NAS Z80 BUS NYT

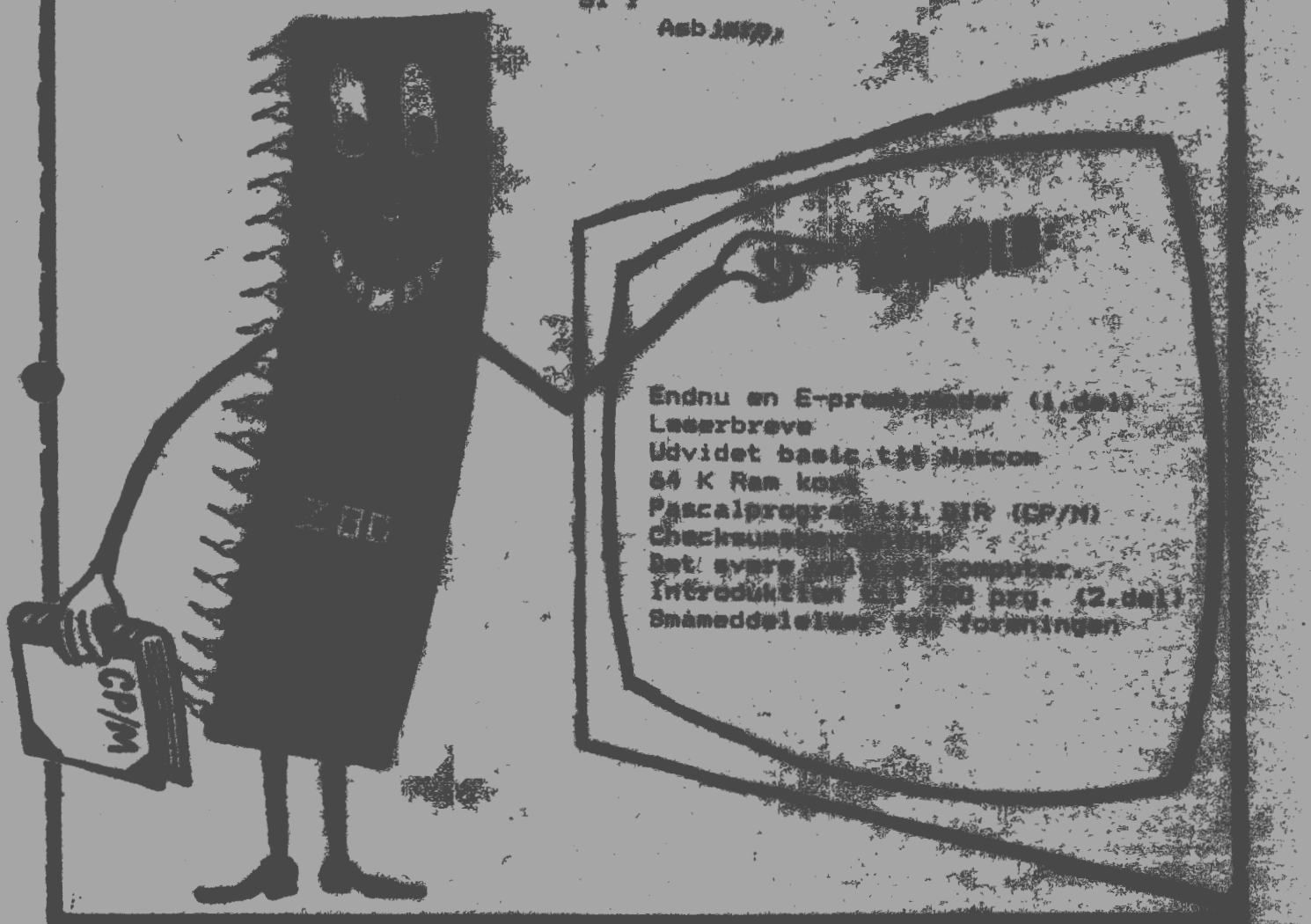
UDGIVET AF NASCOM BRUGERGRUPPE 1. ÅRGANG NR. 3

MARTS 1983

Gode tider ser vi i møde, når talen er om dette blad. Hvis I tæller sider, har det aldrig været større! Vi har haft og haft aftaler med forskellige forfattere om artikler til de næste numre - her vil Jesper fortsætte med introduktion til Z80, en ny E-prom brænder er på trapperne, et eventuelt om NASSYS 1/2, forskellige anvendelser af Plutokort, lydkort og urkort er i vente. Husk ikke at tale om skriverierne i forbindelse med GENERALFORSAMLINGEN den 8. maj. Men der er jo altid plads til endnu et indlæg/læserbrev - så tænk på det i påskeferien og så god påskeferien

Bi'r

Asbjørn.



Endnu en E-prombrænder (1.del)
Læserbreve
Udvidet basic til Nascom
64 K Ram kort
Pascalprogram til BIR (CP/M)
Checksumprogram
Det gyser [illegible] computer.
Introduktion til Z80 prg. (2.del)
Småmeddelelser fra foreningen

ALMINDELIGE OPLYSNINGER OM FORENINGEN

HENVENDELSE TIL FORENINGEN TIL FORRETNINGSFØREREN:

I. SKAVIN
BROHOLMS ALLE 3
2920 CHARLOTTENLUND
Telefon 01 - 64 03 14

Hertil skal rettes henvendelse om indmeldelse, adresseforandring, salg af foreningens materialer (bånd, blade og programmer). Øvrige henvendelser af generel art til formanden, herunder stof og annoncer til foreningens blad.

Indmeldelsesgebyr:	25.00 kr.
Kontingent 1.1.83 - 1.7.83	50.00 kr.

Annoncering for medlemmer er gratis i Z80 NYT. For andre 250 kr. pr. A4-side.

Bestyrelsesmedlemmer:

Formand: (Ans. redaktør)	Asbjørn Lind Sidevolden 23 2730 Herlev
Næstformand:	Jesper Skavin Broholms Alle 3 2920 Charlottenlund
Kasserer:	Erik Hansen Lyngby Kirkestræde 6.1 2800 Lyngby
Sekretær:	Carsten Senholt Blommevangen 6 2760 Måløv
Teknisk red.:	Ole Hasselbalch Vibeskrænten 6 2750 Ballerup

Frank Damgaard
Kastebjergvej 26A
2750 Ballerup

Knud Ytteborg
Dyssegårdsvej 71B
2860 Søborg

NASCOM BRUGERGRUPPE, SIDEVOLDEN 23, 2730 HERLEV GIRO 6742602

tryk: PC Lyngby

Udvidet basic til Nascom.

Hvad med 32 exstra kommandoer til din basic? Det får du ved at anskaffe dig denne ekstra programstump fra England. Den fylder fra starten 8 kilobyte, men efter relocering kommer den ned på 4 kilo. Du får følgende kommandoer:

AUTO	BREAK	CALL	CHECK	COPY	DEC
DELAY	DELETE	EDIT	FIND	GET	HEX
INKEY	INLIN	IF..THEN..ELSE		REPEAT..UNTIL	
PUT	PRINT@	REDUCE	SET	VDU	RENUMBER
TRACE	SPEED	WHILE..WEND		WRAP	XLIST XREF

Her er så en kortfattet, og muligvis ikke helt korrekt oversættelse, men jeg har da forsøgt at få de fleste af kommandoerne til at virke. Der er dog et par stykker, der råder lidt tvivl om.

AUTO. Genererer linienumre efter ENTER 10,10

CHECK. Giver besked om ureffererede linienumre

DELETE. Fjerner linier i programmet 100,110

EDIT. Giver op til 700 karakterer på een linie ikke mere besvær med XO mode.

LINE. Her bruges grafik 'pixel', så der kan tegnes en linie mellem to punkter X1,X2 og Y1,Y2.Argumentet 0 resetter.1 setter, og 2 inverterer.

REDUCE. Fjerner unødvendig tekst, hvis rampladsen er lille. Tre ordrer: 1 fjerner space undtagen i REM DATA og mellem anførselstegn. 2. fjerner REM inde på linien,(med REM i starten kun teksten). 3.Sammentraekning af 1 og 2.

RENUMBER. Renummererer f.eks RENUMBER 100,10

FIND. Søger efter et ord.F.eks FIND"?"

TRACE. Denne kommando skal have argumentet 1 eller 0. Argumentet 1 kan erstattes med et tal (X), hvor 'X' er en forsinkelse på X milliSEK.

XLIST. Lister en linie og giver reference til den.

XREF. Som XLIST, men den specificeret linie listes ikke.

DEC. Konverterer til HEX.

HEX. Konverterer til DEC.

CALL. Her kaldes en subrutine.Første argument er subrutinens adresse i decimal. Ved at bruge følgende argumenter, kan der udveksles værdier til subrutinen.

GET. Denne returnerer ASCII værdien af næste tast, eller hvis ingen tast nedtrykkes, returneres med 0. Ikke noget med at kludre i maskinkode: doke og deek.

INKEY. Som GET, men tast scannes, indtil næste tast nedtrykkes og ASCII værdien er returneret.

INLIN. Er jeg ikke helt på det rene med endnu.

TEST. Tester om en bestemt tast er nedtrykket. Der returneres hvis NOT, og 1 hvis ja. Koden der vises kan aflæses i den manual, der følger til programmet.

PLOT. Setter og Resetter eller inverterer punktet X,Y

PRINT @. PRINT AT. Letter omskrivning af TRS80 programmer til Nascom.

PUT. Argumenter, som er numre, udskrives med deres ASCIIværdier, og STRINGS som meddelelser.

WRAP. Sammen med EDIT kan lange ord deles.

VDU. Samler SCREEN og PRINT under eet.

IF..THEN..ELSE. Taler for sig selv.

REPEAT..UNTIL. Løkke indtil udtryk er sandt.

WHILE..WEND. Taler vel også for sig selv ?

BREAK.+- Kan låse ENTER.

COPY. Som i NASSYS, men i decimal.

DELAY. Argumentet 'n' giver delay i nmilliSek.

SET. Viser om udvidet basic er inde i programmet.

Programmet kan fås ved at skrive til: LEVEL 9 Computing,229 Hughenden Road,High Wycombe,Bucks England.

Hvis man kan lide at arbejde i basic, er det en god ekstra investering, og det er ikke vidre dyrt.

