver man data på blanketter, speciellt framtagna för systemet. Dessa blanketter utgör stansunderlag, som tas om hand av en stansavdelning någonstans. Där stansar man hålkort eller hålremsor. Man kan också från ett tangentbord lägga upp data på band (bandrulle eller kassettband eller skiva). Då använder man bandskrivare eller skivskrivare.

Blanketterna kan också vara utförda för optisk läsning. I så fall läggs data upp på band i en optisk läsare.



stansavdelning



eller eller

till datorn
via posten eller
(med hjälp av terminal) via telenätet

7. Programmet - en arbetsinstruktion

Varje typ av dator reagerar för en speciell uppsättning s k datorinstruktioner (tidigare kallade maskininstruktioner), t ex ”läs från en inenhet”, ”flytta ett tal från primärminnet till aritmetiska enheten”, ”multiplicera två tal”. Programmet, datorns arbetsinstruktion, består av datorinstruktioner. När programmet ska exekveras (=utföras) måste det finnas *lagrat i primärminnet i form av datorinstruktioner, ettor och nollor.*

Instruktionen LÄS PÅ EN INENHET (t ex ett hålkort i hålkortsläsaren) ser ut så här i ett D20-primärminne:

111 111 110 000 000 000 000 001

och MULTIPLICERA TID (som finns i aritmetiska enheten) MED TIMLÖN:

000 100 000 001 101 001 111 000.

Som väl är slipper programmeraren att skriva programmet i denna opraktiska form. De två raddorna av ettor och nollor kan skrivas i ett s k *datorinriktat språk (maskinorienterat språk):*

SCALL, 1;
*,TIMLÖN;

Detta är ett exempel på språket DAC 3 (Datasaab Auto Code).

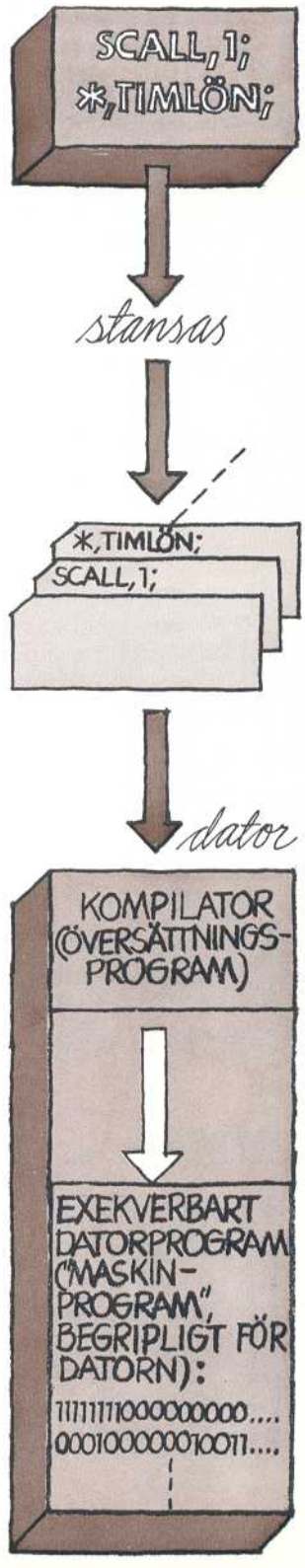
Varje rad motsvarar en datorinstruktion och kan stansas på t ex kort.

Men datorn kan inte reagera på SCALL,1; osv utan detta måste översättas till just de kombinationer av ettor och nollor som datorn är konstruerad för. Översättningen utförs av ett översättningsprogram, en s k *kompilator*. En kompilator är alltså ingen maskin utan ett program, ett ganska komplicerat program, som är dyrbart att utveckla.

Ett datorinriktat språk passar bara de maskiner som det är utvecklat för. DAC3 fungerar bara för D22, D220, D23 och D223.

I slutet av 50-talet gjorde man stora ansträngningar för att utveckla standardspråk, som kunde användas för alla datortyper. Man eftersträvade dessutom att bygga upp språken med hänsyn till de problem, som programmeraren skulle lösa, s k *probleminriktade språk.*

Cobol (Common Business Oriented Language, allmänt affärsorienterat språk) togs fram för administrativ databehandling.



Fortran (FORmula TRANslation, formel översättning) och Algol (ALGOrithmic Language) utvecklades med tanke på teknisk/vetenskaplig databehandling. På Datasaab kompletterades Algol till Algol-Genius (GEnerellt IN-Utmatnings System). Med Geniuskompletteringen blev det möjligt att enkelt beskriva stora datamängder. Varken Algol eller Fortran har sådana möjligheter.

När programmeraren skriver så här i Cobol:

