JERNINDUSTRIENS FORLAG

Mikrodatamat-teknik

Hardware

1983

Instruktioner
Opgaver

Jern- og Metalindustrien

Best. nr. 4043

Forord

Lærebogen anvendes i undervisningen på kurset "Mikrodatamatteknik, trin 1-Hardware", kursuskode 5818.

Metalindustriens Efteruddannelsesudvalg har foranlediget lærebogen optrykt.

Lærebogen er tilrettelagt af faglærere fra Sønderborg tekniske skole i samarbejde med Jernindustriens Forlag.

Instruktionsdelen omfatter teoriinstruktioner om emnerne:

Mikrocomputorens:
- Blokdiagram
- Processorsektion
- Instruktionssæt
- Lagersektion
- Interruptsystem
- Kommunikation

Parallel in/out
Dokumentation

Opgavedelen omfatter teoriopgaver om emnerne:

Måleøvelser
Programmeringsopgaver

Til brug ved undervisningen har lærebogen fortløbende sidenumre nederst på siderne.

Forlaget modtager gerne forslag til ændringer og rettelser fra lærere, elever og andre interesserede.

Metalindustriens Efteruddannelsesudvalg og Jernindustriens Forlag takker Sønderborg tekniske skole og faglærerne for medvirken ved tilrettelæggelsen af denne lærebog.

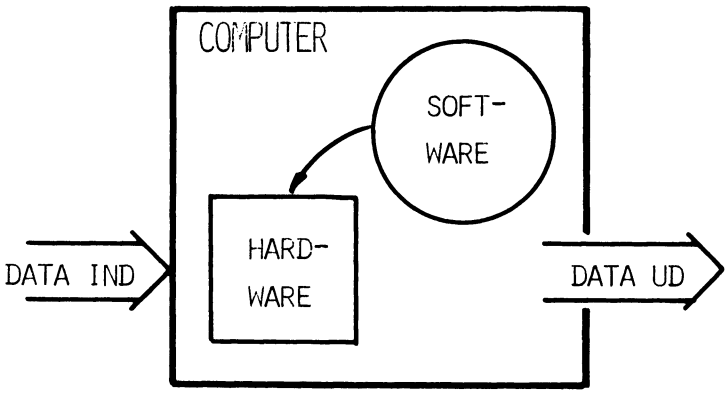
København, juni 1983

JERNINDUSTRIENS FORLAG

Nr.	TEORIINSTRUKTIONER	Side

Nr.	TEORIØVELSER	Side

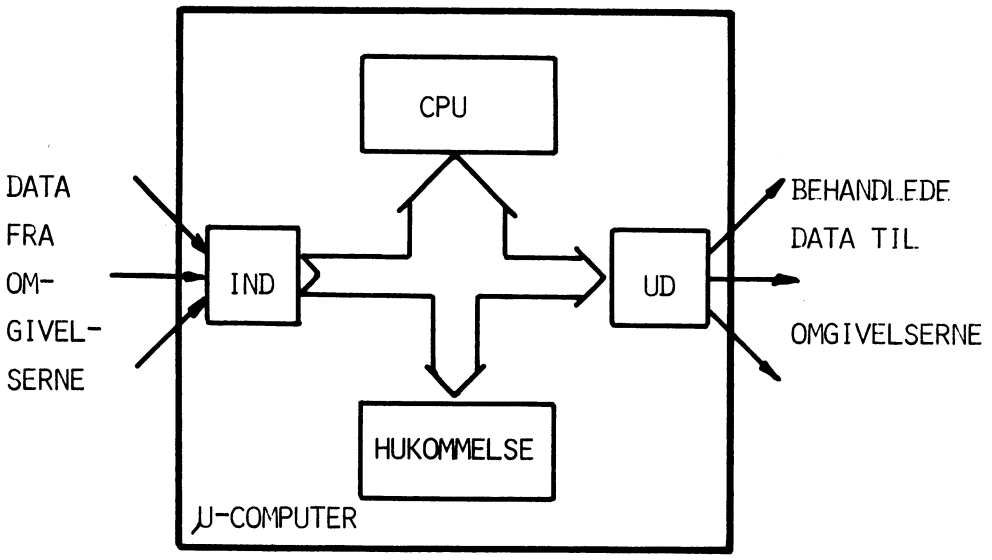
Nr.	TEORIOPGAVER	Side



Hardware/software.

Computeren kan opfattes som bestående af to hovedbestanddele, hardware og software. Ved hardware forstås selve kredsløbet med dets IC'r, ledninger, kontakter o.s.v.. Softwaren er det program der får hardwaren til at udføre den aktuelle funktion. Medens det normalt vil være forbundet med vanskeligheder at ændre hardwaren, vil en ændring af softwaren nemt kunne udføres, og da det er softwaren der bestemmer hardwarens funktion, vil det ses, at en computer vil kunne tilpasses til at løse de forskelligste opgaver.

Af ovenstående fremgår også, at hvor arbejdet med et nyt kredsløb tidligere hovedsaglig lå i værkstedet, foregår en betydelig del af arbejdet nu ved skrivebordet med udvikling af programmer, så meget mere som man ofte køber hardwaren færdig i form af standard kredsløbskort.



CPU'en.

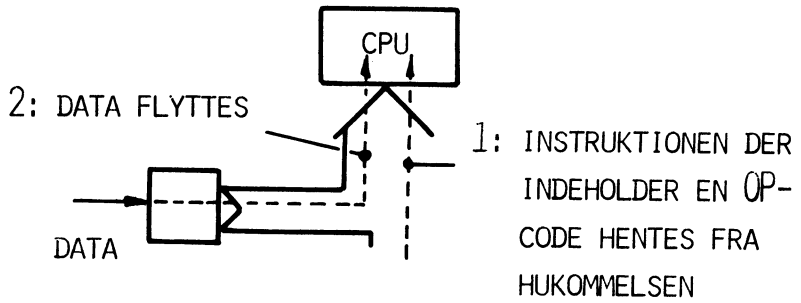
En µ-computer kan opdeles i fire hovedblokke som vist på blokdiagrammet.

Den centrale af disse blokke er microprocessoren (µP) også kaldet CPU'n, hvilket betyder Central Proces Unit.

Det er i CPU'n alle beregninger og beslutninger foregår, ligeledes kontrolleres input-og output kredsene samt hukommelsen af CPU'n.

Op-coder.

CPU'n er "født" med evnen til at udføre et begrænset antal små operationer, der set enkeltvis er uden mening, men som anvendt i rækkefølge kan bringes til at danne en hel proces.



Et eksempel på en af CPU'ens små operationer er at flytte data fra inputkredsen til CPU'n. For at få udført denne dataflytning skal CPU'n påvirkes med et tal, en såkaldt operationskode - (OP-code).

En CPU har alt efter fabrikat omkring 5o OP-coder som tilsammen danner et OP-codesæt.

Hukommelsen.

Hukommelsen har to opgaver i en µ-computer:

1. Den rummer de nødvendige OP-coder til at styre CPU'ens funktion. Denne samling af OP-coder kaldes programmet.

2. Den skal lagre mellemresultater og data.

Langtidshukommelse/programhukommelse.

Til at løse første opgave anvendes en langtidshukommelse, hvis indhold kun vanskeligt kan ændres.

Til gengæld kan apparatet slukkes uden at hukommelsens indhold går tabt. Denne hukommelse kaldes også programhukommelsen.

Korttidshukommelse/Data hukommelse.

Til opgave to anvendes en hukommelse, hvis indhold let kan ændres, men som stiller det krav, at der hele tiden skal være tilsluttet forsyningsspænding. Afbrydes denne blot et øjeblik, går indholdet tabt. Korttidshukommelsen kan findes i CPU'en, og kaldes da ofte "Scratchpad memory", der betyder "Notesblok hukommelse".

CPU'en contra hukommelsen.

Ved opstart af en µ-computer begynder CPU'en med at "bede" langtidshukommelsen om den første OP-code. Denne sendes til CPU'en, der straks eksekverer den. Derefter "beder" CPU'en om den anden OP-code, får den og udfører den, o.s.v.
Som det fremgår er CPU'en nok den der "gør arbejdet", men det er det i hukommelsen, lagrede program der fortæller CPU'en, hvad den skal foretage sig.

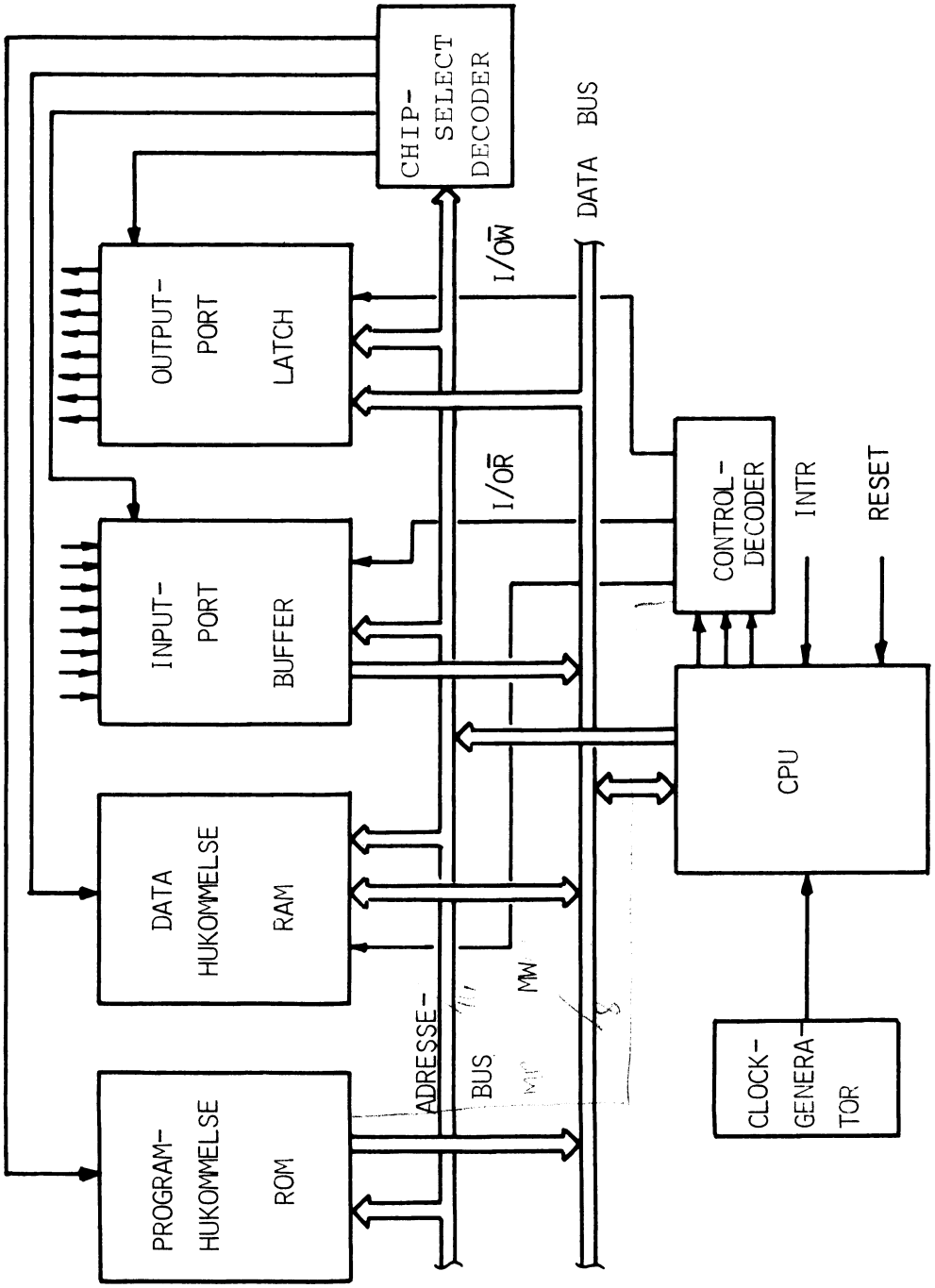
INPUT/OUTPUT.

INPUT- og OUTPUT-kredsene har til opgave at danne kontaktflade (interface) mellem µ-computeren og omgivelserne. De styres af CPU'en til at åbne og lukke når der skal overføres data fra eller til µ-computeren.

Systemdiagram.

Det herunder viste systemdiagram indeholder programhukommelse (ROM), datahukommelse (RAM), CPU med taktgenerator, INPUT-port og OUTPUT-port.

Derudover er de nødvendige kommunikationslinier mellem ovennævnte kredsløb vist.



Hukommelsen.

Som tidligere nævnt kan CPU'en ikke udføre meget selv, medmindre den får de nødvendige ordrer.

Disse ordrer, kaldet programmet er lagret i programhukommelsen.

Krav.

Programhukommelsen skal kunne holde sit indhold i lang tid, også uden der er spænding tilsluttet apparatet, til gengæld er det ikke nødvendigt, at kunne ændre indholdet.

ROM.

