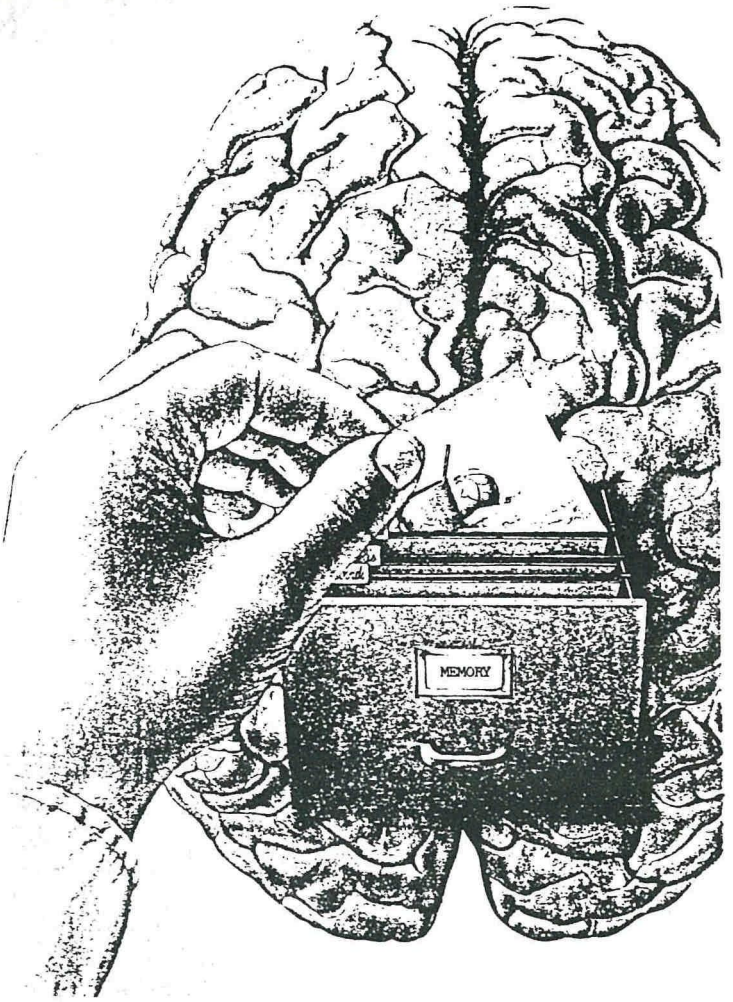
HUSORGAN FÖR
MIKRODATORFÖRENINGEN
STACKEN
Redaktör: Björn Ahlén

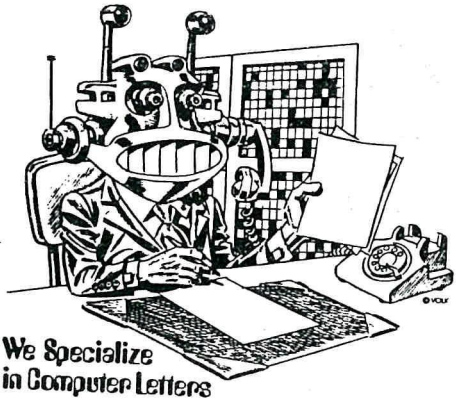
1-80

Bränn hemma

2708

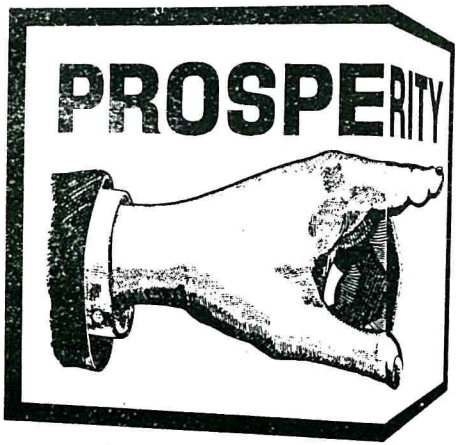


"C" & UNIX



STACKPOINTER

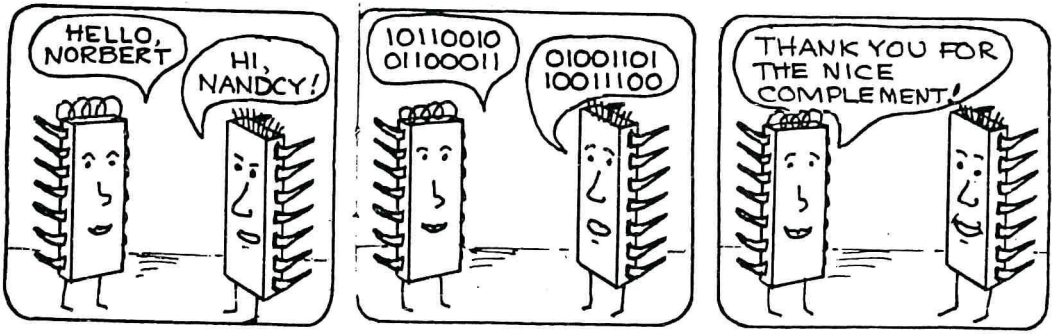
REDAKTÖREN HAR ORDET



BETALA DIN MEDLEMSAVGIFT NU SÅ ATT DU FÅR STACKPOINTER UNDER HELA 1980 !

Du vill väl hålla dig uppdaterad om allt kul som händer i Datorföreningen Stacken! Studiebesök, föredragsserier, TOPS-körningar, operativsystem, kompilatorer, mikrodatorbyggen med Z80, Z8000 m.fl., hårdvarubeskrivningar för modem, PROM-programmerare etc, besök på mässor, utställningar, osv, osv. Det finns helt enkelt väldigt mycket intressant och spännande i Stacken just nu! Var med och njut bland computerholics genom att sätta in endast sjutti pix (70:-) på pg 433 01 15-9 "STACKEN". Du får då Stackpointer under ett år framåt, information om samköp etc och är välkommen på alla möten!

Björn Ahlén, redaktör



I redaktionen: Björn Ahlén, Hans Nordström. Stackpointer utges av datorföreningen Stacken, Box 5079, 141 05 HUDDINGE, pg 433 01-9. Ansv. utgivare: Nils Söderman, Fredriksonsv. 12, HÄGERSTEN.