Jeg vil i nogle senere numre vise et par eksempler.

Ole Hassebalch

Bemærkninger til "INTRODUKTION TIL Z80 PROGRAMMERING"

Her følger så første halvdel af kapitel 2 sammen med nye opgaver og løsninger til opgaverne fra kapitel 1. Da de nye opgaver dækker hele kapitel 2, er det ikke sikkert du kan løse dem alle på baggrund af første halvdel. Opgaverne 7,8,9 og 10 er relevante for første halvdel og resten for anden halvdel, som kommer næste gang.

INTRODUKTION TIL Z80 PROGRAMMERING af Jesper Skavin.

KAPITEL 2 DE FØRSTE SKRIDT I INSTRUKTIONSSÆTTET.

INDLEDNING.

Høj niveau programmering.

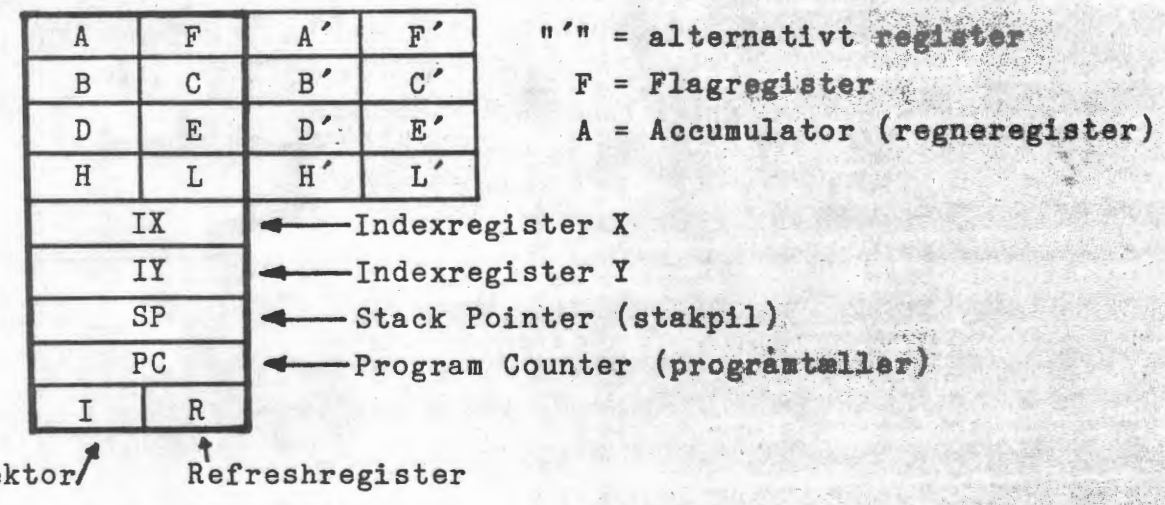
Når man programmerer i et høj niveau sprog som f.eks BASIC eller PASCAL, har man nok at gøre med at få programmet til at opføre sig, som man havde tænkt sig det skulle. Hvad man ikke bekymrer sig om er, hvor i memory de enkelte variable ligger og hvor f.eks. en addition finder sted. Tag f.eks. dette lille program:

```
X = 3
Y = 2
Z = X + Y
PRINT Z
```

Hvor er X ? Hvor er Y ? Hvor foregår additionen ? Hvor gemmes resultatet Z ? Alle disse "hvor" spørgsmål er helt nye for en, når man er ny i assemblerprogrammering. Når man går fra høj- til lavniveauprogrammering, åbner der sig en hel ny begrebsverden og man skal lære at tænke og dermed programmere på en helt ny måde.

Registre i Z80.

"I registrene" er svaret på de fleste "hvor" spørgsmål af ovennævnte type. Hvad er så disse registre for noget ? Et register i Z80 er rent fysisk det samme som en lagercelle i memory, nemlig 8 bit (en byte), som hører sammen, og får derfor et navn. Registrene ligger inde i selve CPU'en hvilket gør, at CPU'en utroligt hurtigt kan hente eller ændre indholdet af et register. Nedenfor er vist samtlige registre i Z80:



Alle registrene i de små kasser er 8 bit brede og de andre er på 16 bit. A,B,C,D,E,H og L registrene kaldes de generelle registre og IX,IY,SP,PC, I og R registrene kaldes de specielle registre. Flagregistret er lidt

specielt derved at kun nogle af bittene bliver brugt. Det er i F-registret de forskellige flag (CY,S,Z,P/V) sidder.

Lad os for en stund glemme de specielle og de alternative registre for at se lidt på de generelle, altså A,B,C,D,E,H og L. De enkelte bit i registrene nummereres fra højre til venstre med nul som det længst til højre og syv som det længst til venstre:

```
Bit nummer:  76543210
             10011010  Register B f.eks.
```

A registret benævnes også akkumulatoren (eng. accumulator) og det er der alle additioner, subtraktioner, AND, OR og XOR operationer finder sted. Akkumulatoren er også altid den ene operand ved ovennævnte operationer, når der er tale om 8 bit. D.v.s. at når man skal lægge to tal sammen, skal det ene altid stå i A, og resultatet bliver placeret i A af CPU'en.

Registrene B og C, D og E, H og L kan slås sammen til tre registerpar (kaldet BC,DE og HL), som så bliver på 16 bit pr. par. Det afhænger af den enkelte instruktion om B f.eks. skal opfattes som et selvstændigt 8-bit register,eller om det er den ene halvdel af BC-registerparret. Foruden at være i stand til at indeholde et stort tal, bruges BC, DE og HL også til at udpege (adressere) cellerne i memory.

De øvrige registre vil blive omtalt, når vi når frem til de instruktioner, der bruger dem.

Memory.

Det er i memory alle variable befinder sig og det er i memory maskinkodeprogrammet med dets data hører hjemme. Som nævnt i kapitel 1 er memory indelt i individuelle bytes, som hver tildeles en adresse. Alle adresser er to bytes lange (16 bit). Hvor meget memory kan man egentlig adressere med to bytes ? Betragt nedenstående tabel:

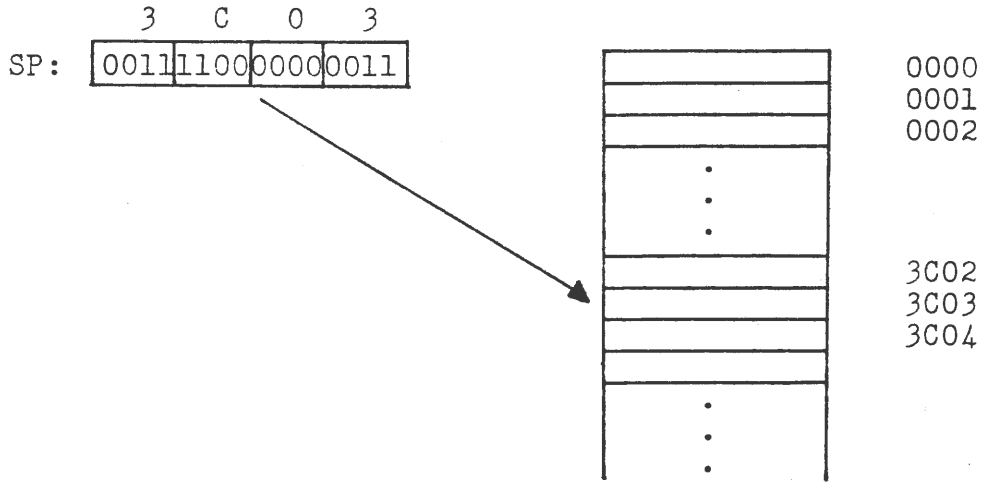
Antal bit:	Memorystørrelse:	Højeste adresse i hex:
8	256 bytes (0-255)	FF
9	512 bytes	1FF
10	1024 bytes = 1K	3FF
11	2048 bytes = 2K	7FF
12	4096 bytes = 4K	FFF
13	8192 bytes = 8K	1FFF
14	16384 bytes = 16K	3FFF
15	32768 bytes = 32K	7FFF
16	65536 bytes = 64K	FFFF

En byte tillader således kun adressering af 1/4 K af memory. Da det i de fleste tilfælde er helt utilstrækkeligt, er det naturligt at bruge to bytes til adresser, hvorved man kan adressere 64K memory.

Vi har allerede været inde på, at BC, DE og HL registerparene kan bruges til at indeholde memoryadresser. Der er imidlertid endnu fire 16 bit memoryadresse-registre i Z80. To af disse, nemlig SP og PC vil vi nu se lidt nærmere på.

Stakpilen og Programtælleren (SP & PC).

Stakpilen (SP) et et 16 bit register, hvis eneste funktion er, at pege på et sted i memory:



I Z80 computeren er stakken blot en del af den sædvanlige memory. Programmøren kan ved at lægge et 16 bit tal i SP-registret, sætte stakken til et vilkårligt område efter eget valg. Men hvad er egentlig en stak ?

En stak er, i bred forstand, en samling af ting, hvor al tilgang og afgang fra bunken kun kan ske fra toppen af. En stabel tallerkner er et godt eksempel på en stak.

Det karakteristiske ved stakken er, at man kun har adgang til den via staktoppen.

Stakken i Z80 sammenhæng er altså en del af lageret, hvor man kan gemme tal (oftest registre) midlertidigt. Den bruges mest i forbindelse med kald af underprogrammer, men den kan også bruges til at gemme mellemresultater i udregninger, eller værdien af en kortlivet variabel. Brugen af stakken til disse formål sparer os for at bruge unikke memoryceller til sådanne flygtige data.

Programtælleren (eng. Program Counter) betegnes med PC og den peger

også hele tiden på et sted i memory, nemlig der hvor CPU'en skal læse den næste instruktion. PC bruges da også mere af CPU'en internt end af programmøren. PC kan således ikke bruges til at gemme noget i.

Notation.

Lad os tænke os vi har en variabel i lageret og vi kalder den VAR. Da VAR befinder sig et eller andet sted i memory, har den en adresse. Den har selvfølgelig også en værdi. Så når vi bruger navnet VAR, hvad mener vi så egentlig, adressen eller værdien ? For BASIC-programmøren er der ingen tvivl. Han kender ikke noget til sine varibles adresser, så når han bruger VAR, mener han værdien.

