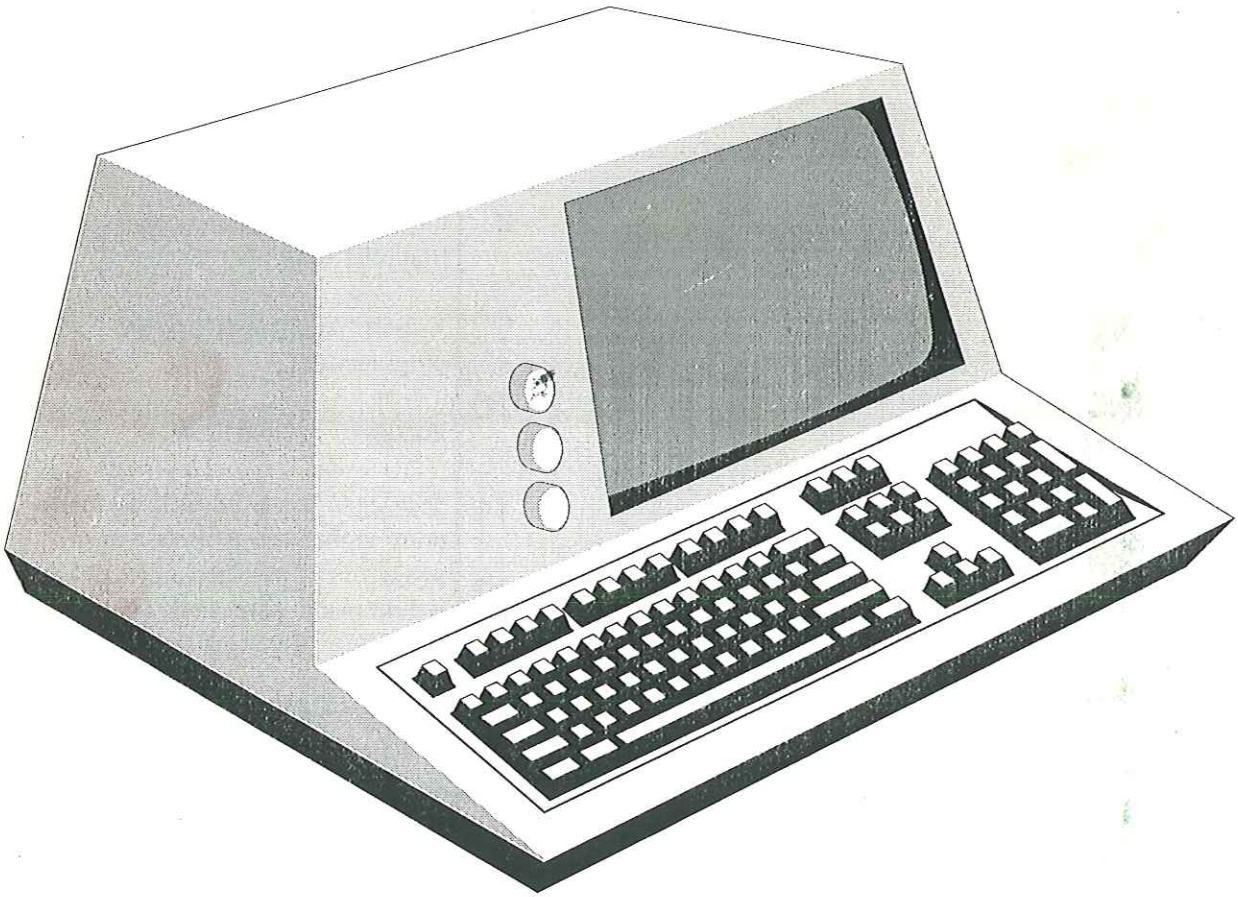
RELIK

Tidningen för 8-bitars datorer och dess användare

Nummer 1, September 1996 Pris: 20 SEK



INNEHÅLL

KONTAKTER!

Har du en produkt till en åtta-bitars dator som du vill ha rescenserad? Har du artiklar som skulle passa i Relik, eller har du lust att skriva ett par? Har du synpunkter på tidningen?

Hör av dig till redaktionen!

RELIK

Relik är en tidning om åtta-bitars datorer, skriven av användare för användare. Arbetet är helt idéellt, och eventuellt överskott från försäljningen går till förbättringar av tidningen.

Redaktör: Jens Örtenholm
Ansvarig utgivare: Jens Örtenholm

Adress:
Relik
c/o Jens Örtenholm
Råbystigen 48
197 31 Bro

Telefon red: 070-5927747

WWW:
http://www.nethosting.com/~emotion/relik.html

E-mail: jens.ortenholm@pc-programs.se

Postgiro: 770118-0239

Tidningen kostar 20 SEK inklusive porto inom Sverige. Betala in pengarrna på postgirokontot och ange tydligt namn, adress och vilket nummer av tidningen du vill beställa.

PD-disketter kostar 30:- styck, eller 30:- för den första disketten och 10:- / styck för de övriga vid beställningar över 2 st. Betala in pengarna på postgirokontot och ange tydligt namn, adress och vilken/vilka disketter du vill ha.

Medverkande i detta nummer: Jens Örtenholm, Peter Karlsson och Peter Frank.

Välkommen till Relik Nummer 1

Jag heter Jens Örtenholm, och jag är redaktör för denna nystartade tidning avsedd för ägare till åtta-bitars datorer. Detta första nummer innehåller tyvärr bara material om och kring Commodore 64:an, samt väldigt litet om 128:an. Detta beror på att de är de enda fungerande åtta-bitars datorer jag har tillgång till just nu, och jag har inte kunnat få tag på artiklar om andra datormärken, vilket jag hoppas kunna korrigera till nästa nummer med artiklar kring bl.a Spectrum.

Jag är för närvarande 19 år gammal och fyller 20 i Januari. Mitt användande av åtta-bitars datorer började tidigt, ungefär i 6-7 års åldern, då far min släpade hem en VIC-20 från jobbet. VIC-20:n byttes snabbt ut mot en C64, faktiskt en av de släpbara SX-modellerna, och jag har sedan dess i stort sett alltid haft och brukat en C64:a eller C128:a.

En del av er känner säkert igen mitt namn från tidningen Åtta Bitar, där jag kring ett halvår eller så skötte månadens PD-diskett, och även skrev en och annan artikel. Det slutade jag med när mina maskiner gick sönder, och det tog ett tag att laga dem. När det väl var färdigt så var tidningen Åtta Bitar mer eller mindre försvunnen.

Förutom att pyssla med tidningen så är jag även coder och allt-i-allo i en demogrupp kallad ANGELS, mer eller mindre verksam (just nu mindre) på ett flertal datorer, däribland 64:an och 128:an. Jag tillbringar även en hel del tid på internet, och med massor av annan aktivitet bakom tangentbordet. För att försörja mig så jobbar jag som programmerare och utbildare av blivande programmerare, och har också skrivit ett par böcker. Mina främsta programmeringsspråk (förutom 6510 assembler) är C++ och 80x86 assembler, och denna programutveckling sker alltså på PC:n.

Jag avser att i Relik presentera intressanta artiklar om datorerna vi älskar, samt givetvis en del om dess omgivningar. En del artiklar om programmering och programmeringsteknik kommer jag att skriva, och redan i detta nummer startar en artikelserie för att lära sig grunderna i assembler-programmering på 64:an. Rescensioner av spel, program och hårdvara kommer också finnas med i den mån vi får tag på någonting intressant att rescensera.

Naturligtvis kommer vi försöka nosa reda på de senaste nyheterna kring våra datorer. När det gäller 64:an och 128:an ska jag i framtiden försöka få tag på material kring GEOS. Funderingar rör sig också kring intervjuer med olika människor, bl.a medlemmar i demogrupper och andra personer som har med datorerna att göra.

Vi kommer självklart ta en noggrann titt på alla era frågor och brev som kommer in, och insändare kommer vi lägga med i tidningen så länge som det finns plats över (och innehållet inte är alltför tveksamt). Vi tar gärna emot artiklar som kan passa in i tidningen, och publicerade artiklar belönas med ett gratis-ex av det nummer som de kom med i.

Internet är någonting som vi ska försöka hålla ett vakande öga på, och självklart då de delar som har åtta-bitars anknytning. En BBS-lista, en internet-länk-lista samt PD-disketter ska vi också försöka prestera.