Disse egenskaber findes i en ROM. ROM er en forkortelse af Read Only Memory, det vil sige en hukommelse, hvorfra der kun kan læses, d.v.s. hentes information.

ROM'ens indhold fastlægges allerede ved fabrikationen, hvorfor der skal anvendes et stort antal af samme type, hvis det skal være rentabelt.

PROM.

PROM'en er en fætter til ROM'en. Det tilføjede P står for Programable, d.v.s. programmerbar af brugeren. Denne kan ved hjælp af et programmeringsudstyr fremstille et enkelt styk til en foreliggende opgave, men som ved ROM'en kan indholdet ikke ændres når først programmeringen er sket.

EPROM.

Sidste skud på stammen er EPROM'en. E står for Erasable, altså sletbar. EPROM'en kan programmeres af brugeren på samme vis som PROM'en, men hvor det er en mekanisk ændring, der finder sted i PROM'en, er det blot en opladning af nogle kondensatorer, der sker i EPROM'en.

Finder brugeren på et senere tidspunkt ud af at programmet skal ændres, kan EPROM'ens indhold slettes ved af belyse den med kraftigt ultraviolet lys gennem vinduet.

Derefter kan der ske en ny programmering.

RAM.

Til lagring af DATA, det vil sige de signaler, som µ-computeren behandler, skal anvendes en hukommelse hvor man til enhver til kan lagre data, og hente dem igen.

Den såkaldte RAM har denne egenskab. RAM står for Random Access Memory, hvilket betyder "hukommelse med tilfældig adgang", altså en hukommelse, man kan komme med data til, og få dem fra på et vilkårligt sted og tidspunkt. RAM lageret benævnes også ofte read write memory, læse-skrive, hukommelse eller scratch-pad memory, notesblok hukommelse.

EAROM

Electrical Alterable ROM, er en "krydsning" mellem en ROM og en RAM, idet den kan slettes elektrisk og genprogrammeres medens den sidder i opstillingen.

OUTPUT-port.

OUTPUT-porten kan betragtes som en del af RAM-lageret, dog således at der kun kan gå data fra CPU'en til porten. Til gengæld vil disse data være tilgængelige for eksterne anvendelser.

INPUT-port.

INPUT-porten er i princippet kun 8 parallelle kontakter der slutter når porten adresseres. Derved bliver der forbindelse mellem data-indgangene og data-bussen og dermed CPU'en.

Takt-generator.

Taktgeneratoren (clockgeneratoren) leverer styreimpulser til CPU'en, og bestemmer dermed den tid det tager at udføre en instruktion. Typisk er frekvensen mellem 2oo KHz og 5 MHz. Den nominelle frekvens er givet af den anvendte CPU.

Busser.

Som vist på systemdiagrammet kommunikerer de enkelte enheder via nogle fælles ledningsbundter. Et ledningsbundt som fører signaler med samme mission kaldes en "bus".
Adressebussen består af typisk 16 og databussen af 8 parallelle ledninger. I steder for at tegne alle disse ledninger i diagrammet bruges ofte at tegne en enkelt streg og angive antallet af ledere, eller tegne et bredt bånd og med klar tekst angive arten og antallet af ledninger.

Med pile kan man angive retningen af signalerne.

Adressebussen.

Når CPU'en under eksekveringen af et program skal have tilført en ny opcode fra ROM'en eller data fra RAM'en eller inputporten, sender den via adressebussen et bitmønster ud, hvorved den ønskede memorylokation adresseres.

Som vist i systemdiagrammet bliver adressen tilført parallelt til både ROM, RAM, INPUT- og OUTPUTPORT, men kun en af blokkene reagerer, da ingen adresse kan (må) gå igen hos flere blokke, hvilket varetages af chip-select decoderen.

Databussen.

Via databussen overføres data! Data kan være:

Opcoder fra ROM'en til CPU'en Faste data fra ROM'en til CPU'en Variable data fra I-port til CPU'en Variable data fra RAM'en til CPU'en	MR eller IOR LOW
Data fra CPU'en til RAM'en Data fra CPU'en til O-porten	MW eller IOW LOW

Som det kan ses kan flere enheder sende signaler over databussen. For at undgå forstyrrelser er det nødvendigt at indrette kredsløbet således, at der kun er en af gangen, der sender, medens alle andre enten lytter eller er koblet ud. Den sidste mulighed betegnes "den tredie stilling", hvorfor man kalder denne form for logik "3-state logik", eller "TRI-STATE logik".

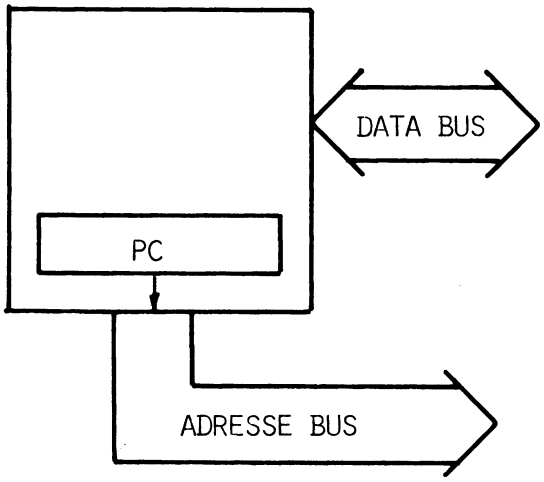
Kontrolbussen.

Kontrolbussen omfatter bl.a. clockpulser og læse/skrive kontrollene, MR. MW, IOR og IOW.

Læse/skrive signalerne kommer fra CPU'en og har til opgave at styre RAM-memoryen og I/O portene.

CPU. - Simpel blokdiagram.

CPU'en har til opgave at styre afviklingen og udføringen af et program. Til styringen har CPU'en en programtæller (PC). Denne tæller holder styr på hvor langt CPU'en er nået i et program. Hver gang programtælleren anvendes, forhøjes den automatisk med een. Ved hjælp af specielle instruktioner og/eller hardware signaler kan programtællerens værdi blive udskiftet med en anden værdi. Efter reset indholder PC værdien øøøø. PC'en er normalt den der leverer signalet til adressebussen, d.v.s. at programtællerens værdi er lig med en adresse. Adressebussen er der hvor CPU'en fortæller hvorfra den vil læse (Read) eller skrive (Write) data, der transporteres via databussen.

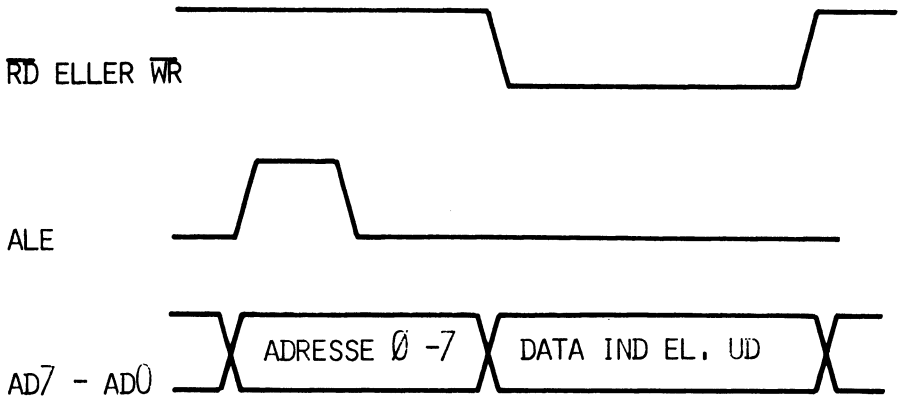


RW, WR.

For at fortælle omgivelserne om CPU'en vil læse eller skrive data, har den nogle kontrol signaler, der ikke kun fortæller retningen af data på databussen, men angiver også hvornår data henholdsvis er eller skal være stabile. Disse styreledninger hedder RD (read) og WR (write).

Multiplexning.

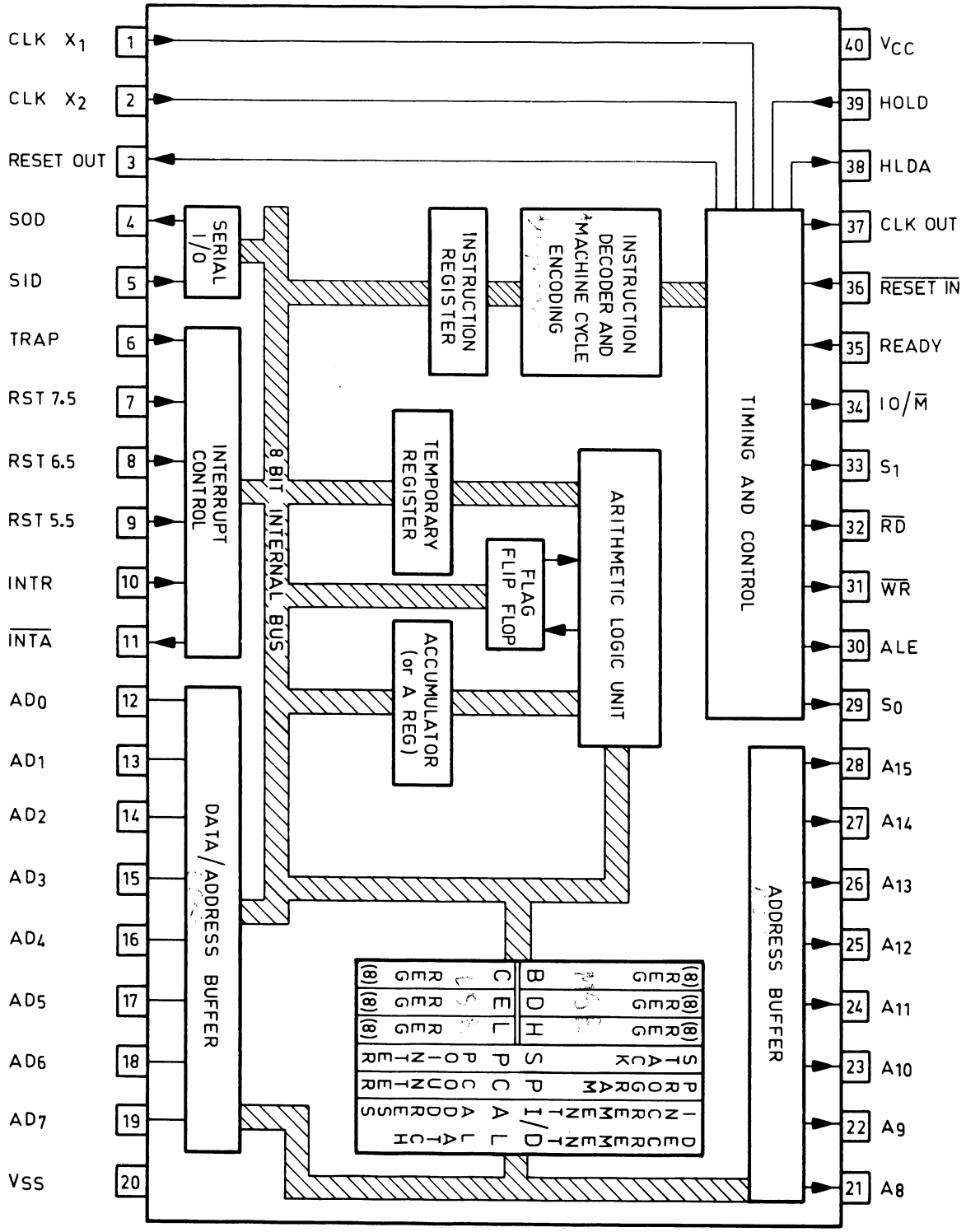
I 8085 er PC og adressebus 16 bit bred, databussen er 8 bit bred. De 8 mindst betydende adressebit og data bit'ene er ført ud på de samme 8 ben på IC'en og bliver tidsmultiplexet ud. For at fortælle hvornår disse 8 ben har adresse information, findes der et ekstra ben, der fortæller hvornår adresseinformationen er stabil, dette ben hedder ALE. Data angives stabile med $\overline{WR}$ signalet og skal være på databussen, når $\overline{RD}$ signalet indikerer det. D.v.s. ALE fortæller hvornår de 8 laveste adressebit er på adresse/databussen og $\overline{RD}$/$\overline{WR}$ angiver, hvornår CPU'en har eller ønsker at få data på adresse/databussen.