Maskinkodeprogrammøren er nødt til at tage sig af såvel værdi som adresse. Det er de enkelte Z80 instruktioner som afgører om det er det ene eller det andet. Nogle instruktioner tager sig af værdierne, mens andre vedrører adresserne. Men for at kunne skelne mellem adresse og værdi, er det nødvendigt at indføre en ny notation. Hvis vi lader "n" betegne en 8 bit værdi og "nn" en 16 bit værdi, vil vi bruge:

nn	til at betegne en adresse
og (nn)	til at betegne indholdet (værdien) af adresse nn

F. eks. hvis nn = 3B6A og der i adresse 3B6A står 7E, så er (3B6A)=7E. Bemærk at nn altid er 16 bit, mens (nn) altid er 8 bit.

Mnemonics.

"Mnemonics" er en engelsk betegnelse for noget der understøtter hukommelsen (altså den menneskelige). I Z80 sammenhæng er mnemonics den forkortelse, man bruger for de enkelte instruktioner. For CPU'en er en instruktion blot et bitmønster, kaldet en OP-code (engelsk for OPeration-code). De binære værdier for OP-coden skrives oftest på hexadecimal form. For de fleste mennesker er en samling cifre, binære eller hexadecimale, svære at relatere noget til. Alle Z80 instruktionerne udtrykkes derfor i mnemonics, som er korte ordlignende forkortelser, som omsættes til binære tal af et program, kaldet en assembler.

Efterhånden som de enkelte instruktioner bliver præsenteret, vil jeg bruge de af Zilog anvendte mnemonics, og i enkelte tilfælde give OP-coden dertil. En samlet oversigt over alle instruktionerne kommer på et senere tidspunkt.

8-BIT FLYTTEINSTRUKTIONER.

Register ↔ Register.

Indholdet af ethvert 8-bit register kan kopieres over i ethvert af de andre 8-bit registre. Instruktionen, der kopierer A til D, ser sådan ud:

LD D,A

"LD" er en forkortelse (mnemonic) for det engelske ord LOAD, som i denforbindelse kan oversættes til "flyt". "D" er det register flytningen sker TIL og "A" er det register der flyttes FRA. Nu er der i virkeligheden tale om en kopiering, for A-registeret er urørt efter instruktionen er udført. Denne klasse af instruktioner kan skrives under et med

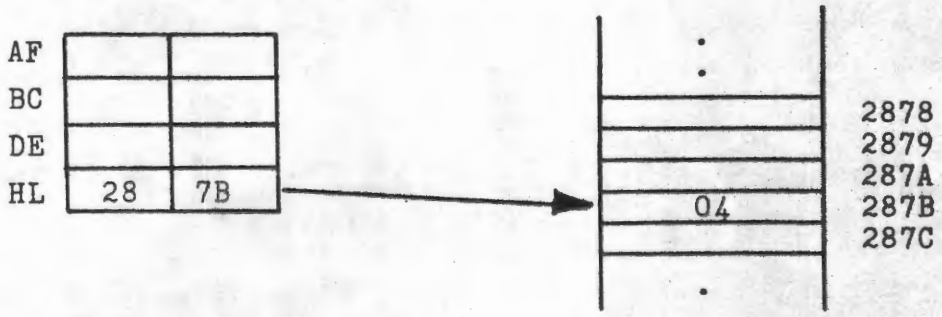
LD r,r´ hvor r og r´er et af A,B,C,D,E,H,L

OP-coden for LD D,A er 57H og alle de andre kombinationer har også sin egen OP-code. Men har man en assembler, behøver man ikke tænke på OP-coder, da assembleren kender samtlige OP-coder og indsætter dem i stedet for vores mnemonics.

Register ↔ Memory.

8-bit størrelser kan også flyttes mellem registrene og memory, men det er specielt flytning mellem A og memory som er mest udviklet på Z80. Flytning mellem de andre registre og memory er temmeligt begrænset. Lad os her kigge på den sidstnævnte gruppe.

Som nævnt tidligere kan registerparene BC, DE og HL bruges som pegere (pointere) i memory. HL er langt det mest brugte par til dette formål. Lad os antage at vi har følgende situation:



HL peger på adresse 287B i memory, som indeholder værdien 4. Denne værdi kan flyttes (læs: kopieres) over i ethvert af 8-bit registrene med instruktionen:

LD r,(HL) r = A,B,C,D,E,H,L

Denne instruktion læses som: Kopier indholdet af memorycelle,hvis adresse står i HL,over i registeret angivet ved r. F.eks, hvis r = C, så vi får LD C,(HL), vil C-registret indeholde tallet 4, ved ovenstående situation, og intet andet vil være ændret.

Den modsatte flytning er selvfølgelig også mulig med instruktionen:

LD (HL),r r = A,B,C,D,E,H,L

Bemærk at det kun er HL, der kan bruges som pointer, når man skal flytte til et vilkårligt register. Dog kan IX og IY også benyttes, men herom senere. Indskrænker vi os til kun at flytte til/fra A-reg. er der flere muligheder.

Register A ↔ Memory.

De nye muligheder består i,at vi nu kan bruge BC og DE som pointere. Dette giver os følgende instruktioner:

```
LD   A,(BC)
LD   A,(DE)
LD   (BC),A
LD   (DE),A
```

Hvad vi mangler nu er at kunne flytte værdien af en varibel, når den er givet ved sit navn:

A ← (VAR)
(VAR) ← A

Begge disse flytninger kræver mere end blot en OP-code. Foruden at fortælle CPU'en, at vi ønsker at hente værdien af en variabel, er vi også nødt til at sige hvilken, vi ønsker. Før, da vi flyttede fra en celle i memory, svarede vi på spørgsmålet "Hvilken ?" med "Den som udpeges af HL (eller BC eller DE) registerparene". Denne gang er vi nødt til selv at oplyse om adressen i selve instruktionen:

```
LD   A,(VAR)
```

Vi giver assembleren navnet på variablen. Assembleren oversætter dette navn til en 2-bytes adresse og sætter den ind i stedet for "VAR". (Den har en såkaldt symboltabel netop til dette formål).

Lad os sige at variablen VAR er gemt i adresse 23F3. Denne adresse vil da blive en del af instruktionen, så da 1 byte kræves til OP-coden og 2 til adressen, bliver hele instruktionen på 3 bytes. Her er mulighederne:

```
LD  A,(nn)
LD  (nn),A
```

nn står for variabelnavnet i mnemonics og for adressen i instruktionen.

Hvis nn=23F3 vil assembleren oversætte LD A,(23F3) til: 3A F3 23 og LD (23F3),A til: 32 F3 23 . Bemærk den omvendte rækkefølge af de to bytes i adressen. Den laveste byte står først. Dette er et gennemgående træk hos Z80.

OPGAVER OG ØVELSER til KAPITEL 2.

7. Skriv en sekvens af instruktioner, som ombytter indholdet af D og E registrene.

8. Hvad vil B-registret indeholde efter udførelse af nedenstående instruktion ? (de -12 er decimalt): LD B,-12

9. Antag at HL og de nedenfor givne memoryceller indeholder de viste værdier:

HL	20	3D

.	
.	
00	2039
42	203A
00	203B
F3	203C
39	203D
27	203E

Hvad vil A-registret indeholde efter disse to instruktioner er udført:

```
LD  L,(HL)
LD  A,(HL)
```

10. Skriv i hexadecimal den instruktion der flytter indholdet af variablen MINVAR (adresse = 34F3) ind i A-registret. (Husk det omvendte format).

11. Forklar effekten af instruktionen: LD E,(IX+12)

12. Forklar forskellen mellem disse to instruktioner:

```
LD  HL,(SPOT)        LD  HL,SPOT
```

13. Givet dette indhold af AF og BC: AF: 0402H BC: 4020H
Hvad vil de samme registre indeholde efter følgende instruktioner er udført:

```
PUSH  AF
PUSH  BC
POP   AF
POP   BC
```

14. Kan man forvente at få det samme tal tilbage efter disse instruktioner er udført:

```
PUSH  HL
LD    SP,HL
POP   HL
```

15. Givet memorycellerne:

.	
.	
02	6F32
03	6F33
04	6F34
05	6F35
.	

a) Hvilken instruktion sætter stakken til adresse 6F32 ?

b) Hvad vil BC og SP indeholde efter denne instruktion er udført:

```
POP  BC
```

16. Hvad indeholder C efter udførelsen af nedenstående instruktioner og det viste memory:

```
LD A,3
LD (SPOT),A
LD IX,(SPOT)
LD C,(IX-5)
```

SPOT:

21	23FE
22	23FF
23	2400
24	2401
25	2402
26	2403

LØSNINGER TIL OPGAVERNE FRA KAPITEL 1. OPG. NR. 1 TIL 6.

Da det vil være for omfattende at beskrive alle udregningerne, gives kun de enkelte facits.