Allt detta kommer givetvis att utökas allt eftersom vi jobbar med tidningen. Meddela oss era förslag! Är det någonting som saknas, så tveka inte att skriva ett brev eller på annat sätt höra av er. Och skicka in era artiklar! De behövs!

Vill ni kontakta redaktionen, så ta en titt på sidan med innehållsförteckningen. Där står allt ni behöver veta. Då så, tills nästa gång vi ses - lev gott!

Jens Örtenholm

Internet och 64:an

Peter Karlsson

Det finns en hel del att hämta på Internet för den som har en C64 i program- och informationsväg, stora databaser med spel och demos och diskussionsforum speciellt avsatta för C64/C128-användare. Oftast kräver det dock att man har tillgång till en PC eller en MAC för att hämta hem filerna, för att sedan föra över dem till C64:an med Big Blue Reader, StarCommander eller liknande. För den som bara sitter med en C64 eller C128 kan det lätt kännas hopplöst, men det är det inte.

Det absolut enklaste sättet att koppla upp sig är om man har tillgång till en Internet-leverantör som ger möjlighet att koppla upp dig via ett vanligt terminalprogram till ett Unix-skal, d.v.s du ringer upp, uppger namn och lösenord, och får en Unix-prompt där du kan skriva kommandon som utförs på servern. På det sättet kan du köra alla "tunga" program för e-post, diskussionsmöten, World Wide Web (WWW), IRC och allt vad man vill köra direkt via Unix-datorn, och använda C64:an som en "dum" terminal. Man kan även installera program på Unixdatorn som gör det möjligt att via ett vanligt överföringsprotokoll (Xmodem, Ymodem, Zmodem eller Kermit) hämta hem filerna till C64:an för vidare körning.

Problemet är att inte alla Internet-leverantörer av olika skäl, oftast säkerhetsskäl, inte ger möjlighet att använda Unix-shell. Det gäller att fråga innan man tecknar abonnemang.

Det finns dock även andra sätt. Under utveckling till C64:an är nämligen ett program som gör det möjligt att koppla upp sig till Internet via SLIP-protokollet, vilket i stort sett alla Internet-leverantörer stöder. Det finns betaversioner tillgängliga för att köra Telnet och IRC, jag kan personligen inte kommentera dem, eftersom jag inte har haft möjlighet att testa. Även en World Wide Web-visare är under utveckling.

Vad finns det för C64-relaterat på Internet?

Först och främst, Åtta Bitar finns naturligtvis på Internet. ÅB:s hemsida på World Wide Web kan nås på adressen *http://www.mds.mdh.se/%7Edat95pkn/8bitar*. Där finns information om hittills utgivna tidningar, med smakprov ur dessa, samt länkar (hänvisningar) till en hel del annan intressant C64-information.

För den som är intresserad av att diskutera C64 finns diskussionsforumet (newsgroupen) *comp.sys.cbm* på Usenet. Där diskuteras alla Commodores åtta-bitars datorer, och en hel del kunnigt folk deltar, bl.a kända hackers, medarbetare på CMD (Creative Micro Designs, företaget bakom bl.a JiffyDOS och C64-hårddiskarna), och andra intresserade personer. Varje månad publiceras en mycket omfattande FAQ där mycket intressant information finns att finna.

På IRC finns kanalen #c-64 där denna dator diskuteras. Den som gillar snabba svar kan där i realtid diskutera med andra intresserade.

För den som vill hämta hem program och demos finns det stora arkiv med filer till C64:an (FTP-arkiv), en lista över sådana publiceras i comp.sys.cbm. Det mesta finns att hämta, bara man vet var man ska leta (vilket är svårt i början!).

Vad är då UNIX?

Unix är ett operativsystem, ursprungligen skrivet för stordatorer, med sina rötter ända tillbaka till slutet av 1960-talet, då Bell Laboratories utvecklade den första versionen. Det har sedan utvecklats i många versioner, men

Peter Karlsson kan nås på email: dat95pkn@idt.mdh.se eller adress: Peter Karlsson, Kaserngatan 16B, vån 1, 723 47 Västerås

redan 1971 innehöll det i stort sett alla finesser som finns i Unix idag.

Från början skrevs Unix för en dator kallad PDP7, men man ville snart att det skulle flyttas över på andra datorer, vilket var ett problem. Detta löste man genom att skapa ett programmeringsspråk, känt som *C*, i vilket man skrev operativsystemet så systemoberoende som möjligt. Man lade maskinspecifika saker vid sidan om, och vips hade man ett system som kunde köras på valfri dator (om man hade en C-kompilator, och kunde fixa till de systemberoende sakerna, naturligtvis).

Unix är ett operativsystem som från grunden är skrivet som ett fleranvändarsystem, vilket var naturligt på stordatorer, för att flera skulle kunna köra samtidigt. Man hade multikörning som tillät flera program att köra samtidigt. Detta till skillnad från exempelvis MS-DOS på PC-datorer, som är rena enanvändarsystem, och inte mer än med speciella hjälpprogram ens hjälpligt kan köra flera program samtidigt.

Nåväl, allt det här behöver man inte veta för att kunna använda Unix, men det är alltid trevligt att ha lite bakgrundsfakta. För att återknyta till det jag skrev om i början, Unix-skal, så ska jag kort beskriva hur man styr Unix.

Unix är kommandoradsstyrt, det innebär att man skriver kommandon, ett och ett, till datorn, som sedan behandlar dessa efter bästa förmåga. Detta till skillnad från grafiska system, som GEOS, där du med en mus pekar och klickar i menyer **(Red anm: Det finns även grafiska system till UNIX, bl.a X/ Windows)**. Ett Unix-skal är helt enkelt ett program som visar en prompt, d.v.s en förfrågan efter ett kommando, på skärmen och väntar på att man gör något. Hur prompten ser ut beror på hur man ställt in det, men oftast brukar den visa aktuell katalog, eller något i den stilen.

För att kunna använda ett Unix-skal, måste man veta lite om vilka kommandon man kan använda. Eftersom jag är långt ifrån en Unix-expert ska jag bara ta några grundläggande kommandon, vilka jag hoppas är tillräckligt för att kunna styra Unix, och komma ut på Internet.

Jag räknar med att du kommit förbi inloggningen, man måste ange namn och lösenord för att komma in, och har en prompt framför dig på skärmen. Det första kommandot, och det man troligtvis använder mest, heter ls (förkortning för list) (observera att alla kommandon alltid skrivs med små bokstäver), och det visar en lista över filerna i den aktuella katalogen (ungefär som ett LOAD "$",8 eller DIRECTORY). Det finns en massa skojiga finesser man kan ha för sig med ls, men de tänker jag inte gå in på. Den som är intresserad kan titta i onlinemanualen.

Onlinemanualen ja, det är nog det kommando man använder näst mest. Det är väldigt enkelt att använda, man skriver ordet "man" följt av det kommando man vill ha hjälp om, t.ex "man ls" för att få upp en beskrivning på ls. Dessa beskrivningar är oftast rejält omfattande, och man har sällan nytta av mer än hälften av alla de olika varianter och inställningsmöjligheter som finns.

En grupp av kommandon som också är bra att ha är de som används för att navigera i katalogträd, filsystemet i Unix är nämligen uppbyggt som katalogträd. I stället för att ha alla filer i samma katalog (som på en vanlig C1541-diskett), har man olika kataloger för olika filer. De är uppbyggda i olika hierarkier, vilket gör det lätt att hitta, t.ex finns alltid alla användarnas kataloger under /home (/-tecknet används för att avgränsa mellan kataloger). De kommandon som används är chdir, rmdir och mkdir.

Chdir (kan förkortas till cd) växlar mellan kataloger. Det går att skriva en absolut sökväg, hela vägen från rotkatalogen (t.ex "cd /foo/bar"), men det går även att skriva en sökväg relativ från den aktuella katalogen, om du t.ex står i /foo och skriver "cd bar", hamnar du i /foo/ bar. Det går att gå till moderkatalogen med "cd .." (i detta fallet kommer du då tillbaka till / foo).

De två andra kommandona, rmdir och mkdir, tar bort gamla respektive skapar nya kataloger. "mkdir baz" skapar katalogen baz under den aktuella, och "rmdir baz" tar bort den igen.