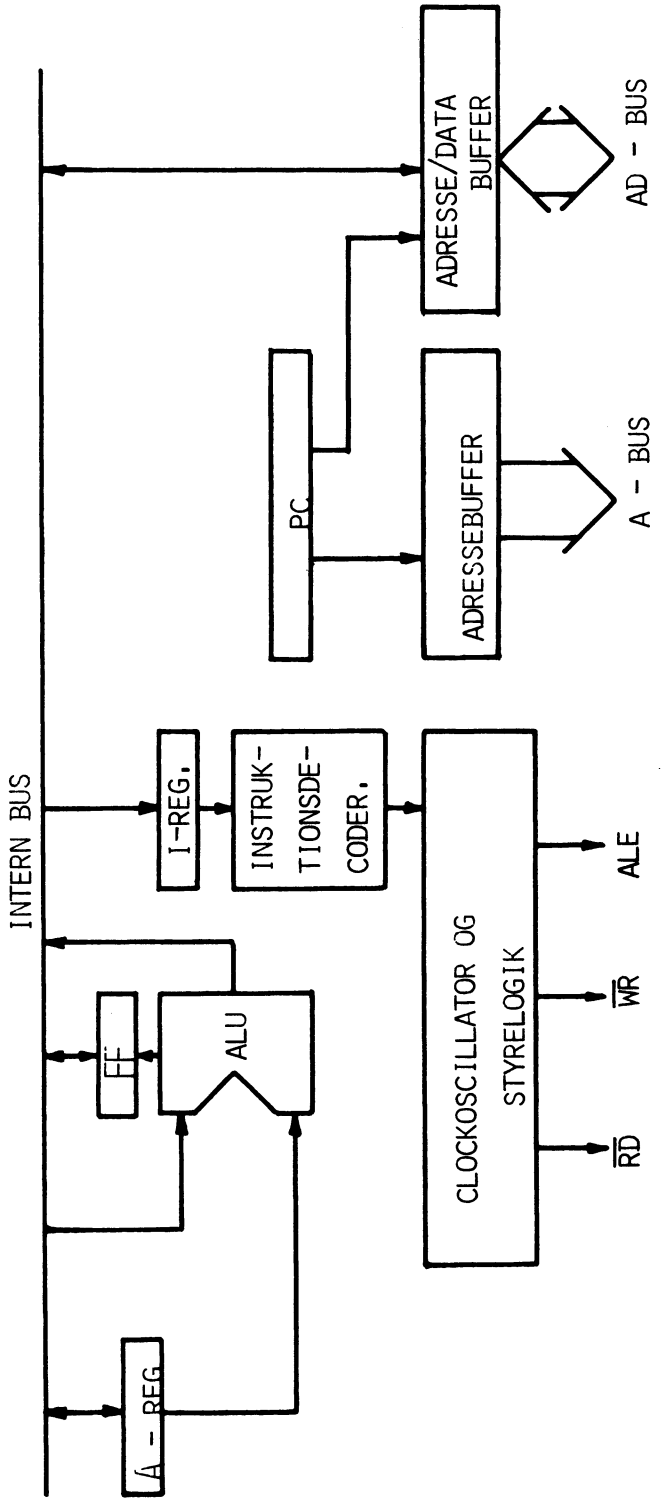
Instruktionsfetch

Første data som CPU'en henter (fetsch) efter reset anbringer CPU'en i et instruktionsregister, hvor CPU'en vil tyde bitmønsteret v.h.a. instruktionsdekoderen. Bitmønsteret fortæller CPU'en om der er tale om en 1 byte, 2 bytes eller 3 bytes instruktion. CPU'en starter nu det til bitmønsteret hørende program.

INTEL 8085 LAYOUT



SIMPELT BLOKDIAGRAM AF CPU, 8085.



Microprogram.

Microprogrammet er det program der internt i CPU'en, får den til at afvikle hver instruktion som CPU'en kan udføre.

Når CPU'en har hentet og udført en instruktion, vil den igen hente et nyt bitmønster ind i instruktionsregisteret. Den første byte i en instruktion kaldes opcoden. Opcoden er den byte, der skal anbringes i instruktionsregisteret.

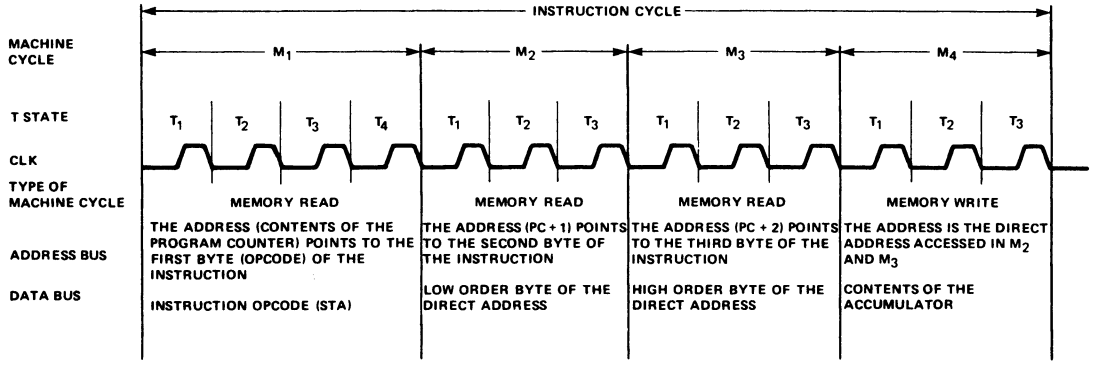
CPU'ens arbejdshastighed og timing er styret af en oscillator. For alle CPU'er findes der en maksimal arbejdsfrekvens, ligeledes findes for en hel del CPU'er også en minimum frekvens.

Interne Reg.

Når data skal andet end flyttes, anvender CPU'en en ALU (arithmetic logic unit),hertil hører også et arbejdsregister der kaldes akkumulatoren (A-reg.) samt nogle FF'ere, der fortæller noget om det resultat, der sidst er beregnet af ALU'en.

Instruktionscyclus.

En instruktioncyclus består af hentning (Fetsch) af en instruktion og udføring (execute) af instruktionen. Instruktionscyclen opdeles i nogle (1 til 5) maskincycler. Antallet af maskincycler svarer til det antal gange CPU'en har brug for at hente eller anbringe data i den eksterne memory. Hver maskincyclus kan igen deles ned i fra 3 til 6 state (trin), der hver især svarer til en tid på en intern clockperiode. For en 8085 er den interne clockfrekvens lig med halvdelen af den frekvens der tilsluttes x_1 og x_0.
Benet CLK out på 8085 er lig med CPU'ens interne clockfrekvens, blot forsinket nogle nanosekunder.



Maskincyclus.

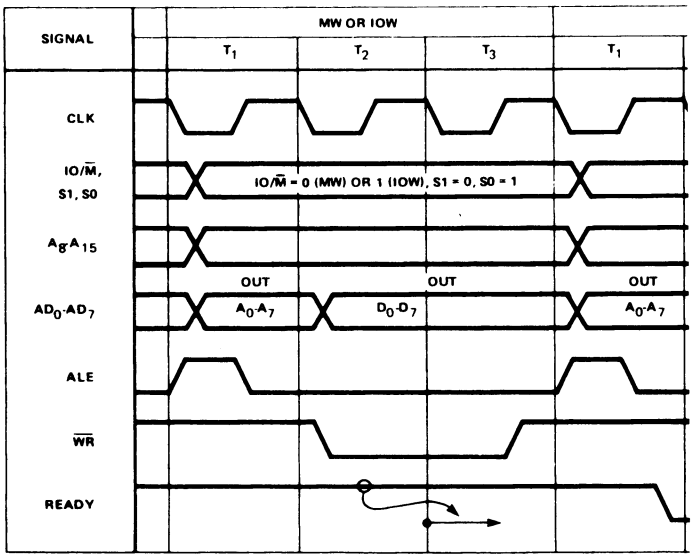
Til hver instruktion hører et bestemt antal og type maskincycler. Med benene S_1, S_0 og IO/M angiver 8085 hvilken maskincyclus type der er igang med at bearbejde. Der er 7 forskellige maskincyclus typer, der angives på følgende måde:

IO/$\overline{M}$	S_1	S_0	Maskin cycle type
T.S.	Ø	Ø	Halt
Ø	Ø	1	Memory write
Ø	1	Ø	Memory read
Ø	1	1	Opcode fetch
1	Ø	1	I/O Write
1	1	Ø	I/O Read
1	1	1	Intr. Acknowledge

T.S. = Tri state

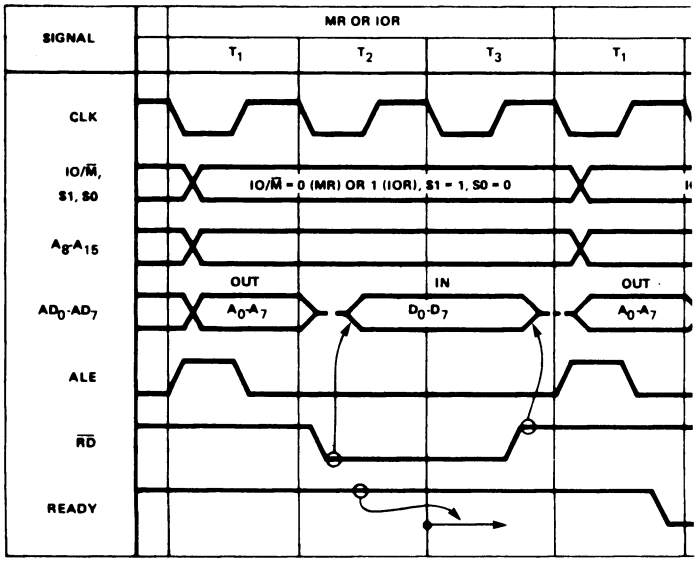
MW, IOW.

En Memory eller IO write (MW, IOW) maskin cycle består af 3 state. I første state udsender CPU'en den adresse, hvor data skal anbringes. I state 2 udsendes data og WR ledningen går low. I state 3 udsendes data og WR ledningen går high. Under alle tre state udsendes S_1= Ø, $S_Ø$=1 samt IO/$\overline{M}$ informationen der er "1" ved IO og "O" ved M.



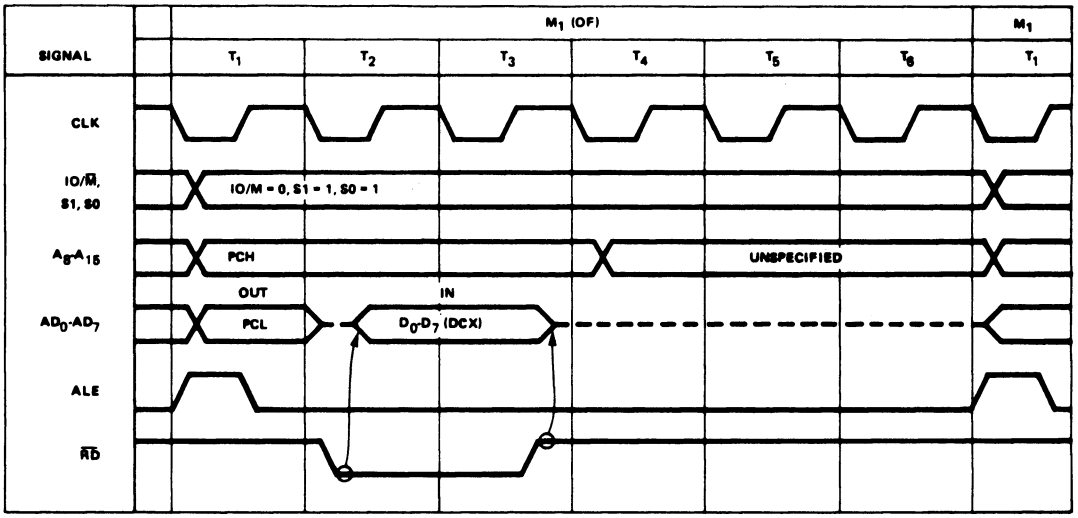
MR, IOR.

En Memory eller IO read (MR, IOR) har også 3 state men afviger fra MW ved at det er RD og ikke WR der går low under T_2 og high igen under T_3. Ligeledes er signalerne på $S_\emptyset$ og S_1 ændret til $S_\emptyset = \emptyset$ og $S_1 = 1$.

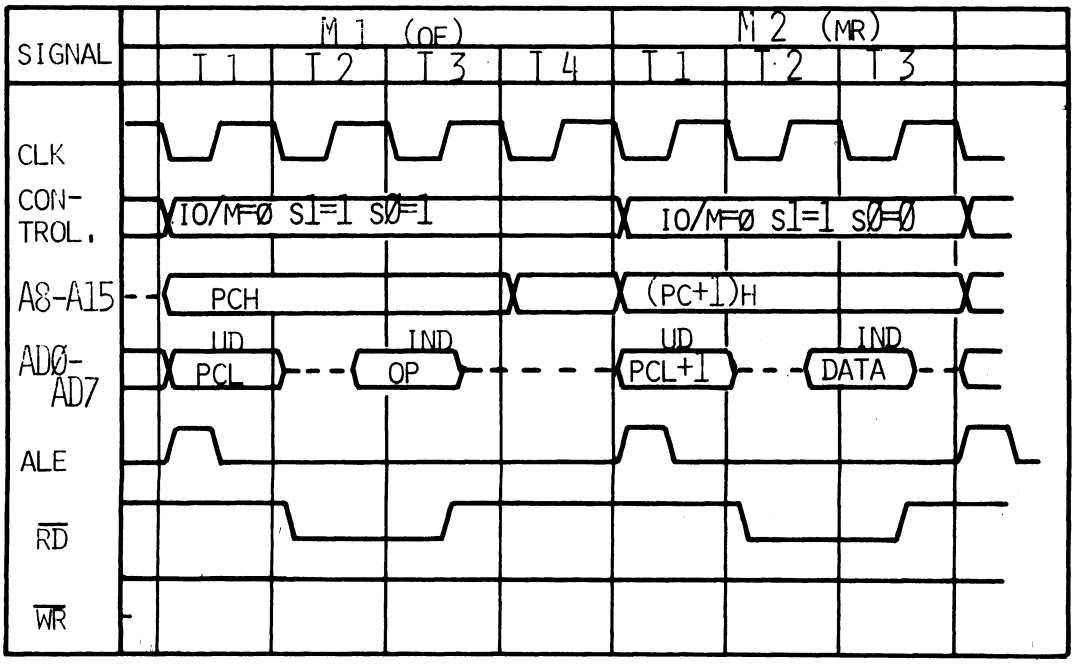


OF.