ÅRSMÖTE

DAGORDNING FÖR ARSMÖTE TORSDAGEN DEN 10 APRIL 1980

I FÖRENINGEN STACKEN

§ 1. Mötet öppnas
§ 2. Val av två personer att jämte ordförande justera mötesprotokollet. Justeringsmännen skall tillika tjänstgöra som rösträknare under mötet
§ 3. Val av ordförande för mötet
§ 4. Val av sekreterare för mötet
§ 5. Tillkännnagivande av vid mötet uppgjord röstlängd
§ 6. Frågan om mötet är stadgeenligt utlyst
§ 7. Frågan om dagordningens godkännande
§ 8. Framläggande av styrelse- och kassaberättelse
§ 9. Framläggande av revisionsberättelse
§ 10. Frågan om styrelsens ansvarsfrihet för det gångna arbetsåret
§ 11. Val av styrelsemedlemmar samt revisorer med suppleant
§ 12. Val av ledamöter till styrelsevalberedningen
§ 13. Behandling av styrelseförslag
13.1 Förslag till mötesdagar: Första torsdagen i varje månad utom 8/5 1980 och 8/1 1981. Årsmöte hålles 2 april 1981
13.2 Styrelsen föreslår att följande grupper aktiveras bestående av 3 personer vardera:
- Mjukvarugrupp
- Hårdvarugrupp
- Teknisk grupp

§ 14. Behandling av budget för innevarande år
§ 15. Fastställande av medlemsavgifter för året efter det år under vilket årsmötet hålls
§ 16. Synpunkter på verksamheten för innevarande år
§ 17. Övriga frågor
§ 18. Mötets avslutande

G A I V
=======

Den 15:e Mars fick Sverige besök av den eminenta franska konstnärsgruppen GAIV, som sysslar med elektronmusik och datakonst. Med ett franskt universitet som hemmabas turnerar dessa datorentusiaster land och rike kring, och håller konsert med sina 400 kg datorer och synthesizers. det var på Tekniska museet som de satte upp sin utrustning och spelade och gjorde otroliga saker på en färgmonitor inför en häpen publik. Tillställningen försiggick i samband med "HEJ DATOR"-utställningen på tekn. museet. (Den utställningen hade inte så mycket att visa för en datorentusiast, tyckte jag. Bara en byte: FF)

Två dagar senare hölls ett seminarium innan man packade ihop och drog vidare. Många människor var där bl.a. från EMS, elekronmusikstudion i Stockholm, för att höra fransoserna berätta om sina datorer och program. Man använder inte bara de små systemen (bl.a. en Heath LSI-11) utan även större saker på sitt universitet. Man arbetar rätt mycket i fransk tradition med att processa naturliga ljud. Man kör bl.a. jätteprogram som fungerar som perfekta filter.

Efteråt blev det diskussion och demonstration av hårdvaran. Dessa personer påstår sig vara mer datorentusiaster än musiker och konstnärer. Programmet ADVENTure var tydligen välkänt även i Frankrike! Överhuvud taget var det allmäna intrycket att man nog gärna ville göra saker i realtid med specialbyggd hårdvara, men det kostar både tid och francs, så man krunchar ofta sina vågformer med mjukvara istället.

På seminariet hörde jag även ett rykte som sa att den PDP-11 som TV inköpt för att göra grafik till kärnkrafsomröstningen ska sedan användas till mer konstnärlig datorgrafik. Verkar lovande!

/Perre L

LOKALPROBLEMET

Som ni kanske märkt, är vi utan lokal för tillfället. Den gamla hemtrevliga källaren under NADA har återgått till att användas som terminalrum, p.g.a. att terminalsalen på 4:e våningen ska byggas om. Ombyggnaden blir inte klar förrän i december, påstås det från vanligtvis välunderrättade källor. Det innebär att vi måste försöka låna någon annan lokal. Alla förslag är givetvis välkomna. STACKENs verksamhet är ju beroende av att vi har någonstans vi kan samlas och ha våra grejor. Ta en funderare och prata med folk! /Perre L

Feedback

ABC-KLUBBEN HAR STARTAT!

25 januari hölls det första mötet på KTH med ca 200 närvarande. Pga det enorma intresset beslöt man dela upp sig i 12 olika intressegrupper:

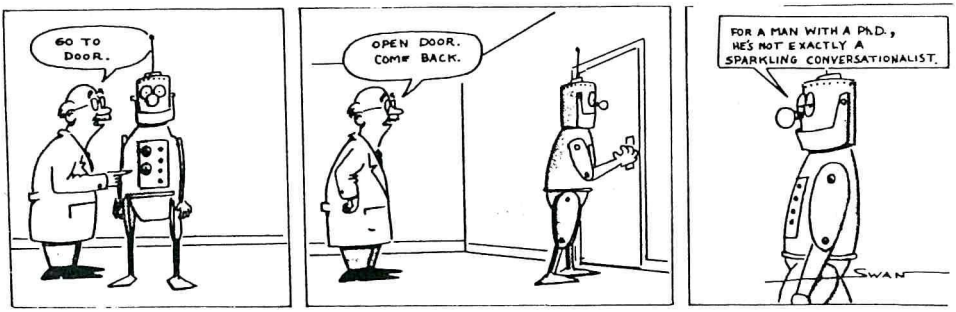
Sport & Föreningar
Undervisning
Styr & Mätteknik
Datakonst & Musik
Amatörradio
Nybörjare
Datakommunikation
Textbehandling
Spel & Grafik
Programmeringsteknik
Företagare
Matematik & Statistik

Nästa möte som är ett årsmöte hålls tisdagen den 22 april kl. 19.00 i sal F1, KTH (Lindstedtsv 22).

Efter en massa prat hit och dit, återbud, uppskjutningar, etc. etc. blev det i alla fall av, resan till LYSATOR, alltså. Strax före kl. 9.00 på lördagsmorgonen kunde den morgonpigge stockholmaren skåda en märklig syn ett 20-tal nymornade datorentusiaster PIGGA och på påtagligt GLATT HUMÖR samlades i en buss bakom Ahlens. Strax därefter drog bussen iväg med kurs mot datastaden Linköping. Redan några minuter efter start kom ivriga diskussioner igång om allt från ett numeriskt museiföremål (som utgjorde en present till LYSATOR) till features i LISP. Efter ett snabbt interrupt på ESSO Motor Hotel i Norrköping där vi blev servade med en tidig lunch, susade bussen in i Linköping. Vår chauffös, Inger, visade sig på styva linan och lyckades tråckla in bussen på en parkeringsplats vid LiTH, där en skateboardburen LYSARE tog emot och lotsade oss till LYSATORS huvudlokal som låg i andra änden av byggnaden, dit vi nådde efter en lång och stärkande promenad.

Så hade äntligen 2 av Sveriges större datasällskap strålat samman. Vi överräckte vår "present", och blev sedan på gott studiebesöksmaner uppdelade i tre grupper som lotsades runt till sevärdheterna av kunniga personer. Och nu fick vi verkligen se oss mätta på datorer av alla slag! Allt från LYSATORS egna gamla D21- och D22-system som upptog en hel lokal (bullret och värmen därinne var rätt påfrestande) till små persondatorer modell APPLE II och ABC80. Vi fick se det vittomsusade operativsystemet UNIX gående på en PDP-11, och den stora starka LISBETH (en DEC-20 med TOPS-20 monitor och INTERLISP, bl.a.). Men vi blev nog trots allt mest imponerade av den uråldriga Datasaab D21-an. Den är stor som attan och ser ut som en dator SKA göra, iallafall enligt skämtteckningarna. Massor med skåp, radskrivare och bandaggregat. Den har inga skivminnen, utan man använder n st. bandaggregat med entums band istället. Vi blev bl.a. visade en ALGOL-GENIUS kompilering där det behövdes FYRA band igång samtidigt. En klart imponerande syn! Eftersom det var så många som såg på, gick naturligtvis flera band av, och Martin NIL passade på att ta ett par komprometterande bilder av en lysare insnärjd i avslitna band. LYSATORS senaste kelgris på D21-an var en plotter (även den i jätteformat) där man ritade tjusiga kurvor inför allas hänförda blickar. Här fick vi se prov på ett fint TEAMWORK. (en person hade grejat OS:et, en kompilatorn, en plottern och ytterligare en hade klurat fram de matematiska formlerna till de tjusiga kurvorna.) Vi fick även veta att man implementerat LISP (!) på D21an. Andra attraktioner fanns att beskåda i LYSATORS "huvudlokal" (där kaffebryggaren var belägen). En prototyp av D5:an åtdrog sig tyvärr bara ett måttligt intresse, utkonkurrerad av APPLET och ABC80:n. Här fick vi återigen se prov på lysarnas programmeringsenergi. Monitorn på APPLET hette TOPS-20 och betedde sig till synes precis som LISBETH. ABC80-systemet hade nyligen fått en floppy, så något liknande är väl att vänta även på den.