OPGAVE 1

a) $101_{bin} = 5_{dec}$ b) $1101_{bin} = 13_{dec}$ c) $11101_{bin} = 29_{dec}$
d) $101011_b = 43_d$ e) $10000000_b = 128_d$ f) $11001010_b = 202_d$
g) $10001110_b = 142_d$ h) $11111001_b = 249_d$ i) $00010010_b = 18_d$
j) $01110011_b = 115_d$ k) $111000100_b = 452_d$ l) $1010101011_b = 683_d$

OPGAVE 2

a) $6_d=6_h=0110_b$ b) $14_d=E_h=1110_b$ c) $127_d=7F_h=01111111_b$
d) $280_d=118_h=000100011000_b$ e) $542_d=21E_h=001000011110_b$
f) $1077_d=435_h=010000110101_b$ g) $4095_d=FFF_h=111111111111_b$
h) $8702_d=21FE_h=0010000111111110_b$ i) $15430_d=3C46_h=0011110001000110_b$
j) $43751_d=AAE7_h=1010101011100111_b$ k) $65552_d=10010_h=00010000000000010000_b$
l) $70980_d=11544_h=00010001010101000100_b$

OPGAVE 3

a) 00000111=7, -7=11111001 b) 00010001=17, -17=11101111
c) 00010111=23, -23=11101001 d) 00110000=48, -48=11010000
f) 01111111=127, -127=10000001

OPGAVE 4

a) 00001001=9 b) 00011001=25 c) 11111011=-5 d) 0010011=39
e) 11110010=-14 f) 11010000=-48

OPGAVE 5

a)
```
00001011   11      +00001111  +15
00011010   26      CY=0, V=0
```

b)
```
00010001   17      +11101011  +(-21)
11111100   -4      CY=0, V=0
```

c)
```
00101110   46      -00001100  -12
00100010   34      CY=0, V=0
```

d)
```
01101000  104      +00110111  + 55
10011111  159      CY=0, V=1
```

e)
```
10111101  - 67     -01101011  -107
01010010  -174     CY=0, V=1
```

f)
```
10111101  - 67      +01101011  +107
1 00101000   40     CY=1, V=0
```

OPGAVE 6

a) AND: 00000000 CY=0 Z=1 S=0 P/V=1 (paritet)
OR: 11010101 CY=0 Z=0 S=1 P/V=0
XOR: 11010101 CY=0 Z=0 S=1 P/V=0

b) AND: 10110100 CY=0 Z=0 S=1 P/V=1
OR: 11111111 CY=0 Z=0 S=1 P/V=1
XOR: 01001011 CY=0 Z=0 S=0 P/V=1

c) AND: 00010110 CY=0 Z=0 S=0 P/V=0
OR: 11111111 CY=0 Z=0 S=1 P/V=1
XOR: 11101001 CY=0 Z=0 S=1 P/V=0

d) AND: 10100010 CY=0 Z=0 S=1 P/V=0
OR: 11110111 CY=0 Z=0 S=1 P/V=0
XOR: 01010101 CY=0 Z=0 S=0 P/V=1

e) AND: 00010000 CY=0 Z=0 S=0 P/V=0
OR: 10011101 CY=0 Z=0 S=1 P/V=0
XOR: 10001101 CY=0 Z=0 S=1 P/V=1

f) AND: 00000000 CY=0 Z=1 S=0 P/V=1
OR: 11110001 CY=0 Z=0 S=1 P/V=0
XOR: 11110001 CY=0 Z=0 S=1 P/V=0

g) AND: 00000000 CY=0 Z=1 S=0 P/V=1
OR: 11111111 CY=0 Z=0 S=1 P/V=1
XOR: 11111111 CY=0 Z=0 S=1 P/V=1

h) AND: 11001111 CY=0 Z=0 S=1 P/V=1
OR: 11001111 CY=0 Z=0 S=1 P/V=1
XOR: 00000000 CY=0 Z=1 S=0 P/V=1

Læserbrev. Af : Ole Vilmann
Lersøparkalle 37 I.
2100 Kbh Ø.
(01) 205958.

Kender I de prioder,hvor man sidder og har lyst til at skrive et eller andet program, men ikke kan finde på en ide? Det gør jeg, men det er nu ikke grunden til at jeg skriver. Derimod, hvis en eller anden sidder med ovennævnte problem, er der en ide på vej, som jeg ikke selv har tid til at realisere p.g.a. examensprojekt på DTH.

Baggrund:

I industrien, benytter man, eller kunne godt tænke sig at benytte en såkaldt skærmgenerator når man er i forbindelse med administrativt programmel. En skærmgenerator er et assemblerprogram, der udfra specifikation af udseende af et skærmbillede kan generere et andet assemblerprogram, der igen kan fortage styring af variable til og fra forud spec. felter på skærmbilledet. Styringen går ud på at checke, om det er tilladelige variabeltyper der ønskes uskrevet i forud spec. felter, og udskrive variablene til de rigtige felter.

For dem der ikke ved, hvad et skærmbilled er: Dette er et fast billed på skærmen, som hele tide er uændret. Dette vil sige, at der er forklarende tekst med efterfølgende felt til indtastning / udskrivning af en variabel. Der er måske nederst et specielt felt til fejludskrifter. Der er vedlagt et skærmbilled eksempel.

Anvendelsesområde:

Af anvendelses område for NASCOM 1/2 kan nævnes de database programmer der er lavet, programmer der ikke benytter sig af alt for mange ind / uddata, programmer der benyttes af ikke indviede personer (skærmbilledet styrer, hvor værdierne skal indtastes).

Opbygning af skærmbilledgenerator:

I det følgende beskrives de dele af skærmgeneratoren der er nødvendige, samt hvilke funktioner den ønskes indholdt.

Som nævnt må skærmgeneratoren bestå af to dele:

a) Generator del, som generer et færdigt underprogram ud fra udseende af billedet, og ud fra hvilke tekster og

felter (typer) der ønskes.

b) Det færdige underprogram (genereret af a)), som kan kaldes af andre programmer (f.ex. PASCAL-runtime package).

a) Inddata:
   1) Rammer.
   2) Alfanummeriske felter.
   3) Nummeriske felter.
   4) Tekster.

b) Underprogram:
   1) Specifikation af feltnummer.
   2) Check af felttype contr. variabeltype.
   3) Læse / skrive til / fra felt.

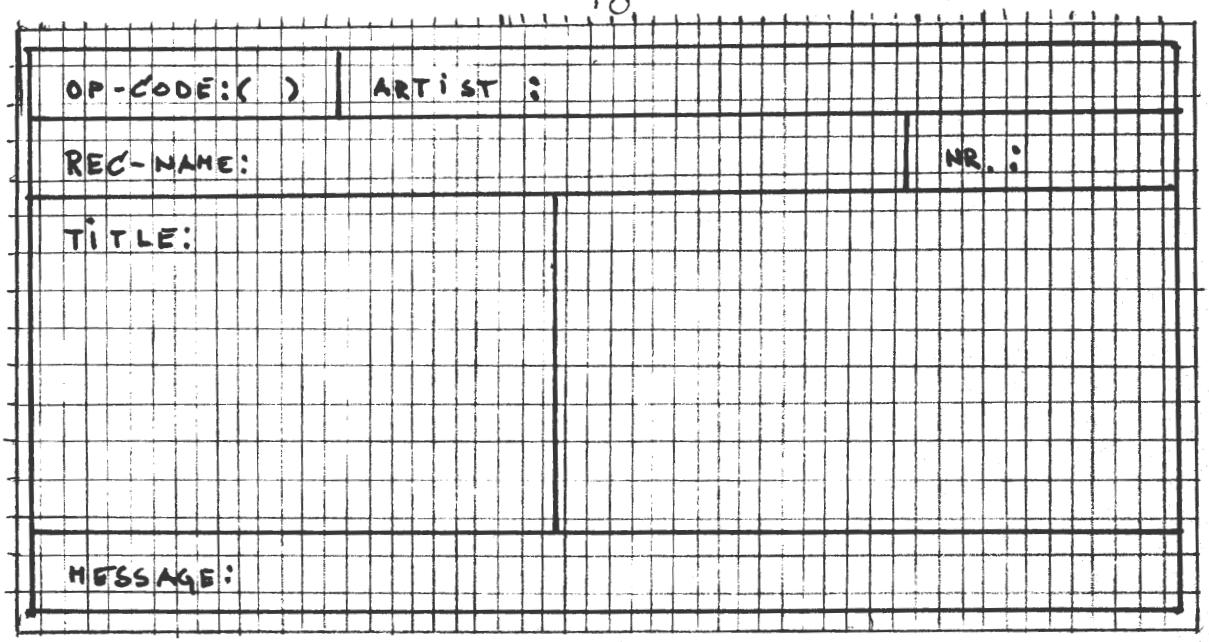
Det er meningen, at når først skærmbilledet er i produktionskørsel, skal det checke cursor position, samt styre al skrivning til / fra skærmen. Det må f.ex. ikke være muligt at flytte rundt med curser ex.

Hvis der er nogle der interesserede i ovenstående program, er de velkomne til at ringe til mig.

Efter dette brev, er det min hensigt, at beskrive en ny PASCAL-copiler (HISOFT PASCAL VERS. 4) som er indkøbt fra England i en af de følgende numre.

MVH. Ole Vilmann. 365.

18



16

Eksempel på skærmbilled. Her er eksemplet tilknyttet et plade-bibliotek. (Database, hvor posterne ikke overstiger størrelsen af billedet).

64k RAM

Til byggevejledningen omtalt i N.N. Nr.9 1982, har jeg nogle indvendinger.

For det første :

Jeg savner oplysninger om fabrikat og hastighedsselectering.

Nogle fact fra 64k ram data blade :

fabrikat	No.		t_{RAS}	t_{RP}	v_{il}	t_{RAC}	t_{CAC}	refs. adre.	time mS
Motorola	MCM6665	-15	150	120	-2	150	75	128	2
		20	200	140	-2	200	110	128	2
		25	250	190	-2	250	145	128	2
Fujitsu	MB8264	-10	100	90	-1	100	50	128	2
		12	120	100	-1	120	60	128	2
Hitachi	HM4864	-2	150	100	-1	150	100	128	2
		12	120	100	-1	120	60	128	2
		15	150	100	-1	150	75	128	2
		20	200	120	-1	200	100	128	2
		3	200	120	-1	200	135	128	2
NEC	uPD4164	-3	150	100	-1	150	100	128	2
		2	200	120	-2	200	135	128	2
		1	250	150	-2	250	165	128	2
Intel	I 2164	-15	150	120	-2	150	85	128	2
		20	200	135	-2	200	110	128	2
		25	250	175	-2	250	135	128	2
Toshiba	TMM4164	-2	120	90	-1	120	80	128	2
		3	150	100	-1	150	100	128	2
		4	200	120	-1	200	135	128	2
Texas	TMS4164	-15	150	100	-1	150	100	256	4
		20	200	120	-1	200	135	256	4
National	NMC4164	-1	140	120	-1	120	65	256	4
		2	165	130	-1	150	90	256	4
Mostek	MK4564	-15	150	100	-2	150	85	128	2
		20	200	135	-2	200	115	128	2
		25	250	165	-2	250	145	128	2
Mitsubishi	M5K4164	-15	150	100	-1	150	75	128	2
		20	200	120	-1	200	100	128	2
OKI	MSM3764	-12	120	90	-1	120	80	128	2
		15	150	100	-1	150	100	128	2
		20	200	120	-1	200	135	128	2

I mostek microelectronic data book 82/83 står der under Z80 refresh : $\overline{RFSH}$ indikerer at A_0-A_6 på adresse bussen indeholder en refresh adresse til dynamisk ram. A_7 er logisk "0". A_8-A_{15} er indholdet af "I" registret.