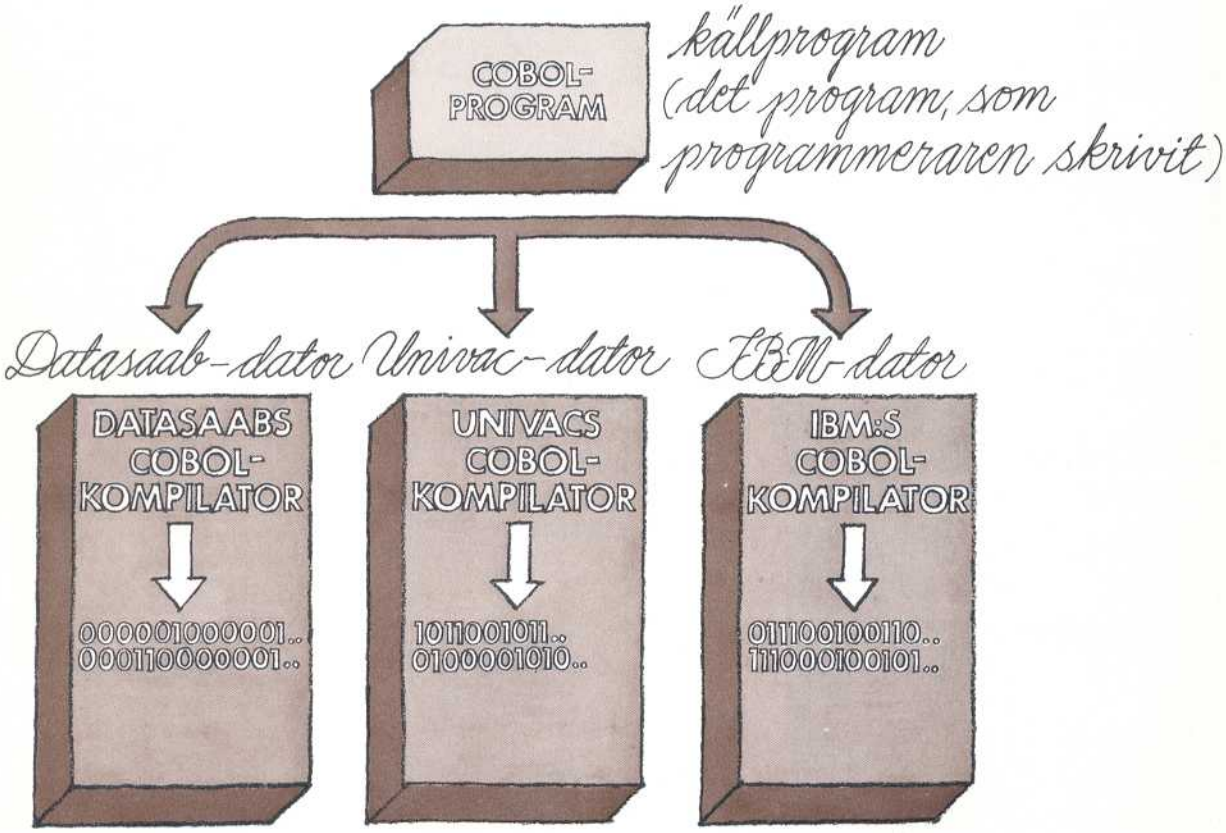
MULTIPLY TID BY TIMLÖN GIVING LÖN.

vill hon eller han att datorn ska göra följande:

- hämta tidsuppgift från en minnescell, som programmeraren kallar TID, och lägga in denna aritmetiska enheten
- multiplicera det som finns i aritmetiska enheten (alltså TID) med den timlöneuppgift, som finns i en minnescell, som programmeraren kallar TIMLÖN
- lagra resultatet av multiplikationen i en minnescell, som programmeraren kallar LÖN

Cobol-meningen ovan måste översättas, kompileras, till sådana kombinationer av ettor och nollor, som datorn reagerar för på avsett sätt. Varje datortyp eller datorfamilj måste ha sin Cobol-kompilator, eftersom deras inre funktioner skiljer sig åt.

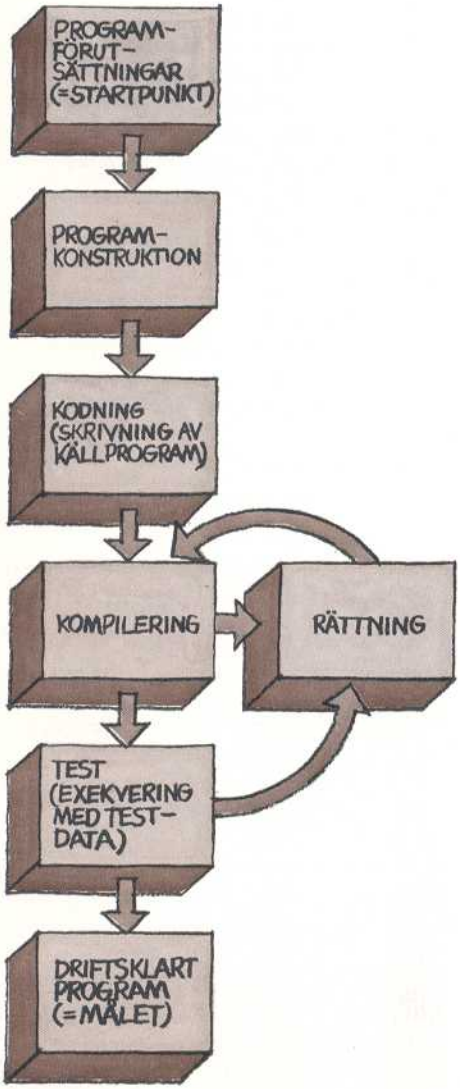
I princip kan ett och samma program, skrivet i ett probleminriktat språk, användas utan ändringar i olika datortillverkares system. Tyvärr förekommer en del små avvikelser från standard hos samtliga tillverkare. Därför måste man i viss utsträckning anpassa programmen, om man ska byta datorfabrikat. Denna anpassning kallas ofta för programkonvertering.