Med detta fåtal kommandon kan man komma ganska långt, men inte till Internet. För det behöver man program. Det kan ju exempelvis vara trevligt att kunna skriva e-post till någon (t.ex till mig, och skriva vad du tycker om artikeln). För det används oftast programmet Pine, i alla fall är det det som är vanligast, och det jag har erfarenhet av. Det finns andra, t.ex "mail" som ingår i Unix och som är ganska så kasst.

Om man startar upp Pine (genom att skriva "pine" vid prompten) får man upp en trevlig informationsskärm. Programmet brukar fråga om man vill skicka efter en informationstext, det kan vara en bra idé att svara ja på den frågan. Därefter kommer man till huvudmenyn, vilket man i fortsättningen kommer att hamna i så fort man startar.

För att skriva ett brev väljer man "Compose new message". Skärmen kommer då att övergå till brevskrivarläge, och markören hamnar på raden som det står "To" på. Där skriver man mottagarens e-post adress. Raderna "Cc" och "Bcc" kan med fördel lämnas tomma (de är till för att skicka kopior på brevet till andra intresserade), men raden "Subject" fylls lämpligen i med en beskrivning på vad brevet handlar om.

När detta är gjort är det bara att skriva så att tangenterna glöder! Avsluta med control-X (tryck och håll nere control-tangenten och tryck sedan på X). Du får en fråga om du vill skicka meddelandet,

svarar du ja susar det förhoppningsvis iväg till mottagaren.

För att se om du har fått några brev väljer man "Folder index" från huvudmenyn. I bästa fall får man då upp en lista på brev man fått (annars får man en i stort sett tom skärm). Man kan stega igenom listan och trycka Enter när det brev man vill läsa är markerat.

Pine är betydligt mer avancerat än så här, men jag lämnar de mer avancerade finesserna till läsaren. Pine har en omfattande onlinehjälp som kan nås genom att trycka på ? (eller på control-H om man håller på att skriva något). Läs gärna den!

Det var allt för denna gången, i nästa avsnitt tänkte jag ta upp lite om hur man läser newsgroups i Pine och i tin, samt hur man "surfar" med Lynx.

Detta har bara varit en kort sammanfattning om vad man kan göra med en C64 och Internet. Eftersom jag inte riktigt vet vad ni läsare är intresserade av, så har jag skrivit det ganska allmänt. Det finns mycket mer att berätta, och väldigt mycket jag inte har nämnt i den här korta artikeln.

Skriv gärna e-post eller vanliga brev till mig med förslag om vad jag ska skriva i fortsättningen. Min e-postadress är *dat95pkn@idt.mdh.se*, och min vanliga adress är som följer:

Peter Karlsson
Kaserngatan 16B, vån 1
723 47 Västerås

(Red anm: Relik kommer att ha en hemsida uppe ungefär då detta nummer är färdigt. Den kommer inte vara komplett på något vis, och den kommer byggas ut ganska kraftigt den närmaste tiden. Testa: http://www.nethosting.com/~emotion/relik.html)

1750 REUn ÄR DÖD - LÄNGE LEVE 1750 REUn!

Peter Frank

Så kanske man skulle kunna säga i likhet med när seriefiguren Fantomen "återuppstår" genom sin son, för 1750 RAM-expansionen gör Comeback!

Inte mindre än **Chip Level Designs** i USA har förstått att det finns ett skriande intresse för dessa, och har därför slagit två flugor i en smäll. Dels har de gjort den mindre än originalet - inte mycket större än en "vanlig" cartridge såsom TFC eller Action Replay, och dels har de gjort den strömsnålare genom bättre kretsar som drar just mindre ström. Nu ska en C64-ägare av gamla årsmodellen med transformatorer på 5 Volt / 1.5 Ampere slippa byta transformator bara för att REUn kräver det. "Super 1750 Clone", namnet som den går under, är 100% kompatibel med den gamla 1750 REUn och fungerar med andra ord i både C64-, C128- och C128 CP/M-läge. Även GEOS torde inte ha problem. Minnet är på 512 kb som det ska vara, men historien förtäljer inget om det går att uppgradera den till mer som man kan med original-REUn. En 28-sidig engelsk manual ingår - vilket tyskarna inte gillar. De hade föredragit (gissa) en tysk dito.

Säljs av:
Tyska CEUS Computersysteme
Fritz-Reuter Strasse 31
4353 Oer-Erkenschwick
Tfn# 02368/5310-0

Pris: **289 Dmark** (C:a 1200 SEK)

TIO I TOPP FEL PÅ C64

Peter Frank

Här är en lista över de tio vanligaste felsymptomen en Commodore 64 (och till stor del även C128) kan få, och vad som brukar fela. Den vanligaste orsaken till att datorn går sönder är att man kopplar in datorprylarna när datorn är påslagen. Det behövs inte mycket för att en dator ska tröttna och packa ihop (gå sönder).

GLÖM INTE ATT STRÖMMEN TILL DATORN SKA VARA AVSLAGEN OM DU SKA LAGA DATORN ELLER SÄTTA I / TA UR SLADDAR ELLER PERIFERIUTRUSTNING!

Notis!

"Load Error" är en ganska vanlig åkomma som de flesta råkat ut för och beror vanligtvis på att bandstationens läshuvud inte ligger på samma position som när bandets innehåll skrevs. Detta märks vanligen på band som sparas med "Band-Turbo". Orsaken är att vanligen används 300 baud, men turbo höjer detta till det tiodubbla och blir då känsligare för felmarginalerna. Endast en tiondels millimeter används, och det är justeringar på hundradels millimeter som behövs.

Antalet laddningsfel på dessa kassetter kan minskas genom att man använder sig utav två bandspelare. En som är "rätt" justerad, som man använder till att spara filerna. Dessa laddas sedan in på samma bandstation eftersom justeringar ej behövs då. Vill ett program inte laddas in använder du den andra bandstationen som du justerar tills att du fått in det och sedan kan spara det på den andra "korrekta" bandstationen!

För att inte missbruka att rycka i och ur de två bandstationerna kan man

införskaffa (eller bygga, om man är händig nog) en s.k. "CloneKit" som gör att man kan koppla in två bandstationer på samma bandspelarport. Fördelen med denna är dessutom att du kan kopiera dina filer utan att använda LOAD/SAVE på datorn. Tryck bara på PLAY på den ena bandstationen och PLAY/RECORD på den andra! För att det ska fungera förutsätts att du inte får LOAD ERROR när du vanligen laddar in, då får du göra ovanstående justeringar som jag nämnde ovan.

I värsta fall kan bandet ha blivit utsatt för magnetisk åverkan (d.v.s bandet kanske har legat på en högtalare), stark sol eller värme. Innehållet på bandet kan då vara förlorat för alltid då det blivit förvanskat.

FÖRVARA ALLA MAGNETMEDIA PÅ RÄTT SÄTT. RENGÖR LÄS/SKRIVHUVUDET MED JÄMNA MELLANRUM SÅ HÅLLER DINA VIKTIGA PROGRAM BETYDLIGT LÄNGRE!

Död, ingen bild.
Säkringen i transformatorn.

Joysticken fungerar inte.
Fel på joystick eller 6526-chipet.

Tangentbordet fungerar inte.
Fel på 6526-chipet.

Ingen bild.
Fel på chip 4164, 6510, 6526, 6569 eller 7406.

Diskdrive-porten fungerar inte.
Fel på chip 6526 eller 7406.

Userporten fungerar inte.
Fel på chip 6526 eller 7406.

Bandspelaren fungerar inte.
Fel på 6510 eller 7406, alternativt skyddsdiod eller transistor i bandspelaren eller säkring trasig i datorn. Se även notis.

Konstiga tecken på skärmen.
Fel på chip 4164 eller 6569.

Bild på monitorn men inte på TV.
Fel på RF-modulatorn.

Tom, blå ram på TVn.
Fel på BASIC-ROM eller 6569.

VIDEOTEXT PÅ C64:an

Peter Frank

Nu kan du se Videotext (text-tv) på C64:an. Det är tyska SCANNTRONIC som säljer hård/mjukvaran till datorn.