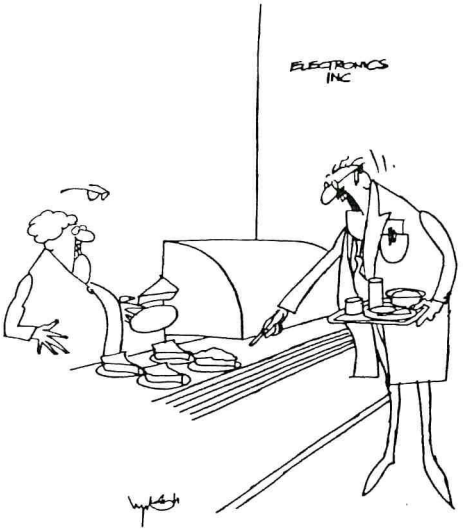
När så alla varit runt till de olika attraktionerna, blev vi bjudna på kaffe och bullar. Därefter vidtog lite mer fria aktiviteter, där var och en kunde ägna sig mer åt sitt eget favoritgebit. Det pratades en hel del om LiTH:s Programmeringsspråk LiTHP (ursäkta, LISP) och relaterat stuff, såsom PROLOG och LISP-maskiner. LYSATOR är flitiga på att implementera detta eminenta språk, det finns numera även på deras egen "uppfinning", 16-bitslice-maskinen LYS-16.



Klockan 5 beordrades som planerat samling i bussen för en tripp ner på stans nyöppnade kineskrog, där vi blev trakterade med vällagad mat och gott vin. Stället var välvalt, maten superb. (fem byte: FF FF FF FF FF). Stämningen steg ytterligare, och planer på en invers folkförflyttning började ta form. Efter desserten (gissa vad?) blev vi bussade tillbaka till skolan och LYSATOR. Mera shop talk, ide´utbyte, och klockan susade snabbt iväg mot 20-strecket, då avfärden enligt hyreskontraktet på bussen var spikad. En sista samling av hela gänget, LYSATORS ordf. Sören Tirfing överräckte en avskedpresent: en tjusig kurva producerad av D21-ans plotter, och så iväg i flygande fläng.

Efter denna genomlyckade heldag, håller vi nu på att diskutera ett returbesök för LYSATOR till STACKEN. Ännu är det inte för sent att komma med ideer till aktiviteter. Dessutom bör vi i anständighetens namn fått fram åtminstone en temporär lokal, där vi kan hänga upp presenten!

/Perre L



"...Pi!!.... That's the quantity I forgot to factor in!!!"

STUDIEBESÖKET PÅ IBM

En torsdagkväll härförleden fick vi gratis buss-skjuss av Traveller AB till Järfälla och makro(dator)företaget IBM. Efter att ha blivit kontrollräknade några gånger, med checksumma och carry, fick vi komma in och titta på Många Makalösa Maskiner. Järfällafabriken tillverkar bl. a. radskrivare som är mer svenska än Volvo; 70% svenska delar i radskrivaren mot 50% i Hisingstraktorn. Det är snabba rackare som tillverkas. För att få tillräckligt snabba hammar-rörelser har man tagit ballistiken till hjälp (läran om kul(-)rörelse). Utan vare sig turbo eller EBK kommer man upp i prestanda som i ett annat sammanhang skulle motsvara följande: Du åker i ett tåg som färdas med 200 km hastighet. Längs banvallen går ett spjälstaket med 20 cm mellan spjälorna. På andra sidan spjälstaketet finns det en äng med olika sorters blommor. Tag nu i farten och plocka enbart en viss sorts blomma, en mellan varje spjäla! Aningen stressat kanske....

För materialkontroll och liknande hade man ett högst förträffligt svepelektronmikroskop med zoom och tillhörande minidator. Man kunde zooma in en bacill och se att bacillen hade en födelsemärksliknande prick av främmande substans på ryggen. Nu kommer det goaste: Ratta in hårkorset på bacillens födelsemärke och skriv 10 GLURR, KRUX X,Y som instruktion till minidatorn. Efter någon minut visar sig då på en grafisk skärm en spektralanalys och texten "18,9% Fe, 23,7% Cu" etc. Man påstod sig ha löst många besvärliga materialproblem med den här apparaten. Tro det? Den här metoden får väl närmast betraktas som fusk. Människan ska arbeta i sitt anletes svett eller ...?

Efter rundvandring i fabriken fick vi komma in i det allra heligaste: datacentralen. Den stod givetvis inte det övriga efter. Bl. a. hade man ett s.k. "massminne". När IBM talar om massminne... 3850 Mass Storage Unit var en burk stor som två Alfaskåp (Nej Alfa Romeo gör inga skåp, men Volvoskåp då!) innehållande 1000 bandrullar a 50 miljoner tecken. För den händige behövs varken räknedosa eller bambuskott (sticka) för att räkna ut att det blir ju för bövelen FEMTI GIGABYTE online. Per burk. Ujujuj! Det blir ju 12 miljoner fullskrivna A4-sidor.

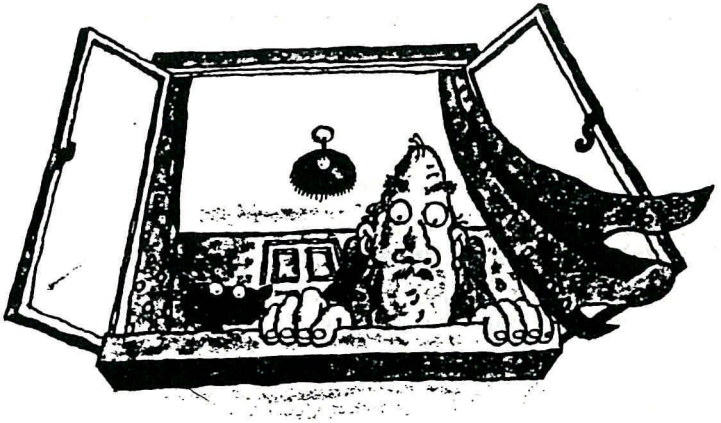
3600 km pyjamaspapper (Stockholm-Istanbul).
2400 kartonger papper.
20 lastbilar!

Nej förresten, vi vill dumpa maskinläsbart. Ta ut det på remsa istället. Hoppsan, det blir visst 127 000 km, drygt 3 varv runt jorden. Vad sa du? Är snabbstansen trasig? Rulla fram Dunderklumpen då, den gamla kära Teletype-33:an. Dom brukar vara outslitliga, vilket kan behövas om man som här skall hålla på i 158 1/2 år!