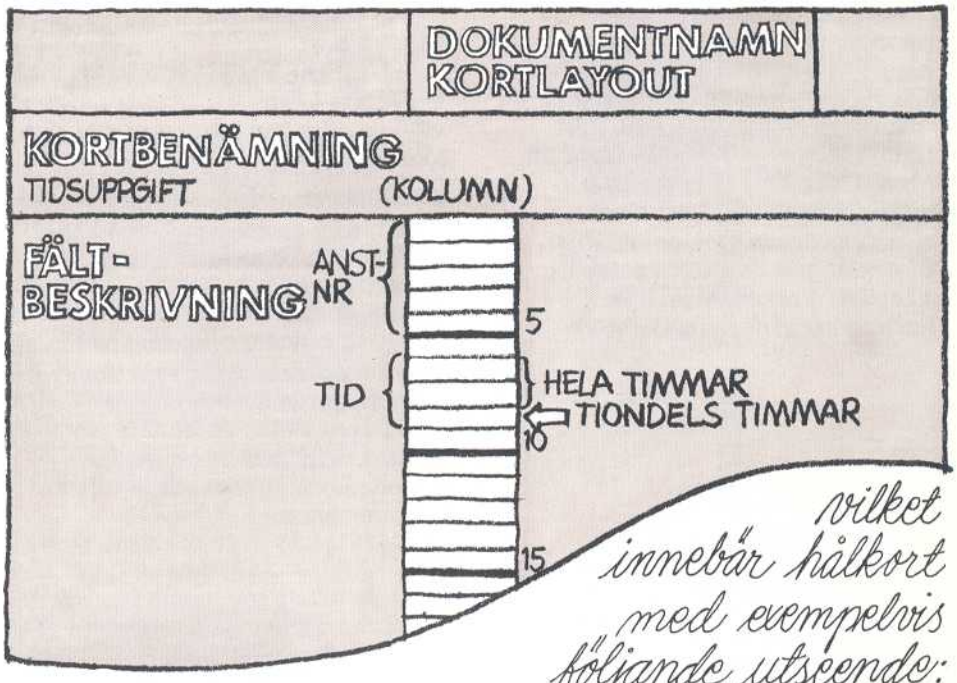
8. Programmera - är det svårt det?

I och för sig är inte programmering svårare än annan verksamhet bara man har lärt sig det! Och att lära sig grunderna i ett programspråk tar en eller ett par veckor. Att lösa svåra problem med väl konstruerade program kan dock kräva några års erfarenhet.

Man kan särskilja olika steg i programmerarens arbete, t.ex. som i figuren nedan.



programmerarens arbete



vilket innebär hålkort med exempelvis följande utseende:

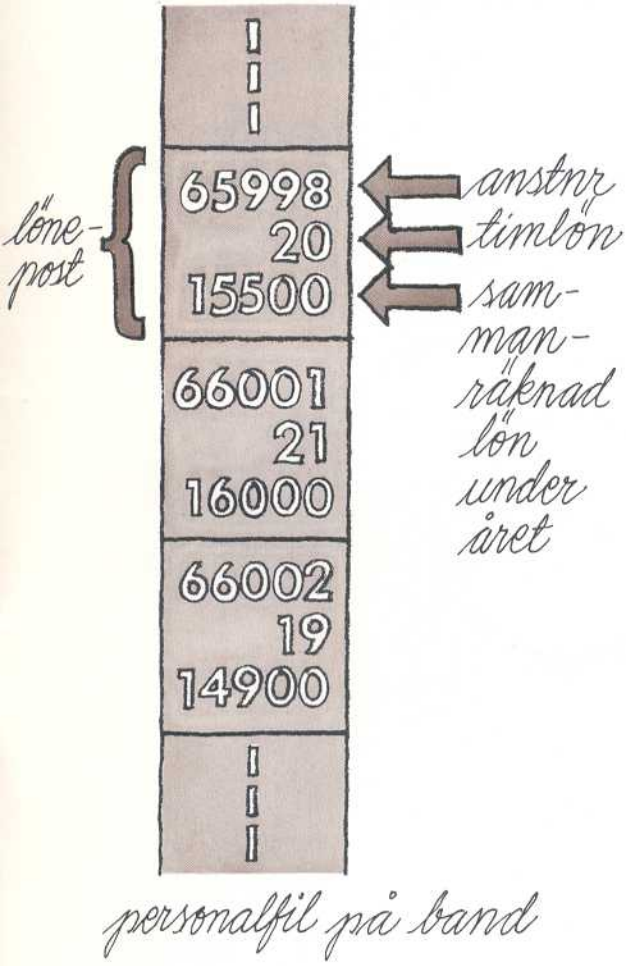
65998 405

Programförutsättningar

Programförutsättningarna kommer från systemeraren, vars arbete vi ska beröra i nästa avsnitt. De omfattar bl.a. detaljerad beskrivning av in- och utdata, hur data ska bearbetas och eventuella kontroller.

Låt oss förutsätta att tiduppgifts-data ska läggas på hålkort i vår lilla lönerutin från tidigare avsnitt. Dessa data kan beskrivas på en blankett enligt figuren.

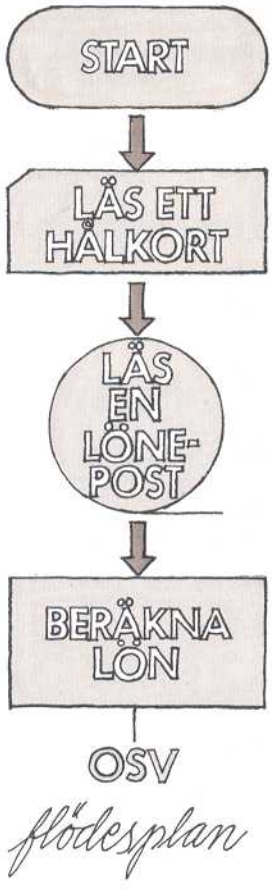
Blanketter finns också för beskrivning av filer (=register) på band eller skiva och radskrivarlistor. Figuren med lönerutinens personalfil på band visar detta i ''klartext''. I praktiken har man som bekant många fler uppgifter i lönesammanhang som skatteavdrag, uttagen semester osv. men detta ändrar ingenting i princip.