I maskincyclen opcode fetch (OF) anvendes der fra 4 til 6 state, de første tre state anvendes til at læse opcoden, det efterfølgende state anvendes til at tyde opcoden, hvorunder CPU'ens microprogram afgør om der er brug for yderligere to state for at udføre instruktionen eller en del af instruktionen efterfulgt af andre maskincycler.

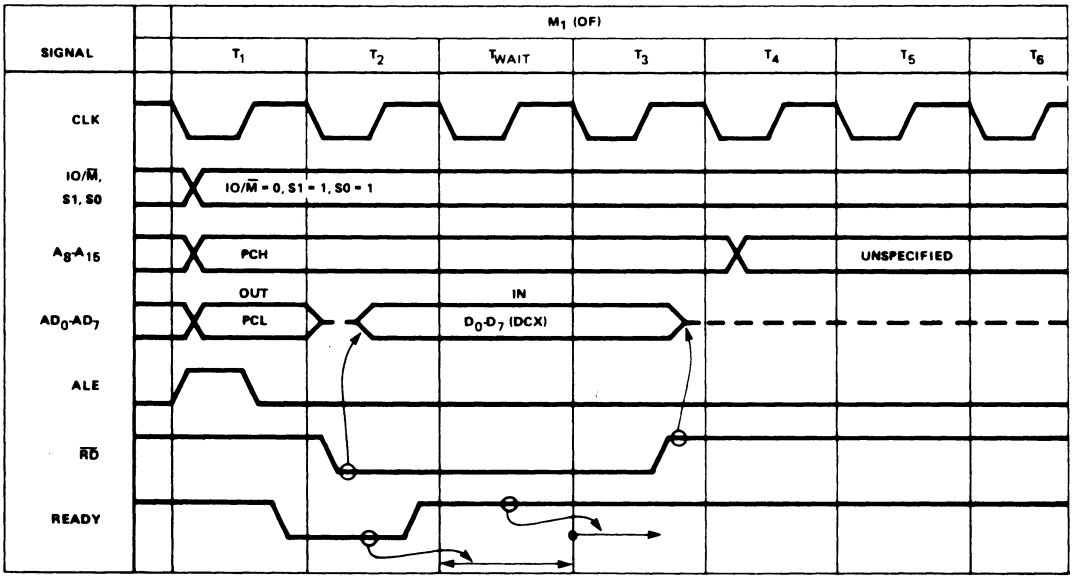


En instruktion som ANI DATA vil bestå af to maskincycler, en OF og en MR.

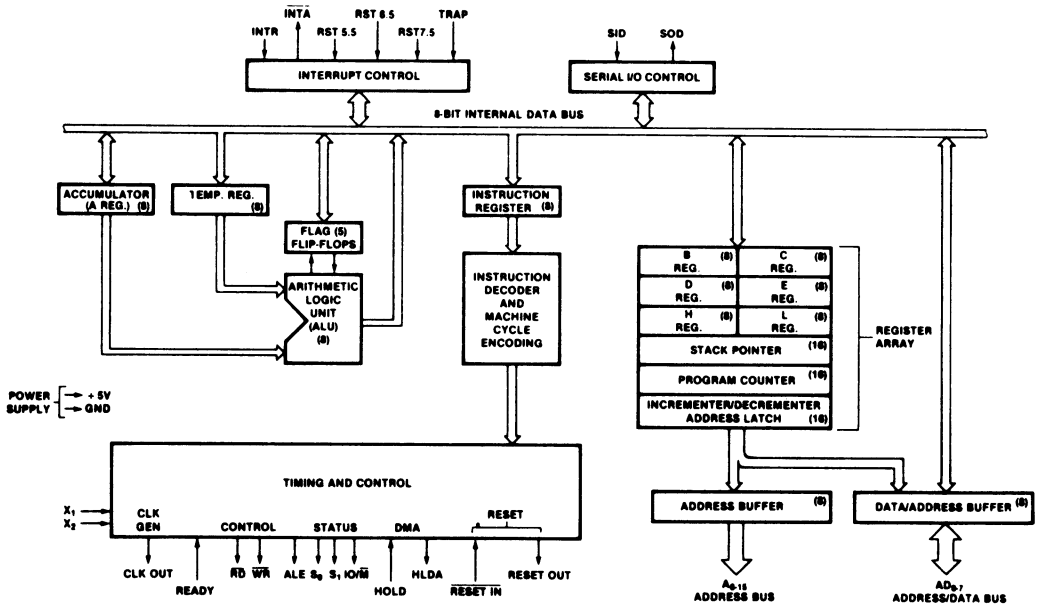


Twait.

Når CPU'en skal arbejde sammen med langsomme memory eller portkredsløb, kan der mellem T_2 og T_3 indskydes nogle ventetrin (Twait). For at T_3 indledes skal READY benet være high inden den positive clock out flanke, der går forud for indledningen af T_3.



Et blokdiagram af 8085's struktur ser således ud:



CPU'en består af tolv adresserbare 8 bit registre. Fire af disse arbejder altid som to 16 bit registre. Det er PC (Program Counter) og SP (Stack Pointer). Seks andre registre kan valgfrit for programmøren anvendes som 8 bit eller 16 bits registerpar (rp), disse registre hedder som 8 bits registre B, C, D, E, H og L og som 16 bits registre BC, DE og HL. Disse registre anvendes universelt. HL registret kan dog bruges som datapointer, d.v.s. HL kan indeholde et 16 bit mønster, der er et udtryk for den adresse, hvor data skal hentes eller placeres. Et andet 8 bit register, der anvendes som resultatregister ved aritmetiske, logiske og in/out instruktioner er akkumulatoren (A-reg.). Det sidste 8 bit register er flag registeret, i dette register anvendes kun de fem af bit'ene. Ved aritmetiske og logiske funktioner påvirkes bit'ene, idet hver bit for sig angiver noget om resultatet.

Interrupt.

I en 8085 er der to forskellige måder, hvormed CPU'en kan interruptes. Den ene er ved anvendelse af INTR og $\overline{\text{INTA}}$ benene. Når et eksternt kredsløb ønsker at påkalde sig opmærksomheden, genererer det et logisk "1" på INTR benet. Et "1" på dette ben vil få CPU'en til at stoppe PC, hvis processoren er enablet (gjort mulig) for interrupt (EI). Derefter vil der indledes en interrupt acknowledge (erkendelse) maskincyclus. Dette ligner en OF maskin cyclus blot med den forskel, at benet $\overline{\text{INTA}}$ anvendes i stedet for $\overline{\text{RD}}$ benet. D.v.s. at der nu skal genereres en opcode på databussen timet v.h.a. af $\overline{\text{INTA}}$ signalet, denne opcode kan være RST instruktion bestående af 1 byte eller en CALL bestående af 3 byte, der skal genereres i takt med $\overline{\text{INTA}}$ signalet. Med RST instruktionerne starter CPU'en på forud bestemte startadresser, der er indeholdt i opcoden.

Instruktion	Startadresse
RST Ø	ØØØØ H
RST 1	ØØØ8 H
RST 2	ØØ1Ø H
RST 3	ØØ18 H
RST 4	ØØ2Ø H
RST 5	ØØ28 H
RST 6	ØØ3Ø H
RST 7	ØØ38 H

Med CALL instruktion kan der startes på enhver adresse i systemet.

Den anden måde at generere interrupt er med TRAP, RST 5.5, RST 6.5, RST 7.5 benene. Ved signal på et af disse ben vil CPU'en interruptes og derefter selv generere en RST instruktion uden hjælp af eksterne kredsløb. Forskellen på disse ben er, hvor CPU'en vil starte restart rutinen.

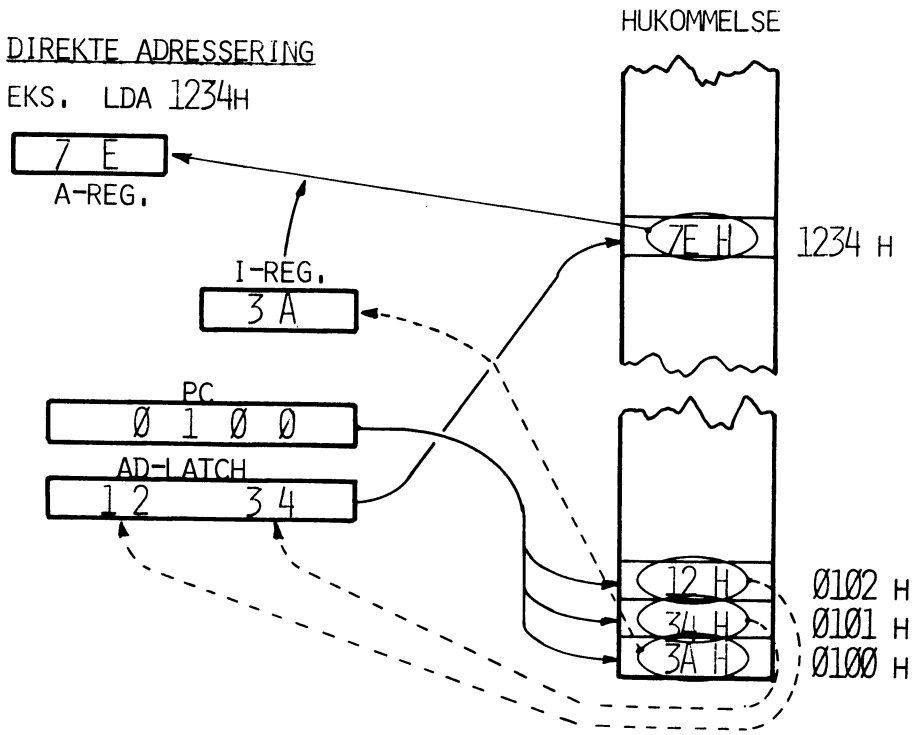
Ben	Startadresse
TRAP	ØØ24 H
RST 5.5	ØØ2C H
RST 6.5	ØØ34 H
RST 7.5	ØØ3C H

Der er ligeledes også den forskel at TRAP ikke kan blokeres med DI eller SIM instruktionerne.

Adresseringsmåder.

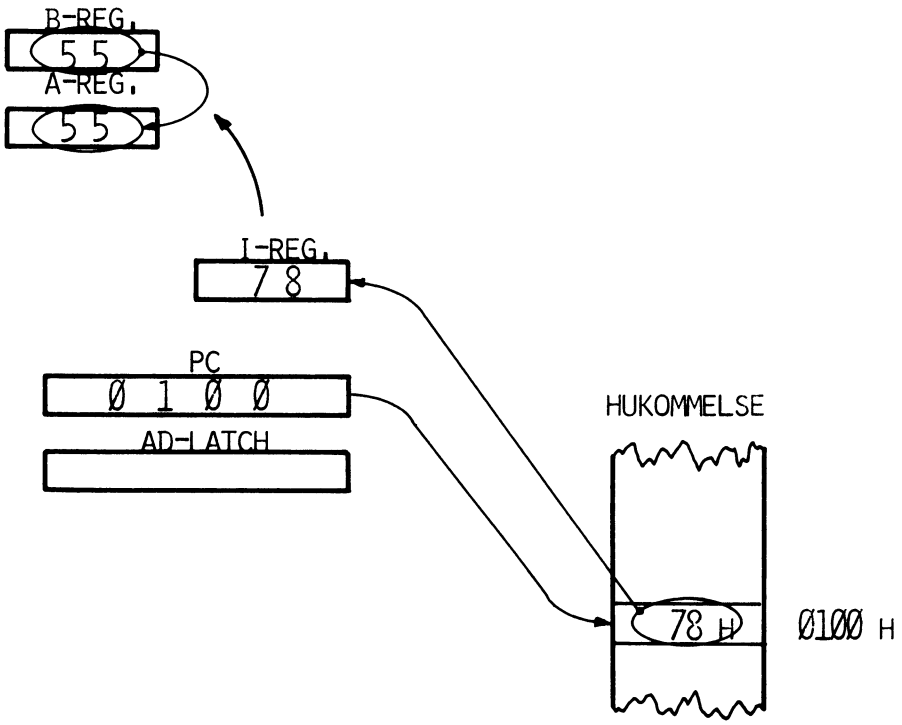
8085'en har fire måder til adressering af data i lageret eller i registrene:

Ved DIREKTE ADRESSERING bruges 2 byte til selve adressen, således at byte 2 er de 8 mindst betydende bits af adressen og byte 3 de 8 mest betydende bits.