Nåja, IBM hade bot för utskriftsproblemet också. Med hjälp av Laserskrivaren 3800 kunde man tömma en låda papper på 12 minuter, dvs vi klarar av utskriften på ca 20 dygn i stället. 3 veckor låter klart vilsammare än 158 1/2 år! Givetvis kunde den skriva vilka typfonter som helst. I princip kunde man t.ex. få ut PL/I Reference Manual i tysk frakturstil om man ville.

Efter alla häftiga maskiner var det gott att smörja kråset med en makrofika med allehanda rätter. Ett klart föredöme för kommande studiebesöksföretag (Hint, hint...).

/Björn Ahlen



ABC80

-de många möjligheternas mikrodator. Tack vare utbyggnadskorten

Till ABC 80 finns ett rikt utbud av utbyggnadskort i Europatormat, såväl för 4680-bussen som den rena ABC-bussen. Korten för de båda bussarna är pinkompatibla beträffande I/O-kort men inte minneskort. Detta innebär att Du måste använda minneskort ur 4680-serien till DataDisc 80 och kort ur ABC-serien till FD2. Till båda systemen finns dessutom separata expansionschassin för inmontering av korten.

ABC-SERIEN

ABC RAM	8 K statiska RAM för expansion av ABC 80:s primärminne *Art.nr 55 50950-01*
ABC PIO	32 bit parallell I/O TTL-kompatibel *Art.nr 55 50952-01*
ABC SIO	Seriell två-kanalers I/O för synkron kommunikation. Programmerbar överföringshastighet *Art.nr 55 50953-01*
ABC IEC	Generell IEC-buss (IEEE 488) för tillkoppling av t ex mätinstrument (HP-IB) *Art.nr 55 50954-01*
ABC ADC	12-bitars A/D-omvandlare. 32 analoga ingångar *Art.nr 55 50955-01*
ABC DAC	12-bitars D/A-omvandlare. 2 analoga utgångar *Art.nr 55 50956-01*

DataBoard 4680-SERIEN

4680/2055	8 k statiskt RAM för expansion av ABC 80:s primärminne *Art.nr 55 50900-01*
4680/3032	8 k EPROM för fast lagring av t ex drivrutiner *Art.nr 55 50903-01*
4680/4006	32 ut/16 in TTL-kompatibel I/O för avläsning av t ex mätinstrument och styrning av yttre enheter *Art.nr 55 50903-01*
4680/4007	8 reläutgångar för styrning av lampor, kontaktorer, reläer etc. *Art.nr 55 50904-01*
4680/4008	16 optoingångar för t ex data-insamling *Art.nr 55 50905-01*
4680/2081	Färgvideo-RAM. View-data kompatibel. Färgmonitor eller modifierad FTV kan användas *Art.nr 55 50910-01*

Utöver dessa finns ytterligare ett 50-tal kort ur DataBoard 4680-serien t ex AD/DA-omvandlare och IEC-buss. Tips om användningsområden får Du i boken "Bygg ut ABC 80 med DataBoard 4680".

...och de praktiska tillbehören

LUXOR

Luxor AB, Division Datorer, 591 83 Motala

UNIX - ett OS med många "!"

På annat ställe i detta nr berättas om hur vi i Data City (Linköping) fick stifta bekantskap med terminaloperativsystemet UNIX. Den första versionen av UNIX togs fram 1969-70 på en PDP-7 dator hos Bell Laboratories i USA. Enligt vissa källor skrevs det av två skottar med COBOLfingrar (nerslitna stumpar, du vet). Kommandospråket utmärker sig nämligen av en viss korthuggenhet typ "uisge ja tk".

Om man bortser från vissa detaljer som att PRINT heter "cat" och att RENAME heter "mv" (move) så är det ett mycket intressant operativsystem. Det är unikt, därför att det har skrivits av användare, inte av dataloger. Det är fullt av eleganta, nytänkta lösningar på programutvecklarens vardagsproblem. Några huvudpunkter;

1. Hierarkiskt filsystem.
2. Allting är filer. Terminalen är en fil, minnet är en fil, periferienheterna är filer, ja t.o.m. filerna är filer!
3. Multipla processer. Fleranvändare.
4. "Pipes" för kommunikation mellan processer/kommandon.
5. Inbyggda funktioner av diverse märkliga slag, t.ex. textformattering, fotosättning, stavfelsrättning av vanlig text, sortering m.m.
6. Det mesta är skrivet i högnivåspråket "C", varför det är tämligen portabelt.

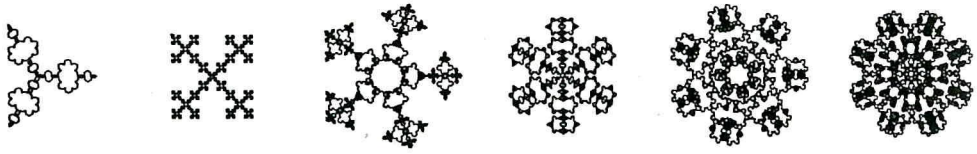
I UNIX finns flera olika programspråk som standard, bl.a. naturligtvis "C" som används för att skriva operativsystemet. F.ö. finns bl.a. assembler, Fortran, PASCAL, Snobol, APL, Ratfor (Rationell Fortran) och bc, ett interaktivt C-liknande språk men med aritmetik i obegränsad precision(!). UNIX innehåller också någonting man försiktigtvis kallar "en dialekt av BASIC". Se bara följande:

```
10 comment this is a rem
20 for i=1 10 prompt i
30 goto 3*i+10
40 if a<b<c
50 print "Ge x och y :"
60 x=expr()
70 y=expr()
80 fi
90 hypo=1000(X,Y)
100 prompt "Hypotenusan :; hypo
110 done
1000 comment beräkna hypotenusan ur tvenne kateter
1010 return (sqr(arg(1)*arg(7)+arg(2)*arg(21)))
```

Den vanligaste maskinen för UNIX torde vara PDP-11 men versioner finns för bl.a. Interdata 8/32, IBM 360/370, Honeywell 6000, SEL80, Nova, Eclipse. Från vanligtvis välunderrättat håll meddelas att Zilog håller på och skriver ihop ett häftigt UNIX-system för Z8000. T.o.m en Z80-version kallad OMNIX, för floppysystem, har nyligen annonserats i Byte.

Vi hoppas kunna återkomma mera till detta operativsystem i ett följande nummer.

/Björn Ahlen



"C"

C är ett intressant programmeringsspråk för främst system-programmering. Det togs fram 1973-74 på Bell laboratories i USA för att bl.a. skriva terminaloperativsystemet UNIX. (Se separat artikel i detta nr.)

Varför heter det "C" ? Jo, naturligtvis därför att det bygger på det tidigare programmeringsspråket "B"! C som programmeringsspråk är ganska kortfattat utan att därför vara så svårläst som t.ex. APL. Det har naturligtvis faciliteter för strukturerad programmering, modularisering, pekarhantering, egna typer osv, som krävs av ett modernt programmeringsspråk.

