HÅLL UT!!

I DECEMBER KOMMER DET NYA NUMRET AV SPECTRAVIEW'.

D.V.S. ETT

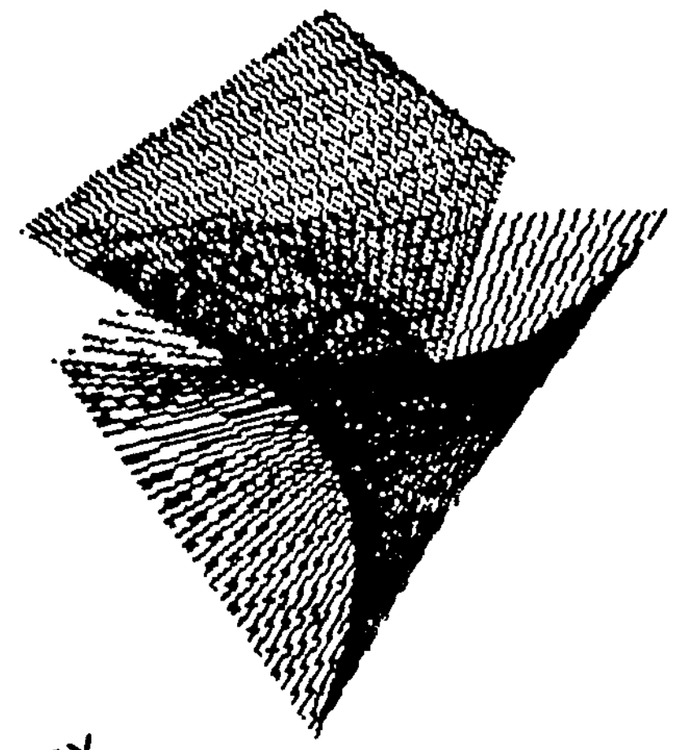
JULNUMMER

DÄR PRESENTERAS EVENTUELLA VINNARE I VÅR STORA I&-TÄVLING.

DET BLIR PROGRAMLISTNINGAR OCH TIPS M.M. M.M. M.M.

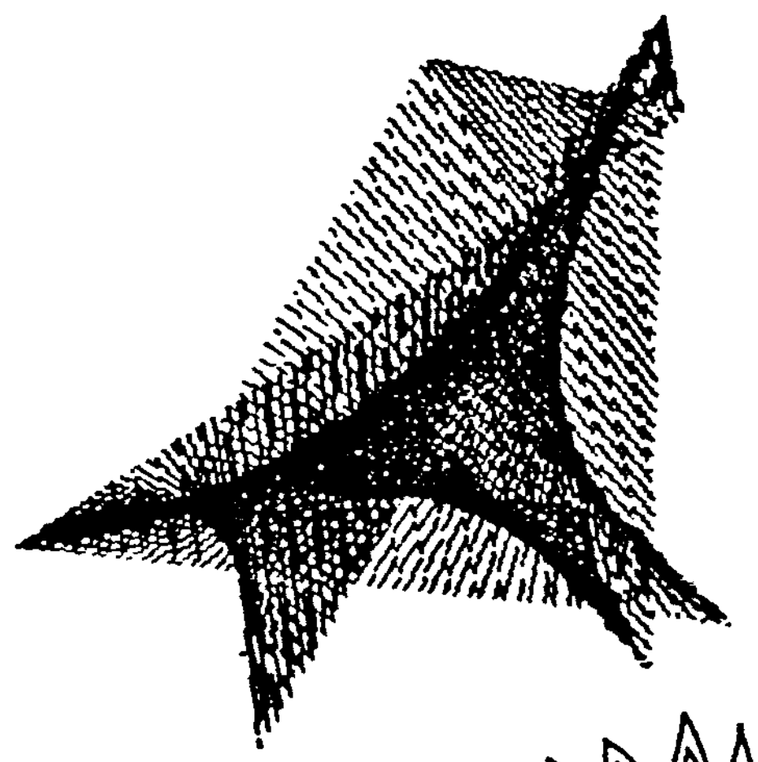
SPECTRAVIEW

NR 4-5 ÅRG 1



MASKINKODSSKOLA

```
D000: 7E          IGEN  LD A,(HL)
D001: FE00              CP 00
D003: C8                RET Z
D004: DF                RST 18
D005: 23                INC HL
D006: 18 F8             JR IGEN
SUCCESSFUL ASSEMBLY !!
```



NSUK

32-SIDIGT DUBBELNUMMER

A N N O N S E R

Medlem DC vill köpa en SV 601 Super Expander till billigt pris. Hon heter Kajsa Söderström och har telefonnummer 018-32 22 59.

Medlem 17B vill köpa/byta program. Han heter David Munck och har telefonnummer 08-96 41 64.

Om ni har ett adventurespel att sälja för högst 60 kr. så kan ni skriva och berätta om det för:
Per Aranda
Klingsbro, Ringstorp
585 90 LINKÖPING

Just det, har ni något adventurespel så kan ni väl skicka in det till oss för test/publicering, det finns ett akut behov av adventures. Om det är bra kan det mycket väl hända att RONEX i Malmö vill sälja det.

Säljes: SV-318 paket, Quickshot II, Riktigt tangentbord, Basic på spectravideo, Maskinspråksmanualen, Användarhandledning. Garanti till 24/12. Ring 046/20 98 61

Böcker säljes. Basic på spectravideo, Grafik ljud sprites, Basicboken för spectravideo. Ring Erik Liljencrantz 0150/271 95

Dessutom har vi två litet mer kommersiella annonser (Sådana får man sätta in till priset 200 kr/helsida, 125 kr/halvsida och 75 kr/kvartssida):

Esthetic Software heter nordiska spectravideoklubbens egen mjukvaruleverantör efter det att klubben har blivit skild från Spectrasoft HB. De har hittills gett ut två program. (Medlemmar i NSVK har 15% rabatt om de anger namn+medlemsnummer)

HUNCH.MAN är ett spel där man skall springa/hoppa/ta sig igenom olika faror för att rädda en prinsessa. Spelet är så uppbyggt att ju längre man kommer desto svårare blir det och desto mer avancerad blir grafiken. Det kostar 75:-

ESPRITE är ett program som mycket snabbar upp skapandet av speritar. Några plus: man kan spara spritar för att senare hämta tillbaks dem, man kan få programmet att skriva ner ett BASICprogram direkt på band/kassett som laddar in spritarna via datasatser. (Så slipper man skriva av spritedata som vid vissa andra program.) Det kostar 75:-

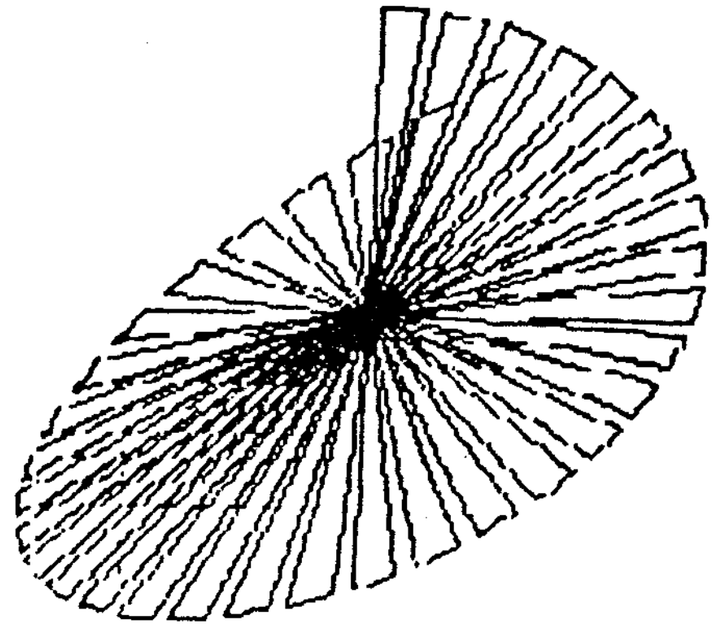
Om du vill ha programmen på disk i stället för kassett så tillkommer 35:- oberoende av om du är medlem eller inte.

Från medlem 116 Erik Liljencrantz har vi fått en tabell som visar hur tangentbordet tolkas av datorn. Följande tio bytes innehåller tangentbordets status. I vanliga fall är alla bittar satta, men om någon tangent trycks ned så nollställs den bitten.

Värde:	128	64	32	16	8	4	2	1
Bit:	7	6	5	4	3	2	1	0
Adress:	Tangent:							
FD80	7	6	5	4	3	2	1	0
FD81	-	.	E'	,	Ä	ö	9	0
FD82	G	F	E	D	C	B	A	+
FD83	O	N	M	L	K	J	I	H
FD84	W	V	U	T	S	R	Q	P
FD85	JU	<=	Ü	<	Å	Z	Y	X
FD86	JV	CR	ST	ESC	RG	LG	CT	SH
FD87	JN	INS	CLS	F5	F4	F3	F2	F1
FD88	JH			SEL	CL	DEL	=>	SP

Förkortningar: JU,JV,JN,JH: Joystick Upp, Vänster, Ner, Höger. CR=Enter. ST=STop. RG=Right Graph. LG=Left Graph. CT=CTrl. SH=SHift. SEL=SELect. CL=Caps Lock. SP=SPace (Mellanslag). Att E':t inte är äkta beror på att tidningen är skiven på ett engelskt ordbehandlingsprogram till spectravideo, Just Write Jr. Detta finns dock översatt till svenska ute i affärerna.

VÅR stora medlemsåldersräkning äravgjord. Resultatet blev: 18 år i medelålder. (Ganska exakt) Därför har vi tänkt utlysa en tävling. Den som på roligaste sättet använder talet 18 vinner ett pris som tills vidare är hemligt. Allt från att man är ettan som skall skjuta sig igenom en massa nior för att nå fram till åttan, till ett gyllene blinkande 18 som reser sig ur sjön som i en tecknad film är vad vi väntar oss... Vår adress finns på första sidan.



M A S K I N K O D S S K O L A Del 2

Det här med maskinkod är något som kanske inte är det lättaste att lägga upp pedagogiskt. Därför tar vi det hela uppdelat på flera klumpar, som kan hänvisa till varandra. Första klumpen handlar om register.

REGISTER är något som används i maskinkod. Det finns 10 register i speckan som vi skall gå in på litet närmare. De är:
A Accumulatorn. Det viktigaste registret. Inblandat i alla jämförelseoperationer och de flesta räkneoperationerna.
B B-registret. Ett ganska vanligt register. Används speciellt som räknare i vissa funktioner (Ex. DJNZ d).
C C-registret. Ett ganska vanligt register. Inblandat som pekare när det gäller IN och OUT-instruktioner.
D D-registret. Ett helt vanligt register.
E E-registret. - " - - " -
F Flaggregistret. Innehåller enskilda bittar som visar om resultatet blev noll, negativt, om operationen utföll med minnessiffra eller overflow.
H H-registret. Helt vanligt.
L L-registret. Helt vanligt det också.
IX Indexregister X. Kan användas som pkare o.dyl. i stället för HL.
IY Indexregister Y. Se IX.

Med "Helt vanligt register" menas ett register som kan innehålla en byte, som kan flyttas mellan register eller sparas i en adress. Om man kan göra något speciellt annat med registret anges det ovan. IX och IY är tvåbytes register.