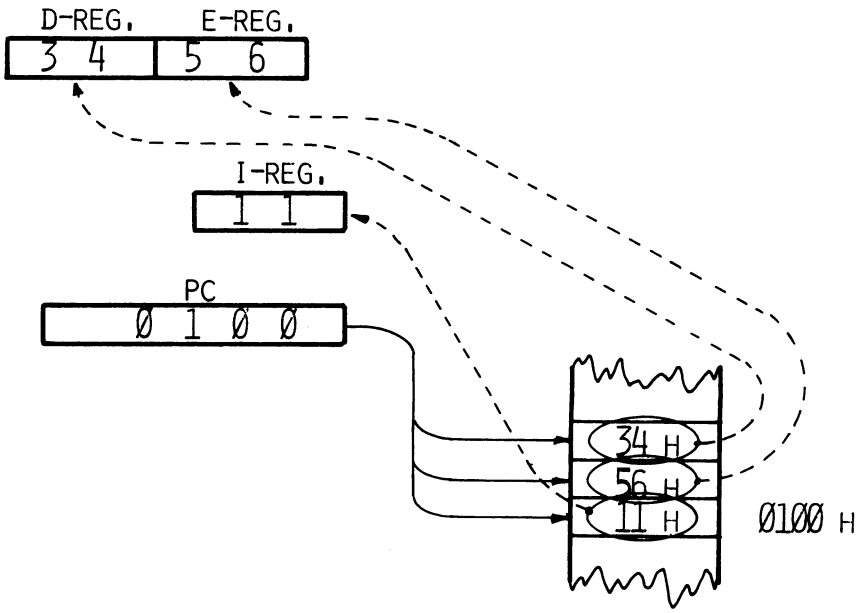
Ved REGISTER-ADRESSERING angiver selve op-koden det register, der skal benyttes til operationen.

EKS. MOW A,B



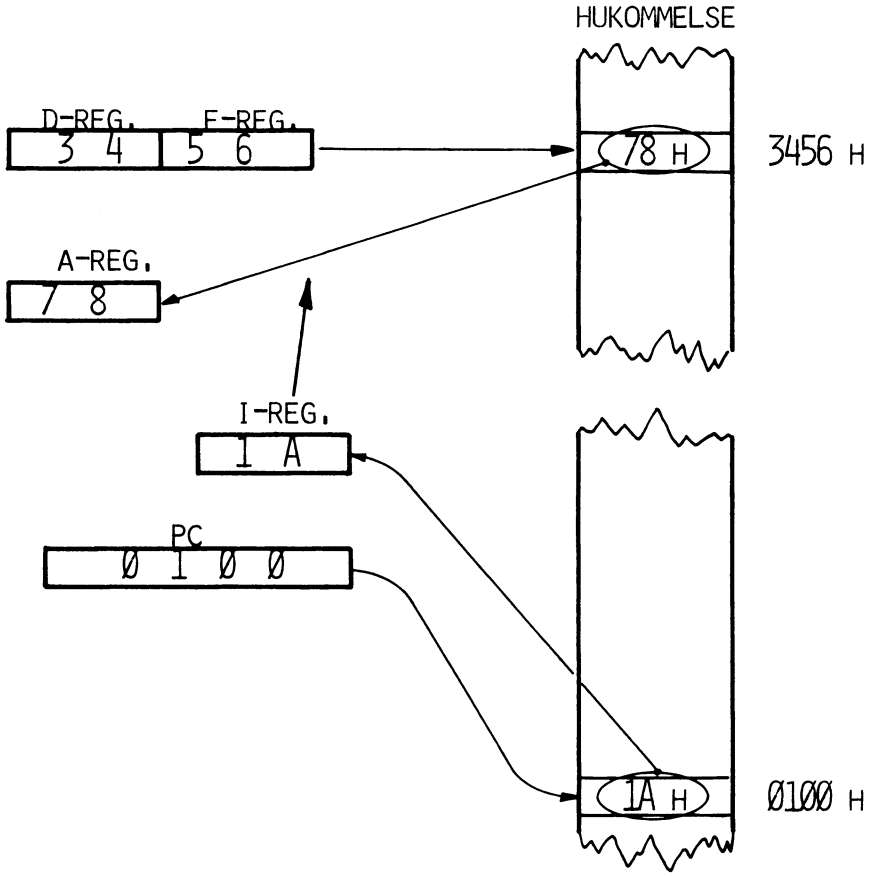
Ved "IMMEDIATE" ADRESSERING indeholder instruktionen også data. Instruktionslængden er 2 eller 3 bytes.

EKS. LXI D,3456H



Ved REGISTER INDIREKTE ADRESSERING angiver selve op-koden det registerpar, f.eks. (H.L.), som indeholder adressen for data, der skal benyttes ved operationen. Register H indeholder den mest betydende byte af adressen.

EKS. LDAX D



Når et program køres, udføres instruktionerne i den rækkefølge, de står i programmet, med mindre dette ændres på grund af en hop-instruktion eller et interrupt-signal. En hop-instruktion angiver den adresse, der skal hoppes til på to måder, nemlig:

Ved DIREKTE HOP-ADRESSERING bruges 2 bytes til adresse, som ved andre direkte adresserende instruktioner.

Ved REGISTER INDIREKTE HOP-ADRESSERING angiver op-koden det registerpar, der indeholder hop-adressen.

RST-instruktionen er en speciel hop-instruktion, der anvendes ved interrupt af programmet. Instruktionen er på 8 bits, hvoraf tre bruges som adressefelt. Dette felts værdi ganges automatisk med 8, og programmet udføres fra den adresse, der svarer til dette tal.

STATUS - FLAG.

Der er fem resultatvisende flag, der virker i forbindelse med udførelsen af instruktionerne i 8085'en. De kaldes henholdsvis ZERO, SIGN, PARITY, CARRY og AUXILIARY CARRY. De kan indeholde "1" eller "0" afhængig af, om de er i stilling SET eller RESET.

Når en instruktion påvirker flags, gælder følgende regler, medmindre andet er specificeret i specielle tilfælde.

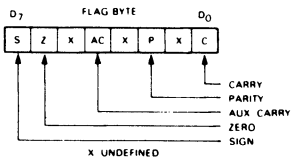
ZERO: Hvis en instruktion giver resultatet 0, sættes ZERO-FF'en, i modsat fald resættes den.

SIGN: Får samme værdi som den mest betydende bit i akkumulatoren.

PARITY: Får værdien 1 ved lige paritet for det tal, der står i akkumulatoren.

CARRY: Hvis en instruktion giver en mente (ved addition) eller en låner (ved subtraktion eller sammenligning) sættes Carry-FF'en i stilling 1, 1 modsat fald i stilling 0.

AUXILIARY CARRY: Virker som Carry for de 4 mindst betydende bit. (Anvendes oftest ved BCD-regning).



SYMBOLER OG FORKORTELSER.

I det efterfølgende forklares symboler og forkortelser, der anvendes ved beskrivelse af 8085'erers instruktionssæt.

Symbol	Betydning
Akkumulator	Register A
Adr.	16-bits adresse
Data	8-bits data størrelse
Data 16	16-bit data størrelse
Byte 2	Byte 2 i en instruktion
Byte 3	Byte 3 i en instruktion
Port	8-bit adresse for en I/O komponent
r, r1, r2	Et af registrene A,B,C,D,E,H,L

DDD, SSS Et bitmønster der henviser til et af registrene A, B. C. D. E. H, L idet

DDD står for modtagerregister (destination)

SSS står for det register, der hentes data fra (source).

DDD eller SSS	REGISTER NAVN
111	A
000	B
001	C
010	D
011	E
100	H
101	L

rp Står for et af følgende registerpar, idet det mest betydende er anført forrest. BC, DE, HL og SP, der står for stackpointer registeret (16 bits).

RP Et bit-mønster, der henviser til et af registerparene B, D, H, SP:

RP	Registerpar
00	B-C
01	D-E
10	H-L
11	SP

rh Det første register i et registerpar, f.eks. B,D,H.

rl Det andet register i et registerpar, f.eks. C,E,L.

PC 16-bits program counter register. For den mest betydende del af PC anvendes PCH (high), og for den mindst betydende del PCL (low).

SP 16-bit stack-pointer register. SPH pg SPL anvendes her ligesom ved PC.

r_m Bit m i et register f.eks. den 5'te bit noteres r_5. (mest betydende længst til venstre).

Z,S,P,
CY,AC Flagene: ZERO, ZIGN, PERITY, CARRY og AUX.CARRY.

( ) Noteres f.eks. (H), menes indholdet af register H.

(( )) Noteres f.eks. ((H)), menes indholdet i lageret på den adresse H peger på.

A Et tal i RST instruktionen fra o-7 (Dec.).

AAA Et bitmønster på 3 bit fra OOO - lll.

DATA TRANSPORT-GRUPPE.

Instruktionerne i denne gruppe transporterer data til og fra registre og lager.
FLAG-FF'erne påvirkes IKKE af instruktionerne i denne gruppe.

MOV r1, r2 (Move register)

(r1) ←—— (r2).
Indholdet af register r2 kopieres ind i register r1.

O 1 D D D S S S

Maskincycles:	1
Antal clockpulser:	4
Adressering:	register (skal angives)
Påvirkede flag:	ingen.

MOV r,M (Move from memory to register)

(r) ←—— ((H) (L)).
Registre H og L indeholder adressen på den lagerlokation, hvis indhold kopieres ind i register.

O 1 D D D 1 1 O

Maskincycles:	2
Antal clockpulser:	7
Adressering:	indirekte via registerpar H.
Påvirkede flag:	ingen.

MOV, M,r (Move from register to memory)

((H) (L)) ← (r)

Register r's indhold kopieres over i lageret på den adresse, der står i registrene H og L.

0	1	1	1	0	S	S	S

Maskincycles: 2
Antal clockpulser: 7
Adressering: indirekte via registerpar H.
Påvirkede flag: ingen.

MVI r,data (Move immediate)

(r) ← (byte 2)

Indholdet af byte 2 i instruktionen kopieres ind i register r.

0	0	D	D	D	1	1	0
data							

Maskincycles: 2
Antal clockpulser: 7
Adressering: data hører til instruktionen.
Påvirkede flag: ingen.

MVI M, data (Move memory immedate)

((H) (L)) ← (byte 2).

Indholdet af byte 2 i instruktionen kopieres over i lageret på den adresse, der står i registrene H og L.

0	0	1	1	0	1	1	0
data							

Maskincycles: 3
Antal clockpulser: lo
Adressering: data hører til instruktionen og indirekte via registerpar H
Påvirkerede flag: ingen

LXI rp, data 16 (Load register pair immediate)

(rl) ← (byte 2)
(rh) ← (byte 3)

Byte 2 i instruktionen kopieres ind i registerpar rp's mindst betydende del, og byte 3 i den mest betydende del.

O	O	R	P	O	O	O	1
low-order data							
high-order data							

Maskincycles: 3
Antal clockpulser: 10
Adressering: data hører til instruktionen
Påvirkede flag: ingen.

LDA adr (Load akkumulator direct).
(A) ← ((byte 3) (byte 2)).

Byte 3 og byte 2 indeholder adressen på den lagerlokation, hvis indhold kopieres ind i akkumulatoren.

O	O	1	1	1	O	1	O
low-order adr.							
high-order adr.							

Makincycles: 4
Antal clockpulser: 13
Adressering: direkte
Påvirkede flag: ingen.

STA adr (Store akkumulator direct)

((byte3) (byte 2)) ← A

Akkumulatorens indhold kopieres over i lageret på den adresse, der angives af byte 3 og byte 2 i instruktionen.

0 0 1 1 0 0 1 0
low-order addr
high-order addr

Maskincycles: 4
Antal clockpulser: 13
Adressering: direkte
Påvirkede flag: ingen.

LHLD adr (Load H and L direct)

(L) ← ((byte 3) (byte 2))

(H) ← ((byte 3) (byte 2) + 1)

Indholdet i lageret med den adresse, byte 3 og byte 2 peger på, kopieres ind i register L, og dets indhold med adressen 1 højere kopieres ind i register H.

0 0 1 0 1 0 1 0
low-order addr.
high-order addr

Maskincycles: 5
Antal clockpulser: 16
Adressering: direkte
Påvirkede flag: ingen

SHLD (Store H and L direct).

((byte 3) (byte 2)) ← (L)

((byte 3) (byte 2) +1)) ← (H).

Indholdet i register L kopieres over i lageret på den adresse, byte 3 og byte 2 peger på.
Register H's indhold kopieres over i lageret på den efterfølgende adresse.

0 0 1 0 0 0 1 0
low-order addr.
high-order addr

Maskincycles:	5
Antal clockpulser:	16
Adressering:	direkte
Påvirkede flag	ingen.