Ett första C-program:

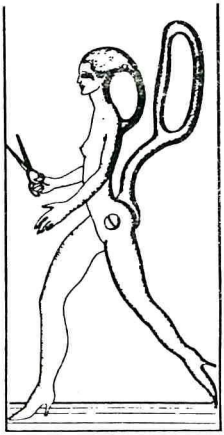
Program 1.

```
main() ä
      printf("C-VITAMIN")
å
```

Ett C-program består av ett antal "funktioner", motsvarande procedurer i PLZ, PASCAL m.fl. Det måste alltid finnas en main-funktion. Parenteserna efter main betecknar en tom parameterlista. "ä" och "å" (vänster och höger fiskmås) markerar funktionens början och slut.

Program 2.

```
main() ä
          int TAL1, TAL2, SUMMA
          TAL1=123; TAL2=456;
          SUMMA=TAL1+TAL2;
          printf("SUMMAN BLIR %d ön", SUMMA)
å
```

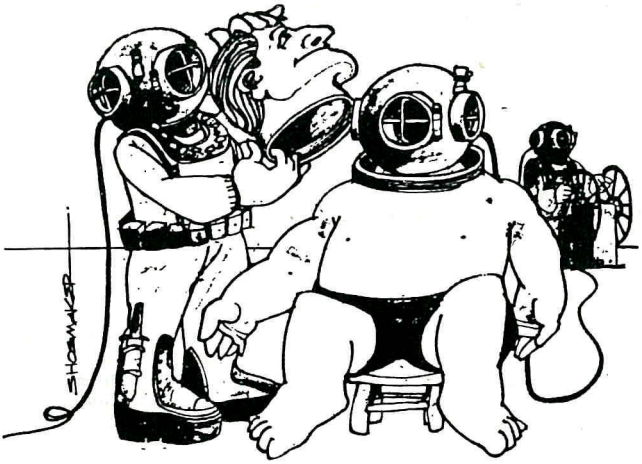


Int betyder att de följande talen deklareras att vara av typ integer.
"%d" i printf markerar plats för ett tal som skal skrivas ut decimalt (!).
"ön" betyder att vi skall ge ett "newline"-tecken, dvs radframmatning.

Program 3.

```
main()ä
      char tkn
      while(tkn=getchar()!='öö')
            putchar(tkn);
å
```

Här läser vi från en fil till en annan tills End-of-file. EOF signaleras i C av att getchar returnerar tecknet 'öö' dvs ASCII "NUL". "!=" betuder skilt ifrån, dvs "<>" i andra språk. While-satsen har syntaxen "while (uttryck) sats", där sats naturligtvis kan vara flera satser om man så önskar. Funktionen getchar anropas som här läser ett tecken till tkn. Uttrycket (tkn=getchar()) får värdet av tilldelningen, dvs det lästa tecknet. Detta jämförs nu med 'öö', dvs End-of-file. När uttrycket blir falskt, ramlar vi ur whileloopen. PRESTO!



C har också ett annat styrvillkor som kallas "villkorligt uttryck".

```
a<b ? a : b;
```

betyder: om a<b är sant får hela uttrycket värdet a, annars värdet b.

För att sätta x till min(a,b) dvs. det minsta av talen a och b kan vi skriva antingen

```
if (a<b)
     x=a;          eller          x=(a<b ? a:b);
 else
     x=b;
```

Program 4.

```
while (( tecken=getchar()) != '\0')
     putchar (( 'A' <= tecken && tecken <= 'Z')
               ? tecken-'A'+'a' : tecken );
```

Detta program läser en textfil och skriver till en annan fil med alla stora bokstäver översatta till små! (Skillnaden mellan stora och små bokstäver är en konstant, 'A'=hex 41 och 'a'=hex 61, 'B'=hex 42, 'b'=hex 62 o.s.v.)

"C" har två intressanta operatorer, "++" (inkrementera) och "--" (dekrementera). Dessa kan användas både före och efter en variabel för både pre- och post-inkrement t.ex.

Antag att k har värde 5.

```
x = ++k
```

ökar k med 1 och lägger värdet (6) i x.

```
x = k++
```

lägger värdet av k (5) i x och ökar sedan k med 1.

Fler intressanta operatorer

"=-" och "=+"

x =- 5 betyder minska x med 5. x =+ 5 öka x med 5.
x = -5 betyder x tilldelas värde -5. x = +5 x tilldelas 5.

"C" har också flyttal i både enkel och dubbel precision. Alla beräkningar utförs i dubbel precision.

Strängar kan deklareras enl.

```
char *msg "Detta är en sträng ön";
char *keyword ÄÖ ä
     "if", "else", "for", 0 ö;
```

Som avslutning tar vi nu ett ordentligt "C"-program. Med litet kännedom om Fortran ser du säkert vad programmet gör! (Hi)

/Björn Ahlén

```
int messg[2],mutex[2];  /* messg[0] for read, messg[1] for write*/

main()
{ pipe(messg);pipe(mutex);
write(mutex[1],"x",1);
  if (fork()==0) {        /* Child-1 */
    child('1', "I'm number one");
    exit(0);}
  else if (fork()==0) { /* Child-2 */
    child('2', "I'm number two");
    exit(0);}
  else {                  /* Parent */
    parent();
    parent();
    exit(0);}}

child(no, mes)
char no, *mes;
{ int i;
char ch;
read(mutex[0],&ch,1);
  write(messg[1], &no, 1);
  sleep(1);
  i=0;
  while (mes[i]) {
    write(messg[1], &mes[i], 1);
    i= i+1;}
  write(messg[1], "\0", 1);
  write(mutex[1],"x",1);}

parent()
{ char ch;
  read(messg[0], &ch, 1);
  printf("Message from %c\n", ch);
  while (-1) {            /* -1 = TRUE */
    read(messg[0], &ch, 1);
    if (ch=='\0') goto ex;
    printf("%c", ch);}
ex:
  printf("\n");}
```

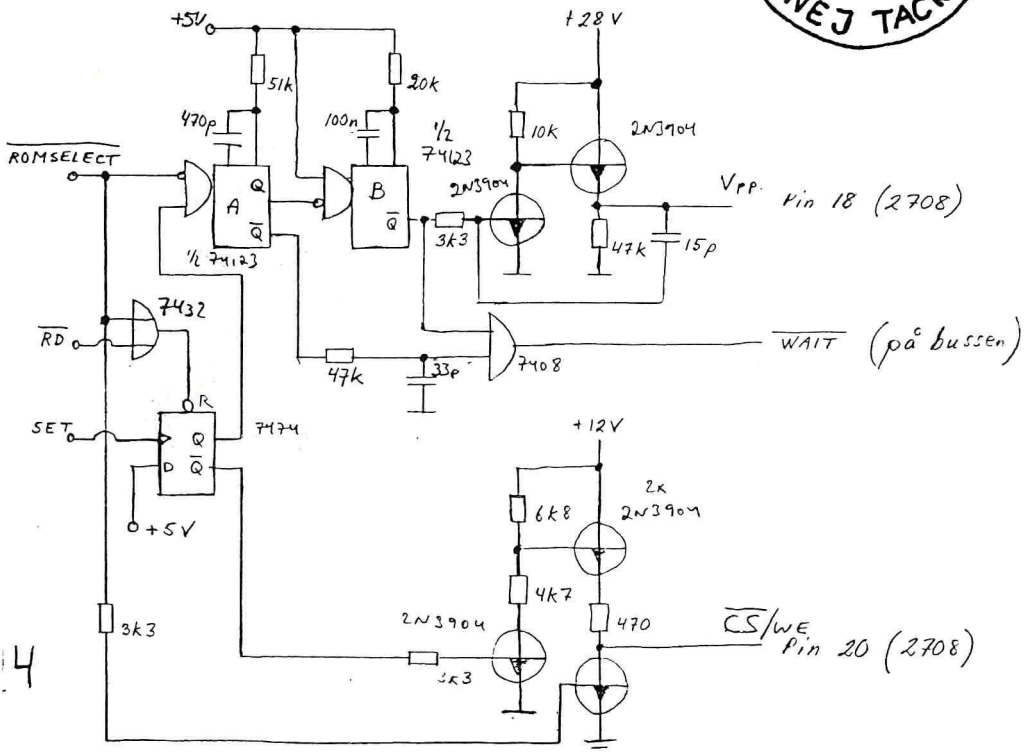
hårda varupaket

Enkel 2708-programmerare.