Register kan dessutom användas två och två som sextonbittarsregister. Dessa register är följande:

BC används mest som räknare, men ibland också som pekare.
DE används ibland som pekare, men mest för mellanlagring av data som senare skall antingenbearbetas i annat register eller adderas till ett annat register.
HL är det viktigaste registerparet. Det används som minnespekare och vid sextonbittars räkning.

Till IX och IY kan man lägga till en "Displacement" (d). Detta räknas fram på samma sätt som displacement i relativa hoppinstruktioner.

Flaggorna i flaggregistret går också att använda när man skall göra en test (Hoppa om resultatet blev noll etc.).

HOPPINSTRUKTIONER används för att förflytta sig i programmet (jfr. GOTO). Adressen kan antingen anges direkt (JP F30AH) eller relativt med sk. displacement (JR d). Detta är snabbare, men man kan bara hoppa 127 bytes framåt eller 128 bakåt.

Displacement räknas fram enl. följande:
Hopp framåt: Byten direkt efter sista byten i instruktionen är byte 00, nästa byte 01 osv. till 7F.
Hopp bakåt : Sista byten i instruktionen är FF, den innan den FE osv. till 80. För den som känner till vad tvåkomplement är låter det nog bekant.

Displacement för IX och IY räknas fram på samma sätt, fast innehållet i registret räknas som "första byten efter instruktionen".

VILLKOR kan anges i hopp eller dyl. Det anges på det sättet att instruktionen utförs om flaggan som står efter instruktionen är satt. De flaggor som kan förekomma är:

C Gör instr. om Carry är satt.
M Gör instr. om resultatet blev negativt.
NC Gör instr. om Carry inte är satt.
NZ Gör instr. om Zero inte är satt.
P Gör instr. om resultatet blev positivt.
PE Gör instr. om det blev jämn paritet.
PO Gör instr. om det blev udda paritet.
Z Gör instr. om Zero är satt.

ARITMETISKA instruktioner är tex. ADD eller SBC. ADD adderar två tal. Ex. ADD A,C. Där adderas C till A och resultatet lagras i A. Om resultatet skulle bli mer än åtta bittar lagras de åtta minst signifikanta bittarna i A och den nionde i Carry. Om alltså ett resultat blev större än 255 så sätts Carry. Om resultatet (i det här fallet innehållet i A) skulle bli noll så sätts Zeroflaggan. Dessa är de två viktigaste flaggorna. ADC gör samma sak, fast Carry läggs också till resultet. (Om Carry är satt blir resultatet 1 större.) SUB är subtraktion, och SBC är subtraktion där också Carry dras ifrån.

Instruktionen CP är en instruktion som inte påverkar något mer än flaggregistret. Det får det värde det skulle fått om man skrivit SUB. SUB påverkar accumulatorn och flaggregistret. E. om man skulle skriva SUB 07 så dras 7 ifrån accumulatrn och flaggregistret påverkas beroende på resultatet. Om man i stället skulle skrivit CP 07 skule accumulatorn varit oförändrad, men Zero skulle vara satt om accumulatorn innehöll 7. Läs igenom det igen så fattar du nog.

DEC och INC är två operationer som påverkar flaggregistret om det gäller en åttabittars operand, annars inte. DEC minskar ett register/innehållet i en adress med 1, och INC ökar med ett.

LOGISKA funktioner som AND, OR och XOR finns också. De ger resultat mellan accumulatorn och något register/innehållet i en adress. Resultatet hamnar i accumulatorn. Andra logiska funktioner är NEG och CPL. NEG inverterar alla bittar i accumulatorn (Lika med XOR FF). CPL omvandlar talet i accumulatorn till ett negativt tal i tvåkomplementform. En funktion för att nollställa accumulatorn är XOR A, vilket går nästan dubbelt så fort som LD A,0. Dessutom tar den upp bara en byte, LD A,0 tar upp två.

LADDA register/minnesadresser med värden gör man genom instruktionen LD d,k. D står för destination och k för källa. Ingen LD-instruktion påverkar flaggregistret. Om det är en adress som man skall ladda till/från skall den anges inom parentes. Ex. LD (HL),B sparar B:s värde i den adress som HL pekar på. LD

HL,(FA30) laddar L med vad som ligger i FA30 och H med vad som ligger i FA31 (se förra numret). LD D,0FH laddar D med 15 decimalt.

STACKEN är mycket använd i maskinkod. Den kan liknas vid en burk där man kan lägga och hämta saker. Det som senast lades dit är det som först kommer ut. Denna typ av minne brukar kallas för LIFO-minne (Last In First Out) till skillnad från en kö, som är ett LILO-minne (Last In Last Out). Sparar data på stacken gör man genom att skriva PUSH registerpar. Man får tillbaks registerparet med POP registepar. Även A och F räknas som ett registerpar AF. För att läsa av flaggregistrets värde kan följande rutin användas: PUSH AF EX (SP),HL LD A,L POP HL
Vad denna rutin gör är att först spara AF på stacken, sedan byta det sista värdet på stacken (det som just sparades) med värdet i HL. L får då F:s värde. Sedan laddas A med L, och genom POP HL så återställs HL. Resultatet blir att A laddas med innehållet i F.

Stacken används också av CALL och RET-instruktionerna. De kan liknas vidGOSUB och RETURN. CALL hoppar till ny adress och sparar den gamla på stacken. RET hoppar till den adress som ligger sist i stacken. GLÖM ALLTSA INTE KVAR NAGONTING PA STACKEN, DET KAN FA ALLVARLIGA KONSEKVENSER. Detta går också att utnyttja om man vill att datorn skall göra rutinen som HL pekar på. Då skriver man PUSH HL RET. Då sparas HL på stacken för att sedan hämtas tillbaks som den adress datorn hoppar till vid RET.

BITMANIPULERING kan man också göra i maskinkod. Dels finns det instruktioner för att sätta (SET), släcka (RES) och testa (BIT) bittar, sedan finns det också varierande shiftfunktioner. Dessa beskrivs bäst av bilderna nedan. Ex. BIT 1,A sätter Zero om Bit 1 (Värde 2) i acumulatorn är noll, annars släcks Zero.

Nå, nu har vi skrivit vår snabba och minnessnåla maskinkod. Hur skall vi nu mata in den i datorn och sparka igång den? Det finns två sätt att lagra maskinkod i minnet. Det ena är i en sträng (lämpligt för korta snuttar som gör saker man inte skulle kunna göra i BASIC) och det andra POKE:at på plats. För att kunna lagras i en sträng måste maskinkoden vara max. 255 tecken lång. Dessutom bör den vara mycket hastighetsökande, ex.vis nedåtscrollning, dumpa/hämta en massa data från videominnet o.dyl. Sedan tar man och lägger sin maskinkod i strängen, och varje gång man skall kalla på maskinkoden skriver man DEFUSR1=VARPTR(MK$):P=USR1(0)
Här förutsätts att det var MK$ som var maskinkodsvariabeln. OBS! Glöm inte att CLEAR:a tillräckligt med utrymme för strängarna.

Om man skall POKE:a in maskinkoden skall man först reservera en del minne som datorn inte skall kunna komma åt. Det gör man med kommandot CLEAR. Ge CLEAR <strängutrymme>,<Början-av-fria-bytes> (Det fanns väl ingen som inte visste att om man får out of string space så kan man reservera mer utrymme för strägar med den första parametern efter CLEAR (200 vid uppstart) ?)
<Början-av-fria-bytes> är vid uppstart F500H (D5B8H). Värdet inom parentes gäller om du har Disk Basic inne i maskinen. Om du vill reservera 256 bytes för eget bruk (ex. maskinkod...) kan du skriva CLEAR 200,&HF400 (CLEAR 200,&HD4B8). Om du behöver anvanda strängar med mer än 200 tecken skall du ändra 200 till något annat. Nu kan inte datorn skriva något mellan F400 F4FF (D4B5 och D5B4). Sedan är det bara att POKE:a in maskinkodsprogrammet med början på F400. Detta förutsätter att maskinkodeär max. 256 bytes lång, annars får man reservera mer minne genom att sätta ett lägre tal som andra parameter i CLEAR. OBS! Skriv innågonting ovanför F4FFH, F500H till FFFFH används av datorn som systemvariabler.

Här har vi ett litet maskinkodsprogram som genererar ett "laserpistolsljud".

```
F400:  START   PUSH AF            F5
F401:          PUSH BC            C5
F402:          PUSH HL            E5
F403:          LD C,84H           0E84
F405:          LD A,07H           3E07
F407:          OUT (80H),A        D380
F409:          LD A,FEH           3EFE
F40B:          OUT (C),A          ED79
F40D:          LD HL,0FFFH        21FF0F
F410:  IGEN    LD A,01H           3E01
F412:          OUT (80H),A        D380
F414:          OUT (C),H          ED61
F416:          XOR A              AF
F417:          OUT (80H),A        D380
F419:          OUT (C),L          ED69
F41B:          EXP (SP),IX        DDE3
F41D:          EXP (SP),IX        DDE3
F41F:          EXP (SP),IX        DDE3
F421:          EXP (SP),IX        DDE3
F423:          DEC HL             2B
F424:          LD A,H             7C
F425:          OR L               B5
F426:          JR NZ,IGEN         20E8
F428:          POP HL             E1
F429:          POP BC             C1
F42A:          POP AF             F1
F42B:          RET                C9
```

Vad detta program gör är att först på stacken spara de register som kommer att användas, sedan sätter det upp mixern till ljud ur kanal A (lika med SOUND 7,254) och sätter HL till OFFFH (4095 decimalt), vilket är den lägsta ton ljudgeneratorn kan producera. Sedan sätts register i i ljudchippet lika med H (mest signifikant byte) och register 0 till L. Därefter utförs en kort fördöjning för att ljudet inte skall gå för snabbt. Sedan minskas HL, och prograet kollar om HL har kommit till noll (Enda gången H OR L är noll är när HL ä noll.), och om det inte har gjort det hoppar programmet relativt tillbaks och producerar en litet högre ton. Om HL är noll återställs registrena från stacken och programmet hoppar tillbaks till basic. Här kommer detta program med BASIC-laddare.

```
10 CLEAR 200,&HF400 ' DETTA ANDRAS TILB400 FÖR DEM MED DISK.
20 AD=&HF400 ' DETTA MED.
30 READ A$ : IF A$<>"*" THEN POKE AD,VAL ("&H"+A$)  :  AD=AD+1  :
GOTO 30
```

```
40 DEFUSR1=&HF400 ' ANDRAS TILL B400 ODU HAR DISK.
50 P=USR1(0)
60 END                                                    DATA
70
F5,C5,E5,0E,84,3E,07,D3,80,3E,FE,ED,79,21,FF,0F,3E,01,D3,80,ED,61
,AF,D3,80,ED,69,DD,E3,DD,E3,DD,E3,DD,E3,2B,7C,B5,20,E8,E1,C1,F1,C
9
```

VAR 135 % SÄKER PÅ ATT RAD 70 ÄR RÄTT INMATAD ! SPARA FÖR SÄKERHETS SKULL PROGRAMMET INNAN DU KÖR DET !!