LDAX rp (Load accumulator indirect).

(A) ← ((rp))

Indholdet i lageret på den adressen, registerpar rp peger på, kopieres ind i akkumulatoren. Kun registerpar B (B og C) og D (D og E) må anvendes.

0 0 R P 1 0 1 0

Maskincycles:	2
Antal clockpulser:	7
Adressering:	indirekte via registerpar rp.
Påvirkede flag:	ingen.

STAX rp (Store accumulator indirect).

((rp)) ← A

Akkumulatorens indhold kopieres over i lageret på den adresse, registerpar rp indeholder.
Kun registerpar B (B og C) og D (D og E) må anvendes.

0 0 R P 0 0 1 0

Maskincycles:	2
Antal clockpulser:	7
Adressering:	indirekte via registerpar rp.
Påvirkede flag:	ingen.

XCHG (Exchange H and L with D and E).

(H) ↔ (D)

(L) ↔ (E)

Indholdet i registrene H og L bytter plads med indholdet i registrene D og E.

1 1 1 0 1 0 1 1

Maskincycles:	1
Antal clockpulser:	4
Adressering:	register (skal ikke angives)
Påvirkede flag:	ingen.

ARITMETISKE GRUPPE.

Instruktionerne i denne gruppe udfører aritmetiske operationer på data hentet fra registre og lageret.

FLAGENE PÅVIRKES, med mindre andet er anført. Ved udførelsen af instruktioner, der benytter subtraktion, anvendes 2-komplement regning, idet CARRY-FLAGET benyttes til låneregister (borrow).

ADD r (Add register to accumulator)

(A) ← (A) + (r).

Indholdet i register r adderes til indholdet i akkumulatoren. Resultatet placeres i akkumulatoren.

1 0 0 0 0 S S S

Maskincycles: 1
Antal clockpulser: 4
Adressering: register (skal angives).
Påvirkede flag: alle.

ADD M (Add memory to accumulator)

(A) ← (A) + ((H) (L))

Indholdet i lageret på den adresse registrene H og L peger på, lægges til indholdet i akkumulatoren. Resultatet placeres i akkumulatoren.

1 0 0 0 0 1 1 0

Maskincycles: 2
Antal clockpulser: 7
Adressering: indirekte via registerpar H
Påvirkede flag: alle

ADI data (Add immediate to accumulator)

(A) ← (A) + (byte 2)

Indholdet i byte 2 af instruktionen lægges til akkumulatoren. Carry-FF'en deltager ikke i additionen, men påvirkes af resultatet.
Resultatet placeres i akkumulatoren.

1 1 0 0 0 1 1 0
data

Maskincycles: 2
Antal clockpulser: 7
Adressering: immediate
Påvirkede flag: alle.

ADC r (Add register to accumulator with carry).

(A) ← (A) + (r) + (CY)

Indholdet i register r og carry FF'en lægges til indholdet i akkumulatoren. Resultatet placeres i akkumulatoren.

1 0 0 0 1 S S S

Maskincycles: 1
Antal clockpulser: 4
Adressering: register (skal angives)
Påvirkede flag: alle.

ADC M (Add memory to accumulator with carry)

(A) ← (A) +((H) (L)) + (CY)

Indholdet i lageret på den adresse, registrene H og L peger på og carry FF'en lægges til indholdet i akkumulatoren. Resultatet placeres i akkumulatoren.

1 0 0 0 1 1 1 0

Maskincycles: 2
Antal clockpulser: 7
Adressering: indirekte via registerpar H

ACI data (Add immediate to accumulator with carry).

(A) ← (A)+(byte 2) + (CY)

Instruktionens byte 2 og carry-FF'ens indhold lægges til akkumulatoren. Resultatet placeres i akkumulatoren.

1 1 0 0 1 1 1 0
data

Maskincycles: 2
Antal clockpulser: 7
Adressering: data hører til instruktionen.
Påvirkede flag: alle.

SUB r (Subtract register from accumulator)

(A) ← (A) - (r)

Indholdet i register r trækkes fra akkumulatorens indhold. Resultatet placeres i akkumulatoren.

1 0 0 1 0 S S S

Makincycles: 1
Antal clockpulser: 4
Adressering: register (skal angives).
Påvirkede flag: alle.

SUB M (Subtract memory fra accumulator)

(A) ← (A) - ((H) (L))

Indholdet i lageret på den adresse, registrene H og L peger på, trækkes fra akkumulatoren. Resultatet placeres i akkumulatoren.

1 0 0 1 0 1 1 0

Maskincycles: 2
Antal clockpulser: 7
Adressering: indirekte via registerpar H.
Påvirkede flag: alle.

SUI data (Subtract immediate from accumulator).

(A) ← (A) - (byte 2)

Instruktionens byte 2 og indholdet af akkumulatoren subtraheres. Resultatet placeres i akkumulatoren.

1 1 0 1 0 1 1 0

data

Maskincycles: 2
Antal clockpulser: 7
Adressering: immediate
Påvirkede flag: alle.

SBB r(Subtract register and borrow from accumulator).

(A) ← (A) - (r) - (CY)

Indholdet i register r og carry-FF'en trækkes fra indholdet i akkumulatoren. Resultatet placeres i akkumulatoren.

1 0 0 1 1 S S S

Maskincycles: 1
Antal clockpulser: 4
Adressering: register (skal angives).
Påvirkede flag: alle.

SBB M (Subtract memory and borrow from accumulator)

(A) ← (A) - ((H) (L)) - (CY)

Indholdet i lageret på den adresse, registrene H og L peger på, og CY-FF'en trækkes fra indholdet i akkumulatoren. Resultatet placeres i akkumulatoren.

1 0 0 1 1 1 1 0

Maskincycles: 2
Antal clockpulser: 7
Adressering: indirekte via registerpar H
Påvirkede flag: alle.

SBI data (Subtract with borrow from accumulator immediate).

(A) ← (A) - (byte 2) - (CY)

Instruktionens byte 2 og carry-FF'ens indhold trækkes fra indholdet i akkumulatoren. Resultatet placeres i akkumulatoren.

1 1 0 1 1 1 1 0
data

Maskincycles: 2
Antal clockpulser: 7
Adressering: immediate
Påvirkede flag: alle.

INR r (Increment register)

(r) ← (r) + 1.

Indholdet i register r øges med 1.

NB: CY-FF'en påvirkes ikke.

0 0 D D D 1 0 0

Maskincycles:	1
Antal clockpulser	4
Adressering:	register (skal angives)
Påvirkede flag:	Z, S, P, AC.

INR M (Increment memory).

((H) (L)) ← ((H) (L)) + 1.

Indholdet i lageret på den adresse, registrene H og L peger på øges med værdien 1.

NB: CY-FF'en påvirkes ikke.

0 0 1 1 0 1 0 0

Maskincycles:	3
Antal clockpulser:	lo
Adressering:	indirekte via registerpar H
Påvirkede flag:	Z, S, P, AC.

DCR (Decrement register).

(r) ← (r) - 1.

Indholdet i register r mindskes med værdien 1.

NB: CY-FF'en påvirkes ikke.

0 0 D D D 1 0 1

Maskincycles:	1
Antal clockpulser:	4
Adressering:	register (skal angives)
Påvirkede flag:	Z, S, P, AC.

DCR M (Decrement memory)

((H) (L)) ← ((H) (L)) - 1.

Indholdet i lageret på den adresse, registrene H og L peger på, mindskes med værdien 1.

NB: CY'FF'en påvirkes ikke.

0 0 1 1 0 1 0 1

Makincycles: 3
Antal clockpulser: 10
Adressering: indirekte via registerpar H
Påvirkede flag: Z, S, P, AC.

INX rp (Increment register pair)

(rh) (rl) ← (rh) (rl) + 1.

Indholdet i registerpar rp øges med værdien 1.
NB: ingen flag påvirkes.

0 0 R P 0 0 1 1

Maskincycles: 1
Antal clockpulser: 6
Adressering: registerpar (skal angives).
Påvirkede flag: ingen.

DCX rp (Decrement register pair)

(rh) (rl) ← (rh) (rl) - 1.
Indholdet i registerpar rp mindskes med værdien 1.
NB: Ingen flag påvirkes.

0 0 R P 1 0 1 1

Maskincycles 1
Antal clockpulser: 6
Adressering: registerpar (skal angives)
Påvirkede flag: ingen.

DAD rp (Add register pair to H and L)

(H) (L) ← (H) (L) + (rh) (rl).

Indholdet i registerpar rp lægges til registerpar H.
Resultatet placeres i registerpar H.
NB: Kun CY-FF'en påvirkes, og kun hvis en mente opstår som den 17'ende bit.

0 0 R P 1 0 0 1

Maskincycles:	3
Antal clockpulser:	10
Adressering:	registerpar (skal angives)
Påvirkede flag:	CY.

DAA (Decimal adjust accumulator).

Akkumulatorens 8-bit indhold justeres til 2 4-bit binærkodede decimaltal efter følgende regler:

1. 6 lægges til akkumulatoren, hvis den binære værdi for de 4 mindst betydende bit er større end 9 eller hvis AC-flaget indeholder værdien 1.

2. 6 lægges til akkumulatorens 4 mest betydende bit, hvis deres binære værdi er større end 9 eller hvis CY-flaget indeholder værdien 1.

NB: Alle flag påvirkes.

0 0 1 0 0 1 1 1

Maskincycles:	1
Antal clockpulser:	4
Adressering:	ingen
Påvirkede flag:	alle.

LOGISKE GRUPPE:

Instruktionerne i denne gruppe foretager binære logiske operationer på data fra registrene, flagene og lageret. Flagene påvirkes med mindre andet er anført.

ANA r (AND register with accumulator).

(A) ← (A) . (r).

Indholdet i akkumulatoren og i register r AND'es logisk med hinanden (ensplacerede bit benyttes). Resultatet placeres i akkumulatoren.

NB: CY'FF'en 0-stilles og AC-FF'en 1-stilles.

1 0 1 0 0 S S S

Maskincycles:	1
Antal clockpulser:	4
Adressering:	register (skal angives)
Påvirkede flag:	alle.

ANA M (AND memory with accumulator)

(A) ← (A) . ((H) (L)).

Indholdet i lageret på den adresse registrene H og L peger på og akkumulatorens AND'es logisk med hinanden (ensplacerede bit benyttes).

Resultatet placeres i akkumulatoren.

NB: CY-FF'en 0-stilles, og AC-FF'en 1-stilles.

1 0 1 0 0 1 1 0

Maskincycles:	2
Antal clockpulser:	7
Adressering:	indirekte via registerpar H
Påvirkede flag:	Alle.

ANI data (AND with accumulator immediate).

(A) ← (A) . (byte 2).

Instruktionens byte 2 og akkumulatorens indhold AND'es logisk med hinanden (ensplacerede bit benyttes). Resultatet placeres i akkumulatoren.

NB: CY-FF'en 0-stilles og AC-FF'en 1-stilles.

1 1 1 0 0 1 1 0

data

Maskincycles: 2
Antal clockpulser: 7
Adressering: immediate.
Påvirkede flag: alle.

XRA r (Exclusive OR register with accumulator).

(A) ← (A) ⊕ (r).

Indholdet i register r og akkumulatoren bliver EXOR'et logisk med hinanden (ensplacerede bit benyttes) Resultatet placeres i akkumulatoren.

NB: CY- og AC-FF'erne 0-stilles.

1 0 1 0 1 S S S

Maskincycles: 1
Antal clockpulser 4
Adressering: register (skal angives)
Påvirkede flag: alle.

XRA M (Exclusive OR memory with accumulator).

(A) ← (A) ⊕ ((H) (L)).

Indholdet i lageret på den adresse, registrene H og L peger på og akkumulatoren bliver EXOR'et logisk med hinanden (ensplacerede bit benyttes).

Resultatet placeres i akkumulatoren.

NB: CY- og AC-FF'erne 0-stilles.

1 0 1 0 1 1 1 0

Maskincycles: 2
Antal clockpulser: 7
Adressering: indirekte via registerpar H.
Påvirkede flag: alle.

XRI data (Exclusive OR with accumulator immediate).

(A) ← (A) ⊕ (byte 2).

Instruktionens byte 2 og akkumulatorens indhold EXOR'es logisk sammen (ensplacerede bit benyttes).
Resultatet placeres i akkumulatoren.

NB: CY- og AC FF'erne O-stilles.

1	1	1	0	1	1	1	0
data							

Maskincycles: 2
Antal clockpulser: 7
Adressering: immediate.
Påvirkede flag: alle.

ORA r (OR register with accumulator).

(A) ← (A) + (r).

Indholdet i register r og akkumulatoren bliver OR'et logisk med hinanden (ensplacerede bit benyttes)
Resultatet placeres i akkumulatoren.

NB: CY- og AC-FF'erne O-stilles.

1	0	1	1	0	S	S	S

Maskincycles: 1
Antal clockpulser: 4
Adressering: register (skal angives)
Påvirkede flag: alle.