Följande superenkla koppling gör det möjligt att programmera 2708 direkt på plats. Med hjälp av 74123 en-skjutare hänger man CPU:n (Z-80) i WAIT med adress och data utlagda och skickar ut en programmeringspuls på ca 0,7 ms. Eftersom nästan alla använder dynamiska RAM, kan det vara lämpligt att efter 0,7 ms WAIT raska på litet med refreshen. Därför gör man efter varje programmeringspuls en tom loop, DJNZ $. Någon intresserad som hinner vira ihop ett kort att användas av Stackens medlemmar?

```
        LD    A,255
PROG1   LD    HL,(STARTADRESS)
        LD    BC,EPROMSIZE
        LD    DE,EPROMADRESS
LOOP    LDI
        PUSH  BC
        LD    B,100
        DJNZ  $                 ; 100 refresh pulser
        POP   BC
        JP    PE,LOOP
        DEC   A
        JR    NZ,PROG1
        RET
```

2708 – PROGRAMMERARE



MASKINSPRÅK PÅ ABC80 -KAN DE' VA' NÅT?

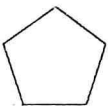
Den programmerare som verkligen vill utnyttja sin dator til 110% kommer snabbt att få behov av att skriva åtminstone vissa delar av sitt program direkt i maskinspråk. Det kan gälla tillfällen då man t.ex. behöver göra tidskritiska operationer. Exempelvis ligga och vänta på någon statussignal, och sedan snabbt läsa in data från en yttre enhet. Eller en musikrutin som lägger ut en frekvens på en datautgång, där man vill ha en exakt kontroll över frekvens och tid. Eller en snabb sorteringsrutin eller sökrutin.

På en persondator typ ABC80 är det mycket lätt att komma åt maskinnivån från BASIC. Dessutom har vi att göra med en av de trevligare 8-bit mikroprocessorerna, en Z80. Och vi vet ganska väl vad som finns var i minnet. Med POKE, PEEK och CALL kan vi skriva och läsa direkt i minnet, samt anropa maskinspråksrutiner.

Låt oss nu säga att vi har kopplat en ABC80 till en våg, där vi kan läsa in vikten från ett interface som är kopplat till en I/O-port. Men det är bråttom att läsa in data, för det är flera siffror som kommer snabbt efter varandra. Så vi skriver en liten rutin i assembler för att sköta den detaljen. Resten av programmet skriver vi naturligtvis i BASIC.

När assemblerprogrammet är skriven, ska det ASSEMBLERAS, dvs. översättas från instruktioner typ "LD A,(HL)" och "CALL FOOBAR" till motsvarande maskininstruktioner. Detta kan man för det mesta göra för hand. Det är bara att räkna ut eller slå upp varje instruktion för sig. Är det ett större program (mer än 10 rader) kan det vara bra att ha ett program som gör detta, en ASSEMBLER. Men det GÅR att "handassemblera" längre program. Det gäller bara att inte göra några fel.

När så programmet är assemblerat, ska det läggas in byte för byte någonstans i minnet. Nu kan vi göra på flera olika sätt. Större datorsystem har oftast någon form av LOADER, ett program som läser från en fil och lägger upp kod i minnet. Man kan göra något liknande på ABC80, dvs. låta huvudprogrammet läsa från en fil och lägga upp rutinen. Detta är särskilt användbart när vi har ont om utrymme i huvudprogrammet. Ett exempel på hur det kan se ut:



```
100 A%=65408% : N%=17% : REM STARTADRESS & ANTAL BYTES SOM SKA LADDAS
110 OPEN "FOO.D1" ASFILE 1%
120 FOR I%=0% TO N%-1% : INPUT #1%,D% : POKE A%+I%,D% : NEXT I%
130 CLOSE 1%
```

En liten nackdel med denna metod är att det blir fler filer att hålla reda på, vilket kan bli ganska jobbigt om man inte har en diskettenhet, utan måste ha filen på kassett. En variant är då att lägga maskinspråksprogrammet i DATA-satser. Programmet blir ganska likt föregående exempel:

```
100 A%=65408% : REM STARTADRESS
110 RESTORE 1000
120 READ N% : REM ANTAL BYTES SOM SKA LADDAS
130 FOR I%=0% TO N%-1% : READ D% : POKE A%+I%,D% : NEXT I%
.
.
1000 DATA 17
1010 DATA 33,0,0,219,0,203,103,192,219,32,203,111,200,33,37,252,201
```

Tyvärr går det åt ganska mycket minne för att lagra data på det här viset, i stort sett en byte per tecken. Ett mer minnessnålt lagringssätt är att lägga data direkt i POKE-satser. Så här:

```
100 A%=65408% : REM STARTADRESS
110 POKE A%+0%,33%,0%,0%,219%,0%,203%,103%,192%,219%,32%
120 POKE A%+10%,203%,111%,200%,33%,37%,252%,201%
```

Då lagras varje byte som ska "pokas" in som ett heltal, vilket endast tar upp två byte. Tyvärr blir nog programmet en aning mer svårläst.

När vi nu har lagrat upp en maskinspråksrutin, vill vi anropa den. På ABC80 finns det två nästan likadana funktioner som gör det: CALL( ) och CALL (,). De är funktioner och ska användas t.ex. som SQR( )-funktionen. Exempel:

```
100 X%=CALL(A9%)
```

Om A9%=65408% så hoppar programmet då till adress 65408 och utför maskinspråksprogrammet som förhoppningsvis ligger där. När vi senare kommer till maskininstruktionen RET (dvs 201 decimalt) återgår vi till vårt BASIC-program, och variabeln X% får värdet av innehållet i registerpar HL i Z80-processorn. Ibland har vi ingen glädje av detta, men oftast kan det vara bra att kunna överföra värden från maskinspråksprogrammet. Vi kan även överföra ett värde till maskinspråksprogrammet. Då använder vi en liknande konstruktion:

```
100 X%=CALL(A9%,P3%)
```

Här fungerar det precis som i exemplet ovan, bara med den skillnaden att värdet av variabeln P3% kommer att läggas i register DE i Z80-processorn vid anropet. Vi kan alltså överföra ETT heltal in till rutinen, och få tillbaka ETT heltal vid återhoppet. Vill vi överföra fler, måste vi "poka" undan dessa någonstans i minnet. Då är det smidigt att låta den överförda parametern istället utgöra ADRESSEN till var våra data finns. Sedan är det programmets sak att gräva upp dessa. Tekniken går givetvis att använda i båda riktningarna, både vid inhopp till maskinspråksrutinen och återhopp till BASIC-programmet.