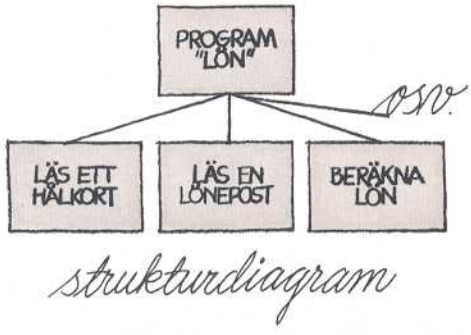
personalfil på band

Programkonstruktion

Programmeraren ska börja med konstruktion av programmet. Han/hon gör en ''ritning'' över hur instruktionerna ska genomlöpas. Ritningen kan vara utförd som en flödesplan eller som ett strukturdiagram. Därefter tar kodningen vid.



flödesplan



strukturdiagram

Kodning

Kodning innebär att programmeraren skriver programmet (källprogrammet) i det aktuella programspråket. När programmet sedan kompileras, kan kompilatorn upptäcka brott mot språkreglerna, så kallade formella fel. Dessa måste rättas, annars blir programmet inte entydigt. Det rättade programmet måste kompileras på nytt.

Test

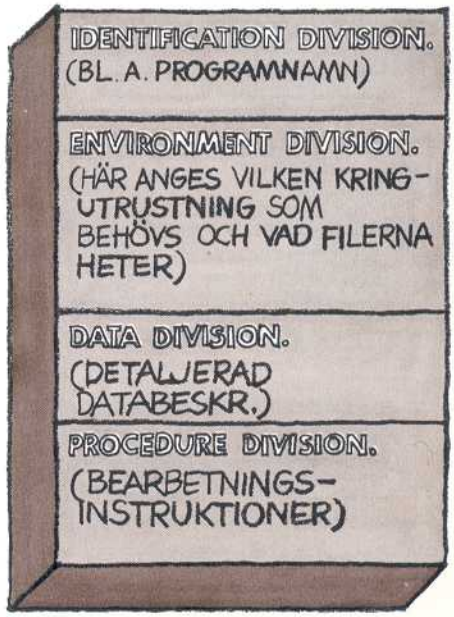
Så tar testverksamheten vid. Programmet körs då med testdata, så att programmeraren kan upptäcka s.k. logiska fel, tankemissar. Dessa kan inte kompilatorn avslöja.

Omsorgsfull programkonstruktion — A och O

Om man lägger ned mycket arbete på programkonstruktionen, kommer kodning och test att utgöra en mindre del av programutvecklingen. Detta gäller speciellt om s.k. programstrukturering utförs metodiskt. Annars kan enbart rättning och test ta halva tiden i anspråk för programutveckling, ibland mer.

Ett Cobol-program

ser ut i princip som i figuren.



cobol-programmets delar

Här är ett utdrag ur PROCEDURE DIVISION för vår lönerutin, något förenklad:

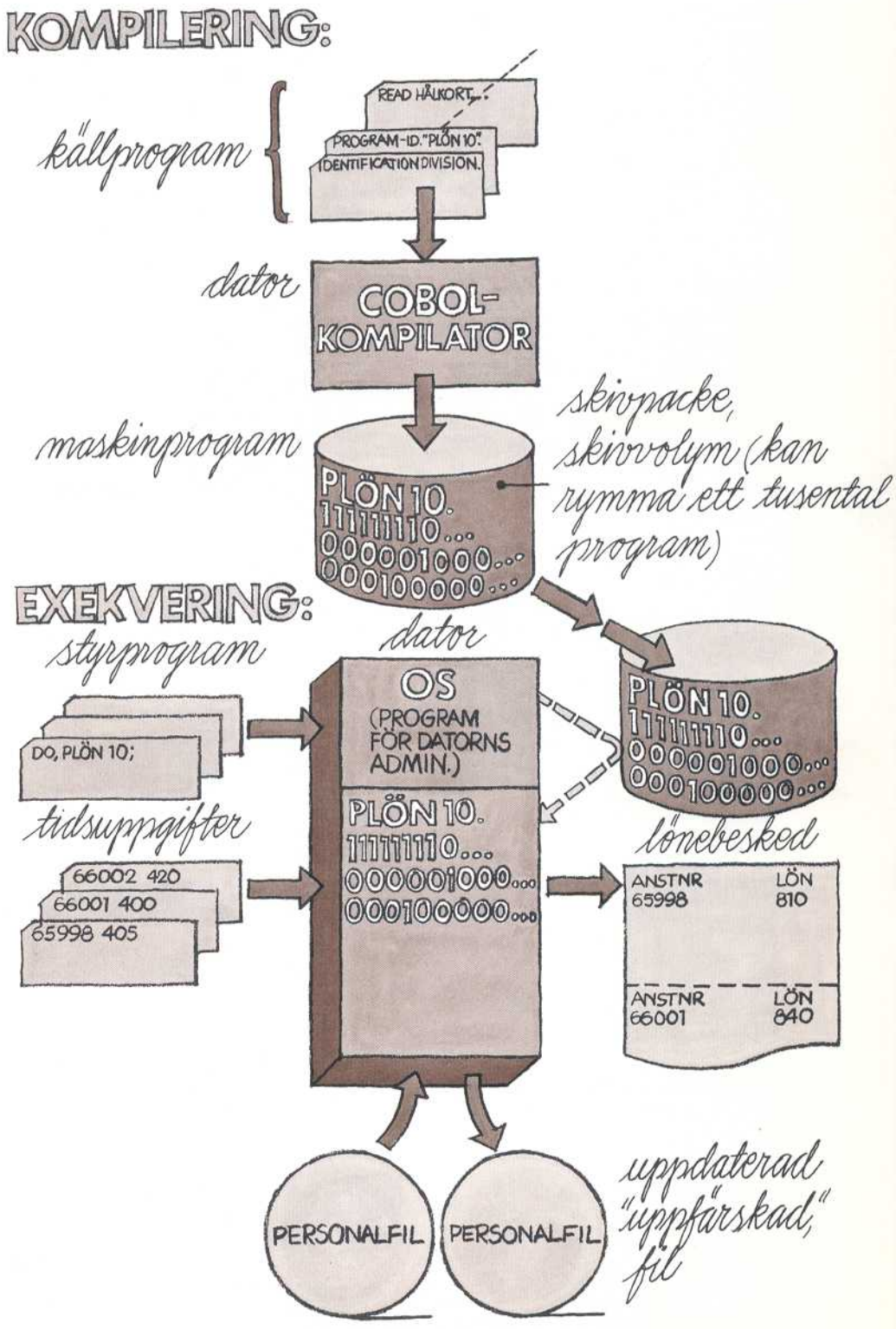
```
LÄS.  READ HÅLKORT AT END
        GO TO SLUT.
      READ PERSONALFIL AT
      END GO TO SLUT.
      MULTIPLY TID BY TIM-
      LÖN GIVING LÖN.
      ADD LÖN TO TOTAL-
      LÖN.
      - - - - - - - - - - -
      - - - - - - - - - - -
      WRITE LÖNEPOST.
      WRITE RADPOST.
      GO TO LÄS.
SLUT.- - - - - - - - - - -
```