Det kan her konkluderes at Z80 CPUIKKE kan refreshe 64k ram fra National eller Texas. Dog kan der indføges et specielt kredsløb til adresse A_7 under refresh.

Her skal siges at dynamiske ram's også kan refreshes ved at læse eller skrive i hver række i rammerne tilstrækkeligt tit, da vil der ikke blive nogle udfald af data.

Anvender man en ramtype der fordre 256 row addres refresh, og tænker man sig en lille ventesløjfe på f.eks. 1 minut, der kun benytter nogle få addresser, vil op til 50% af ramlagrets indhold gå tabt.

For dynamiske rammer gælder det at det er leakstrøm som aflader lagersellernes ladning, denne leakstrøm halveres for hver gang temperaturen falder ca. 10°c. Det vil med andre ord sige at ved en ram omgivelses temperatur på 35°c, vil en ram der fordre 2 mS refresh time ved 70°c, normalt ikke få data udfald hvis der går op til 7 mS imellem hver gang den enkelte række bliver addresseret i rammen.

Det skal dog siges, at det er maximum tider, og at rammerne oftest er noget bedre.

For det andet :

$\overline{MREQ}$ "off" (høj) for Z80 CPU ved clock = 4 MHz ligger imellem 95 nS og 125 nS, imellem en "M1" og refresh cycle. Hertil tilføjes en spredning på optil $\pm$ 20 nS, ialt imellem 75 nS og 145 nS, stammende fra buffere og gates inden $\overline{MREQ}$ signalet bruges til $\overline{RAS}$ på rammerne.

Ved at se i tabellerne (t_{RP}) konstateres det at ingen af de her nævnte rammer får deres krav opfyldt.

For det tredje :

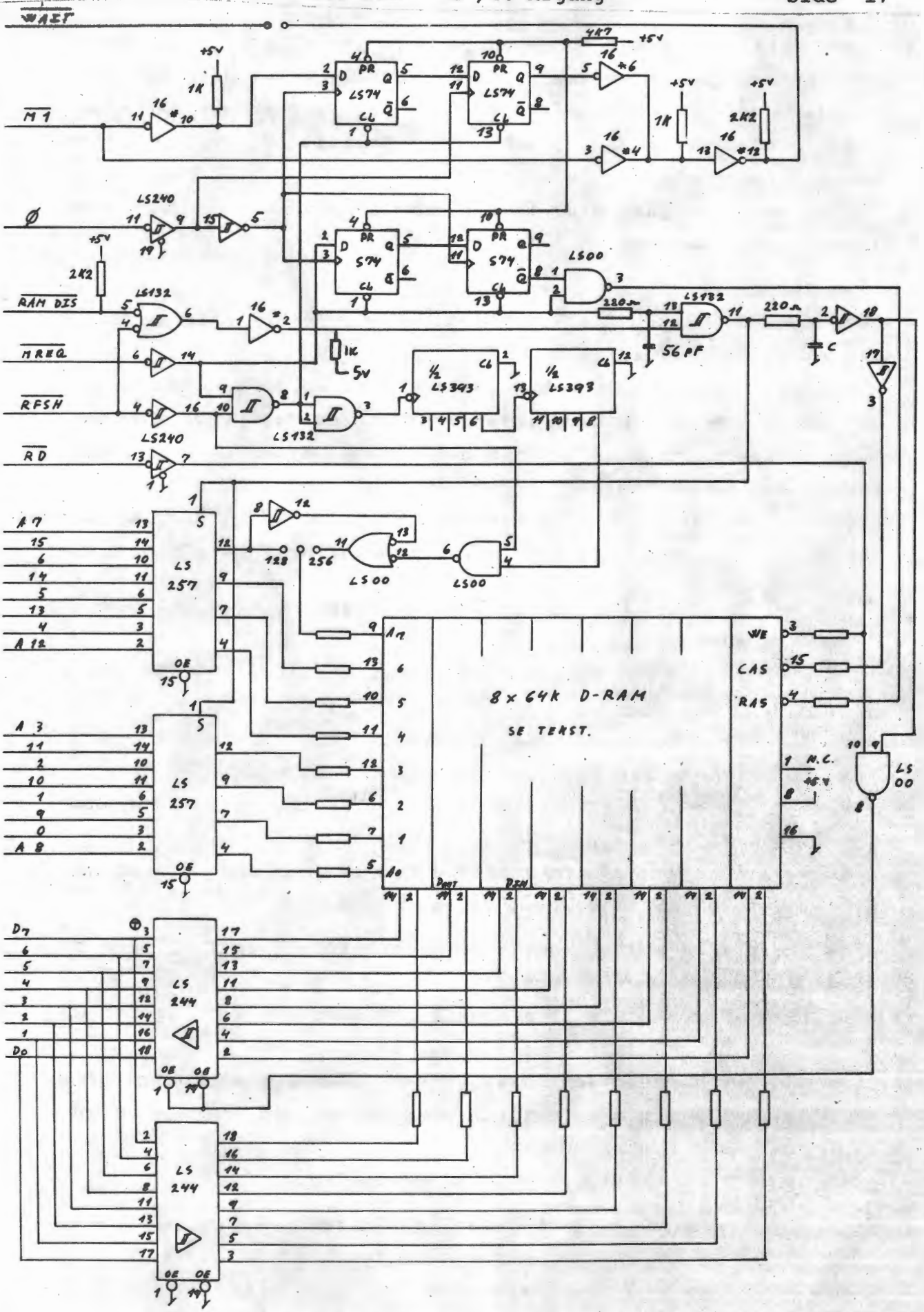
Konstruktionen af delay line med en flok buffere i serie, må betejnes som uheldig, idet delayet er meget udefineret

til multiplexer

En løsning som den her viste ville give nogle mere veldefinerede delays.

For det fjerde :

Formentlig alle signalindgange (64k ram's) forsynes med en seriemodstand, hvis værdi først endelig kan bestemmes ved måling på det aktuelt udlagte print, hvis rammerne skal køre optimalt hastighedsmæsigt. En sådan måling kræver at osciloskop og probebåndbredden er større end 100 MHz. Årsagen til at sætte en serie modstand, er at ingen spikes må komme under den i tabellen nævnte negative spænding (V_{il}). Indgangene på rammerne er mos og derfor kun kapasitive, Denne kapasitet sammen med sprennings-



* open collector

Ⓣ three-state

Jørgen M-Nielsen

83-03-04

C: ca 100 pF ved 128 adr. ca. 220 pF ved 265 adr. refresh

selvinduktionen giver en resonanakreds der slås an af de stejle flanker fra TTL kredsene. Ved fornuftig printudlæg kan modstandene være 10-68 ohm, ved dårlig 0 og 5v forbindelser eller 74s-- kredse noget større modstande. Et godt printudlæg har et Ov og 5v gitter, over hele printet.

Hvis denne ram parameter ikke overholdes kan det medføre ødelæggelse af rammerne.

For det femte :

Den anvendte latch (ls375) til A_{12}-A_{15} er overflødig og kun til extra fejlkilde.

Ved forespørgsel hos Mostek igennem Semicap, erkendte man at den latch som Mostek har foreskrevet var en "bøf", de angivne timing diagrammer i aplication note, Z80 - dynamisk ram, eksisterer kun på papiret.

Både Zilog, SGS og Mostek, specifiserer for Z80 4 MHz, at addressen til enhver tid er stabil min. 65 nS efter $\overline{MREQ}$ går høj.

Det her medfølgende diagram stiller følgende krav til 64k D-ram kredse :

1. Hvis det er 256 adr. refresh, skal jumperen $^{128}/_{256}$ stå i 256.
2. t_{RP} ($\overline{RAS}$ precharge time) mindre end 230 nS.
3. t_{RAS} ($\overline{RAS}$ pulse with) mindre end 220 nS.
4. t_{RAC} ($\overline{RAS}$ access time) mindre end 180 nS, med wait kredsløbet 285 nS.
5. t_{CAC} ($\overline{CAS}$ access time) mindre end 100 nS, med wait kredsløbet 205 nS. Det er en M1 wait cycle.

Ved ram kredse med 256 adr. refresh shal t_{RAC} og t_{CAC} være 25 nS mindre end de her anførte.

Der er medregnet delay fra 1 buffer i addresse ledningerne og 1 buffer i data ledningerne til cpu'en.

Jeg uddyber gerne ovennævnte, og skal bemærke, at kredse ofte er noget hurtigere end fabrikanterne lover, med undtagelse af rammerne, da disse er udmålt.

Diagrammet har været opbygget og kontroleret.