Var ska vi nu lägga upp våra maskinspråksprogram och poka undan våra data? Tittar vi på en minneskarta på ABC80, ser vi att det finns lite ledigt minne högst upp. Men det är bara 128 bytes, och den arean vill man gärna använda till andra saker, t.ex. någon slags "COMMON"-area att spara värden på viktiga variabler vid CHAIN av andra BASIC-program. Den fiffige kommer även att upptäcka att det finns 64 bytes i bildminnet som aldrig används. Men att använda DET är verkligen inte att betrakta som "elegant programmering"!

Det finns dock två andra sätt att få ledigt minne i ABC80. Det ena är att flytta "taket" eller "golvet" i den area som normalt används till BASIC-program. På adress 65052 och 65053 ligger "BOFA", dvs en pekare till golvet, och på adress 65063 och 65064 ligger adressen till taket. Genom att ställa om dessa med POKE kan man stjäla minne som man sedan får ha ifred. Det enklaste är nog att lyfta golvet. Det kan göras antingen som ett kommando eller som en sats i ett program. Antag att vi vill sätta BOFA till 50000. Då kommer kommandot att se ut så här:

```
POKE 65052,50000,SWAP%(50000) : NEW
```

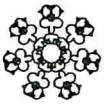
Och i ett program kan vi göra så här:

```
1000 POKE 65052,50000,SWAP%(50000) : CHAIN ""
```

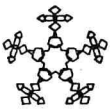
Observera att vi i båda fall måste nollställa BASIC med NEW. Satsen CHAIN "" ger samma effekt som kommandot NEW. Vill vi, går det utmärkt att köra vidare i ett nytt program med ex.vis CHAIN "KAKA2".

Den här metoden är dock ganska omständig: vi måste dela upp BASIC-programmet, ibland har vi en printerrutin lagrad under golvet, och det kanske finns extra minne inkopplat? I så fall ligger golvet från början MYCKET längre ner. Jag brukar föredra att använda metod nummer två: TA UPP MINNE PÅ HEAPEN.

I den area som finns tilldelad för BASIC-program, läggs själva BASIC-programmet upp som en internkod från golvet och uppåt. Det finns en pekare som visar var det tar slut, som heter EOFA och ligger på adress 65054 och 65055. den är dock ganska ointressant. Från taket och nedåt växer stacken, som är en minnesarea som används till bl.a. tillfällig lagring av mellanresultat. Här lagras t.ex. återhoppsadresser till GOSUB-RETURN och återhoppsadressen vid CALL. Ovanpå själva BASIC-koden ligger HEAPen. Det är en minnesarea som används till att lagra variabler (tal och strängar) när BASIC-programmet körs. När man skriver RUN så skapas heapen innan BASIC-programmet går igång. Då reserveras alltså minnesutrymme till alla variabler, dock inte sådana som behöver DIMensioneras. I ABC80-BASIC är det nämligen tillåtet att ha en variabel eller ett uttryck i en DIM-sats. Och då vet BASIC inte i förväg hur mycket minne som går åt. Så det väntar den med till vi utför DIM i programmet.



På adress 65056 och 65057 ligger pekaren HEAP som talar om var heapen slutar. (HEAP pekar egentligen på första lediga byte EFTER heapen). Här kan vi lägga upp maskinspråksprogram eller minnesareor för POKE och PEEK. Det är bara att läsa HEAP-pekaren, addera den mängd man vill ha, och sedan skriva tillbaka HEAP. Eftersom vi inte i förväg vet VAR vi kommer att få minne, måste vi spara undan det gamla värdet på HEAP i en variabel för att senare hitta dit. Dessutom måste vi se upp så vi inte kör igenom taket! Vi måste faktiskt ta till en säkerhetsmarginal till stacken också. ca. 500-1000 bytes brukar räcka. Så här kan det då gå till att reservera allt kvarvarande minne till en minnesarea som ska användas t.ex. till en stor bufferarea till en editor:



```
100 S%=PEEK(65063)+256%*PEEK(65064) : REM TAKET
110 H%=PEEK(65056)+256%*PEEK(65057) : REM HEAP
120 IF S%-H%<1024% THEN PRINT "Minnet räcker inte (snyft)" : STOP
130 A%=H% : REM SPARA STARTADRESSEN
140 H%=S%-1024% : REM 1024 BYTES UNDER TAKET
150 POKE 65056,H%,SWAP(H%) : REM SKRIV IN NYA HEAP
160 B%=H%-1% : REM SPARA SLUTADRESSEN
```

Eftersom ett heltal tar upp 2 byte och POKE bara skriver ut 1 byte, frågar man sig kanske "vilken av dom?". Svar: den lägsta. Så för att lägga ut båda, använder vi funktionen SWAP%( ) som kastar om den höga och låga byten i ett heltal. Ett exempel till: vi vill lägga in en maskinspråksrutin på heapen:

```
100 RESTORE 1000 : READ N% : REM ANTAL BYTES
110 S%=PEEK(65063)+256%*PEEK(65064) : REM TAKET
120 H%=PEEK(65056)+256%*PEEK(65057) : REM HEAP
130 IF S%-H%-N%<1024% THEN PRINT "Minnet räcker inte (sorry)" : STOP
140 A%=H% : REM SPARA STARTADRESSEN
150 FOR H%=H% TO H%+N% : READ D% : POKE H%,D% : NEXT H%
160 POKE 65056%,H%,SWAP%(H%) : REM SKRIV IN NYA HEAP
.
.
1000 DATA 17
1010 DATA 10,33,  <O.S.V.>
```

Lägg märke till att när vi gått ur slingan på rad 150, kommer H% att vara ett mer än sista värdet på H% i FOR-slingan. Så när vi sparar undan H% på rad 160 kommer HEAP att peka på första LEDIGA byte.

Den här metoden att lägga upp maskinspråksprogram har en allvarlig nackdel: vi vet inte i förväg VAR någonstans den kommer att hamna. Därför måste vi spara undan startadressen i t.ex. A% för att hitta dit. Det innebär även att vi tyvärr inte kan använda instruktioner av typen "JP <adr>" ty vi vet inte i förväg vart vi ska hoppa. Jump Relative -intruktionen går dock utmärkt att använda, och det räcker gott när man programmerar Z80-processorn.

Som synes finns det inga patentlösningar för att lägga upp maskinspråksrutiner, utan man får anpassa metoden efter omständigheterna. De ovan beskrivna metoderna borde dock täcka de allra flesta situationerna på ABC80. Den som vill lära sig mer om sådana saker rekommenderas att läsa boken "Avancerad programmering på ABC80" av Anders Isaksson och Örjan Kärrsgård. Om du tycker om att experimentera, kan du försöka ladda ett maskinspråksprogram direkt i bildminnet! Till sist ett gott råd: spara för allt i världen undan programmet på kassett eller diskett INNAN du provkör det! Även om du bara har ändrat någon till synes obetydlig detalj! Det finns nämligen alla chanser att göra fel, och det brukar resultera i att maskinen bryter ihop, och programmet går förlorat.

/Perre L.