En vanlig programstorlek för ett Cobol-program är ett tusental rader, som stansas på lika många hålkort.

Vi ska sluta det här avsnittet med att beskriva hur ett program hamnar på en programfil och sedan körs eller exekveras. Tester, rättningar och förnyade kompileringar utelämnar vi här.

Maskinprogrammet lagras på ett sekundärminne tillsammans med andra program under ett programnamn som måste vara unikt, i vårt fall PLÖN10 (se figuren). För att få PLÖN10 exekverat måste man tala om för datorn att det är just *det* programmet som ska startas upp. Detta, bland andra uppgifter, anges i ett s.k. styrprogram med DO, PLÖN10;

Då läser datorns operativsystem (OS, programsystem för datorns administration) in PLÖN10 från sekundärminnet till primärminnet och startar upp programmet.



9. Systemering - systematiskt sunt förnuft

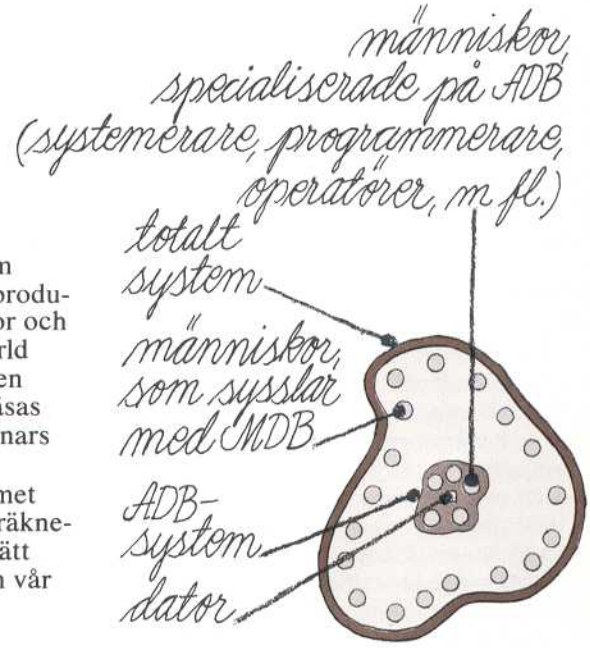
Datorn – anden i flaskhalsen?