För den som vill programmera mer i maskinkod rekommenderas boken "Programming the Z 80" av Rodnay Zachs från Zybex förlag. Dessutom rekommenderas Maskinkodsmanualen till Spectravideo av Andrzej Felczac från Ronex. Den förutsätter att man kan maskinkod, med ger en hel del matnyttigt.

Illustrationerna av shiftfunktionerna kräver kanske sin förklaring. Vad man ser är ett register/en byte med åtta bittar. I vissa fall förekommer Carry (rutan med C i) eller en nolla som skiftas in (rutan med en nolla i). Dessutom förekommer litet fler funktioner än vad som här visas, ex. RL är RR fast med pilarna åt andra hållet.

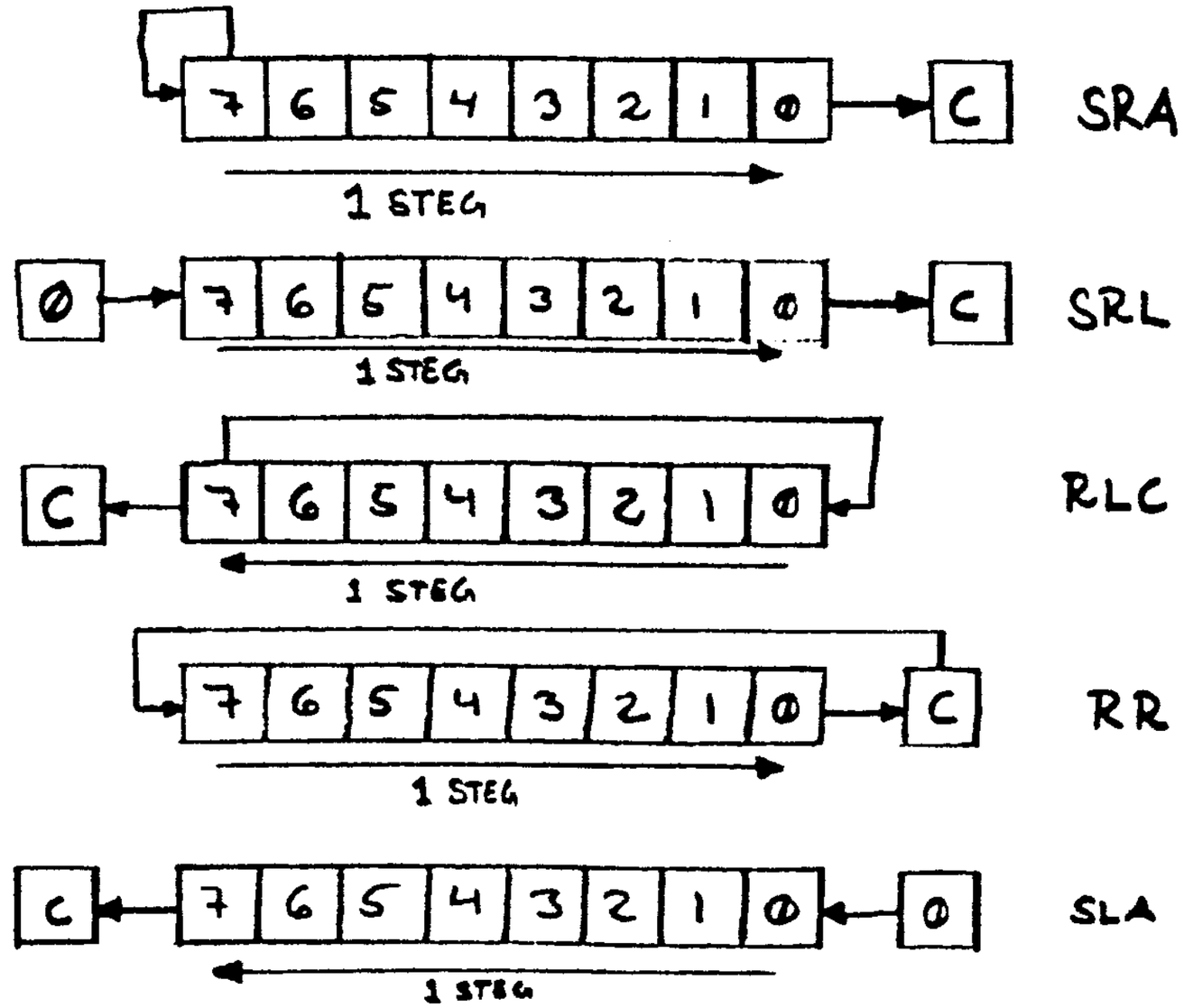
Ex. Vid SRA blir Carry lika med bit 1, bit 1 lika med bit 2... bit 6 lika med bit 7 och bit 7 behåller dessutom sitt värde.

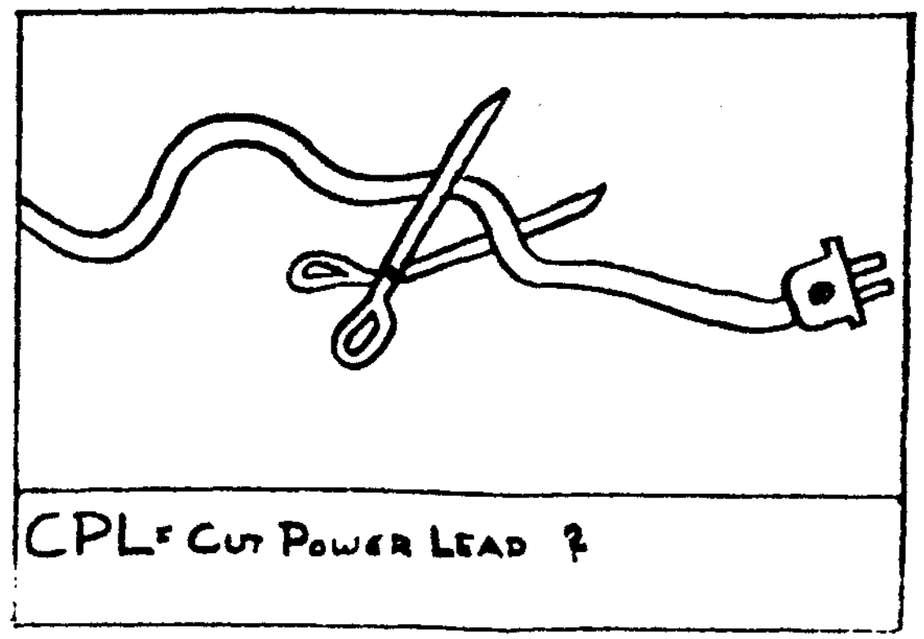
Det här med minne skall dessutom diskuteras litet närmare. I en 318 börjar minnet på adress C000. Denna byte är alltid 0. Sedan ligger den area där BASICprogram och variabler ligger upp till F500 (D5B8). Från och med den adressen ligger systemvariabler och annat, där får man alltså inte skriva program. Däremot kan dessa variabler användas, både i maskinkod och i BASIC. Några användbara variabler skall vi nedan gå in på.

Adress (hex)	Längd	Användning
F540	1	Sista printerinstruktion. 0 om det var line feed, annars print.
F542	1	Vilken device skrift skall gå till. 0: skärmen, 1: skrivare.
F997	2	Pekare till fildata.
F9B4	2	Sista adressen för BSAVE.
FA06	1	Funktionstangentvisning. FF för att visa dem, 00 för att inte göra det.
FE38	1	Caps Lock. 0=av, 20H=på.
FE58	2	Startadressen för binärfilen.
FE79	3	Interrupthook. Här ligger vanligtvis C9 00 00 (RET), men du kan här lägga in en JMP om du vill att nån rutin skall utföras var 1/50 sekund. Den rutinen skall sedan avslutas med RET.

En 328 har minne som ovan, fast startadressen är 8000H för RAMet. Dessutom har en 328 32K RAM som man måste trixa litet för att

Om du får Syntax Error när du kallar på en maskinkodsrutin beror det på det sätt datorn för över data mellan BASIC och maskinkod. Ha för vana att alltid börja dina program med PUSH AF PUSH BC PUSH DE PUSH HL, och ha ett ställe där alla återhopp till BASIC sker. Om man vill hoppa åter även från andra ställen skall dessa hoppa hit. Stället skall se ut så här: POP HL POP DE POP BC POP AF RET.





KOMPLEMENT TILL MASKINKODSSKOLAN

Om man programmerar i maskinkod finns det några olika rutiner i ROM som det kan vara praktiskt att använda. Några av dem skall vi ta upp här.

Den första är en rutin som skriver ut tecknet som ligger i A. Den når man enklast genom att skriva RST 18 i programmet. Rutinen använder bla. adress F542 som står förklarad i maskinkodsskolan.

För att jämföra HL och DE finns det också en enbytes-instruktion. RST 20 heter den. Den sätter C om HL>DE och Z om HL=DE.

Om man vill at datorn skall vänta på en tangent, vars kod sedan hamnar i A kan man skriva CALL 003E.

Om man vill kolla om ctrl-stop är nedtryckt kan man använda CALL 005C. Den ger C satt om ctrl-stop är nedtryckt.

Om man vill producera ljud skall man först OUT:a soundregistret (0-13) på adress 88H och sedan out:a data på adress 8CH.

Om man vill skriva i en bestämd position i videominnet så skall man använda följande rutin:

(HL innehåller adress, A innehåller data)

```
PUSH BC
LD C,81H
OUT (C),L
SET 6,H
OUT (C),H
DEC C
OUT (C),A
POP BC
RET
```

Den kallas på med CALL. Om man vill skriva flera bytes så kan man använda följande rutin. HL=varifrån i minnet data skall hämtas, B=hur många bytes som skall skrivas, DE=var i videominnet data skall placeras.

```
PUSH BC
LD C,81H
OUT (C),E
SET 6,D
OUT (C),D
DEC C
OTIR
POP BC
RET
```

Data som ligger i minnet från HL och B bytes uppåt skrivs.

Om man i stället vill läsa ur videominnet ser rutinen för en byte ut så här:

```
PUSH BC
LD C,81H
OUT (C),L
OUT (C),H
IN A,(84H)
POP BC
RET
```

Och för flera bytes (HL=var data skall placeras, DE=varifrån i videominnet data skall hämtas, B=hur många bytes):

```
PUSH BC
LD C,81H
OUT (C),E
OUT (C),D
LD C,84H
INIR
POP BC
RET
```

Ja, det var väl en del matnyttigt. Nu hoppas jag att ni skickar in riktigt många maskinkodsspel till oss för test/publicering. Lycka till !

G R A F I K K L A D D ? ? ?

När man ritar med grafikmode 2 på speckan så har ni säkert märkt att det blir litet "kladdigt" om man använder många färger. Vad detta beror på, och hur man kan utnyttja detta, skall vi gå in på i den här artikeln. Först skall vi lära ut hur det videochip spectravideo har genererar sina bilder. Det har fyra olika grafikmoder, av vilka vi kan komma åt tre med SCREEN. I grafikmode 0 (Vanlig textmode) är minnet organiserat enl. följande:

0-959: Tecknen som visas på skärmen.

2048-4095: Hur tecknen ser ut.

Tecknen som visas på skärmen ligger i följande kod:

0: Mellanslag 1: ! 2: " 3: #
4: ¤ o.s.v. till 94: ü
Sedan kommer reverserade tecken från 96: Mellanslag 97: !
o.s.v.
Sist kommer de grafiska tecknen med början på kod 192.