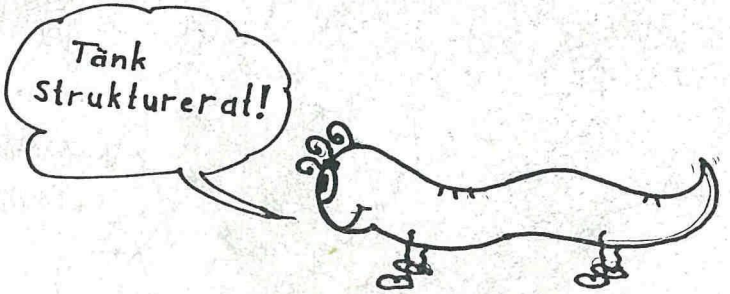

ABC 80:s MINNESKARTA

Följande är en beskrivning av hur ABC 80 använder tillgänglig minnesarea. Här kan Du hitta utrymme att lägga in maskinspråksrutiner och adresseringar av extra minneskort.

DECIMAL ADRESS			HEX ADRESS	OKTAL ADRESS
65535				
65408	128 BYTES LEDIGT FÖR POKE		FF80H	377:200
65046	ENKLA VARIABLER		FE16H	376:026
	SYSTEMVARIABLER			
64768			FD00H	375:000
64512	CASBUF 1	DOSBUF 7	FC00H	374:000
64256	CASBUF 2	DOSBUF 6	FB00H	373:000
64000		DOSBUF 5	FA00H	372:000
63744		DOSBUF 4	F900H	371:000
63488		DOSBUF 3	F800H	370:000
63232		DOSBUF 2	F700H	367:000
62976		DOSBUF 1	F600H	366:000
62720		DOSBUF 0	F500H	365:000
	STACK			
	16 KB RAM ARBETSMINNE[X)]			
49152			C000H	300:000
	16 KB RAM EXTERNT MINNE			
32768			8000H	200:000
	1 KB RAM BILDMINNE[X)]			
31744			7C00H	174:000
	1 KB ROM (PRINTER-OPTION)			
30720			7800H	170:000
	1 KB (LEDIGT)			
29696			7400H	164:000
	1 KB ROM (IEC-OPTION)			
28672			7000H	160:000
	4 KB ROM (FLEXSKIV-OPTION)			
24576			6000H	140:000
	8 KB ROM (LEDIGT)			
16384			4000H	100:000
	16 KB ROM[X)] BASIC			
0			0	0

HUVUDBRY

Felicia, Olga & Ofelia tänker skaffa sig var sitt modem. Så de går till hårdvarubutiken "MOSBAREN" där de inhandlar var sin SIO för 100 kr var =300 kronor totalt. Just när de klivit ut ur butiken kommer expediten på att de ju var medlemmar i STACKEN och därför skulle ha fått sammanlagt 50 kr. i rabatt. Så ha skickar ut springpojken Per Rornik med 50 kr i tior att betalas tillbaka till Felicia, Olga och Ofelia. På vägen kommer den samvetslöse typen Per Rornik på att det blir svårt att dela 5 tior på tre personer, så han stoppar två i fickan och ger våra intet ont anande vänner 10 kr. var. Så de hade alltså betalat 90 kr. var = 270 kr tillsammans. Plus 20 kr. till Per Rornik = 290 kr. Men vart tog den sista tian vägen?



Senaste skvallret (TT,UPI,STP,DDT,LSD):

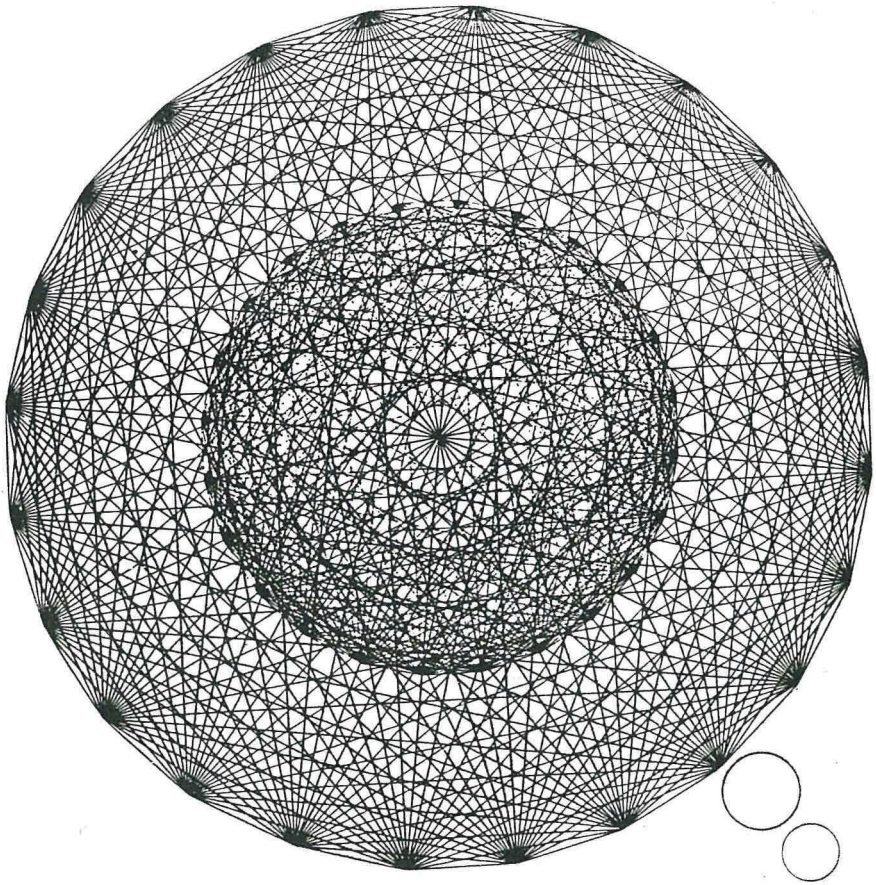
QZ (förkortning för Stockholms Datamaskincentral (Se fotnot!)) har enligt vanligtvis vällunderrättad källa köpt IBM-kompatibel hårdvara från AMDAHL till ett belopp av 11 megaspänn. Den ska ersätta QZ´s gamla obsoleta IBM-junk. Det lär vara den första riktigt stora ordern för AMDAHL i Konungariket Sverige och har betecknats som "genombrott" av insatta personer.

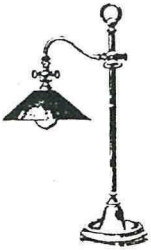
(Historisk fotnot: Stockholms Datamaskincentral förkortades i forna dar med SD (eller något liknande, de historiska källorna är osäkra). Men då kom någon och sade: SD (eller va de nu var) är MITT registrade varumärke! Och då fick SD (QZ, alltså) hitta på något annat istället. Man sade: "Låt oss hitta på något som OMÖJLIGT kan vara upptaget!" Och så blev dt QZ istället för SD. (Den som brukar använda tangentbord, upptäcker snart att tecknen Q och Z ligger lite speciellt till.) Det har sagts att QZ är förkortning för "Quality and Zest", men det är nog en efterkonstruktion.

SÄLJES !

Säljes 1 st 800-system med låda och nätagg (32k RAM). (Pga studier? Red:s anm)

Ring Per Lindberg tel 08 / 43 42 58





say you saw it

in

STACK POINTER