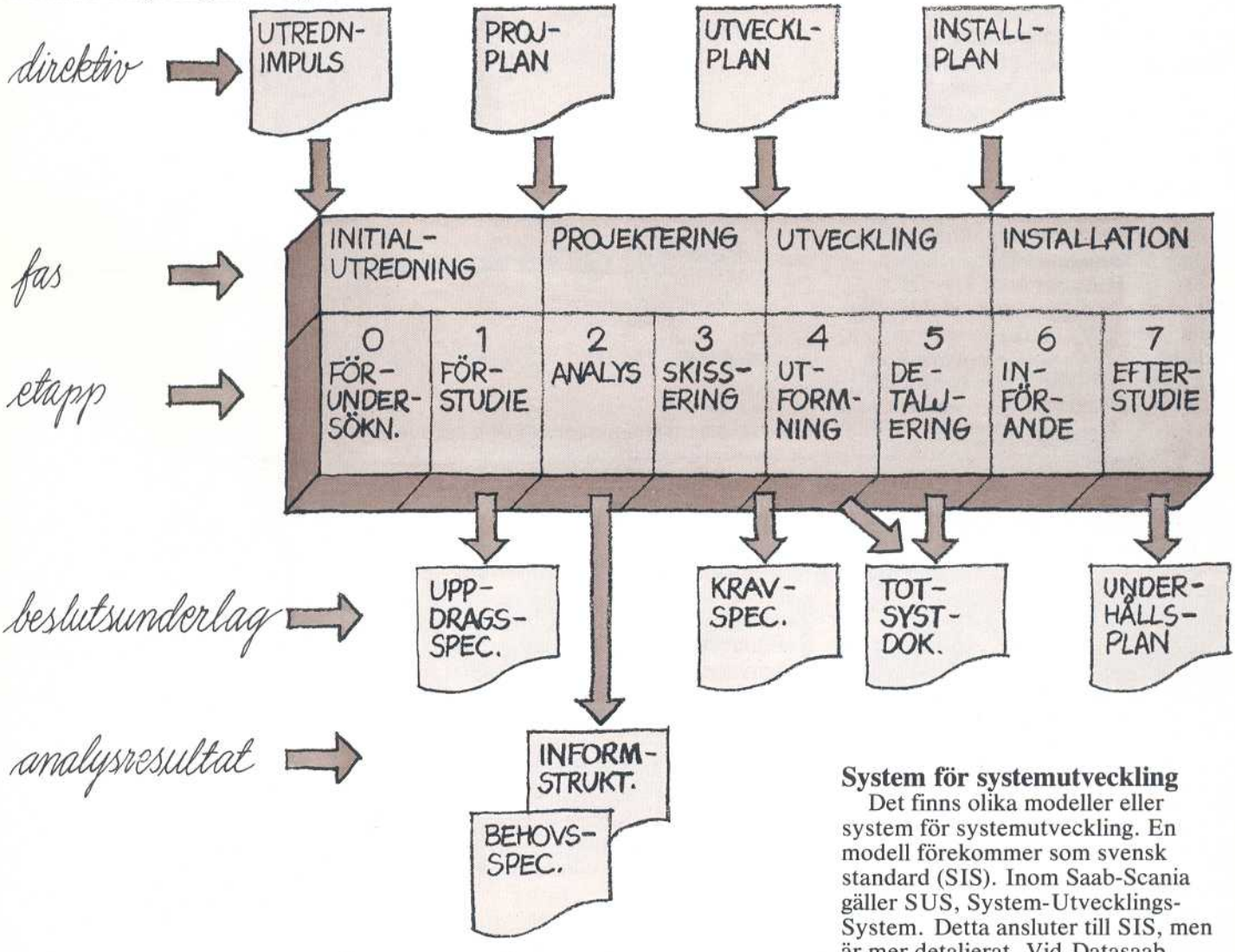
Målsättningen för systemmännens, eller systemerarens, arbete är att få ett totalt system att fungera bra. Med totalt system menar vi allt som sker från och med då data föds till och med då data tolkas till avsedd information. Här räknar vi alltså in även manuell databehandling, MDB.

Man kan inte nog understryka betydelsen av systemanpassning till människan. Tyvärr förekommer inte denna anpassning alltid, vilket resulterar i dåligt fungerande system.

Det är i systemets utkanter som saker och ting verkställs. Där producerar man varor, levererar varor och utför tjänster. Systemets omvärld utgörs alltid av människor. Även det dataskrivna brevet måste läsas och *förstås* av en människa, annars är det värdelöst.

Datorn har i det totala systemet i princip samma status som en räknemaskin eller en skrivmaskin. Rätt använd kan datorn fungera som vår lydiga ande i flaskan.





förenklad översikt av ZSUS

System för systemutveckling

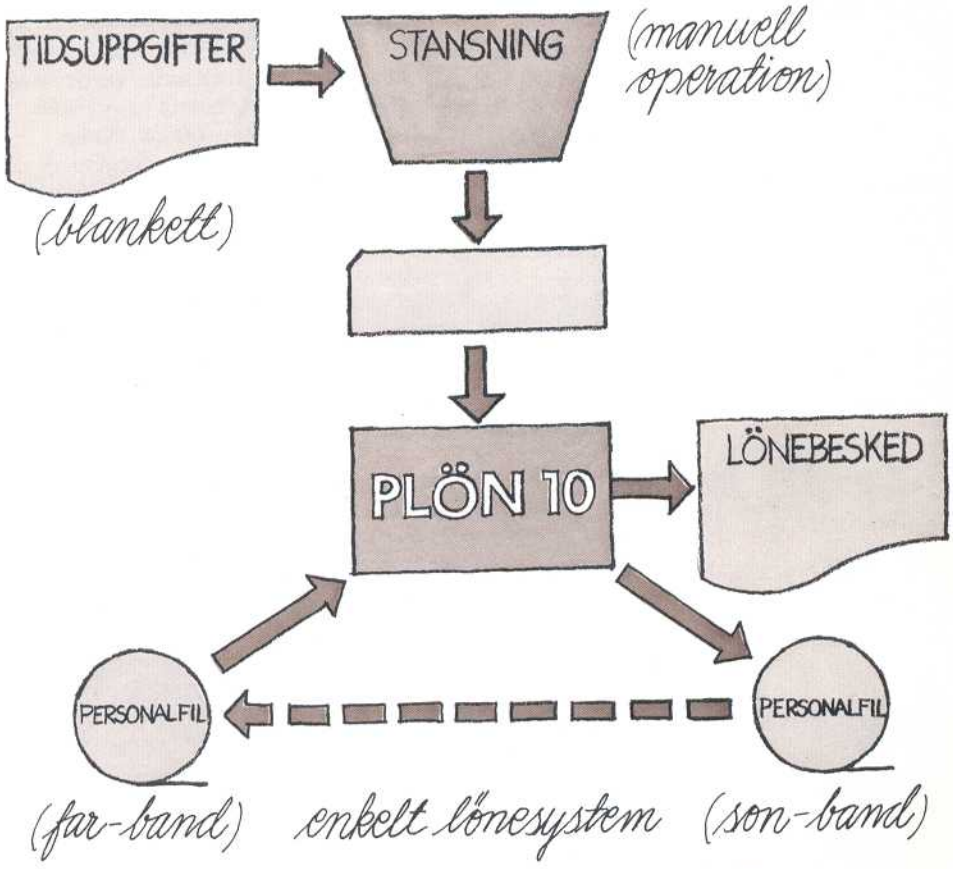
Det finns olika modeller eller system för systemutveckling. En modell förekommer som svensk standard (SIS). Inom Saab-Scania gäller SUS, System-Utvecklings-System. Detta ansluter till SIS, men är mer detaljerat. Vid Datasaab har viss förenkling och bearbetning av SUS resulterat i ZSUS.