För att se vilka tecken som finns kan man skriva
FOR T=0 TO 255:VPOKE T,T:NEXT T

Kanske borde det tilläggas att man läser ur en adress i videominnet med VPEEK(adress) och skriver med VPOKE (adress),värde.

Hur de vanliga bokstäverna ser ut går att ändra på genom att ändra mellan 2048 och 4096. Där ligger informationen om hur tecknen ser ut i samma ordning som koderna representerar, fast varje tecken tar upp 8 bytes. Alltså, hur mellanslag ser ut ligger i adresserna 2048-2055, ! i 2056-2063. Följande lilla porgram kan användas för att ändra teckenuppsättningen. Observera att de två minst signifikanta bittarna (bit 0 och 1) inte syns!

```
10 REM ÄNDRA TECKEN
20 INPUT"Vilket tecken";A$
30 CLS:WIDTH 39:LOCATE 0,0:PRINTA$:T=VPEEK(1)
40                          CLS:FORZ=2048+T*8                          TO
2055+T*8:B$=BIN$(VPEEK(Z)):B$=STRING$(8-LEN(B$),48)+B$:PRINTUSING
"? #       #";B$:NEXT
50   LOCATE   0,0:FORZ=2048+T*8   TO   2055+T*8:INPUT     B$:VPOKE
Z,VAL("&b"+B$):NEXT Z:CLS:FORZ=0TO959:VPOKEZ,T:NEXT
60 A$=INPUT$(1):CLS:GOTO20
```

Vad detta program gör är att först ta in ett tecken från tangentbordet, sedan skriva ut det längst upp till vänster på skärmen. I och med att WIDTH 39 används är detta position 1. Sedan läses denna position av för att få ut vilken kod tecknet har (Den skiler sig ifrån ASCII-koden). Detta tecken hämtas och visas på skärmen i binär form. Sedan INPUT:as de åtta -värena igen, det är då man kan göra sina ändringar, och lagra dem i videominnet.

I SCREEN 1 är uppbyggnaden mer komplex. Håll i er så kommer

beskrivningen

```
0-2047:       1:a tredjedelen av teckengenratorn
2048-4095:    2:a tredjedelen av teckengenratorn
4096-6143:    3:e tredjedelen av teckengenratorn
6144-6399:    1:a tredjedelen av skärmen
6400-6655:    2:a tredjedelen av skärmen
6656-6911:    3:e tredjedelen av skärmen
6912-7039:    Sprite attribute table
8192-10239:   1:a tredjedelen av färgtabellen
10240-12287:  2:a tredjedelen av färgtabellen
12288-14335:  3:e tredjedelen av färgtabellen
14336-16383:  Spritarnas utseende
```

Vi börjar med att konstatera att SCREEN 1 är uppbyggd av 3 skärmar som tar hand om en tredjedel av skärmen var. Varje skärm har en egen teckengenerator, färggenerator och skärm med 256 tecken. Detta ger att hela skärmen består av 768 tecken, närmare bestämt inordnade i 24 rader om 32 tecken. Om man använder grafiken med LINE, PSET och med mera så är från början teckengeneratorerna och färgtabellerna satta till noll. Skärmarna är alltid lika med (adressen)AND 255 (alltså den minst signifikanta byten). När man sedan skall sätta en punkt så räknar man fram vilket tecken som skall ändras i vilken skärm, och ändrar tecknets UTSEENDE, inte var tecknet befinner sig.

Detta medför att det rent teoretiskt är möjligt att använda SCREEN 1 som textskärm, med färg o.s.v. Detta är dock inte problemfritt, så i chippet finns möjligheter att använda en fjärdgrafikmode som är speciellt avsedd för 24x32 teckens färgtext. Dengår dock inte att komma åt med basic, utan där får man använda maskind.

Om man nu skulle vilja göra en figur, ex.vis en meteor, så går man tillväga ungefär som i SCREEN 0. Dock räknas alla åtta bittarna. Dessa decimala data skall användas för en meteor:

```
Bit     7 6 5 4 3 2 1 0   Decimalt
Värde 1286432168 4 2 1

          * *   * *        54
        * *     * * * *    111
        * * * *   * *      123
      * *     * * * *      226
      *   *   * *   *      173
      * * * * * * * *      255
        * * *   * *        118
            * * * *        30
```

Ett program som lägger in meteoren som tecken 1 i skärm 0 kan se ut så här:

```
10  REM METEOR
20  COLOR 0,0,0:SCREEN 1
30   FOR T=0 TO 7
40    READ A%
50    VPOKE 8+T,A%
60   NEXT T
70  DATA 54,111,123,226,173,255,118,30
80  GOTO 80
```

Om du nu kör programmet blir skärmen bara svart. Det beror på att tecknet inte har någon färg. Nu skall det få det. Man anger vad varje byte tecknet består av skall ha för färg. De fyra mest signifikanta bittarna anger vilken färg de punkter som är satta i tecknet skall ha, de fyra minst signifikanta vilken färg de icke satta skall ha. Om man skall ange färg på tecknet kan man komplettera programmet med följande rader:

```
80   FOR T=0 TO 7
90    READ B%
100   VPOKE 8192+8+T,B%
110  NEXT T
120 DATA 96,96,96,96,96,96,96,96
130 GOTO 130
```

Koden 96 för färgen är 6*16+0, 6 är mörkröd, 0 är genomskinlig. Alltså kommer meteoren att vara röd med den bakgrundsfärg som anges som sista parameter i COLOR. Om man nu vill att punkterna i meteoriten skall vara gula kan man ändra 96 till 96+10=106 för de bytes där meteoriten går över hela byten (Annars kommer bakgrunden också att bli gul, vilket inte var meningen). Rad 120 lyder då:

```
120 DATA 96,96,106,106,106,106,96,96
```

Om man nu vill ha hela skärmen full av meteorer tror ni kanske att det bara är att vpoka in dem mellan 6144 och 6911. Vi skall se vad som händer. Lägg till dessa rader:

```
130  FOR T=6144 TO 6911
140   VPOKE T,1
150  NEXT T
160 GOTO 160
```

Vad nu då? Bara den översta tredjedelen blev fylld med meteorer. Det beror på att vi bara har definierat meteoren i 1:a tredjedelen av teckengeneratorn. För att definiera den i de andra två, lägg till följande:

```
53    VPOKE 2048+8+T,A%
56    VPOKE 4096+8+T,A%

103   VPOKE 10240+8+T,B%
106   VPOKE 12288+8+T,B%
```

Provkör nu programmet. Nu har du framför dig en hel skärm fylld med gula och röda meteorer. Om du vill kan du ändra så att meteorerna får olika färg på olika tredjedelar av skärmen. För att få ett lättläst program kan man då dela upp färgpokningen på tre DATAsatser, men det är inte nödvändigt.

Om man vill sätta ut en meteor på en speciell plats är adressen man skall VPOKEa tecknets kod (i det här fallet en 1:a) Y*32+X+6144. 0<=Y<=23 , 0<=X<=31. Detta medger att man definierar olika figurer som man sedan ritar upp en bakgrund med. Om figurera sedan är sprites kan man lätt känna av var de befinner sig genom att känna av vilket/vilka tecken som finns där spriten befinner sig. I sådana fall är adressen INT(Y/8)*32+INT(X/8)+6144. X och Y skall ligga inom grafikskärmen.

Den skärm man får fram genom att använda dessa metoder har också den fördelen att den går att scrolla genom att ändra på 768 tecken i stället för det sextondubbla (12288), vilket skulle gå långsamt även i maskinkod. I för sig går inte 768 teckens scroll speciellt snabbt i BASIC, men läs maskinkodsskolan här i tidningen och använd vad du får lära dig.

Att resultatet blir kladdigt när man ritar med LINE och många färger beror på att varje byte i teckengeneratorn bara kan ha två färger som representeras av tänd och släckt bit. Om man försöker rita med fler färger inom samma byte blir resultatet att de två senaste färgerna används.

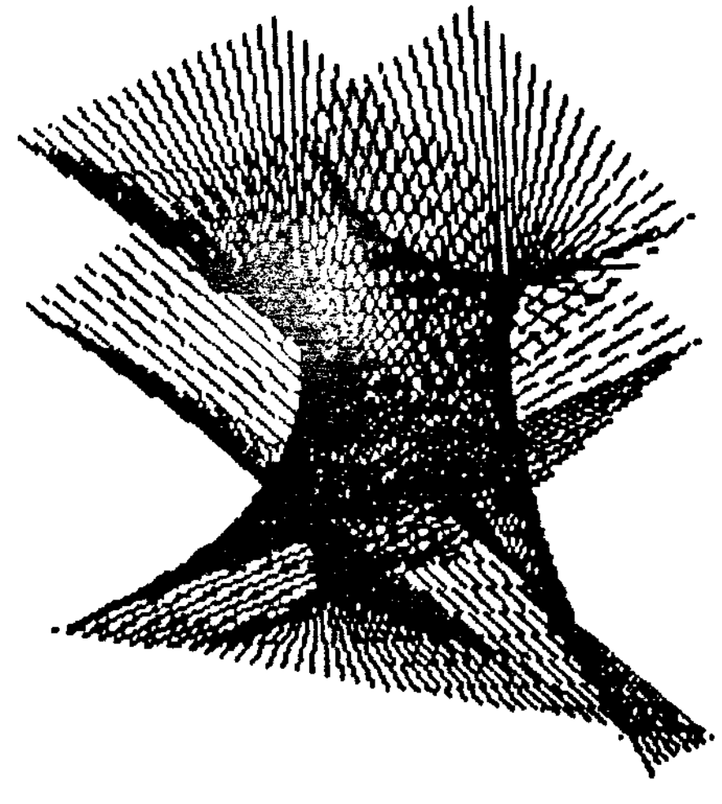
Om man nu inte vill använda det sätt att bygga upp en bakgrund som har beskrivits här, utan använder LINE, DRAW e.dyl. kan det hända att man tröttnar på att titta på när datorn ritar. Det går då att släcka skärmen genom följande rad:

```
OUT 129,x:OUT 129,129
```

x skall vara 162 om man använder 16x16 sprites, annars 160.
Om man vill tända skärmen igen skriver man samma sak fast x=224 för 16x16 sprites, och x=222 för 8x8 sprites.

Om man kör med förstorade sprites lägger man till 1 till dessa värden. OBSERVERA ATT DESSA VÄRDEN ENBART GÄLLER SCREEN 1 !