För mer information eller beställning:

Scanntronic / Mugrauer GmbH
Parkstrasse 38
8011 Zorneding-Pöring
Tfn#: 08106/22570

Pris: **298 Dmark**

RELIK PD DISKETT #1

Jens Örtenholm

Jag har i all hast lyckats peta ihop en PD-Diskett till detta första nummer av Relik. Innehållet på den är som följer:

MATEMATICA / REFLEX
Ett mycket trevligt demo, visar att det gör att göra demos med bra design också på 64:an.

DAWNFALL / OXYRON
Ett enfils-demo, innehåller en del sevärda rutiner. Hyfsat snyggt!

Turbo Assembler 7.0 / Alter
Gamla hederliga Turbo Assembler, fixad av Lynchbit / Alter. Behövs för de som tänker hänga med i Assembler-skolan.

TurboDOS Pack
En samling med olika turboladdare som kan relokeras till lämplig minnesplacering.

Mr Thomas / Vibrants
En gammal men bra musiksamling från två Vibrants-musiker, nämligen Drax och Laxity.

Som ni ser så är det mesta på disketten demo-relaterat, men så snart jag har fått ordning på allting här (det HAR varit rörigt och stressigt att få ihop #1 ☺) så ska jag sätta ihop PD-disketter med lite allmänna prylar på, liksom disketter för 128:an och för GEOS. Vi får se hur långt materialet räcker.

För att beställa denna diskett, sätt in 30 SEK på postgiro 770118-0239 mottagare Jens Örtenholm, och ange tydligt vilken diskett du vill ha (fast just nu finns det ju bara en ☺) och ditt namn, adress och telefonnummer, så dimper den ned i din brevlåda ett par dagar senare.

NTSC TURBO

Peter Frank

Det finns olika sätt att hotta upp sin Commodore 64 vad gäller hastighetsmässigt sätt. Och då syftar jag inte på olika kassett- och diskett-turbos. Man kan faktiskt få sina program att gå fortare INTERNT också. Jag kommer här att klargöra ett av dem samt hur man praktiskt kan göra det.

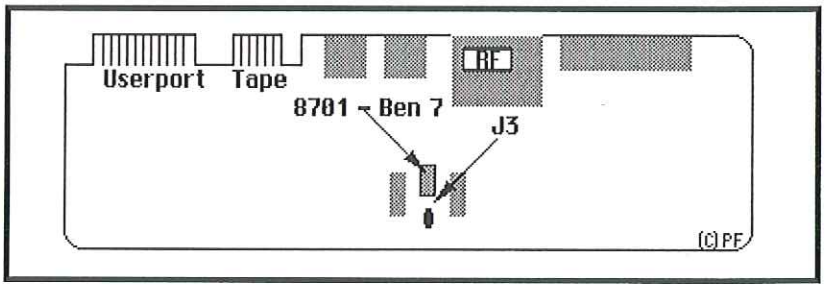
Ett av de mindre omtalade på C64:an är PAL-NTSC switchen. För att förstå det så får vi förklara vad som är skillnaden mellan PAL och NTSC.

PAL står för Phase Alternate Line och är en större Europeisk standard (men bland annat Frankrike har SECAM som är ganska snarlik PAL). Bilderna på en TV-utsändning är 25 bilder per sekund och en bild har 625 rasterlinjer. C64:ans CPU hastighet i PAL är 0,98525 MHz.

NTSC står för National Television System Committee och är standarden i USA och Japan. Den visar 30 bilder per sekund och en bild har 575 rasterlinjer. C64:ans CPU hastighet i NTSC är 1,02273 MHz.

VAD ÄR DET SOM GÖR ATT DET BLIR "TURBO" ??

Genom att ändra om från PAL till NTSC och låta bli att byta ut kristallen kan man göra en hastighetsökning, hur stor ökning beror på vad du kör för program. Kretsen vi manipulerar är 6567 VIC och är själva videokretsen. Den använder en 8MHz "dot" signal-in på ben 22 för att generera en 1MHz fas 0 klocka signal-ut på ben 17. Denna signalen används för att driva själva 6510 CPU:n. Nu torde det ju inte bli så stor skillnad mellan PAL och NTSC (0,98 resp 1,02 MHz)...ca 3,7% men det blir faktiskt mer. Kristallen som genererar signalen är på 17,73442 MHz, och eftersom vi inte byter ut den mot en 14,31818MHz kristall som egentligen ska vara där i NTSC så blir det hela 19,2% snabbare!



Nu finns det även baksidor med detta. Det första är att bilden "försvinner" om man använder en vanlig TV, eftersom frekvensen för bilden inte är den korrekta längre. Rådet är här då att skaffa sig en billig RGB-monitor, vilket även har den fördelen att du får en skarpare bild. Det andra är att alla program inte vill veta av sådant här. Kommunikationsprogram (även kallat terminalprogram) kan få skräp vid sändning och mottagning då timingen heller inte stämmer. 6526 CIA #2 liksom några andra kretsar reagerar (och följer med) på skillnaden. Vissa terminalprogram kan dock fungera BÄTTRE än innan just för att de är inställda för NTSC systemet och 1,02 MHz (men eftersom man inte ser vad som händer i C64-läget är "turbon" ganska meningslös här). GEOS är ett program som tydligen totalt förnekar och vägrar att köra med denna "turbo".

På en C128 i 128-läget kan man genom att använda 80-kolumner utnyttja "turbon" bättre eftersom man kan se vad man gör. Detta för att bilden för 80-kolumner går via RGB-utgången och inte via RF-modulatorn.

Men hur gör man nu denna "NTSC Turbo" ??

Jo, på följande sätt:

Signalen vi manipulerar går från 8701 kretsen, närmare bestämt ben 7. Därifrån går den bl.a vidare till

6567VIC och slutligen till 6510 CPU:n.

C64: Tyvärr så finns det så många olika versioner på detta moderkort (över sju stycken) att det kan skilja sig från dessa exempel och just ditt kort. Skulle så vara (8701 finns inte) rekommenderar jag att hellre låta det vara än att riskera att man sedan sitter med en defekt dator.

Det första du ska göra är att leta upp en 16 bens IC-krets som har beteckningen 8701. På de nya C64C med det lilla kortet (och få kretsar) finns den ungefär mitt på kortet, till höger om ljudkretsen (8580 eller 8581, Se även bild!). C:a två centimeter lite snett ner till höger finns en markering benämnd J3 (bredvid en mindre trimpotentiometer). Observera att här är den benämnd J3...på andra modeller är den benämnd J2 och på C128 J1!

JUST FÖR ATT DET ÄR OLIKA PÅ OLIKA VERSIONER PÅPEKAR JAG ÅTERIGEN ATT HELLRE LÅTA BLI OM DU ÄR OSÄKER!!

Gå till "Dags för lödkolven" för fortsättning.

C128 och C128D (plast): På C128an är det enklare att finna kretsen 8701 som vi ska leta efter. Det finns två "metallburkar" på moderkortet. Den övre till vänster ska vi öppna. Du ser att inuti är den delad i två sektioner. Den till vänster innehåller bl.a Video-RAM. Den högra innehåller 6567VIC samt 8701. Från 8701 (U28) ben 7 går signalen till emittorn på transistor Q4. Mellan dessa finner du J1.

DAGS FÖR LÖDKOLVEN

Om du funnit jumpern så ser du att bryggan är "full" av lödtenn. Ta en bra tennsug eller en tennfläta (finns i de flesta butiker som säljer elektronik-komponenter) samt din varma lödkolv och sug bort det.

Du ser sedan två halv-öar med vilket du tar en tvåpolig sladd och löder dit en pol på vardera ö. På andra sidan sladden sätter du en tvåpolig ministrömbrytare med vilken du sedan kan skifta mellan NTSC och PAL.

FEL ??

Är bilden där oavsett du ändrar switchen eller ej, så har du råkat löda ihop sladdarna på öarna. Bort med dem, och gör om det med mindre tenn.

Är bilden inte där oavsett du ändrar switchen eller ej, så kan det vara brott på sladden, söndrig switch, eller så kan någon komponent på kortet ha gått sönder.

Denna artikel är förhoppningsvis så enkel och utförligt skriven att även en novis ska kunna förstå. Däremot får ni själva lära er att hantera en lödkolv! Vi tar inga ansvar för ingreppen, men ovanstående har testats och fungerar.