Med venlig hilsen

Jørgen M-Nielsen o2 32 88 45

Nascom no 142 (efter 20^{30})

```
Filnavn: COMPCAT.PAS
PROGRAM KATALOG;      (* COMPAS Version 1.0 *)
(*$R-A+*)

(* ------------------------ Global blok ------------------------------ *)
CONST
     DISKSIZE = 384;      (* Bør være diskkapacitet - SYS *)
     MAXDIR = 128;        (* Hardware-afhængig som ovenfor *)


TYPE
     BYTE = 0..255;
     CHAR = STRING(.1.);
     STR3 = STRING(.3.);
     STR8 = STRING(.8.);
     STR12 = STRING(.12.);
     STR60 = STRING(.60.);
     STR80 = STRING(.80.);

     DIRENTRY = RECORD
                  NAVN: STR8;
                  EXT: STR3;
                END;

     DIRLIST = ARRAY(.1..MAXDIR.) OF DIRENTRY;
     LISTE = ARRAY(.1..MAXDIR.) OF STR12;
     KBYTE = ARRAY(.1..MAXDIR.) OF INTEGER;

VAR
    SUM,N,I,J,K1,K2: INTEGER;
    D: DIRLIST;
    DIRL: LISTE;
    KB,KB1: KBYTE;
    CH1,CH: CHAR;
    F: TEXT;
    LPT,STOP: BOOLEAN;

(* ---------------------- Global blok slut ----------------------------- *)
```

```
FUNCTION STORT_BOGSTAV(C:CHAR): CHAR;
BEGIN
  IF (C>='a') AND (C<='å') THEN
    STORT_BOGSTAV := CHR(ORD(C)-32)
  ELSE
    STORT_BOGSTAV := C;
END;

FUNCTION JA(T: STR60): BOOLEAN;
VAR C: CHAR;
    ST: STR80;
BEGIN
  ST :=' '+T+' (J/N) ? ';
  REPEAT
(*    GOTOXY(0,22);
æ GOTOXY, CLREOL og CLREOS: Hardwareafhængig å
    CLREOL;*)          (* Brug dem kun på dit eget system *)
    WRITE(ST+' ');     (* Bemærk, at 'flettede' kommentarer kræver 2 symboler *)
    GET(C);
    C := STORT_BOGSTAV(C);
  UNTIL (C='J') OR (C='N');
  JA := (C='J');
  (*GOTOXY(0,22);
  CLREOL *)
END;

FUNCTION BDOS(FUNK: BYTE; PARM: INTEGER): BYTE;
BEGIN
  CODE  $3A,FUNK,$4F,$ED,$5B,PARM,$CD,$05,$00,$32,BDOS
END;

PROCEDURE GETDIR(DRIVE: CHAR; VAR DIR: DIRLIST; VAR NUMENTRY: INTEGER);
VAR
    I,J: INTEGER;
    S1,S: STR8;
    DMABUF: ARRAY(.0..3,0..31.) OF BYTE;
    FCB: ARRAY(.0..35.) OF BYTE;
```

```
(* ------------------ Lokal procedure for GETDIR ------------------ *)

PROCEDURE KONVERT;
VAR P: INTEGER;
BEGIN
  S1 := '';
  FOR P:=1 TO LEN(S) DO
    BEGIN
      IF S(.P.) = '&' THEN
                  S1 := S1+'&'
                  ELSE
                    S1 := S1+S(.P.);
    END;
  S := S1;
END;


(* ------------------------------------------------------------------ *)

BEGIN (* GETDIR *)
  I := BDOS(26,ADDR(DMABUF));
  FCB(.0.) := ORD(DRIVE) - $40;
  FOR I:=1 TO 11 DO
    FCB(.I.) := $3F;
  FOR I:=12 TO 35 DO
    FCB(.I.) := 0;
  NUMENTRY := 0;
  I := BDOS(17,ADDR(FCB));
  WHILE I<255 DO
    BEGIN
      NUMENTRY := NUMENTRY+1;
      S := '';
      FOR J:=1 TO 8 DO
        S := S+CHR(DMABUF(.I,J.));
      KONVERT;
      DIR(.NUMENTRY.).NAVN := S;
      S := '';
      FOR J:=9 TO 11 DO
        S := S+CHR(DMABUF(.I,J.));
      DIR(.NUMENTRY.).EXT := S;
      I := BDOS(18,ADDR(FCB))
    END
END;
```

```
PROCEDURE HOVED(S:STR60);
BEGIN
(*  GOTOXY(0,0);
  CLREOS; *)
  WRITELN(S,CH,':');
  WRITELN;
  FOR I:=1 TO 4 DO
    WRITE('Navn    .Type  Kb   ');
  FOR I:=1 TO 4 DO
    WRITE('---------------   ');
END;

FUNCTION FILESIZE(DRIVE: CHAR; N: DIRENTRY): INTEGER;
VAR
    I: INTEGER;
    F: REAL;
    FCB: ARRAY(.0..35.) OF BYTE;
BEGIN
  FCB(.0.) := ORD(DRIVE)-$40;
  FOR I:=1 TO 8 DO
    FCB(.I.) := ORD(N.NAVN(.I.));
  FOR I:=9 TO 11 DO
    FCB(.I.) := ORD(N.EXT(.I-8.));
  FOR I:=12 TO 35 DO
    FCB(.I.) := 0;
  I := BDOS(35,ADDR(FCB));
  F := FCB(.33.)+SWAP(FCB(.34.));
  FILESIZE := ROUND(F);
END;

PROCEDURE SORT(VAR LIST:LISTE; VAR INDX: KBYTE; ANT: INTEGER);
VAR
   G,H,I,L,N: INTEGER;
BEGIN
  FOR N:=1 TO ANT DO
    BEGIN
      L := 0;
      H := N;
      WHILE (H-L)>1 DO
        BEGIN
          G := (H-L) DIV 2+L;
          IF LIST(.N.) >= LIST(.INDX(.G.).) THEN
                        L := G
                        ELSE
                          H := G;
        END;
```

```
      FOR I:=N-1 DOWNTO H DO
        INDX(.I+1.) := INDX(.I.);
      INDX(.H.) := N;
    END;
END;

PROCEDURE DIR(C: CHAR);
LABEL EXIT;
VAR FZ: INTEGER;
    CH: CHAR;
BEGIN
(*  GOTOXY(0,0);
  CLREOS; *)
  HOVED('Sorteret katalog i drev ');
  SUM := 0;
  GETDIR(C,D,N);
  FOR I:=1 TO N DO
    BEGIN
      DIRL(.I.) := D(.I.).NAVN+'.'+D(.I.).EXT;
      KB1(.I.) := I;
    END;
  SORT(DIRL,KB1,N);
  STOP := FALSE;
  FOR I:=1 TO N DO
    BEGIN
      IF KEYPRESS THEN      (* Dersom der trykkes ESC under kørsel, stopper *)
         GET(CH);           (* udskriften, og der hoppes tilbage til start. *)
      STOP := ORD(CH)=27;   (* Tilsvarende kan indføjes i printer-proceduren *)
      IF STOP THEN GOTO EXIT;
      FZ := (FILESIZE(C,D(.KB1(.I.).))+7) DIV 8;
      KB(.I.) := FZ;
      SUM := SUM+FZ;
      WRITE(DIRL(.KB1(.I.).),FZ: 5);
      IF I AND 3 = 0 THEN    (* Kan erstattes af:  IF I MOD 4 = 0 THEN *)
        WRITELN
      ELSE
        WRITE('   ');
    END;
  WRITELN;
  WRITELN;
  WRITELN('Anvendt ca. :',SUM:5,' Kb');
  EXIT:
END;
```

```
PROCEDURE PRINTER;
VAR TXT: STR60;
    F  : TEXT;
    I  : INTEGER;
BEGIN
(*  GOTOXY(0,22);
  CLREOL;*)
  REWRITE(F,'LST:');
  WRITE(F,CHR(24),CHR(29),CHR(27),CHR(54),CHR(27),CHR(66));
  TXT := '';
  (*GOTOXY(0,22);
  CLREOS;*)
  WRITELN;
  WRITE('Evt. tekst: ');
 (*  GOTOXY(20,22); *)
  BUFLEN := 39;
  READLN(TXT);
  WRITELN(F,CHR(31),TXT,CHR(29)); (* Overskrift er "mellemstor" *)

(* --------  Bemærk forskel/lighed med PROCEDURE HOVED ------------- *)
  FOR I:=1 TO 77 DO
    WRITE(F,'-');
  WRITELN(F);
  FOR I:=1 TO 4 DO
    WRITE(F,'Navn    .Type  Kb   ');
  WRITELN(F);
  FOR I:=1 TO 4 DO
    WRITE(F,'----------------   ');
  WRITELN(F,CHR(27),CHR(56));
(* ------------------------------------------------------------------ *)
  FOR I:= 1 TO N DO
    BEGIN
      WRITE(F,DIRL(.KB1(.I.).),' ',KB(.I.): 4);
      IF I AND 3 = 0 THEN
        WRITELN(F)
      ELSE
        WRITE(F,'   ');
    END;
  WRITELN(F);
  WRITELN(F);
  WRITELN(F,'Anvendt ca. ',SUM:5,' Kb');
  FOR I:=1 TO 8 DO  (* 8 angiver antal linier til næste printer udskrift *)
    WRITELN(F);
  CLOSE(F)          (* CLOSE(F,LOCK) ikke nødvendig her *)
END;
```

```
PROCEDURE LPNORM; (* Resætter printer til normalstatus *)
VAR F: TEXT;
BEGIN
  REWRITE(F,'LST:');
  WRITE(F,CHR(30),CHR(27),CHR(54),CHR(27),CHR(66));
  CLOSE(F,LOCK)
END;

FUNCTION GENTAG: BOOLEAN;
BEGIN
  (*GOTOXY(0,22);
  CLREOL;*)
  WRITE('Drev (A) eller (B) ');
  GET(CH);
  WRITELN;
  CH := STORT_BOGSTAV(CH);
  GENTAG := (CH='A') OR (CH='B')
END;

BEGIN
(*  GOTOXY(0,0);
  CLREOS;
  GOTOXY(0,20);*)
  WRITE('KATALOG: Tastes andet end A eller B, afbrydes programmet.');
  WHILE GENTAG DO
    BEGIN
      DIR(CH);
      IF NOT STOP THEN
        LPT := JA('Ønskes udskrift på (Microline 82A) printer');
      IF LPT AND (NOT STOP) THEN
        PRINTER;
    END;
  IF LPT AND (NOT STOP) THEN
    LPNORM;
(*  GOTOXY(0,22);
  CLREOL *)
END.
```

GG 369.

Læserbrev: Af Ole Vilmann. mnr. 365.

Emne: Brugen af medlemsbladet NASCOM-NYT.

Den serie af artikler der synes at skulle udgives i Z80-nyt :"Introduktion til Z80 programmering", er efter min mening uheldig. Derved, at der er mange af os Z80-nyt læsere der er "hjemme" i maskinprogrammering, hvorved disse artikler er værdiløse for os. Det forslag jeg har er, at disse artikler trykkes som særtryk og sælges eller gives til de medlemmer der skulle være interesserede. Jeg mener, at disse artikler optager alt for meget plads i NASCOM-NYT (se bl.a. sidste nummer). Jeg vil udtrykke min sympati for dette projekt. Forfatteren skriver, at der er en håbløs mangel på dansk litteratur om dette emne. Dette vil jeg give ham ret i, men der findes derimed mange virkelig gode på Engelsk.

Checksumsberegning.

En sen nattetime sad jeg og kikkede lidt på en udlæsning, som jeg havde fået skrevet ud. Formatet var udskrevet i INTEL format, og det var et kort lille Basicprogram. INTEL formatet er lidt specielt, men ligner det rigtige lidt. Jeg har udskrevet hele formatet, så Du kan se, hvordan det er opbygget. De to sidste tegn på hver linie er checksummen, men hvad bruges den til? Jo det siger navnet jo, men hvordan regnes den ud? Man lægger samtlige hexadecimale tal sammen, undtagen checksum. Lad os tage den første linie i udskriften.