Ha så roligt med grafiken !



```
=255ORH!(OTHENA(2)=0:PUTSPRITE2,(0,192),7,2:RETURN30ELSEA(2)=A(2)+1:
=H!-(A(2)*.01):RETURN 32
56 REM--------
57 IF A(3)=0 THEN I!=184
58 PUTSPRITE 2,(A(3),I!),3,2
59 IF A(3)=255THENA(3)=0:PUTSPRITE2,(0,192),3,2:RETURN30ELSEA(3)=A(3)+1:I!=I!-
(3)*3E-03):RETURN 32
60 REM--------
61 PUTSPRITE 2,(A(4),120),4,2
62 IF A(4)=255THENA(4)=0:PUTSPRITE2,(0,192),4,2:RETURN30ELSEA(4)=A(4)+1:RETURN
2
63 REM--------
64 IF A(5)=0 THEN F!=30
65 PUTSPRITE 2,(A(5),F!),9,2
66 IF A(5)=255ORF!>192THENA(5)=0:PUTSPRITE2,(0,192),9,2:RETURN30ELSEA(5)=A(5)+
F!=F!+(A(5)*.01):RETURN 32
67 REM--------
68 IF B(5)=250 THEN A!=0
69 PUTSPRITE 2,(B(5),A!),11,4
70 IF B(5)=0THENB(5)=250:PUTSPRITE2,(0,192),11,4:RETURN30ELSEB(5)=B(5)-1:A!=A!+
5:RETURN 32
71 REM--------
72 IF B(4)=250 THEN B!=190
73 PUTSPRITE 2,(B(4),B!),14,4
74 IF B(4)=0THENB(4)=250:PUTSPRITE2,(0,192),14,4:RETURN30ELSEB(4)=B(4)-1:B!=B!-
5:RETURN 32
75 REM--------
76 IF B(3)=250 THEN C!=100
77 PUTSPRITE 2,(B(3),C!),12,4
78 IF B(3)=0 THENB(3)=250:PUTSPRITE2,(0,192),12,4:RETURN30ELSEB(3)=B(3)-1:C!=C!
.3:RETURN 32
79 REM--------
80 IF B(2)=250 THEN D!=100
81 PUTSPRITE 2,(B(2),D!),8,4
82 IF B(2)=0THENB(2)=250:PUTSPRITE2,(0,192),8,4:RETURN30ELSEB(2)=B(2)-1:D!=D!+.
+(B(2)>100):RETURN 32
83 REM--------
84 IF B(1)=250 THEN E!=50
85 PUT SPRITE2,(B(1),E!),5,4
86 IF B(1)=0 THEN B(1)=250:PUTSPRITE 11,(0,192),5,4:RETURN 30ELSEB(1)=B(1)-1:E!=
E!+.4:RETURN 32
87 REM--------
88 FOR TT=1 TO 5:A(TT)=0:NEXT
89 FOR TT=1 TO 5:B(TT)=250:NEXT
90 PUT SPRITE 2,(0,192),0,2:AH=AH+1
91 SOUND 9,0:SOUND 7,247:SOUND 6,26:SPRITE OFF
92 FOR Z!=15 TO 0 STEP -.08:PUT SPRITE 1,(X,Y-1),Z!,3:SOUND 8,Z!
93 NEXT
94 PUTSPRITE 1,(0,192),Z!,3:SOUND 7,239:SOUND 9,16:SOUND 6,18:RETURN30
95 DATA ,,,,,,,,,,,,,,1,,3,128,1,,,,,,,,,,,,
96 DATA ,,,,,,,,,,,1,,1,,1,,14,224,1,,1,,1,,,,,,,,,
97 DATA ,,,,,,,248,,32,2,112,3,232,,104,,48,,120,,,,,,,,,,,
98 DATA ,,,,10,192,17,72,,,48,165,35,50,7,152,155,192,64,,140,216,40,140,,,41,16
0,5,8,4,128
99 DATA ,,,,,,31,,4,,14,64,23,192,22,,12,,30,,,,,,,,,,,,
100 SOUND 7,255:SOUND 9,0
101 LOCATE20,30:PRINT"Du prickade";AH;
102 IF AH=1 THENPRINT"helikopter"ELSE PRINT"helikoptrar"
103 LOCATE 20,40:PRINT"under 120 sekunder"
104 A$=INPUT$(1):IFA$=" "THEN105ELSE104
105 FORTT=0TO4:PUTSPRITETT,(0,192),0,0:NEXT
106 LINE(18,18)-(200,50),1,BF:PSET(25,30),6:PSET(130,40),2:PSET(80,25),4
107 FORTT=1TO5:A(TT)=0:B(TT)=250:NEXT
108 AH=0:SP=1:X=128:Y=96:RETURN 28
```


Vi har testat diverse saker som har med Spectravideo att göra och skall i den här spalten skriva vad vi tyckte om dem. Vi har ett betygssystem enl. följande:

SPEL : <1-5>/<1-5>/<1-5> (1=Urkasst, 5=Fenomenalt) Det första visar hur originell och annorlunda ide'n är, det andra vad man har gjort av den (Grafik, ljud etc.) och det tredje hur roligt spelet är.

ANNAT : <1-5>/<1-5>/<1-5> (1=Inte ett dugg lyckat, 5=Jättebäst) Det första visar hur nyttig ide'n är, det andra hur användarvänligt det är och det tredje vad slutresultatet blev.

Observera att vid bedömningarna har ingen hänsyn tagits till priset !

SPEL:

Turboat.....3/5/4 (Action) Man åker med en båt först genom ett försvarssystem, sedan i en skärgård där man samlar radarstationer, sedan flyr man genom ett annat försvarssystem. Kul.

Sasa.....4/5/4 (Action) Man är en robot som skall klara sig genom olika skärmar med varierande faror. Kul.

Telebunny.....5/4/4 (Action) Man är en kanin som skall lotsa en sköldpadda ut ur en farofylld labyrint. Kul och annorlunda.

Old mac farmer4/4/3 (Action) Man är en kinesisk farmare som skall skörda sitt ris, men en massa monster jagar en. Medelbra pacmanvariant.

Tetra horror..4/5/3 (Action) Man skall först samla bränsle och sedan knata in i en gammal inkastad fylld med dödskallar och annat läbbigt. Roligt i början men enformigt efter ett tag.

Kung fu master4/3/3 (Action) Man skall slå ner folk med sparkar eller slag, sedan skall man leta efter en nyckel. Kul men enformigt.

Ninja.....5/5/4 (Action) Man är en ninja som skall ta sig igenom en försvarslinje och samla skatter. Man kan kasta pilar eller slåss med svärd beroende på vilken skärm man är på. Kul och omväxlande

Number game...3/3/4 (Strategi) Gillar man matte skall man köpa det här spelet. Det går ut på att ta nummer på ett sådant sätt att datorn inte kan ta så många. Är roligare än det låter.

Compatibility.4/4/3 (???) Detta program bygger på biorytmprincipen, och jämför en person med en annan och visar i diagram hu pass bra man passar ihop intelektuellt, emotionellt och fysiskt (Om man nu tror på biorytm).

Juno lander...3/3/3 (Action) Man skall landa sin rymdkapsel på JUNO utan att krascha. Enformigt.

Biorytm.....2/3/2 (???) Om man tror på biorytmprincipen kan det här programmet vara bra att räkna fram sina kurvor med. Annars är det ganska dött.

Car ace.....2/1/1 ("Action") Man skall köra bil mellan punkt A och B utan att krocka. En normal spectravideoanvändare kan göra ett bättre själv.

NYTTOPROGRAM:

Font editor...4/3/3 Man kan med det här programmet skapa ett eget teckenset som man kan använda. Tyvärr försvinner det vid SCREEN samt kassettrutinerna är långsamma så slutbetyget dras ner.

Star words....2/4/2 Man skall skjuta ner alla bokstäver som inte passar in i ett ord och hämta den rätta. Inte så värst användbart.

Financial calculator4/4/3 Ett program att ha när man räknar ut räntor och amorteringar. Det som skulle varit bäst att ha av det som finns inbyggt är uträkning av förlust p.g.a. värdeminskning. Tyvärr är tabellerna för det avpassade för det amerikanska skattesysemet och går alltså inte att använda hemma i kära gamla Svedala.

Spectra diary.4/4/2 Ett enkelt almanacksprogram som fungerar hyfsat. Men det är enklare att skriva upp möten etc. i en vanlig almanack.

Acutype.....4/3/3 Ett program för träning av skrivmaskinsskrivning. Tyvärr så har det några smärre missar.

SAKER:

Graphics tablet 4/4/3 Ett grafiskt ritbord med tillhörande program (I manualen står också hur man kan läsa av det i BASIC. Tyvärr kände det av fel ibland vilket gjorde att kanske hela bilden blev förstörd. Dessutom står det i manualen att det skall gå att få ut bilden både på en Seikosha GP-100 och en Epson-kompatibel (Ex. Admate DP-80) skrivare. Dock fick man inte den valmöjligheten där man skulle fått det, utan skrivaren började spruta tecken som antagligen var kontrolltecken för en Seikosha.

Basic referensmanual på Spectravideo 4/4/4 Denna bok tar upp alla (nästan) BASIC-kommandon och hur de fungerar på Spectravideo. En ny omarbetad version är i skrivande stund på väg.

Maskinspråksmanualen 5/3/4 Den här boken tar upp hur man anropar maskinspråk, hur man använder de dolda 32 K-na i en SV 328, var olika systemvariabler och ROMrutiner ligger i minnet etc. Det den inte lär ut är maskinspråksprogrammering, utan det får man lära sig i spectraview eller någon annan bok. Den är inte heller pedagogiskt upplagd, utan är indelad i olika sektioner beroende på vad texten behandlar.

Colecoadapter 5/5/5 Denna tillåter att man spelar colecos fina (men dyra) spelkassetter i sin Spectravideo. Fungerar helt klanderfritt. Pluggas in i spectravideons expanderport, ingen super-/mini expander behövs.

Det var väl allt för den här gången, men vi hoppas att kunna komma igen nästa gång med fler nya tester.

F I L H A N T E R I N G P Å S P E C T R A V I D E O

Att lagra data på band eller kassett är ett moment som inte speciellt utförligt tas upp i instruktionsboken. Vi skall dock här gå igenom filhantering på kassett och disk.

För att lagra data på kassett skall man först öppna en s.k. FIL, där man sedan skriver ner sina data. Det gör man genom att ge kommandot OPEN "cas:<Namn>" FOR OUTPUT AS # 1 . När datorn kommer till den instruktionen kommer den att skriva ut det välkända Press play and record on tape. Då skall man ha i en blank kassett i bandspelaren och trycka på record+play. Sedan snurrar bandet ett litet tag och stannar. Datorn fortsätter nu med nästa instruktion. Nu är det möjligt att skriva data i filen genom att använda PRINT # 1,<Variabler/siffror/strängar> . När man är färdig med att skriva sina data skall man ge instruktionen CLOSE som skriver ut data som eventuellt finns kvar i bufferten och sedan stänger filen. Följande program skriver talen 1-20 i en fil på kassett.

```
10 REM DEMO # 1
20 OPEN "cas:Number" FOR OUTPUT AS # 1
30 FOR NUMMER=1 TO 20
40 PRINT # 1, NUMMER
50 NEXT NUMMER
60 CLOSE
70 END
```

Nåväl, nu finns data på kassetten. Hur får man det därifrån tillbaks in i datorn igen ?
De med slutledningsförmåga har säkert redan listat ut att man först byter ut OUTPUT till INPUT i OPEN-satsen, sedan läser man data med INPUT # 1,<Variabel> (,<Variabel>,<Variabel> ...)
För att kolla om de data man läste var de sista i filen (filslut) använder man funktionen EOF (1) som ger 0 om filen fortsätter och -1 om den tar slut. Ett program för att öppna, läsa och skriv ut vad som finns i en fil följer nedan:

```
10 REM DEMO # 2
20 CLS:CLEAR 300
30 PRINT "SPOLA KASSETTEN TILL FILBÖRJAN OCH    TRYCK PÅ EN TANGENT."
40 A$=INPUT$(1)
50 OPEN "CAS:" FOR INPUT AS # 1
60 INPUT # 1, A$
70 PRINT A$
80 IF NOT EOF(1) GOTO 50
90 CLOSE
100 END
```

Att vi har satt strängar efter INPUT-satsen baeror på att det går att läsa in data som skrivits numeriskt i en sträng, men om man försöker läsa en sträng in i en nummervariabel får man felmeddelandet Type mismatch error. Pröva med att först köra program 1, spola tillbaks kassetten och sedan köra program 2.