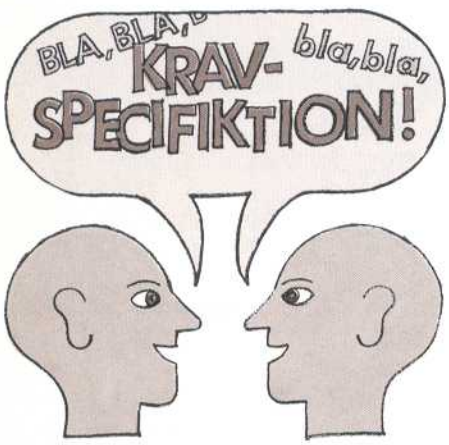
Systemeraren eller gruppen av systemerare måste vara helt på det klara med användarens krav. Därför måste en kravspecifikation utarbetas i intimt samarbete med användaren.

I slutet av varje etapp redovisas etappresultatet för en beslutsgrupp med representanter från användaren. Beslutsgruppens uppgift är att bedöma om det är lönsamt eller ur andra synpunkter önskvärt att fortsätta systemutvecklingen.

I systemdokumentationen ingår en så kallad systemflödesplan eller, som den också kallas, dataflödesplan. Den visar hur data flyter genom systemet. Där framgår också vilka manuella operationer som måste utföras och vilka program som måste köras.

Vårt lilla lönesystem skulle kunna beskrivas i en dataflödesplan enligt figuren. På son-bandet kan bl.a. lagras ackumulerad (hittills hopsummerad) lön för året. Härvid får vi en uppdaterad personalfil. Den streckade pilen markerar att sonbandet används som indatamedium vid nästa körtillfälle. Då fungerar detta band som far-band.





Inför det sista avsnittet i den här broschyren vill vi först presentera en

Sammanfattning

Gamle kamrerns kulram byttes på 60-talet ut mot en med transistorer. Kopplade man in den transistoriserade kulramen till ett vägguttag kunde man lägga ihop två och två en miljon gånger per sekund. Det blev så mycket siffror att man tog miniatyrporträtt på datalistorna för att gamle kamrern skulle få plats. Så skruvade man ihop skrivmaskiner med TV-apparater till dataskärmar som kopplades till telenätet. Då kunde man komma åt datorn den vägen. Alltsammans, inklusive gamle kamrern, komponerade man ihop till ett totalsystem.

10. Datorn i arbete

Multikörning

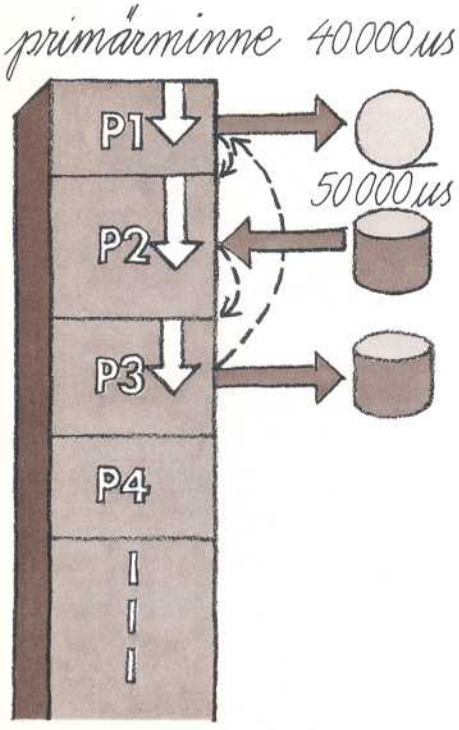
Allt reaktionssnabbare elektronikkomponenter (t.ex. transistorer och IC-komponenter) ser dagens ljus. Man skulle i dag gärna vilja skynda på elektronerna, som masar omkring i trådarna med knappt ljusets hastighet (1 080 000 000 km/tim). Elektromekaniken kan inte hinna med i svängarna. Därför skulle centralenheten (som är elektronisk) under ett programgenomlopp i stort sett bara få vänta på kringutrustningen (som är elektromekanisk). Vid multikörning utnyttjar man centralenheten mycket bättre. Då har man flera program i primärminnet, se figuren. När program 1 har avverkat ett antal instruktioner, ska programmet ha en skvätt data (ett block) utmatade på band. Det tar ca 40 000 mikrosek. Men på den tiden kan D22 utföra t.ex. 13 000 additioner. Därför tar centralenheten itu med program 2 medan program 1 väntar. När program 2 avverkat en del instruktioner ska det hämta data från en skivenhet. Att hitta och överföra data kan ta 50 000 mikrosek. Då startas program 3 upp, som så småningom kanske ska mata ut data på en skiva. Kanhända program 1 äntligen har fått sitt bandblock utmatat. I så fall fortsätts genomloppet från det ställe där det hade satts i väntläge. Det ser operativsystemet till. På det här sättet åstadkommer man ett dator-håll-i-gång, som ger fullt upp att göra för både centralenhet och kringutrustning.