10+70+00+00+FF+55+6A+0A+00+81+20+58+05+31+20+BD+20+31+30+00= 5A5

Har Du en TIprogrammer er det jo let, men lad os nu se lidt på resultatet. Prøv så at skrive det op på denne måde.

	5	A	5	
	0101	1010	0101	
To's komplement til et	1010	0101	1010	vend resultatet. Læg 1 til.
binært tal er lig med	0000	0000	0001	
tallets komplement plus 1	1010	0101	1011	
	A	5	B	Fjern A som er menten.

Hvis Du ikke er klar over, hvordan man lægger 1 og 0 sammen så har jeg vist det her. TIprogrammeren vil vise "FFFFFA5A.

```
************************************
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0          1

      resultat     Carry=mente
```

Menten adderes lige som i decimal addition, til den næste højere bitposition.

Hvis checksummen ikke passer vil der komme et ? når der indlæses i NASCOM. Det samme er tilfældet ved V. V= verify, og det sker også ved V i assembler, CLOAD? og i PASCAL. Det er en stor fordel, for er du ved at køre noget ind i blokke, og der kommer et ? så kør lidt tilbage med båndoptageren og start igen. Det er altså ikke nødvendigt at starte forfra som i mange andre computere.

Der vil i et senere NNyt komme et program, som vil kunne sende og modtage INTELFORMAT. Jeg vil her nævne, at vi nu også kan sende og modtage til de af vore medlemmer som har CP/M . Dette vil også blive beskrevet, så der er noget at glæde sig til.

Skriv til os her i bestyrelsen, hvis Du kunne tænke Dig at få omtalt en eller anden ting, Du ikke kan få oplysning om, så vil vi forsøgeat hjælpe.

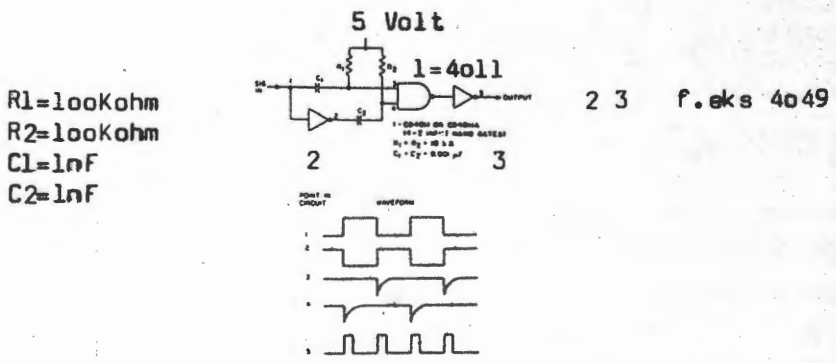
Ole Hasselbalch

Den ombygning af RAM A kort, som Kaj Mortensen beskriver i NN8 gælder også for NASCOM 1(selv om der nok kan findes en lidt kortere ledningsføring), men i grunden er den ikke helt fin i kanten. Ser man efter i databladene, finder man, at ben 18 er chip-select, og ben 20 er output-enable. Hvis man kun drømmer om at bruge 2716, kan det være helt ligegyldigt, men skulle man få den (skøre) ide at sætte 6116 ind i stedet, så bliver modifikationerne lidt mere omfattende, så skal ben 21 forbindes til WR i stedet for 5V, alle ben 18 skal skæres fri fra OV og forbindes til fire forskellige CS (og så kan man lige så godt lægge dem i den rigtige rækkefølge) mens ben 20 alle lægges til RD. Det er ret mange baner at skære over, men det virker altså, i hvert fald på 2MHz, og jeg ser ingen grund til, at det ikke også skulle virke på 4MHz. Og hvad skal man så med det? Ja det er altså en god ting for NASCOM 1 ejere, for det samme kan man også gøre på CPU-kortet, så man kan sætte en 2716 i den ene sokkel og en 6116 i den anden, og lave en omskifter styret af bit 2 i port 0, og tænk det virker spilleme også. Vil I have et diagram?

Christian Laustsen

Frekvensdobler til ur.

Her er vist en simpel løsning på en frekvensdobler.Den indeholder ikke mange komponenter og virker med det samme.



Det svære valg.

Jeg ved at nogle af os gerne så , at jeg brugte NASPEN, men personligt kan jeg bedst lide min gode gamle skrivemaskine, som jeg for år tilbage købte for de penge, som jeg fik for min første artikkel.Se, her er allerede et valg,men det bliver endnu vanskeligere at vælge den rigtige computer.
Sagen er jo den, at skalkøbmand Jensen have sig en, så ser han kun på om den kan dække hans behov, og ikke så meget på om den er servicevenlig.Den skal bare have et godt lagerprogram, samt et eller flere programmer til kontooversigt, samt tilslutning til printer.
Er man tekniker er behovet helt anderledes, og har man nu også en radiosender, ja så melder der sig mange overvejelser.Jeg har lavet en liste, som jeg gerne ser kommenteret,hvis der mangler nogle spørgsmål.
Da jeg mener det er relevant, at vi holder os til Z80,kan noget udelades.
Husk endelig at DINE krav efterhånden stiger, når du kender computeren.

Er der mulighed for andre sprog end BASIC ?f.eks PASCAL COMAL o.s.v.
Hvordan er en evnt GRAFIK?
Kan LAGER udvides ?
Er der PRINTERTILSLUTNING, og er den seriel eller parallel ?
Er der RS232C seriel udgang ?
Kan systemet tilsluttes DISK ?
Kan der udvides til CP/M ?
Kan man assemblere ?
Hvordan er SERVICE ?
Er der et godt PROGRAMBIBLIOTEK ?
Holdes der KURSUS ?
Udkommer der et BLAD ?
Findes der en FORENING ?
Hvad KOSTER den ?
Kan den arbejde SAMMEN med andre Z80 computere,eller bringes til det ?
Kan der indbygges REALTIME ?
Er PIO ført ud ?
Kan der fås udvidelsesmoduler til FARVE, I/Okort og lignende ?
Kan man komme ind i lageret udefra ?
Er den nem at betjene.
Hvordan er erfaringerne fra andre computere af samme type ?
Hvad koster PROGRAMMER ?
Er der NORMALkomponenter isat ?

Hvad med en serie artikler, om netop DIN computer.Jeg ser gerne et par gode artikler.
Lad os se noget om:

NEW BRAIN
TRS 80
LYNCH
ZX 81
VIDIO GENIE
Samt over andre Z80 maskiner. Ring til mig, så vi kan fordele opgaven.

Det jeg er ude på i denne artikkel er en større bredde af vores blad. Dette vil give os flere medlemmer, og et mere alsidigt blad.

Ole Hasselbalch

Svar på læserbrevet fra 365 (se side 22)

Formålet med artikelserien "Introduktion til Z80 Programmering" er netop at hjælpe de mennesker i gang med maskinkodeprogrammering, som har svært ved engelsk, og dem er der vitterligt mange af. Et er at læse en kriminalroman på engelsk, noget andet er at læse og forstå en lærebog om noget så teoretisk som Z80 programmering. En undersøgelse på Københavns Universitet viser, at 40% af de førsteårsstuderende har betydelige læse- og forståelsesvanskeligheder ved deres engelske lærebøger (se Computerworld nr. 6, 15 marts 83 side 21), så det er ikke svært at forestille sig, hvilke problemer mange af foreningens medlemmer har, når de åbner en engelsk lærebog om Z80. Og hvis ikke Z80-NYT skulle være det rette forum til en artikelserie om Z80, så ved jeg ikke hvor. Dog vil jeg give dig ret i, at 13 sider ud af 26 om Z80 programmering er lidt i overkanten, når bladet skal være alsidigt. De mange sider var også ment som en appetitvækker og de kommende artikler vil heller ikke fylde så meget. Hvis de avancerede Z80 brugere synes, at der ikke er noget stof for dem i Z80-NYT (og det synes de åbenbart), skal de bare fatte pennen og skrive hvad netop de sysler med omkring Z80, for jeg håber ikke de har glemt, at de også var novicer en gang.

Jesper Skavin.

ORIENTERING TIL MEDLEMMERNE FRA FORRETNINGSFØREREN:

Da der har været problemer med udsendelse af blade og lister til såvel gamle som nye medlemmer (fra 1983), er nedenfor beskrevet,hvad der er udkommet: I 1982 udkom der 10 numre af Nascom/Z80 Nyt; i 1983 incl. dette nummer 3 blade. Nye medlemmer siden 1. januar 83 har dog kun fået nr. 1 - 3 fra 83. Endvidere blev der i december 82 sendt medlemsliste og programbibliotek til medlemmerne plus lidt senere henholdsvis supplementsliste og rettelsesliste. Medlemmer, der ikke har fået ovenstående, bedes skrive til forretningsføreren. Nuværende medlemsliste er dog desværre sluppet op, men en ny liste forventes udsendt i april måned.

Venlig hilsen

Inga Skavin

Endnu en E-Prom programmeringsenhed.

Den her viste E-Prom programmeringsenhed har den fordel at den direkte kan programmere 2716, 2732 og 2764. Enheden tilsluttes Nascoms PIO, og styres helt af softwaren. Det er envidere simpelt at udvide til 27128 når den bliver aktuel.

Hardware.

Princippet for kredsløbet er at PIOens port B benyttes som data port, medens port A benyttes til styring i overensstemmelse med tabel 1. Kredsenes addresse sættes op ved hjælp af de binære tællere 14040 der resettes og klokkes af bit 0 og 1 i port A.

For en nærmere forståelse af diagrammet er det nødvendigt at se på funktionstabellerne for de enkelte kredse i databøgerne, til hvilke der henvises. Dog skal det bemærkes at da ben 21 skal have 5V ved 2716, medens det er addressebit 11 ved 2732 og 2764 er det forsynet med en switch til 5V, der er styret af Q12 fra 14040. Når 2716 skal programmeres sættes Q12 derfor ved forlods at give tælleren 2048 (800H) klokimpulser. Ligeledes bemærkes det at OE ben 20 styres fra Q2, IC2 og sættes til logisk et ved at give tælleren 8352 (2000H) klokimpulser.