ORA M (OR memory with accumulator)

(A) ← (A) + ((H) (L)).

Indholdet i lageret på den adresse, registrene H og L peger på, og akkumularoren bliver OR'et logisk med hinanden (ensplacerede bit benyttes).
Resultatet placeres i akkumulatoren.

NB: CY- og AC-FF'erne O-stilles.

1 0 1 1 0 1 1 0

Maskincycles: 2
Antal clockpulser: 7
Adressering: indirekte via registerpar H.
Påvirkede flag: alle.

ORI data (OR with accumulator immediate)
(A) ← (A) + (byte 2).

Instruktionens byte 2 og akkumulatorens indhold bliver OR'et logisk med hinanden (ensplacerede bit benyttes). Resultatet placeres i akkumulatoren.

NB: CY- og AC-FF'erne 0-stilles.

1 1 1 1 0 1 1 0
data

Maskincycles: 2
Antal clockpulser: 7
Adressering: imediate.
Påvirkede flag: alle.

CMP r (Compare register with accumulator).
CY, Z: (A) - (r).

Indholdet i register r trækkes fra akkumulatoren. Akkumulatorens indhold forbliver uændret.
Hvis (A) = (r) fås Z-FF'en = 1
Hvis (A) < (r) fås CY-FF'en = 1

1 0 1 1 1 S S S

Maskincycles: 1
Antal clockpulser: 4
Adressering: register (skal angives)
Påvirkede flag: alle.

CMP M (Compare memory with accumulator)

CY, Z: (A) - ((H) (L)).

Indholdet i lageret på den adresse, registrene H og L peger på, trækkes fra akkumulatorens indhold. Akkumulatorens indhold forbliver uændret.

Hvis (A) = ((H) (L)) fås Z-FF'en = 1
Hvis (A) < ((H) (L)) fås CF-FF'en = 1

1 0 1 1 1 1 1 0

Maskincycles: 2
Antal clockpulser: 7
Adressering: indirekte via registerpar H.
Påvirkede flag: alle.

CPI data (compare with accumulator immediate).
CY, Z: (A) - (byte 2).

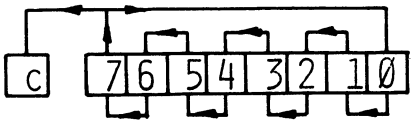
Instruktionens byte 2 trækkes fra akkumulatoren. Akkumulatorens indhold forbliver uændret.

Hvis (A) = (byte 2) fås Z-FF'en = 1
Hvis (A) < (byte 2) fås CY-FF'en = 1.

1 1 1 1 1 1 1 0
data

Maskincycles: 2
Antal clockpulser: 7
Adressering: immediate
Påvirkede flag: alle.

RLC (Rotate accumulator left).

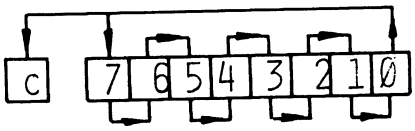


Indholdet i akkumulatoren skiftes én position til venstre, som angivet ovenfor.
NB: kun CY-FF'en påvirkes.

0 0 0 0 0 1 1 1

Maskincycles 1
Antal clockpulser: 4
Påvirkede flag: CY

RRC (Rotate accumulator right).



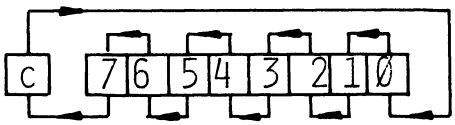
Indholdet i akkumulatoren skiftes én position til højre, som angivet ovenfor.

NB: kun CY-FF'en påvirkes.

0 0 0 0 1 1 1 1

Maskincycles: 1
Antal clockpulser: 4
Påvirkede flag: CY.

RAL (Rotate accumulator left througt carry).

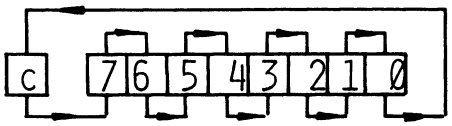


Indholdet i akkumulatoren skiftes én position til venstre igennem CY-FF'en, som angivet ovenfor.
NB: Kun CY-FF'en påvirkes.

0 0 0 1 0 1 1 1

Maskincycles: 1
Antal clockpulser: 4
Påvirkede flag: CY.

RAR (Rotate accumulator right through carry).



Indholdet i akkumulatoren skiftes én position til højre igennem CY-FF'en, som angivet ovenfor.
NB: Kun CY-FF'en påvirkes.

0 0 0 1 1 1 1 1

Maskincycles: 1
Antal clockpulser: 4
Påvirkede flag: CY.

CMA (Complement accumulator)
$(A) \leftarrow (\overline{A})$.
Alle bit i akkumulatoren komplementeres.
NB: Ingen flag påvirkes.

0 0 1 0 1 1 1 1

Maskincycles: 1
Antal clockpulser: 4
Påvirkede flag: ingen.

CMC (Complement carry)
$(CY) \leftarrow (\overline{CY})$
CY-FF'ens indhold komplementeres.
NB: Kun CY-FF'en påvirkes.

0 0 1 1 1 1 1 1

Maskincycles: 1
Antal clockpulser: 4
Påvirkede flag: CY.

STC (Set carry).

(CY) ← 1.

CY-FF'en får indholdet 1.

NB: kun CY-FF'en påvirkes.

0 0 1 1 0 1 1 1

Maskincycles: 1

Antal clockpulser: 4

Påvirkede flag: CY.

HOP-GRUPPE.

Instruktionerne i denne gruppe ændrer normal sekventiel programgennemløb. Flagene påvirkes ikke af instruktionerne i denne gruppe.

Der findes betingede og ubetingede hop-instruktioner.

Ubetingede instruktioner ændrer programtællerens (PC) indhold direkte.

Betingede hop-instruktioner ændrer kun programtællerens indhold, hvis en bestemt tilstand er til stede i status flagene.

Tilstandene er vist herunder:

FLAG-FF	TILSTAND		CCC
NZ	Z = 0		000
Z	Z = 1		001
NC	CY = 0		010
C	CY = 1		011
PO	P = 0	(ulige par.)	100
PE	P = 1	(lige par.)	101
P	S = 0	(plus)	110
M	S = 1	(minus)	111

JMP adr (Jump unconditionally).

(PC)←(byte 3) (byte 2).

Programkontrollen overføres til den adresse instruktionens byte 2 og byte 3 angiver.

1	1	0	0	0	0	1	1
low-order adrr							
high-order adrr							

Maskincycles: 3
Antal clockpulser: lo
Adressering: immediate
Påvirkede flag: ingen.

J condition adr. (Jump conditionally).

Hvis (CCC) fås (PC)← (byte 3) (byte 2)

Hvis den angivne tilstand (CCC) er til stede, overføres programkontrollen til den adresse, der er anført i instruktionens byte 2 og byte 3, i modsat fald fortsætter programmet sekventielt.

1	1	C	C	C	o	1	0
low-order addr							
high-order addr							

Maskincycles: 2/3,
Antal clockpulser: 7/lo
Adressering: immediate
Påvirkede flag: ingen.

CALL adr (Call unconditionally).

((SP) - 1)← (PCH)
((SP) - 2)← (PCL)
(SP) ← (SP) - 2
(PC) ← (byte 3) (byte 2)

Den mest betydende del af næste instruktionsadresse overføres til lageret på den adresse, stackpointeren minus 1 peger på. Den mindst betydende del af næste instruktionsadresse overføres til lageret på den adresse, stackpointeren minus 2 peger på. Stackpointerens indhold mindskes med værdien 2. Programkontrollen overføres til den adresse, byte 2 og byte 3 peger på.

1 1 0 0 1 1 0 1
low-order adrr
high-order adrr

Maskincycles: 5
Antal clockpulser: 18
Adressering: immediate og register indirekte.
Påvirkede flag: ingen.

C condition adr. (Call conditionally.)

Hvis (CCC) fås

((SP) - 1) ← (PCH)
((SP) - 2) ← (PCL)
(SP) ← (SP) - 2
(PC) ← (byte 3) (byte 2).

Hvis tilstanden (CCC) er til stede, sker det samme som anført ved CALL-instruktionen, i modsat fald fortsætter programmet sekventielt.

1 1 C C C 1 0 0
low-order addr
high-order addr

Maskincycles: 2/5
Antal clockpulser: 9/18
Adressering: adresse hører til instruktionen og indirekte via register SP
Påvirkede flag: ingen.

RET (Return unconditionally)

(PLC) ← ((SP))

(PCH) ← ((SP) + 1)

(SP) ← (SP) + 2.

Indholdet i lageret på den adresse, register SP peger på, kopieres over i programtællerens mindst betydende del. Indholdet i lageret på den adresse, register SP plus 1 peger på, kopieres over i programtællerens mest betydende del. Registeret SP's indhold øges med værdien 2.

1 1 0 0 1 0 0 1

Maskincycles: 3

Antal clockpulser: 1o

Adressering: indirekte via register SP

Påvirkede flag: ingen.

R condition (Return conditionally).

Hvis (CCC)

(PCL) ← ((SP))

(PCH) ← ((SP) + 1)

(SP) ← (SP) + 2.

Hvis tilstanden (CCC) er tilstede, sker det samme som anført ved RET-instruktionen, i modsat fald fortsætter programmet sekventielt.

1 1 C C C 0 0 0

Maskincycles: 1/3

Antal clockpulser: 6/12

Adressering: indirekte via registeret SP.

Påvirkede flag: ingen.

RST a (Restart).

((SP) - 1) ← (PCH)

((SP) - 2) ← (PCL)

(SP) ← (SP) - 2

(PC) ← 8 x (AAA)

Denne instruktion virker meget lig CALL, som vist ovenfor. Programtælleren tilføres værdien A gange 8, og programmet fortsætter fra den adresse, der derved opstår.

1 1 A A A 1 1 1

Maskincycles: 3
Antal clockpulser: 12
Adressering: adressen hører til instruktionen.
Påvirkede flag: ingen.

Programtælleren indeholder følgende værdi efter en RST n instruktion.

15	14	13	12	11	1o	9	8	7	6	5	4	3	2	1	o
O	O	O	O	O	O	O	O	O	O	A	A	A	O	O	O

PCHL (Load Program counter with H and L)

(PCH) ← (H)
(PCL) ← (L)

Indholdet i registerpar H kopieres over i programtælleren.

1 1 1 0 1 0 0 1

Maskincycles: 1
Antal clockpulser: 6
Adressering: register (skal ikke angives)
Påvirkede flag: ingen.

STACK - I/O og MASKIN-KONTROL-GRUPPE.

Instruktionerne i denne gruppe udfører input/output operationer, påvirker stack'en og kan ændre indholdet i flag-FF'erne.

FLAGENE påvirkes ikke med mindre andet er anført.

PUSH rp (Push registerpair onto stack).

((SP) - 1) ← (rh)

((SP) - 2) ← (rl)

(SP) ← (SP) - 2.

Indholdet af registerpar rp kopieres over i stacken, idet den mest betydende byte placeres på adressen, der fås ved at trække en fra SP-registeret. Den mindst betydende del af registerparret placeres i stack'en på adressen SP-2. SP'erens indhold mindskes med værdien 2.

NB: Register SP MÅ IKKE ANFØRES for rp.

1 1 R P 0 1 0 1

Maskincycles:	3
Antal clockpulser:	12
Adressering:	indirekte via register SP og registerpar (skal angives).

PUSH PSW (Push processor status word onto stack)

((SP) - 1) ← (A)

((SP) - 2) ← S Z O AC O P L CY

D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0

(SP) ← (SP) - 2.

Instruktionen virker meget lig den foregående instruktion PUSH, idet indholdet i akkumulatoren og status FF'erne kopieres over i lageret, hvor stacken er placeret. Dette angiver SP. SP'erens indhold mindskes med værdien 2.

1 1 1 1 0 1 0 1

Maskincycles:	3
Antal clockpulser:	12
Adressering:	indirekte via register SP
Påvirkede flag:	ingen.

POP rp (Pop off stack to register pair)

(rl) ← ((SP))

(rh) ← ((SP) + 1)

(SP) ← (SP) + 2.

Med denne instruktion fås den modsatte virkning af en PUSH rp. Stackens indhold kopieres tilbage til et registerpar. SP'erens indhold øges med værdien 2.

NB: Register SP MÅ IKKE ANFØRES for rp.

1	1	R	P	0	0	0	1

Maskincycles	3
Antal clockpulser	10
Adressering	indirekte via register SP og registerpar (skal angives).
Påvirkede flag:	ingen.

POP PSW (Pop processor status word off stack to registers).

Instruktionen har modsat virkning af PUSH PSW, idet den genskaber den tilstand i akkumulatoren og status-FF'erne, der eksisterede før PUSH PSW instruktionen blev udført.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
S	Z	0	AC	0	P	1	CY	← ((SP))

(A) ← ((SP) + 1)

(SP) ← (SP) + 2.

1	1	1	1	0	0	0	1

Maskincycles:	3
Antal clockpulser:	10
Adressering:	indirekte via register SP
Påvirkede flag:	alle.