Batch, remote batch, direkt respons och time-sharing

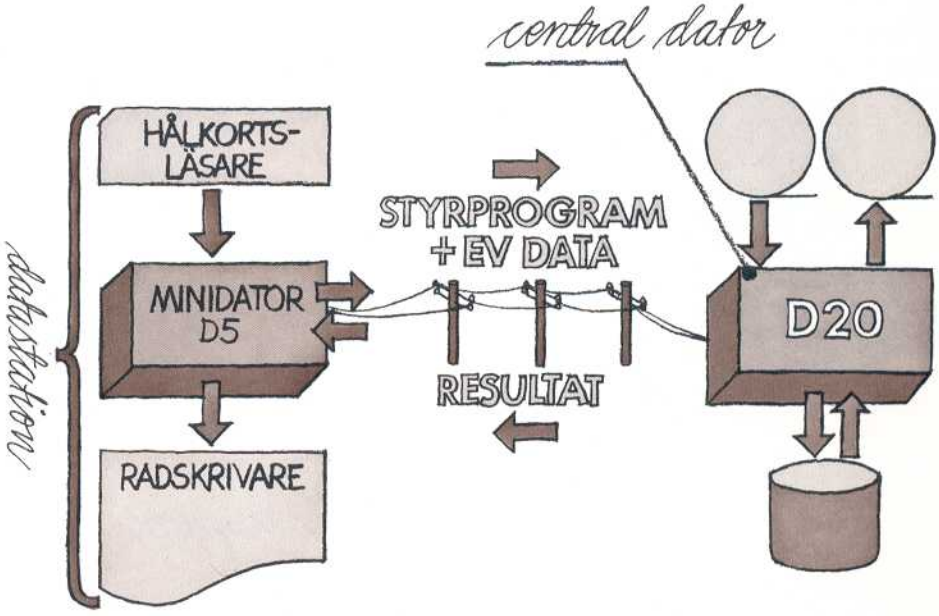
är begrepp som man ofta träffar på i ADB-sammanhang.

Batch-bearbetning kan översättas med satsvis bearbetning. Då lämnar man in till datacentralen en beställning på körning av ett s.k. jobb. Ett jobb kan innebära exekvering av ett eller oftast flera program. I beställningen anger man vilka volymer (=band eller skivpackar) som ska bearbetas och bifogar ett styrprogram. Ofta bifogas också en kortbunt med data. Har man inte datacentralen inom bekvämt räckhåll, måste man alltså ta något slag av postgång till hjälp både för inlämning av batch-jobb och distribution av resultatet.

Remote batch-bearbetning kallas på svenska för satsvis fjärrbearbetning. Det liknar vanlig satsvis bearbetning men i stället för postverket använder man televerket. Man kan då använda en s.k. datastation, se figuren, där en del viktiga enheter medtagits.



multikörning



satsvis fjärrbearbetning (remote batch)

Direkt respons kan också kallas fråge-svars-system. Man kan på en terminal välja ett s.k. responsprogram och skriva en fråga (enligt ett förutbestämt format) som sänds till den centrala datorn, se figuren. Inom någon eller några sekunder skriver responsprogrammet ut svaret på terminalen. I ett direkt responssystem kan man också uppdatera filer, t.ex. en lagerfil, med inlevererad kvantitet av en artikel. Lagerfilen blir då ständigt aktuell, den uppdateras i verklig tid, reell tid. Man talar därför om realtidsbearbetning i detta sammanhang.

Time-sharing (tidsdelning) brukar man kalla system där man hyr ut terminaler och ställer datorresurser till förfogande.

Lagom är bäst och fakturasystemet klarar skivan

Datasaab har bland sina produkter färdiga programsystem för olika viktiga tillämpningar. Vi ska bara nämna ett par som exempel.

Har man en verksamhet, som kräver stor lagerhållning står man inför två ytterlighetsproblem:

1. Med en liten kvantitet av varje artikel i lager riskerar man att en del artiklar tar slut innan man fått hem nya.
2. Med stora kvantiteter av varje artikel krävs en stor kapitalinsats och större lagerutrymmen. Resultat: onödiga ränteförluster på det kapital som satsas på lagerartiklarna.

Kvantiteten av olika artiklar bör alltså inte vara för liten och inte heller för stor. Hur mycket man bör ha i lager beror bl.a. på förväntad åtgång och leverantörens leveranstid. Lagom ingår i ett D20-programpaket för lagerstyrning. Med Lagom kan man t.ex. få fram hur mycket man bör ha i lager för att lagerstorleken just ska bli lagom. I lagerstyrningssystemet ingår också lagerredovisning, framtagning av underlag för inköp m.m.

Datorn i D15-systemet samarbetar alltid med en skivenhet. På skivan lagras data och program, som är åtkomliga från olika arbetsplatser.

I D15-systemet ingår en mängd tillämpningspaket som t.ex. fakturering. Så snart en order leveransbekräftats i D15 orderrutin faktureras den. Fakturautskrift kan man starta upp när man vill. Då skrivs fakturan ut automatiskt med alla erforderliga uppgifter.

Men gamle kamrern, vad sysslar han med nu för tiden? Jo tack, han bär numera slimmade skjortor i djärva mönster och slipsar i matchande färger. Han har fått lite extra spänst i gången. Han är nämligen chef för stanscentralen ...