Software.

Styreprogrammet kan læse, skrive, teste korrekt sletning og programmering af E-Promkredsene.

Man kan læse og skrive til en hvilken som helst byte i kredsen, og man kan læse til et hvilket som helst lageromrade og skrive fra et hvilket som helst lageromrade.

Programmet betjenes ved at skrive:

E prog. add. SP typenummer SP start add. i E-Prom SP slut add.i E-Prom SP start skriv add. SP start læs add.

Programmet viser herved en menu, der har de fire ordrer:
L læs, E test for sletning, S skriv, V verificer programmering.

Ved programmering af en kreds foreslåes det at udføre ordrene i den ovenfor nævnte rækkefølge.

Det bemærkes at det er vigtigt at man indtaster det rigtige typenummer. forsøger man at programmere en anden kreds end den, hvis typenummer man har indtastet, vil den brænde af, da den så får programmeringspænding på en forkert terminal.
Programmet viser typenummeret i overskriften, og afviser andre numre end de tre det er beregnet for.

Programmeringsimpulsen der er 50 mS lang er udført ved hjælp af forsinkelsesrutine der er afstemt til en CPU klokfrekvens på 4MHz.
Anvendes en anden frekvens skal argumentet i programmets linie 167 (addresse 8150) ændres. (4MHz : 1000 Hex; 2MHz : 800 Hex)

Opbygning.

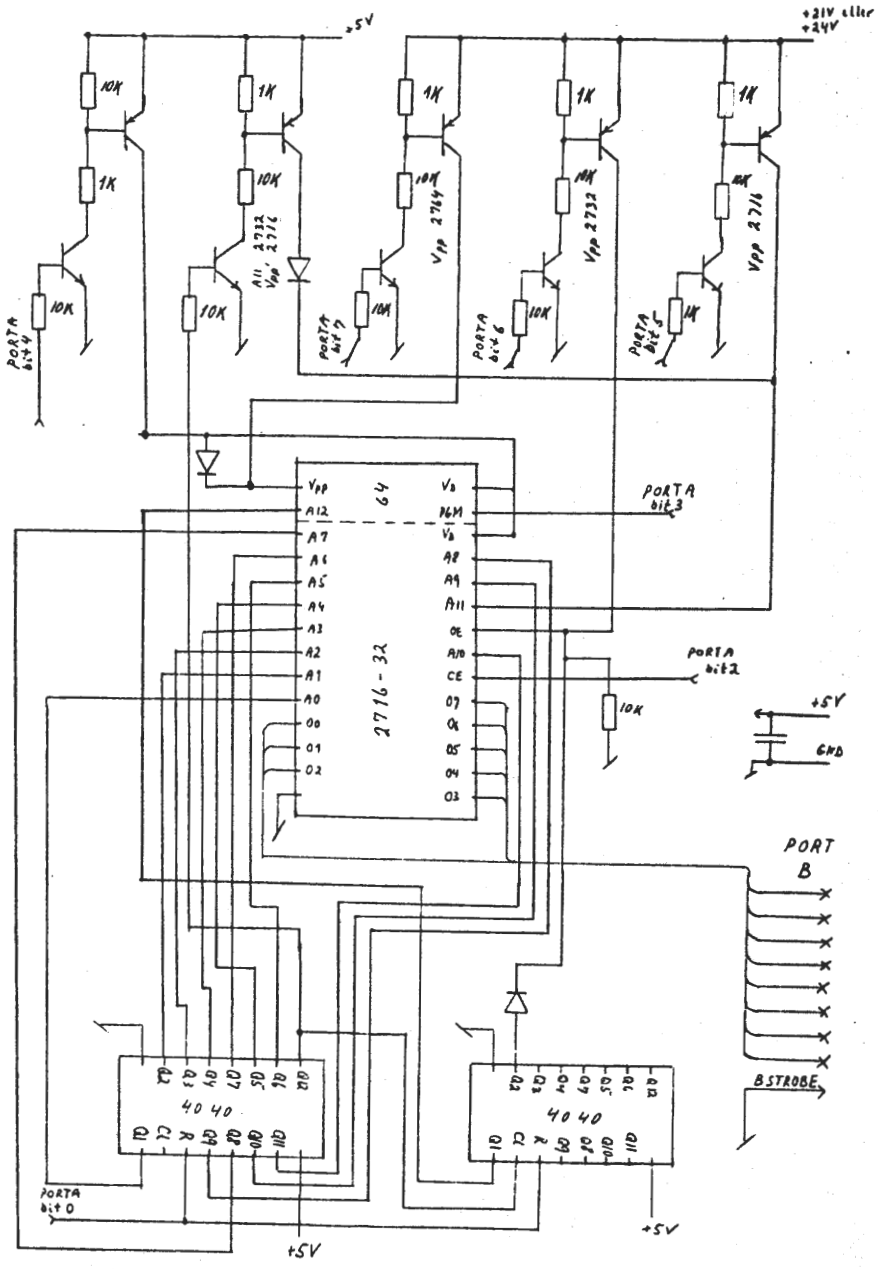
Jeg har bygget kredsløbet op på hulprint og indbygget det i en lille plast kasse med et 25 polet stik. 5 Volt forsyningen kommer fra datamaten, medens programmeringsspændingen der er 21 eller 24V (se endelig efter i databladet) er ført ind fra en separat strømforsyning.

der kun tilsluttes når jeg skal programmere.

Erik Bech OZ2IF

Port A bitnummer	Funktion
0	Reset addressetæller
1	Klok addressetæller
2	Chip select (ben 18)
3	PGM (2764)
4	+5V forsyning
5	Vpp (2716)
6	Vpp (2732)
7	Vpp (2764)

TABEL 1



049
NIELSEN STEEN LÆRKE
ELLEGÅRDSVEJ 7
2820 GENTOFTE
ZILOG M. FLOPPY
OZ 5SO

094
MALMSTRØM TOM
EGEHEGNET 30 2.TH
2850 NÆRUM
-
-
02 80 69 54

314
NIELSEN THOMAS KRAG
HJORTESPRINGPARKEN 17
2730 HERLEV

ÆNDRINGER

VELKOMMEN TIL:

1
381
CHRISTENSEN JENS
BROBYVEJ 50
4180 SORØ

1
382
VENDELBO DANNY
NY MUNKEGADE 9B
8000 ÅRHUS C

383
LARSEN HELGE / CBT-FGRP/BA
HAVVÆNGET 12_
7500 HOLSTEBRO

384
ISSOV BO
NISSEDALEN 4
2740 SKOVLUNDE

385
HANSEN TOM
SAMSØVEJ 45 / VESTERHUS 201
4300 HOLBÆK

386
JENSEN PETER
KINDHESTEGADE 2
4300 HOLBÆK

387
ANDERSEN CARSTEN
DROSSELVEJ 54 / VESTER HASSING
9310 VODSKOV

388
NYBORG MIKAEL
INDELUKKET 5 / VOLLERUP
4200 SLAGELSE

389
CHRISTENSEN TOMMY
BAKKEHØJ 7
7600 STRUER

390
ANDERSEN MADS SIGGÅRD
OLAF POULSENSVEJ 7
2920 CHARLOTTENLUND

391
THESTRUP POUL
AVANALØVEJ 13
5700 SVENDBORG
Evt. gruppe FYN

392
RØRKE PETER
VEDELAVEJ 15
2610 RØDOVRE

393
LUND KARSTEN
ÅSUMSVEJ 661
5240 ODENSE NØ

394
NIELSEN LARS
KILDEBAKKEN 6
5260 ODENSE S

395
HAARUP NIKOLAJ
TURENSENSGADE 9 2.TV
1368 KØBENHAVN K

396
ROSENKILDE HANS
PRÆSTEHUSENE 16
2620 ALBERTSLUND.

ANNONCE ANNONCE ANNONCE ANNONCE ANNONCE ANNONCE ANNONCE ANNONCE

Et brugt anlæg, NCR 7200, sælges for største beløb over 366.00 kr. inden en uge !! Det består af en kombineret skærm/datamaskinen (8080), selvstændigt alfanumerisk tastatur og en kassettestation, der fjernbetjenes. Maskinen er brugt til dataopsamling (recept på apoteket). Dertil yderligere 10 datakassettebånd - verificerede. Henvend dig til Asbjørn Lind. 02 91 71 82 (20-21)

SØGES SØGES SØGES SØGES SØGES SØGES SØGES SØGES SØGES SØGES

Hvis du er blevet træt af dit anlæg (Nascom 2), er der flere, der henvender sig og spøger, om vi har noget til salg !! I øjeblikket venter der helt sikkert tre personer på maskiner, der er rimelige i pris (nedslag på mellem 33 og 50% på nypris uden software). Sæt en annonce i bladet.

A.

nyhed nyhed nyhed nyhed nyhed nyhed nyhed nyhed nyhed nyhed nyhed

Piezodan ApS (01 86 12 17) er blevet forhandler af det kendte engelske software fra Level 9 (se bl.a. udvidelse til basic). Firmaet har fået forskellige spil hjem og venter at udvide sortimentet. Fra samme firma kan leveres et dansk fremstillet lydkort til 735.00 kr. og et REAL-time ur til 355.00 kr. med mulighed for batteriback-up (sættes til PIO'en).

A.

SØNDAGSMØDET d. 10. APRIL 1983. Kl. 13.oo.
Pædagogisk Central. Rustenborgvej 1. Lyngby.

Anders Heilsberg,Polydata,viser den nye Rainbow 100 fra Digital Equipment Corporation,som er et bidrag til mængden af person/kontor-datamater,den har 2 CPU-er,Z80 - 8088,256 K.Ram,2 x 400 KB floppydisks og CP/M.
Anders vil desuden vise software som COMPAS 2,Wordstar,dBase II.
Desuden vil foreningens programbibliotek blive præsenteret med de sidste nyheder til såvel CP/M som NASSYS.
Der vil være mulighed for at få brændt E-PROMMER (2708,2716,TI 2516 2732)----og det vil blive muligt at få udskrevet egne programmer.
Efter alt dette er der almindelig snak og mumlen.

KAFFE _ ØL _ VAND kan købes.