För er som har diskett:

Byt ut cas: mot <drive #>: ex. OPEN "1:Test" FOR OUTPUT AS # 1.
Dessutom finns det två funktioner till, APPEND och om man helt utelämnar FOR. Den förra används om man har skrivit en fil färdigt och sedan vill lägga till något. Då gör man OPEN "<Drive #>:<Namn>" FOR APPEND AS # 1 . Sedan gör man en INPUT # 1, <Variabel> varefter det går att skriva data med PRINT # 1. Den andra är den man mest använder i DISK BASIC, därför att den medger att man kan gå in i mitten av filen och både skriva och läsa på samma gång. Nackdelen är att filen måste delas in i poster med en viss längd. Då skapas en buffert där man skriver i posten, och sedan skrivs posten i filen. Bufferten skapar man genom att använda FIELD 1,<antal> AS <Strängvariabel>(,<antal> AS <Strängvariabel>, ...) . <antal> är det antal tecken som den senare specificerade variabeln skall uppta i posten. I våra fortsatta exempel kommer vi att anta att följande FIELD har gjorts:
FIELD 1, 25 AS NAMN¤ , 25 AS ADRESS¤ , 25 AS POSTNR¤ , 15 AS TEL¤

Sedan (efter att man har gjort FIELDen) öppnar man filen (Ex. OPEN "1:Test" AS # 1). Nu kan man läsa och skriva i bufferten, som i sin tur kan skrivas på skiva eller bli hämtad från skiva. Läser från bufferten gör man som från en vanlig variabel. Ex.
PRINT LEFT¤(NAMN¤,INSTR(NAMN¤," "));" "¦TEL¤
Att skriva i bufferten är däremot litet svårare än i en vanlig variabel. Man använder kommandona LSET och RSET som fungerar enl. följande:

LSET <Buffervariabel> = <Stränguttryck> : <Stränguttryck vänsterställs i <Buffervariabel> som måste vara en variabel i buffern med deklarerad längd. Ev. överflädiga tecken i <stränguttryck> kapas från slutet. Om antalet tecken i <Stränguttryck> är mindre än <Buffervariabel> fylls resterande tecken ut med mellanslag i slutet av <Buffervariabel>.

RSET <Buffervariabel> = <Stränguttryck> : <Stränguttryck högerställs i <Buffervariabel> som måste vara en variabel i buffern med deklarerad längd. Ev. överflädiga tecken i <stränguttryck> kapas från början. Om antalet tecken i <Stränguttryck> är mindre än <Buffervariabel> fylls resterande tecken ut med mellanslag i början av <Buffervariabel>.

Sedan skrivs posten på skivan med PUT # 1,<Postnummer> . <Postnummer> skall vara mellan 1 och 32767. Poster hämtas in i bufferten med GET # 1,<Postnummer> . Om man försöker ladda en post som inte har blivit skriven fås felmeddelandet Read past EOF.

Exempel på ett KORT registrprogram:

```
10 CLS:CLEAR 3000
20 ON ERROR GOTO 420
30 CLOSE : OPEN "1:REGFIL.DAT" AS # 1 : FIELD 1,25 AS NA¤,25 AS AD¤,25
AS PN¤,15 AS TN¤
40 CLS:PRINT "VAD VILL DU GÖRA ?"
50 PRINT:PRINT "1.LÄGGA TILL POST"
60 PRINT "2.ÄNDRA POST"
70 PRINT "3.SÖKA POST"
80 PRINT "4.LISTA POST"
90 PRINT "5.SLUTA"
```



```
100 A¤=INPUT¤(1):IF VAL (A¤)<1 OR VAL (A¤)>5 THEN 100
110 REM ----------- VALET FÄRDIGT -----------
120 ON VAL (A¤) GOSUB 140,220,290,360,440
130 GOTO 40
140 REM ---- INMATNING ----
150 INPUT "NAMN........";I1¤
160 INPUT "ADRESS......";I2¤
170 INPUT "POSTADRESS..";I3¤
180 INPUT "TELEFON.....";I4¤
190 LSET NA¤=I1¤ : LSET AD¤=I2¤ : LSET PN¤=I3¤ : LSET  TN¤=I4¤  :  REM
STOPPA DATA I BUFFERTEN
200 PUT# 1, LOF(1)+1 : REM  lof(1)  GER  LÄNGDEN  PÅ  FILEN  I  POSTER
RÄKNAT.
210 RETURN
220 REM ---- ÄNDRING ----
230 INPUT "VILKET NUMMER SKALL JAG ÄNDRA PÅ ";NU
240 GET# 1, NU : CLS : PRINT "? "¦NA¤ : PRINT  "?  "¦AD¤  :  PRINT  "?
"¦PN¤ : PRINT "? "¦TN¤
250 LOCATE 0,0 : INPUT I1¤ : INPUT I2¤ : INPUT I3¤ : INPUT I4¤  :  REM
TA IN ÄNDRINGAR.
260 LSET NA¤=I1¤ : LSET AD¤=I2¤ : LSET PN¤=I3¤ : LSET  TN¤=I4¤  :  REM
STOPPA DATA I BUFFERTEN
270 PUT# 1, NU : REM SKRIV IN ÄNDRING PÅ SKIVA
280 RETURN
290 REM ---- SÖKNING ----
300 INPUT"NAMN ";SN¤
310 NU=1
320 GET # 1, NU
330   IF   INSTR(NA¤,SN¤)   THEN   PRINT   :   PRINT   USING"###     ß
ß"¦NU,NA¤ : PRINT "     "AD¤ : PRINT "     "PN¤ : PRINT "     "¦TN¤  :
PRINT : A¤=INPUT¤(1) : REM OM NAMNEN STÄMMER, SKRIV UT  OCH  VÄNTA  PÅ
TANGENT
340 IF NOT EOF (1) THEN NU=NU+1 : GOTO 320
350 RETURN
360 REM ---- LISTNING ----
370 NU=1
380 GET # 1, NU
390 PRINT : PRINT USING"###  ß                              ß"¦NU,NA¤ : PRINT
"     "AD¤ : PRINT "     "PN¤ : PRINT : PRINT :  REM  SKRIV  UT  VARJE
POST
400 IF NOT EOF (1) THEN NU=NU+1 : GOTO 380
410 RETURN
420 IF ERR=55 AND ERL=320 OR ERL=380 THEN CLOSE : RESUME 30
430 PRINT ERL¦ERR : ERROR(ERR)
440 CLOSE : END
```

För den som har skrivare borde det inte vara speciellt svårt att ändra raderna 330 och 390 till LPRINT i stället för PRINT. I övrigt borde programmet förklara sig själv. Error-trappingrutinen som är inbyggd skirver ut radnummer, felnummer och på nästa rad ger den rätt error fast i rad 430. Ex:

```
 240  55
Read past EOF in 430
```

Fortsätting kan då göras med GOTO 30

Överflödiga CLOSE gör ingen skada, och orsakar ingen felkod. Därför är det att rekommendera att man gör CLOSE innan man öppnr en fil, för att vara säker på att den int redan är öppen.

Lycka till med filhanteringen !

JR d	18 d
JR NC,d	30 d
JR Z,d	28 d
LD (DE),A	12
LD (HL),B	70
LD (HL),D	72
LD (HL),H	74
LD (HL),n	36 n
LD (IX+d),B	DD 70 d
LD (IX+d),D	DD 72 d
LD (IX+d),H	DD 74 d
LD (IX+d),n	DD 36 d n
LD (IY+d),B	FD 70 d
LD (IY+d),D	FD 72 d
LD (IY+d),H	FD 74 d
LD (IY+d),n	FD 36 d n
LD (mn),BC	ED 43 n m
LD (mn),HL	ED 63 n m
LD (mn),IX	DD 22 n m
LD (mn),SP	ED 73 n m
LD A,(DE)	1A
LD A,(IX+d)	DD 7E d
LD A,(mn)	3A n m
LD A,B	78
LD A,D	7A
LD A,H	7C
LD A,L	7D
LD A,n	3E n
LD B,(IX+d)	DD 46 d
LD B,A	47
LD B,C	41
LD B,E	43
LD B,L	45
LD BC,(mn)	ED 4b n m
LD C,(HL)	4E
LD C,(IY+d)	FD 4E d
LD C,B	48
LD C,D	4A
LD C,H	4C
LD C,n	OE n
LD D,(IX+d)	DD 56 d
LD D,A	57
LD D,C	51
LD D,E	53
LD D,L	55
LD DE,(mn)	ED 5B n m
LD E,(HL)	5E
LD E,(IY+d)	FD 5E d
LD E,B	58
LD E,D	5A
LD E,H	5C
LD E,n	1E n

JR C,d	38 d
JR NZ,d	20 d
LD (BC),A	02
LD (HL),A	77
LD (HL),C	71
LD (HL),E	73
LD (HL),L	75
LD (IX+d),A	DD 77 d
LD (IX+d),C	DD 71 d
LD (IX+d),E	DD 73 d
LD (IX+d),L	DD 75 d
LD (IY+d),A	FD 77 d
LD (IY+d),C	FD 71 d
LD (IY+d),E	FD 73 d
LD (IY+d),L	FD 75 d
LD (mn),A	32 n m
LD (mn),DE	ED 53 n m
LD (mn),HL	22 n m
LD (mn),IY	FD 22 n m
LD A,(BC)	0A
LD A,(HL)	7E
LD A,(IY+d)	FD 7E d
LD A,A	7F
LD A,C	79
LD A,E	7B
LD A,I	ED 57
LD A,R	ED 5F
LD B,(HL)	46
LD B,(IY+d)	FD 46 d
LD B,B	40
LD B,D	42
LD B,H	44
LD B,n	06
LD BC,mn	01 n m
LD C,(IX+d)	DD 4E d
LD C,A	4F
LD C,C	49
LD C,E	4B
LD C,L	4D
LD D,(HL)	56
LD D,(IY+d)	FD 56 d
LD D,B	50
LD D,D	52
LD D,H	54
LD D,n	16 n
LD DE,mn	11 n m
LD E,(IX+d)	DD 5E d
LD E,A	5F
LD E,C	59
LD E,E	5B
LD E,L	5D
LD H,(HL)	66