NÄSTA RELIK?

Detta nummer av Relik blev grymt försenat p.g.a problem med trycket och layouten. Ett par tidigare versioner (inte vackra att se på) skickades ut, men verkade aldrig komma fram (tillsammans med en massa annan post jag skickade samtidigt).

Nästa nummer av Relik kommer ut den 1:a November. Tryckproblemet är nu löst, och kommer så förbli.

Jag reserverar mig för ändringar som beror på postverket och den mänskliga faktorn.

/Red

LISTA PÅ BBS:ER MED INNEHÅLL FÖR ÅTTA-BITARS DATORER

Namn:	*Sysop:*	*Nummer:*	*Datorer:*
Antidote	Taper / Triad	042-76416	64/128
Fosie BBS	Natas	040-269767	64/128
The Studio	Jerry / Triad	0159-31991	64/128
*Utrustning: C64, 2400, 8 MB RamLink (10*1581)*			
Warez Aquarium	Sledge / FLT	08-371360	64/128
*Utrustning: C64, 2400, CMD HD 540 MB / JiffyDOS, 1581, C*Base 3.2*			
WinterMute	Emotion	Tf. Nere	64/128
Utrustning: Pentium 133, Courier v.34+, 4GB HD, CD-ROM			
Hemsida: http://www.nethosting.com/~emotion/winterm.html			

DATORNS MATEMATIK

Jens Örtenholm

Som du säkert vet så jobbar ju datorer med ettor och nollor, så kallade bitar. Dessa är normalt organiserade på ett eller annat sätt, varav den minsta (hårdvarumässigt i datorsammanhang, det finns mindre typer, t.ex en nibble som är på 4 bitar) och mest förekommande typen är en byte, vilket är en samling om åtta bitar (kallas även OKTETT). I övrigt så är de flesta grupperna multipler av åtta (t.ex 16 bitar = 2 * 8, eller 32 bitar = 4 * 8).

För att riktigt förstå hur man kan programmera en dator effektivt, så ska vi ta en titt på de olika talsystemen, och hur man jobbar med dem.

DET BINÄRA TALSYSTEMET

Det binära talsystemet är det mest grundläggande för datorn, det vill säga talsystemet för ettor och nollor. Om vi tar en byte, det vill säga åtta bitar, så representerar de ett värde på följande vis:

```
128  64  32  16  8  4  2  1
1     1   0   1  0  1  0  0
```

Den undre raden är bitarnas värden, och den övre raden använder vi för att räkna. För att då ta reda på vad detta värde är decimalt (det vill säga i vårt vanliga talsystem med basen tio) så lägger vi ihop värdena som står ovanför de "tända" bitarna, alltså ovanför de bitar som är ett.

Detta tal blir då 128+64+16+4 = 212. På varje ställe i datorn där man använder sig av talet 212 så lagrar datorn alltså bitmönstret 11010100.

Men vart kommer då dessa "magiska tal" som står över bitarna från? Jo, de är enkla att räkna fram. Bitarna numreras baklänges (från höger till vänster) och från nummer noll och uppåt, så biten längst till höger har nummer 0, den alldeles till vänster om denna är nummer 1 och så vidare. Utgå från siffran 1 (för bit 0) och multiplicera det med 2 så har du siffran för bit nummer 1. Ta sedan denna siffra och multiplicera med två så har du siffran för bit nummer 2, och så fortsätter du så tills du har fått den mall du behöver. Dessa siffror är alltså av basen 2, det vill säga 2^0, 2^1, 2^2 och så vidare. För att räkna ut hur stort värde ett bitmönster max kan innehålla så tar man alltså 2^antalet bitar. En byte kan alltså som mest innehålla värdet 2^8 = 256 (det är en av anledningarna till att bitarna numreras från 0 och uppåt. 2^0 -> 2^7 blir adderat samma sak som 2^8).

DET HEXADECIMALA TALSYSTEMET

Människan kan få problem med att jobba med ettor och nollor, vilket dock passar utmärkt för en dator. Vi är vana att arbeta med det decimala talsystemet. Nå, det finns ytterligare ett talsystem, det hexadecimala talsystemet med basen sexton, som man kanske kan se som ett slags kompromiss mellan datorns sätt att räkna och människans. Det hexadecimala systemet är mycket enklare att använda, samtidigt som det har ett par fördelar vid datoranvändning jämfört med det decimala.

Det hexadecimala systemet är ganska likt det decimala. Man börjar räkna från 0 och räknar sedan som vanligt, 1, 2, 3, 4, 5, 6, 7, 8, 9...men sedan händer någonting underligt. Istället för 10 följer sedan A, B, C, D, E och F, och först därefter kommer 10.

Bokstäverna A-F står för sifferintervallet 10-15, och det hexadecimala värdet 10 står därför för 16, därav basen sexton. Sedan fortsätter räknandet på samma sätt, d.v.s 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21 och så vidare.

Men vad är det då som är så bra med det här talsystemet? Det verkar ju mest krångla till det hela, om än inte lika mycket som det binära talsystemet. Låt oss ta en titt på en nibble, alltså en bitsamling av 4 bitar. Som jag tidigare nämnde så kan man räkna fram det maximala talet för en bitsamling genom att ta 2 upphöjt till antalet bitar i samlingen, i det här fallet 2^4 = 16...16? Basen 16! Nu börjar bitarna falla på plats. En siffra i det hexadecimala talsystemet blir då alltså en nibble binärt. Det innebär att man lätt och ledigt kan översätta binära tal till

hexadecimala. Låt oss göra ett försök med talet 01100100, och vi delar då skenbart upp det i två nibblar:

```
8 4 2 1     8 4 2 1
0 1 1 0     0 1 0 0
```

Resultat: 4+2 samt 4 vilket ger talet 64.

Som du nyss såg så blir varje nibble i ett bitmönster en siffra i ett hexadecimalt tal. Det hexadecimala talsystemet "råkar" vara uppbyggt på det sättet, och är då alltså väldigt intimt relaterat till det binära talsystemet. Skulle vi till exempel sätta alla bitar till 1 så skulle värdet bli FF, vilket precis är de två siffrornas maximala värde.

En byte kan alltså innehålla värden mellan 0 och FF och ett word (16 bitar) mellan 0 och FFFF (vilket faktiskt är 64 KB, precis lika mycket minne som en C64 innehåller...adresserna ligger alltså mellan $0000 och $FFFF).

Någonting att tänka på när man jobbar med hexadecimala tal är att ett hexadecimalt tal som består av enbart siffror **inte** har samma värde som ett decimalt tal. I exemplet ovan blir alltså resultatet 64 **HEXADECIMALT**. Decimalt blir det 100. Därför brukar hexadecimala tal skrivas på lite olika sätt. Det hexadecimala talet 64 kan skrivas: #$64 (i t.ex assembler på 64:an), $64 (vanligt, finns även med i assemblern men har lite annorlunda betydelse än #$64) eller 0x64 (så skrivs det i programspråket C och även C++).

En färdighet som det är mycket användbart att kunna behärska om man programmerar assembler är att enkelt kunna konvertera mellan hexadecimala tal och decimala tal. Många miniräknare kan stå till tjänst med konverteringarna åt dig, men det finns tillfällen då det kanske inte finns en miniräknare till hands, och då är det alltid bra att kunna göra det i huvudet, eller åtminstone med papper och penna.

Också här ska vi titta på exponenter. Som du vet så är det decimala talsystemet uppbyggt kring basen 10, det vill säga talet 1234 kan representeras av:

```
1 * 1000 = 1 * 10^3
2 *  100 = 2 * 10^2
3 *   10 = 3 * 10^1
4 *    1 = 4 * 10^0
```

Och sedan lägger man enkelt ihop resultaten av detta. På ett liknande sätt kan man konvertera hexadecimala tal till decimala, fast här jobbar man med basen 16 istället. För att konvertera talet 3B till det decimala talsystemet så kan vi göra så här:

```
3 * 16 =  3 * 16^1
B *  1 = 11 * 16^0
```

Och när vi sedan lägger ihop resultatet så får vi 59, vilket är den korrekta decimala representationen av 3B. Observera att vi konverterar bokstäverna i det hexadecimala talet till decimala siffror i högra ledet ovan.

För att konvertera decimala tal till hexadecimala så kan man använda sig av en enkel metod som bygger på att upprepat dela talet med 16, och sedan använda sig av resten från den divisionen. Eftersom vi vet vad 3B motsvaras av decimalt, så låt oss pröva på en konvertering åt andra hållet, alltså att konvertera 59 till det hexadecimala talsystemet.

Steg för steg delar vi talet med sexton, och får genom resten ut det hexadecimala talet baklänges. Det enda vi behöver tänka på under konverteringen är att omvandla alla rester mellan 10 och uppåt till de hexadecimala bokstäverna. Alltså:

```
59 / 16 = 3 rest 11 = B
 3 / 16 = 0 rest  3 = 3
```

Resultat: 3B

Och som du ser så har vi då genom att upprepat ha delat med 16 fått fram svaret baklänges genom resterna efter divisionerna.

ANNONSER

Som vanligt i första numret av en tidning, så är det litet fattigt med läsar-input, speciellt när det gäller annonser. Jag tänkta passa på att förklara hur vi har tänkt det med annonser i tidningen.

Vanliga småannonser från läsare, det vill säga *privatpersoner*, accepteras till en max-längd av 512 tecken per annons, och införs i kommande nummer av tidningen utan kostnad.

Företagsannonser finns i ett par olika varianter:

Notis, som består enbart av text, och lämpligen skickas över till oss i form av textfil (helst ASCII eller MS Word) eller via elektronisk post, men det går också bra med vanlig text på ett pappersark. Max 1024 tecken.

Annons, som består av text och grafik, och max får vara av en halv tidningssidas storlek, och ska då vara utformad så att den motsvarar en övre eller nedre del av en tidningssida. Vi föredrar att få annonsen som fil, helst i ett format som är lätt att montera in i tidningen (vilket som helst av de vanliga grafikformaten går bra, t.ex GIF, PCX, LBM (IFF) etc) men det går **oftast** bra med papperskopia (med oftast menas att kvalitén måste vara så pass bra att en scanner kan läsa in annonsen i godtagbar kvalitet).

Fullsida, som följer samma riktlinjer som för Annons, men istället fyller en hel A4.

Vi jobbar på metoder för att kunna översätta dokument från GeoWrite och GeoPublish till ett lämpligt format så att vi kan ta emot annonser i dessa format, men vi har ingenting färdigt ännu. Vet du hur detta kan göras på ett enkelt sätt, hör av dig!

Vi kan även hjälpa till med formgivningen av en annons till ett mycket lågt pris! Hör av dig för mer information.

LÄR DIG C64 ASSEMBLER

Jens Örtenholm

Här tänkte jag i en serie artiklar lära ut grunderna i assembler-programmering på C64:an, det enda språket som är av intresse om man vill skriva snabba program. Observera att detta enbart är en grundkurs, och alltså inte kommer innehålla särskilt mycket "special-tips", men du kommer förhoppningsvis lära dig såpass mycket assembler att du kan börja programmera och lära dig på egen hand.

För att hänga med i denna snabbkurs så är det ett par saker som jag förutsätter att du har till hands. Det ena är en schysst cartridge, helst en Action Replay (det är den jag kommer att utgå från), men med lite micklande kan du använda vilken annan cartridge som helst som har inbyggd monitor. Det andra är en assembler, vilket du återfinner på detta nummers PD-diskett (Turbo Assembler 7.0). Bra att ha om du verkligen tänker lära dig att programmera assembler (och göra det ofta) är en så kallad bibel, d.v.s en Programmers Reference. Dessa kan vara lite knepiga att få tag på, men det borde finnas såväl privatpersoner som företag som kan sälja ett exemplar till dig.

Jag tänkte då inledningsvis förklara lite grunder i assembler, innan vi gör över till huvuddelen av artikeln som förklarar hur du använder monitorn och assemblern på olika sätt.

Assembler är det mest grundläggande sättet att programmera en dator. Programmen du skriver består av instruktioner som går direkt till processorn och där utförs. Det innebär också att assembler är mycket enkelt uppbyggt, ett lågnivå-språk, och det innehåller inte i grunden exempelvis variabler eller FOR-slingor som högnivå-språk använder (även om det självklart går att åstadkomma sådana saker i assembler). Det som är lite svårt med att lära sig assembler är faktiskt inte att lära sig instruktionerna (de är ganska enkla) utan att pussla ihop det på bästa sätt till effektiva program.

C64:an är, i mitt tycke, en av de bästa datorerna man kan lära sig assembler på. Den är inte onödigt komplicerad eller konstigt uppbyggd, och man får lätt en fin känsla för vad assembler är och hur man använder det. Dessutom ger C64:an massor av övning i *optimering* (omskrivandet av progamkod så att den blir snabbare eller mindre) då det inte finns hur mycket processorkraft som helst att slösa med. Nog med prat, över till instruktionerna.

En instruktion representeras i C64:ans minne som ett eller flera siffertal (eller mer korrekt en eller flera bytes. Är du osäker på bitar, bytes och annat som har med datorns grundläggande matematik att göra, se artikeln **DATORNS MATEMATIK** som finns på sidorna före den här artikeln). Dessa värden talar om för processorn vad det är den ska göra, och (eventuellt) med vilket data i minnet den ska jobba.

För att förenkla för oss dumma människor som hela tiden gör misstag, så har man gett alla dessa värden egna namn som beskriver dess funktion. Man använder sedan speciella program (alltså monitorn eller assemblern) för att översätta dessa namn till de värden som processorn behöver för att veta vad det är den ska göra. Dessa namn kallas vanligast för *mnemonics*.

Förutom datorns vanliga minne så innehåller processorn små minnespositioner som kallas för *register*. Dessa använder processorn när den jobbar internt med beräkningar, för att slippa referera till RAM-minnet hela tiden (det tar längre tid för datorn att hämta ett värde från RAM varje gång den behöver det än att ha det i ett internt register). De register som finns att tillgå i C64:an är A (ackumulatorn), X (indexregister X) och Y (indexregister Y). Dessa register har, som ni snart kommer märka, sina egna speciella funktioner och en del av dem kan inte användas till vissa saker. Många instruktioner använder direkt dessa register, och har dess namn med i namnet på instruktionen.

Låt oss då ta en titt på ett par grundläggande instruktioner.

LDA (LoaD Accumulator): Denna instruktion hämtar ett väre, antingen en konstant angiven direkt i programkoden eller ett värde från datorns minne, och lagrar det i register A, det vill säga ackumulatorn. För att lägga in konstanten 35 (hexadecimalt) i register A så skulle den kompletta instruktionen lyda LDA #$35, och om man vill lägga in innehållet på adressen $1000 i datorns minne så blir den kompletta instruktionen LDA $1000.

Värt att notera är att då man anger värdet med #$ framför, så handlar det alltid om en byte data hexadecimalt som är en konstant som ligger inlagd i programkoden. Med en konstant menar jag ett värde som inte bara ligger någonstans i datorns minne, utan direkt i programkoden (och då bör man egentligen inte ändra det under programmets gång, det vill säga om man inte håller tungan rätt i mun).

Likaså kan man referera till ett värde som ligger någonstans i datorns minne. Det gör man genom att ange adressen till värdet med ett $ framför. Adressen är nästan alltid ett word (mellan $0000 och $FFFF) hexadecimalt. Det finns dock undantag, som vi ska tala om senare.

STA (Store Accumulator): Den här instruktionen tar det värde som ligger i register A och lagrar det på en adress i datorns minne. Om vi t.ex lägger in värdet 35 i A och sedan vill lägga ut det värdet på adress $1000 så skulle vi först lägga in värdet med instruktionen LDA #$35 och sedan använda STA för att lagra värdet med instruktionen STA $1000.

Här ser vi också exempel på att man inte kan använda alla adresseringssätt på alla instruktioner (vissa instruktioner adresserar man inte alls). Man kan till exempel inte skriva STA #$35, eftersom #$35 inte är någon adress där ett värde kan lagras.

LDX, LDY, STX, STY: Dessa instruktioner fungerar precis som LDA/STA, förutom det att de verkar med X- och Y-registren istället för ackumulatorn.