LD L,(HL)	6E
LD L,(IY+d)	FD 6E d
LD L,B	68
LD L,D	6A
LD L,H	6C
LD L,n	2E n
LD SP,(mn)	ED 7B n m
LD SP,HL	F9
LD SP,IY	FD F9
LDDR	ED B8
LDIR	ED B0
NOP	00
OR (IX+d)	DD B6
OR A	B7
OR C	B1
OR E	B3
OR L	B5
OTDR	ED BB
OUT (C),A	ED 79
OUT (C),C	ED 49
OUT (C),E	ED 59
OUT (C),L	ED 69
OUTD	ED AB
POP AF	F1
POP DE	D1
POP IX	DD E1
PUSH AF	F5
PUSH DE	D5
PUSH IX	DD E5
RES 0,(HL)	CB 86
RES 0,(IY+d)	FD CB d 86
RES 0,B	CB 80
RES 0,D	CB 82
RES 0,H	CB 84
RES 1,(HL)	CB 8E
RES 1,(IY+d)	FD CB d 8E
RES 1,B	CB 88
RES 1,D	CB 8A
RES 1,H	CB 8C
RES 2,(HL)	CB 96
RES 2,(IY+d)	FD CB d 96
RES 2,B	CB 90
RES 2,D	CB 92
RES 2,H	CB 94
RES 3,(HL)	CB 9E
RES 3,(IY+d)	FD CB d 9E
RES 3,B	CB 98
RES 3,D	CB 9A
RES 3,H	CB 9C
RES 4,(HL)	CB A6
RES 4,(IY+d)	FD CB d A6

LD L,(IX+d)	DD 6E d
LD L,A	6F
LD L,C	69
LD L,E	6B
LD L,L	6D
LD R,A	ED 4F
LD SP,mn	31 n m
LD SP,IX	DD F9
LDD	ED A8
LDI	ED A0
NEG	ED 44
OR (HL)	B6
OR (IY+d)	FD B6
OR B	B0
OR D	B2
OR H	B4
OR n	F6 n
OTIR	ED B3
OUT (C),B	ED 41
OUT (C),D	ED 51
OUT (C),H	ED 61
OUT (n),A	D3 n
OUTI	ED A3
POP BC	C1
POP HL	E1
POP IY	FD E1
PUSH BC	C5
PUSH HL	E5
PUSH IY	FD E5
RES 0,(IX+d)	DD CB d 86
RES 0,A	CB 87
RES 0,C	CB 81
RES 0,E	CB 83
RES 0,L	CB 85
RES 1,(IX+d)	DD CB d 8E
RES 1,A	CB 8F
RES 1,C	CB 89
RES 1,E	CB 8B
RES 1,L	CB 8D
RES 2,(IX+d)	DD CB d 96
RES 2,A	CB 97
RES 2,C	CB 91
RES 2,E	CB 93
RES 2,L	CB 95
RES 3,(IX+d)	DD CB d 9E
RES 3,A	CB 9F
RES 3,C	CB 99
RES 3,E	CB 9B
RES 3,L	CB 9D
RES 4,(IX+d)	DD CB d A6
RES 4,A	CB A7

JR d	18 d
JR NC,d	30 d
JR Z,d	28 d
LD (DE),A	12
LD (HL),B	70
LD (HL),D	72
LD (HL),H	74
LD (HL),n	36 n
LD (IX+d),B	DD 70 d
LD (IX+d),D	DD 72 d
LD (IX+d),H	DD 74 d
LD (IX+d),n	DD 36 d n
LD (IY+d),B	FD 70 d
LD (IY+d),D	FD 72 d
LD (IY+d),H	FD 74 d
LD (IY+d),n	FD 36 d n
LD (mn),BC	ED 43 n m
LD (mn),HL	ED 63 n m
LD (mn),IX	DD 22 n m
LD (mn),SP	ED 73 n m
LD A,(DE)	1A
LD A,(IX+d)	DD 7E d
LD A,(mn)	3A n m
LD A,B	78
LD A,D	7A
LD A,H	7C
LD A,L	7D
LD A,n	3E n
LD B,(IX+d)	DD 46 d
LD B,A	47
LD B,C	41
LD B,E	43
LD B,L	45
LD BC,(mn)	ED 4b n m
LD C,(HL)	4E
LD C,(IY+d)	FD 4E d
LD C,B	48
LD C,D	4A
LD C,H	4C
LD C,n	OE n
LD D,(IX+d)	DD 56 d
LD D,A	57
LD D,C	51
LD D,E	53
LD D,L	55
LD DE,(mn)	ED 5B n m
LD E,(HL)	5E
LD E,(IY+d)	FD 5E d
LD E,B	58
LD E,D	5A
LD E,H	5C
LD E,n	1E n
LD H,(IX+d)	DD 66 d
LD H,A	67
LD H,C	61
LD H,E	63
LD H,L	65
LD HL,(mn)	ED 6B n m
LD HL,mn	21 n m
LD IX,(mn)	DD 2A n m
LD IY,(mn)	FD 2A n m

JR C,d	38 d
JR (d	d
LD (BC),A	02
LD (HL),A	77
LD (HL),C	71
LD (HL),E	73
LD (HL),L	75
LD (IX+d),A	DD 77 d
LD (IX+d),C	DD 71 d
LD (IX+d),E	DD 73 d
LD (IX+d),L	DD 75 d
LD (IY+d),A	FD 77 d
LD (IY+d),C	FD 71 d
LD (IY+d),E	FD 73 d
LD (IY+d),L	FD 75 d
LD (mn),A	32 n m
LD (mn),DE	ED 53 n m
LD (mn),HL	22 n m
LD (mn),IY	FD 22 n m
LD A,(BC)	0A
LD A,(HL)	7E
LD A,(IY+d)	FD 7E d
LD A,A	7F
LD A,C	79
LD A,E	7B
LD A,I	ED 57
LD A,R	ED 5F
LD B,(HL)	46
LD B,(IY+d)	FD 46 d
LD B,B	40
LD B,D	42
LD B,H	44
LD B,n	06
LD BC,mn	01 n m
LD C,(IX+d)	DD 4E d
LD C,A	4F
LD C,C	49
LD C,E	4B
LD C,L	4D
LD D,(HL)	56
LD D,(IY+d)	FD 56 d
LD D,B	50
LD D,D	52
LD D,H	54
LD D,n	16 n
LD DE,mn	11 n m
LD E,(IX+d)	DD 5E d
LD E,A	5F
LD E,C	59
LD E,E	5B
LD E,L	5D
LD H,(HL)	66
LD H,(IY+d)	FD 66 d
LD H,B	60
LD H,D	62
LD H,H	64
LD H,N	26 n
LD HL,(mn)	2A n m
LD I,A	ED 47
LD IX,mn	DD 21 n m
LD IY,mn	FD 21 n m

LD L,( )	6E
LD L,(IY+d)	FD 6E d
LD L,B	68
LD L,D	6A
LD L,H	6C
LD L,n	2E n
LD SP,(mn)	ED 7B n m
LD SP,HL	F9
LD SP,IY	FD F9
LDDR	ED B8
LDIR	ED B0
NOP	00
OR (IX+d)	DD B6
OR A	B7
OR C	B1
OR E	B3
OR L	B5
OTDR	ED BB
OUT (C),A	ED 79
OUT (C),C	ED 49
OUT (C),E	ED 59
OUT (C),L	ED 69
OUTD	ED AB
POP AF	F1
POP DE	D1
POP IX	DD E1
PUSH AF	F5
PUSH DE	D5
PUSH IX	DD E5
RES 0,(HL)	CB 86
RES 0,(IY+d)	FD CB d 86
RES 0,B	CB 80
RES 0,D	CB 82
RES 0,H	CB 84
RES 1,(HL)	CB 8E
RES 1,(IY+d)	FD CB d 8E
RES 1,B	CB 88
RES 1,D	CB 8A
RES 1,H	CB 8C
RES 2,(HL)	CB 96
RES 2,(IY+d)	FD CB d 96
RES 2,B	CB 90
RES 2,D	CB 92
RES 2,H	CB 94
RES 3,(HL)	CB 9E
RES 3,(IY+d)	FD CB d 9E
RES 3,B	CB 98
RES 3,D	CB 9A
RES 3,H	CB 9C
RES 4,(HL)	CB A6
RES 4,(IY+d)	FD CB d A6
RES 4,B	CB A0
RES 4,D	CB A2
RES 4,H	CB A4
RES 5,(HL)	CB AE
RES 5,(IY+d)	FD CB d AE
RES 5,B	CB A8
RES 5,D	CB AA
RES 5,H	CB AC
RES 6,(HL)	CB B6
RES 6,(IY+d)	FD CB d B6
RES 6,B	CB B0
RES 6,D	CB B2
RES 6,H	CB B4

LD L,(IX+d)	DD 6E d
LD L,A	6F
LD L,C	69
LD L,E	6B
LD L,L	6D
LD R,A	ED 4F
LD SP,mn	31 n m
LD SP,IX	DD F9
LDD	ED A8
LDI	ED A0
NEG	ED 44
OR (HL)	B6
OR (IY+d)	FD B6
OR B	B0
OR D	B2
OR H	B4
OR n	F6 n
OTIR	ED B3
OUT (C),B	ED 41
OUT (C),D	ED 51
OUT (C),H	ED 61
OUT (n),A	D3 n
OUTI	ED A3
POP BC	C1
POP HL	E1
POP IY	FD E1
PUSH BC	C5
PUSH HL	E5
PUSH IY	FD E5
RES 0,(IX+d)	DD CB d 86
RES 0,A	CB 87
RES 0,C	CB 81
RES 0,E	CB 83
RES 0,L	CB 85
RES 1,(IX+d)	DD CB d 8E
RES 1,A	CB 8F
RES 1,C	CB 89
RES 1,E	CB 8B
RES 1,L	CB 8D
RES 2,(IX+d)	DD CB d 96
RES 2,A	CB 97
RES 2,C	CB 91
RES 2,E	CB 93
RES 2,L	CB 95
RES 3,(IX+d)	DD CB d 9E
RES 3,A	CB 9F
RES 3,C	CB 99
RES 3,E	CB 9B
RES 3,L	CB 9D
RES 4,(IX+d)	DD CB d A6
RES 4,A	CB A7
RES 4,C	CB A1
RES 4,E	CB A3
RES 4,L	CB A5
RES 5,(IX+d)	DD CB d AE
RES 5,A	CB AF
RES 5,C	CB A9
RES 5,E	CB AB
RES 5,L	CB AD
RES 6,(IX+d)	DD CB d B6
RES 6,A	CB B7
RES 6,C	CB B1
RES 6,E	CB B3
RES 6,L	CB B5

RES 7,(HL)	CB BE
RES 7,(IY+d)	FD CB d BE
RES 7,B	CB B8
RES 7,D	CB BA
RES 7,H	CB BC
RET	C9
RET M	F8
RET NZ	C0
RET PE	E8
RET Z	C8
RETN	ED 45
RL (IX+d)	DD CB d 16
RL A	CB 17
RL C	CB 11
RL E	CB 13
RL L	CB 15
RLC (HL)	CB 06
RLC (IY+d)	FD CB d 06
RLC B	CB 00
RLC D	CB 02
RLC H	CB 04
RLCA	07
RR (HL)	CB 1E
RR (IY+d)	FD CB d 1E
RR B	CB 18
RR D	CB 1A
RR H	CB 1C
RRA	1F
RRC (HL)	CB 0E
RRC (IY+d)	FD CB d 0E
RRC B	CB 08
RRC D	CB 0A
RRC H	CB 0C
RRCA	0F
RST 00	C7
RST 10	D7
RST 20	E7
RST 30	F7
SBC A,(HL)	9E
SBC A,(IY+d)	FD 9E d
SBC A,B	98
SBC A,D	9A
SBC A,H	9C
SBC A,n	DE n
SBC HL,DE	ED 52
SBC HL,SP	ED 72
SET 0,(HL)	CB C6
SET 0,(IY+d)	FD CB d C6
SET 0,B	CB C0
SET 0,D	CB C2
SET 0,H	CB C4
SET 1,(HL)	CB CE
SET 1,(IY+d)	FD CB d CE
SET ...	...