CMP (CoMPare Accumulator With...): CMP använder man för att jämföra ett värde som redan ligger i ackumulatorn med ett annat. Resultatet indikeras genom förändringen i processorns *flaggregister*, och dessa kan man känna av genom vissa andra instruktioner, främst de i Bxx-serien (t.ex BNE eller BEQ). För att jämföra innehållet i ackumulatorn med 35 så skriver man CMP #$35, och för att jämföra innehållet i ackumulatorn med innehållet på adress $1000 i datorns minne skriver man CMP $1000.

BNE, BEQ, BPL, BMI, BCS, BCC (Branch if...): Branch-instruktionerna är hoppinstruktioner som hoppar från nuvarande position i koden till någon annan plats. Används ofta för att göra slingor med. Instruktionerna betyder i ordning: Branch if Not Equal (hoppa om ej lika), Branch if Equal (hoppa om lika), Branch if Plus (hoppa om större än), Branch if Minus (hoppa om mindre än), Branch if Carry Set (hoppa om carry-flaggan är satt), Branch if Carry Clear (hoppa om carry-flaggan ej är satt).

Carry-flaggan hör också till flaggregistret, och den ska vi prata mer om i nästa nummer.

En nackdel med branch-instruktionerna är att man inte kan hoppa hur långt som helst med dem, närmare bestämt så har de en vidd på ungefär 127 bytes både framåt och bakåt i minnet. För att t.ex hoppa till adress $1030 från adress $1000 (BNE instruktionen ska ligga på adress $1000) med BNE så blir raden BNE $1030, och den hoppar då om resultatet från tidigare jämförelse ej var lika.

JMP (JuMP): Om man behöver hoppa någonstans i koden utan att det beror på en jämförelse, så använder man med fördel instruktionen JMP. JMP $4000 till exempel, hoppar till adress $4000, och den kod som ligger på den adressen fortsätter processorn genast att köra.

NOP (No Operation): Detta är en instruktion som faktiskt inte utför någonting, och används mest för timing. Mer om timing senare i serien.

Nog om instruktioner i detta avsnitt. Låt oss nu ta en titt på hur man enkelt använder monitorn, tätt följt av en grundkurs i Turbo Assembler, för att sedan avsluta med ett par programexempel med förklaringar.

MONITORN

Monitors finns i olika tappningar, och ni förstår väl att jag inte pratar om en bildskärm, utan om ett program med vars hjälp man bland annat kan titta runt i datorns minne. Dels så finns det monitorer som man laddar in som ett program, och sedan startar med SYS Adress. Den mest användbara typen av monitor är dock de som är inbyggde i cartridgar av olika slag, och den monitor som jag kommer utgå från här är den som är inbyggd i Action Replay-cartridgen. De flesta kommandona kommer dock att stämma ganska bra överens med andra monitorer, t.ex The Final Cartridge, men om någonting blir fel och du använder en annan monitor än den i Action Replay, ta en titt i bruksanvisningen för din cartridge och se om det står någonting vettigt där som kan vara till hjälp.

På Action Replay startar man monitorn genom att skriva MON och trycka return, eller eventuellt trycka F8. Man får då veta vissa registers innehåll och lite annan information (som vi inte ska prata om nu) och möts sedan av en punkt-prompt. Det är på den prompten du ger dina kommandon till monitorn, som ofta kan verka ganska kryptiska. Jag ska börja med att förklara fyra grundläggande monitor-kommandon, och sedan bygger vi på den listan allt eftersom.

A (Assemble): Med detta

kommando kan du lägga in din assembler-kod i minnet. Du skriver in instruktionerna en för en, och sedan översätter monitorn dem till rätt koder i minnet. Som parameter ska du ange vart du vill börja skriva in koden, t.ex A 1000 LDA #$10 om du vill lägga in instruktionen LDA #$10 på adress $1000 i datorns minne. När du väl har matat in den första instruktionen så behöver du inte bekymra dig om A-kommandot längre - du skriver bara in en instruktion på varje rad så fixar monitorn resten. Skulle du göra någonting fel så skrivs ett frågetecken ut, och det är bara att undersöka felet och rätta till det. När du är färdig, tryck två return, och du får tillbaka punkt-prompten.

D (Disassemble): Detta kommando använder man för att "avkoda" de siffror som finns på en viss minnesposition i datorn och skriva ut dem på skärmen som mnemonics. Kommandot D 1000 visar t.ex den kod som finns på adress $1000 i minnet. För att kolla vidare i minnet, använd cursor-tangenterna.

Detta kommando kan visa en obegriplig kodsnutt om det nu var så att det t.ex var text som lagrades på den platsen i minnet man tittade på. Text-koderna tolkas då som assembler-kod, och resultatet kan bli lite underligt.

G (Go): Go-kommandot använder du för att starta ett assemblerprogram på en viss adress. G 1000 startar t.ex ett assemblerprogram på adress $1000 i datorns minne.

X (Exit): Detta kommando ger du för att avsluta monitorn.

TURBO ASSEMBLER

En assembler är ett måste för varje riktig programmerare. Assemblern sköter många jobbiga småuppgifter automatiskt när man skriver programmet. T.ex så behöver man inte alltid bry sig om vilken adress i minnet man befinner sig på - man kan använda så kallade labels. Dessutom kan man utan problem lägga till ny programkod mitt i befintlig, utan att behöva skriva om den sista delen av programmet (för ungefär det måste man faktiskt göra i monitorn för att alla adresser ska stämma ordentligt). Man kan också skriva in kommentarer till sin kod, som gör det lättare att komma ihåg vad man sysslar med, och kan vara en ovärderlig hjälpreda när man flera månader senare tittar på koden igen.

En stor nackdel med en assembler är att den, och koden man skriver in, ockuperar minne. Minnet från $8000 och uppåt bör du fortsättningsvis låta bli. Lösningar till detta kan vara att använda en så kallad REU-assembler (och då givetvis också en REU) för att lagra data i extraminnet, eller att assemblera programmet till disk, och sedan ladda in det på en annan dator för att testa det.

Turbo Assembler är den mest använda assemblern, och det är den jag tänker utgå från. De flesta andra assembler är ganska lika den, men jag rekommenderar ändå att använda Turbo Assembler.

När man har startat Turbo Assembler så är det bara att skriva in koden. Det första man bör göra är dock att fastställa vart i minnet koden ska ligga när den assembleras (d.v.s när det du skriver in översätts till riktig kod), och det gör du med *-kommandot. Du skriver *=$1000 (eller någon annan adress) för att följande kod ska hamna på adress $1000 i datorns minne och framåt.

Labels är också bra att använda. Det är ett kort ord, som inte får vara detsamma som något av de ord som assemblern använder, t.ex instruktionsnamn, som ersätter adresser. Ta en titt på följande kod:

```
                *= $1000

test            LDA #$01
                STA $D020
                JMP test
```

Istället för att lista ut vilken adress man ska hoppa till med JMP (om vi vill hoppa tillbaks till början på programmet) så sätter man ut en label och ger den labeln som parameter till JMP. Assemblern översätter sedan det till rätt adress när den assemblerar.

När man har skrivit in ett program så vill man ofta assemblera och/eller provköra det. Assemblerar gör man genom att trycka vänsterpil (alltså den pilen som sitter till vänster om siffran 1 på tangentbordet) och sedan 3. Vill man sedan starta programmet så trycker man på S, annars kan man trycka på någon annan knapp och fortsätta titta i koden. Om man vill både assemblera och starta direkt så kan man trycka F4.

När man väl har startat sitt program så finns inget sätt att återvända till assemblern igen, utan att använda sig av ett litet trick. Först måste du reset:a datorn (INTE stänga av och sätta på den), lämpligtvis med resetknappen på eventuell cartridge. Använder du Action Replay så kan du sedan skriva SYS $9000. Vissa andra cartridgar tillåter inte hexadecimala tal till SYS-kommandot, och har du en sådan (eller kanske inte ens en cartridge över huvud taget) så får du istället skriva SYS 36864.

EXEMPEL

Då var det dags för ett **mycket** enkelt programexempel, som är avsett att skrivas in i Turbo Assembler. Pröva gärna att skriva in det imonitorn också, som en övning, fast då får du översätta labels till adresser istället (alltså till de adresser som labelsarna skulle ha blivit översatta till).

```
                * = $1000
borjan          LDA  #$01
                STA  $D021
                LDA  #$02
                STA  $D021
                LDA  #$03
                STA  $D021
                LDA  #$04
                STA  $D021
                LDA  #$05
                STA  $D021
```

```
LDA  #$06
STA  $D021
LDA  #$07
STA  $D021
JMP borjan
```

Adress $D021 i datorns minne används för att avgöra färgen på bakgrunden. Exemplet visar hur man tar värden och sedan stoppar dem på minnespositioner för att ändra färgen på bakgrunden. Vi tar ett färgvärde i taget och sätter det i $D021, och då ändras bakgrundsfärgen. Sedan hämtar vi nästa värde, och så vidare. När vi har gått igenom ett par färger använder vi en JMP-instruktion för att hoppa tillbaka till början och göra samma sak om och om igen.

Som ett par övningar så kan du pröva att ändra adress $D021 till $D020 istället, och se vad som händer då. Du kan också pröva att stoppa in ett eller flera NOP (som inte tar några parametrar, skriv bara NOP på raden) alldeles efter raderna med STA. Pröva också att utan att titta på exemplet skriva ett liknande program som använder sig av X-registret istället för ackumulatorn.

I nästa nummer ska vi gå igenom ytterligare ett par instruktioner, prata lite om C64:ans minne och gå igenom ett par mer avancerade programexempel. Till dess - öva!

INTERNET-LÄNKAR

Detta är ett par internet-länkar som det inte skadar att ta en titt på:

Relik:
http://www.nethosting.com/~emotion/relik.html

Åtta Bitar:
http://www.mds.mdh.se/%7Edat95pkn/8bitar

FTP:
ftp.funet.fi/pub
(Under katalogen /pub/cbm så finns kataloger med programvara för C64/C128)

ÖVERFÖR FILER MELLAN C64 OCH PC

Peter Karlsson

Det har skrivits en del om våndan med att överföra filer mellan en PC och en C64/C128. Olika lösningar har föreslagits, nollmodemkabel tillsammans med terminalprogram, Big Blue Reader för att skriva direkt till PC-disketter, 64Net som använder PCn som en diskettenhet med mera.

Det finns dock fler sätt, vilka blivit populära nu på sistone, nämligen att koppla in en 1541- eller 1571-enhet direkt till PCn med en specialkabel, och på det sättet överföra data. Det finns flera olika programvaror som kan göra det, och det bästa av allt är att de använder samma kabel, så om man väl byggt en så kan den användas av alla program (i alla fall de jag beskriver här).

Det första programmet som gjordes (som jag känner till) heter X1541, och det var det som var det första som använde den kabel som sedan blivit standard. Just därför kallas kabeln för en "X1541-kabel". Ritning för kabeln medföljer i stort sett varje program som använder den. X1541 i den första versionen är tämligen enkelt uppbyggd. Du har två fönster på skärmen, i det ena ser du din 1541:as innehållsförteckning, och i det andra ser du DOS-katalogen. Du överför filer genom att markera filer i det ena fönstret och välja "Copy" i menyn. Det finns ingen möjlighet att själv ställa in hur filnamnen ska konverteras (en PC använder vanligtvis filnamn på formatet 8.3, max åtta tecken följt av en punkt och sedan max tre tecken), vilket kan ställa till trassel ibland.

Kopieringen görs med 1541:ans standardhastighet, vilket gör att det kan ta en evighet (känns det som) att kopiera över även en liten fil. Men det fungerar, och det är ett stort framsteg gentemot Big Blue Reader, tycker jag. Du kan ju kopiera direkt från din PCs hårddisk, om du vill. Dessutom är du inte begränsad till att använda 5,25"-disketter om du bara har en 1571:a till din C64a. Och BBR fungerar ju inte om du har en 1541.

X1541 har nyligen kommit i en ny version, som inte heter X1541 längre, utan består av två delar, I1541 och U1541. Det ena är en drivrutin som måste laddas in, som ger möjlighet för andra program att utnyttja en tillkopplad 1541:a - ett mycket lovvärt initiativ - och det andra är ett program för att överföra filer. Den här gången styrs det helt från PCns kommandorad, det finns inget menygränssnitt längre. Förutom det som gick att göra tidigare går det nu att kopiera över hela disketter på en gång (till en s.k diskettavbildningsfil (.D64) som används av vissa C64-emulatorer), och även att skriva ut en fil till en vanlig Commodore-skrivare.

Nästa program jag känner till heter TRANS64 och medföljer C64-emulatorn C64s, skriven av Miha Peternel. Detta program kan överföra filer och hela disketter till de tidigare nämnda diskavbildningsfilerna (som C64S sedan kan använda). Programmet kan tyvärr inte användas för tvåvägskommunikation, så du kan inte kopiera tillbaka diskavbildningarna när du väl skrivit dem till PCn.

Mitt favoritprogram i denna genre är dock StarCommander, ett freewareprogram (fritt att använda, de tidigare nämnda är båda shareware, d.v.s kostar pengar att använda) skrivet av Kovacs Balazs från Ungern. För den som använt Norton Commander på PCn är

programmet väldigt intuitivt, det härmar nämligen detta program nästan perfekt. Men även för oss andra är det väldigt lätt att komma igång och arbeta med. På skärmen får man två katalogfönster, som från början visar samma sak, nämligen den aktuella katalogen på PCn. Det går dock via enkla knapptryckningar att byta aktuell enhet, och då ingår 1541:a inkopplad via en X1541-kabel.

Kopieringen är väldigt enkel, man markerar den/de filer som ska kopieras och trycker på funktionsknappen för "Copy" (de aktuella knappfunktionerna finns listade på skärmen, så det är inte svårt att missa). Kopiering kan inte bara göras mellan C64-disken och PCn, det går alldeles utmärkt att kopiera mellan två PC-kataloger eller mellan två 1541-enheter (eller, ska göra, jag har bara en, så jag har inte kunnat testa det). Och förutom detta kan filer även kopieras till/från olika typer av avbildningsfiler, bl.a den diskavbildningsfil (.D64) jag nämnde tidigare, och en s.k bandavbildningsfil (.T64) som i C64S-emulatorn används för att simulera ett kassettband. För andra format, bl.a .P00 som används i emulatorn PC64, medföljer ett lättarbetat konverteringsprogram som körs separat. Det går även att kopiera en hel diskett till en diskavbildningsfil, eller det omvända, att skriva en hel diskavbildningsfil till en diskett. Detta är användbart om man hämtar hela disketter från t.ex Internet. De distribueras nämligen oftast som avbildningsfiler av olika slag.

För den som använder packningsprogrammet ZipCode (packar en 1541-disk till fyra filer) eller arkiveringsprogrammet Lynx (lägger ihop flera sammanhörande filer till en stor fil, utan packning) medföljer program för att manipulera dessa, t.ex kan ZipCode-filer omvandlas till en .D64-avbildningsfil, som sedan kan skrivas ner på en diskett. Detta är mycket behändigt.

Alla dessa program, förutom TRANS64, kan hämtas hem från Internet på de flesta välsorterade ftp-ställen (ftp.funet.fi, ccnga.waterloo.edu, m.fl) för den som vill ha dem.

Det finns ett program till som jag känner till som använder X1541-kabeln som jag inte nämnt ovan, eftersom det fungerar på ett litet annorlunda sätt. Det programmet heter SERVER64, och fungerar omvänt mot de ovanstående programmen, det låter nämligen PCn agera diskettstation, och man kopplar ihop PCn direkt med sin dator, och kan på det sättet direkt läsa och skriva filer på PCns hårddisk. Programmet är fortfarande i betastadiet, och den version jag har testat lyckades jag aldrig få att fungera (det ska finnas en nyare, vilken jag inte lyckats lokalisera ännu), av okänd orsak. Jag har dock läst att andra har kört det med gott resultat. De tidiga betaversionerna är dock väldigt begränsade, det enda som kan användas är LOAD och SAVE, det går inte att lista katalogen, eller att öppna datafiler. För den som inte har råd att köpa 64Net kan detta vara ett alternativ, då SERVER64 så vitt jag vet är gratis att använda.

Allt eftersom de här programmen blir populärare och används mer utbrett, kommer svårigheterna med att föra över data mellan C64:an och PCn att vara borta, och ersatta med lätthanterliga program. En mycket trevlig utveckling, om jag får säga vad jag tycker.