RES 7,(IX+d)	DD CB d BE
RES 7,A	CB BF
RES 7,C	CB B9
RES 7,E	CB BB
RES 7,L	CB BD
RET C	D8
RET NC	D0
RET P	F0
RET PO	E0
RETI	ED 4D
RL (HL)	CB 16
RL (IY+d)	FD CB d 16
RL B	CB 10
RL D	CB 12
RL H	CB 14
RLA	17
RLC (IX+d)	DD CB d 06
RLC A	CB 07
RLC C	CB 01
RLC E	CB 03
RLC L	CB 05
RLD	ED 6F
RR (IX+d)	DD CB d 1E
RR A	CB 1F
RR C	CB 19
RR E	CB 1B
RR L	CB 1D
RRC (IX+d)	DD CB d 0E
RRC A	CB 0F
RRC C	CB 09
RRC E	CB 0B
RRC L	CB 0D
RRD	ED 67
RST 08	CF
RST 18	DF
RST 28	EF
RST 38	FF
SBC A,(IX+d)	DD 9E d
SBC A,A	9F
SBC A,C	99
SBC A,E	9B
SBC A,L	9D
SBC HL,BC	ED 42
SBC HL,HL	ED 62
SCF	37
SET 0,(IX+d)	DD CB d C6
SET 0,A	CB C7
SET 0,C	CB C1
SET 0,E	CB C3
SET 0,L	CB C5
SET 1,(IX+d)	DD CB d CE
SET ...	...

SET 3,(HL)	CB DE
SET 3,(IY+d)	FD CB d DE
SET 3,B	CB D8
SET 3,D	CB DA
SET 3,H	CB DC
SET 4,(HL)	CB E6
SET 4,(IY+d)	FD CB d E6
SET 4,B	CB E0
SET 4,D	CB E2
SET 4,H	CB E4
SET 5,(HL)	CB EE
SET 5,(IY+d)	FD CB d EE
SET 5,B	CB E8
SET 5,D	CB EA
SET 5,H	CB EC
SET 6,(HL)	CB F6
SET 6,(IY+d)	FD CB d F6
SET 6,B	CB F0
SET 6,D	CB F2
SET 6,H	CB F4
SET 7,(HL)	CB FE
SET 7,(IY+d)	FD CB d FE
SET 7,B	CB F8
SET 7,D	CB FA
SET 7,H	CB FC
SLA (HL)	CB 26
SLA (IY+d)	FD CB d 26
SLA B	CB 20
SLA D	CB 22
SLA H	CB 24
SRA (HL)	CB 2E
SRA (IY+d)	FD CB d 2E
SRA B	CB 28
SRA D	CB 2A
SRA H	CB 2C
SRL (HL)	CB 2E
SRL (IY+d)	FD CB d 2E
SRL B	CB 28
SRL D	CB 2A
SRL H	CB 2C
SUB (HL)	96
SUB (IY+d)	FD 96 d
SUB B	90
SUB D	92
SUB H	94

SET 3,(IX+d)	DD CB d DE
SET 3,A	CB DF
SET 3,C	CB D9
SET 3,E	CB DB
SET 3,L	CB DD
SET 4,(IX+d)	DD CB d E6
SET 4,A	CB E7
SET 4,C	CB E1
SET 4,E	CB E3
SET 4,L	CB E5
SET 5,(IX+d)	DD CB d EE
SET 5,A	CB EF
SET 5,C	CB E9
SET 5,E	CB EB
SET 5,L	CB ED
SET 6,(IX+d)	DD CB d F6
SET 6,A	CB F7
SET 6,C	CB F1
SET 6,E	CB F3
SET 6,L	CB F5
SET 7,(IX+d)	DD CB d FE
SET 7,A	CB FF
SET 7,C	CB F9
SET 7,E	CB FB
SET 7,L	CB FD
SLA (IX+d)	DD CB d 26
SLA A	CB 27
SLA C	CB 21
SLA E	CB 23
SLA L	CB 25
SRA (IX+d)	DD CB d 2E
SRA A	CB 2F
SRA C	CB 29
SRA E	CB 2B
SRA L	CB 2D
SRL (IX+d)	DD CB d 2E
SRL A	CB 2F
SRL C	CB 29
SRL E	CB 2B
SRL L	CB 2D
SUB (IX+d)	DD 96 d
SUB A	97
SUB C	91
SUB E	93
SUB L	95

RES 7,(HL)	CB BE
RES 7,(IY+d)	FD CB d BE
RES 7,B	CB B8
RES 7,D	CB BA
RES 7,H	CB BC
RET	C9
RET M	F8
RET NZ	C0
RET PE	E8
RET Z	C8
RETN	ED 45
RL (IX+d)	DD CB d 16
RL A	CB 17
RL C	CB 11
RL E	CB 13
RL L	CB 15
RLC (HL)	CB 06
RLC (IY+d)	FD CB d 06
RLC B	CB 00
RLC D	CB 02
RLC H	CB 04
RLCA	07
RR (HL)	CB 1E
RR (IY+d)	FD CB d 1E
RR B	CB 18
RR D	CB 1A
RR H	CB 1C
RRA	1F
RRC (HL)	CB 0E
RRC (IY+d)	FD CB d 0E
RRC B	CB 08
RRC D	CB 0A
RRC H	CB 0C
RRCA	0F
RST 00	C7
RST 10	D7
RST 20	E7
RST 30	F7
SBC A,(HL)	9E
SBC A,(IY+d)	FD 9E d
SBC A,B	98
SBC A,D	9A
SBC A,H	9C
SBC A,n	DE n
SBC HL,DE	ED 52
SBC HL,SP	ED 72
SET 0,(HL)	CB C6
SET 0,(IY+d)	FD CB d C6
SET 0,B	CB C0
SET 0,D	CB C2
SET 0,H	CB C4
SET 1,(HL)	CB CE
SET 1,(IY+d)	FD CB d CE
SET 1,B	CB C8
SET 1,D	CB CA
SET 1,H	CB CC
SET 2,(HL)	CB D6
SET 2,(IY+d)	FD CB d D6
SET 2,B	CB D0
SET 2,D	CB D2
SET 2,H	CB D4

RES 7,(IX+d)	DD CB d BE
RES 7,A	CB BF
RES 7,C	CB B9
RES 7,E	CB BB
RES 7,L	CB BD
RET C	D8
RET NC	D0
RET P	F0
RET PO	E0
RETI	ED 4D
RL (HL)	CB 16
RL (IY+d)	FD CB d 16
RL B	CB 10
RL D	CB 12
RL H	CB 14
RLA	17
RLC (IX+d)	DD CB d 06
RLC A	CB 07
RLC C	CB 01
RLC E	CB 03
RLC L	CB 05
RLD	ED 6F
RR (IX+d)	DD CB d 1E
RR A	CB 1F
RR C	CB 19
RR E	CB 1B
RR L	CB 1D
RRC (IX+d)	DD CB d 0E
RRC A	CB 0F
RRC C	CB 09
RRC E	CB 0B
RRC L	CB 0D
RRD	ED 67
RST 08	CF
RST 18	DF
RST 28	EF
RST 38	FF
SBC A,(IX+d)	DD 9E d
SBC A,A	9F
SBC A,C	99
SBC A,E	9B
SBC A,L	9D
SBC HL,BC	ED 42
SBC HL,HL	ED 62
SCF	37
SET 0,(IX+d)	DD CB d C6
SET 0,A	CB C7
SET 0,C	CB C1
SET 0,E	CB C3
SET 0,L	CB C5
SET 1,(IX+d)	DD CB d CE
SET 1,A	CB CF
SET 1,C	CB C9
SET 1,E	CB CB
SET 1,L	CB CD
SET 2,(IX+d)	DD CB d D6
SET 2,A	CB D7
SET 2,C	CB D1
SET 2,E	CB D3
SET 2,L	CB D5

SET 3,(HL)	CB DE
SET 3,(IY+d)	FD CB d DE
SET 3,B	CB D8
SET 3,D	CB DA
SET 3,H	CB DC
SET 4,(HL)	CB E6
SET 4,(IY+d)	FD CB d E6
SET 4,B	CB E0
SET 4,D	CB E2
SET 4,H	CB E4
SET 5,(HL)	CB EE
SET 5,(IY+d)	FD CB d EE
SET 5,B	CB E8
SET 5,D	CB EA
SET 5,H	CB EC
SET 6,(HL)	CB F6
SET 6,(IY+d)	FD CB d F6
SET 6,B	CB F0
SET 6,D	CB F2
SET 6,H	CB F4
SET 7,(HL)	CB FE
SET 7,(IY+d)	FD CB d FE
SET 7,B	CB F8
SET 7,D	CB FA
SET 7,H	CB FC
SLA (HL)	CB 26
SLA (IY+d)	FD CB d 26
SLA B	CB 20
SLA D	CB 22
SLA H	CB 24
SRA (HL)	CB 2E
SRA (IY+d)	FD CB d 2E
SRA B	CB 28
SRA D	CB 2A
SRA H	CB 2C
SRL (HL)	CB 2E
SRL (IY+d)	FD CB d 2E
SRL B	CB 28
SRL D	CB 2A
SRL H	CB 2C
SUB (HL)	96
SUB (IY+d)	FD 96 d
SUB B	90
SUB D	92
SUB H	94
SUB n	D6 n
XOR (IX+d)	DD AE d
XOR A	AF
XOR C	A9
XOR E	AB
XOR L	AD

SET 3,(IX+d)	DD CB d DE
SET 3,A	CB DF
SET 3,C	CB D9
SET 3,E	CB DB
SET 3,L	CB DD
SET 4,(IX+d)	DD CB d E6
SET 4,A	CB E7
SET 4,C	CB E1
SET 4,E	CB E3
SET 4,L	CB E5
SET 5,(IX+d)	DD CB d EE
SET 5,A	CB EF
SET 5,C	CB E9
SET 5,E	CB EB
SET 5,L	CB ED
SET 6,(IX+d)	DD CB d F6
SET 6,A	CB F7
SET 6,C	CB F1
SET 6,E	CB F3
SET 6,L	CB F5
SET 7,(IX+d)	DD CB d FE
SET 7,A	CB FF
SET 7,C	CB F9
SET 7,E	CB FB
SET 7,L	CB FD
SLA (IX+d)	DD CB d 26
SLA A	CB 27
SLA C	CB 21
SLA E	CB 23
SLA L	CB 25
SRA (IX+d)	DD CB d 2E
SRA A	CB 2F
SRA C	CB 29
SRA E	CB 2B
SRA L	CB 2D
SRL (IX+d)	DD CB d 2E
SRL A	CB 2F
SRL C	CB 29
SRL E	CB 2B
SRL L	CB 2D
SUB (IX+d)	DD 96 d
SUB A	97
SUB C	91
SUB E	93
SUB L	95
XOR (HL)	AE
XOR (IY+d)	FD AE d
XOR B	A8
XOR D	AA
XOR H	AC
XOR n	EE n