XTHL (exchange stack top with H and L).

(L) ⟷ ((SP))

(H) ⟷ ((SP) + 1)

Indholdet i stack'en på den adresse, SP peger på, og register L ombyttes.
Indholdet i stack'en på den adresse, SP + 1 peger på, og register H ombyttes.

1 1 1 0 0 0 1 1

Maskincycles: 5
Antal clockpulser: 16
Adressering: indirekte via register SP.
Påvirkede flag: ingen.

SPHL (Move H and L to stack pointer).

(SP) ⟵ (H) (L)

Indholdet i registrene H og L (16 bit) kopieres ind i stackpil registeret.

1 1 1 1 1 0 0 1

Maskincycles: 1
Antal clockpulser: 6
Adressering: register (skal ikke angives).
Påvirkede flag: ingen.

IN port. (Input to accumulator)

(A) ⟵ (data)

Data placered på data-busén fra en inputport kopieres ind i akkumulatoren.

1 1 0 1 1 0 1 1
port

Maskincycles: 3
Antal clockpulser: lo
Adressering: direkte (kun 8 bit)
Påvirkede flag: ingen.

OUT port (Output from accumulator)

(data) ← (A)

Data placeres på data-busén fra akkumulatoren, kopieres ud i en output-port.

1	1	0	1	0	0	1	1
port							

Maskincycles: 3
Antal clockpulser: lo
Adressering: direkte (kun 8 bit)
Påvirkede flag: ingen.

EI (Enable interrupt).

Interrupt-systemet gøres aktivt, så interrupt kan lade sig gøre efter næste instruktion.

1	1	1	1	1	0	1	1

Maskincycles: 1
Antal clockpulser: 4
Påvirkede flag: ingen.

DI (Disable interrupt).
Interrupt-systemet gøres inaktivt umiddelbart efter instruktionen DI.

1	1	1	1	0	0	1	1

Maskincycles: 1
Antal clockpulser: 4
Påvirkede flag: ingen.

HLT (Halt and enter wait state)

Processoren stoppes.
Flag-FF'erne påvirkes ikke.

0 1 1 1 0 1 1 0

Maskincycles: 1
Antal clockpulser: 5
Adressering: ingen.

NOP (No operation)

Processoren udfører intet.
Flag-FF'erne påvirkes ikke.

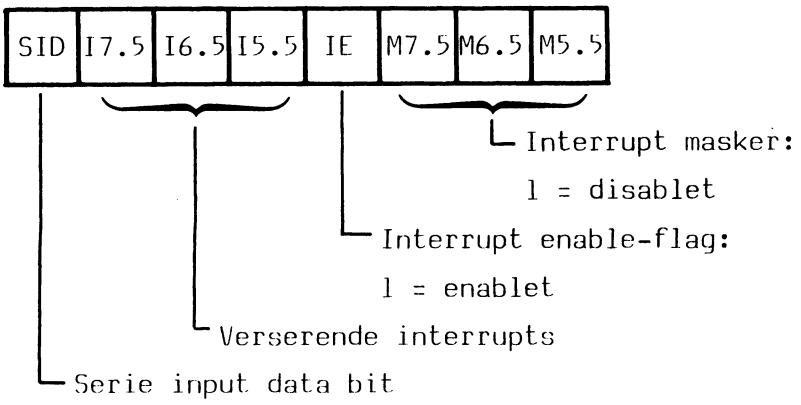
0 0 0 0 0 0 0 0

Maskincycles: 1
Antal clockpulser: 4
Påvirkede flag: ingen

RIM (Read interrupt mask).

RIM instruktionen indlæser 8 databit i akkumulatoren. Det resulterende bit-mønster viser den øjeblikkelige stilling for interrupt-masken, interrupt-flagets stilling, om der er ventende interrupts og en serieinput-bit, hvis der er nogen på ledningen SID.

RIM instruktionen indlæser følgende bitmønster i akkumulatoren:



Masken og ventende (pending) flag refererer kun til RST 5.5, RST 6.5 og RST 7.5 interruptindgangene. IE-flaget refererer til hele interruptsystemet. IE-flaget virker ligesom INTE-indgangen på 8080'eren.
Eb 1-bit i dette flag indikerer, at hele interruptsystemet er aktivt.

0 0 1 0 0 0 0 0

Maskincycles: 1
Antal clockpulser: 4
Påvirkede flag: ingen.

SIM (set interrupt mask).

SIM er en instruktion til flere funktioner. Den anvender akkumulatorens øjeblikkelige indhold til at udføre følgende funktioner:

Indstiller interruptmasken for 8085'erens RST 5.5, RST 6.5 og RST 7.5 interruptindgangen.

Resætter RST 7.5 (edge-triggered indgang).

Sender bit 7 i akkumulatoren ud til serieoutput-data-latch'en (flip-flop).

Man skal være sikker på, at det rigtige bitmønster er indlæst i akkumulatoren, før SIM-instruktionen udføres.

SIM instruktionen opfatter bittene i akkumulatoren, som vist nedenfor:

SOD	SOE	X	R7.5	MSE	M7.5	M6.5	M5.5

Interrupt masker: 0 = enablet, 1 = disablet

Mask set enable: 0 = bit 0-2 ligegyldige.

Reset RST 7.5: 1 = reset RST 7.5 FF.

Serie output enable: 1 = enablet

Serie output data.

Akkumulatorens bit 3 og 6 virker som aktiveringsbit. Hvis bit 3 er 1, er indstilling af masken aktiveret. Bit 0, 1 og 2 kan nu maske (spærre for) eller tillade de respektive RST-indgange at virke. Er bit 0 f.eks. 1, spærres for RST 5.5-indgangen (og omvendt). Hvis bit 3 er 0, har bit 0, 1 og 2 ingen virkning. Dette kan anvendes, hvis man ønsker at sende seriedata ud uden at påvirke RST masknin-gen.

Bemærk at DI-(afbryd interrupt system) instruktionen gør SIM-instruktionen virkningsløs. Uanset maskningen virker RST 5.5 - 6.5 og 7.5 ikke når en DI-instruktion er givet. Man kan anvende RIM-(læs interrupt maske) instruktionen til at bestemme den øjeblikkelige stilling af interrupt-flagene og interrupt-masken.

Hvis bit 6 er 1, er serie-data-output-funktionen aktiv. Processoren låser bit 7 fra akkumulatoren fast i en flip-flop forbundet til SOD udgangen, som en udvendig kompo-nent kan få adgang til. Hvis bit 6 er 0, påvirkes flip-floppen ikke.
Er 1 i akkumulatorens bit 4 sætter RST 7.5-indgangens flip-flop. I RST 7.5-indgangen sidder nemlig en flip-flop, som påvirkes af en ydre komponent med et 0 1 skift. Dette gælder ikke for RST 5.5 og 6.5-indgangene.
RST 7.5's kanttriggede indgang anvendes til at huske sig-naler fra komponenter, der ikke selv kan fastholde signa-let indtil RST 7.5 bliver udført.

RST 7.5 flip-floppen kan resettes af:

1) system-reset,
2) interrupt-signalet konstateres - eller
3) et 1 i akkumulatorens bit 4, og SIM bliver udført.

Det faktum, at en SIM-instruktion kan resette RST 7.5 flip-floppen, tillader, at programmet kan overse en interrupt.

RST 7.5 flip-floppen påvirkes ikke af, hvordan interrupt-masken er sat eller af DI-instruktionen, og kan derfor påvirkes hele tiden. Interrupten RST 7.5 kan dog ikke ser-viceres, når RST 7.5 er bortmasket eller DI-instruktionen er givet.

0 0 1 1 0 0 0 0

Maskincycles: 1
Antal clockpulser: 4
Påvirkede flag: ingen.

Eksempel 1:

Antag, at akkumulatoren indeholder bitmønsteret 00011100. SIM-instruktionen resetter RST 7.5-flip-floppen og setter RST 7.5-interrupt-masken. Hvis en RST 7.5-interrupt står og venter, når denne SIM-instruktion udføres, bliver RST 7.5 ikke udført. Efterfølgende RST 7.5-interruptsignaler er også masket bort og kan ikke blive serviceret, før interruptmasken bliver resat.

Eksempel 2:

Antag, at akkumulatoren indeholder bitmøsteret 11001111. SIM-instruktionen masker RST 5.5, 6.5 og 7.5-interruptmulighederne bort og låser en 1-bit ind i SOD-indgangen.

Modsat hertil vil bitmønsteret 10000111 ikke have nogen virkning, da aktiverings-bit 3 og 6 ikke er på 1.

8085A INSTRUCTION SET SUMMARY

Mnemonic	Description	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock(2) Cycles
		Instruction Code(1)								
MOVE, LOAD, AND STORE										
MOVr1 r2	Move register to register	0	1	D	D	D	S	S	S	4
MOV M,r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r,M	Move memory to register	0	1	D	D	D	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
STACK OPS										
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	12
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	12
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	12
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	12
POP B	Pop register Pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register Pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register Pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	16
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	6
JUMP										
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	7/10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	7/10
JZ	Jump on zero	1	1	0	0	1	0	1	0	7/10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	7/10
JP	Jump on positive	1	1	1	1	0	0	1	0	7/10
JM	Jump on minus	1	1	1	1	1	0	1	0	7/10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	7/10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	7/10
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	6
CALL										
CALL	Call unconditional	1	1	0	0	1	1	0	1	18
CC	Call on carry	1	1	0	1	1	1	0	0	9/18
CNC	Call on no carry	1	1	0	1	0	1	0	0	9/18
CZ	Call on zero	1	1	0	0	1	1	0	0	9/18
CNZ	Call on no zero	1	1	0	0	0	1	0	0	9/18
CP	Call on positive	1	1	1	1	0	1	0	0	9/18
CM	Call on minus	1	1	1	1	1	1	0	0	9/18
CPE	Call on parity even	1	1	1	0	1	1	0	0	9/18
CPO	Call on parity odd	1	1	1	0	0	1	0	0	9/18
RETURN										
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	6/12
RNC	Return on no carry	1	1	0	1	0	0	0	0	6/12
RZ	Return on zero	1	1	0	0	1	0	0	0	6/12
RNZ	Return on no zero	1	1	0	0	0	0	0	0	6/12
RP	Return on positive	1	1	1	1	0	0	0	0	6/12
RM	Return on minus	1	1	1	1	1	0	0	0	6/12
RPE	Return on parity even	1	1	1	0	1	0	0	0	6/12
RPO	Return on parity odd	1	1	1	0	0	0	0	0	6/12
RESTART										
RST	Restart	1	1	A	A	A	1	1	1	12
INPUT/OUTPUT										
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10

Mnemonic	Description	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock(2) Cycles
		Instruction Code(1)								
INCREMENT AND DECREMENT										
INR r	Increment register	0	0	D	D	D	1	0	0	4
DCR r	Decrement register	0	0	D	D	D	1	0	1	4
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	6
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	6
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	6
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	6
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	6
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	6
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	6
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	6
ADD										
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
SUBTRACT										
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
LOGICAL										
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive OR register with A	1	0	1	0	1	S	S	S	4
ORA r	OR register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive OR memory with A	1	0	1	0	1	1	1	0	7
ORA M	OR memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive OR immediate with A	1	1	1	0	1	1	1	0	7
ORI	OR immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
ROTATE										
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
SPECIALS										
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
CONTROL										
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
DI	Disable Interrupt	1	1	1	1	0	0	1	1	4
NOP	No operation	0	0	0	0	0	0	0	0	4
HLT	Halt	0	1	1	1	0	1	1	0	5
NEW 8085A INSTRUCTIONS										
RIM	Read Interrupt Mask	0	0	1	0	0	0	0	0	4
SIM	Set Interrupt Mask	0	0	1	1	0	0	0	0	4

NOTES: 1. DDS or SSS: B 000, C 001, D 010, E 011, H 100, L 101, Memory 110, A 111.
2. Two possible cycle times. (6/12) indicate instruction cycles dependent on condition flags.

Microcomputerens lagersektion.

Lageret (hukommelsen) i en µ-computer har følgende opgaver:

1. Det skal indeholde programmet, også efter en strømafbrydelse.
2. Det skal kunne modtage, opbevare og aflevere data fra processen.

Programhukommelsen:

Til den 1. type hukommelse anvendes en af følgende tre typer kredse:

ROM (Read only memory)
PROM (Programable read only memory)
EPROM (Erasable programable read only memory)

ROM.

ROM'en anvendes normalt kun som programhukommelse, hvor der er tale om et stort antal ensartede udstyr, idet programmeringen foregår hos IC-fabrikanten som et led i fremstillingsprocessen.

PROM.

PROM'en minder i sin opbygning om ROM'en, men har ved leveringen det samme logiske niveau på samtlige bitpositioner. Dette kan ændres af brugeren, idet han ved at programmere hukommelsen kan ændre det logiske niveau på de ønskede steder. Ved programmeringen sker der en fysisk ændring i kredsen (en forbindelse smelter), hvorfor processen er irevercibel.

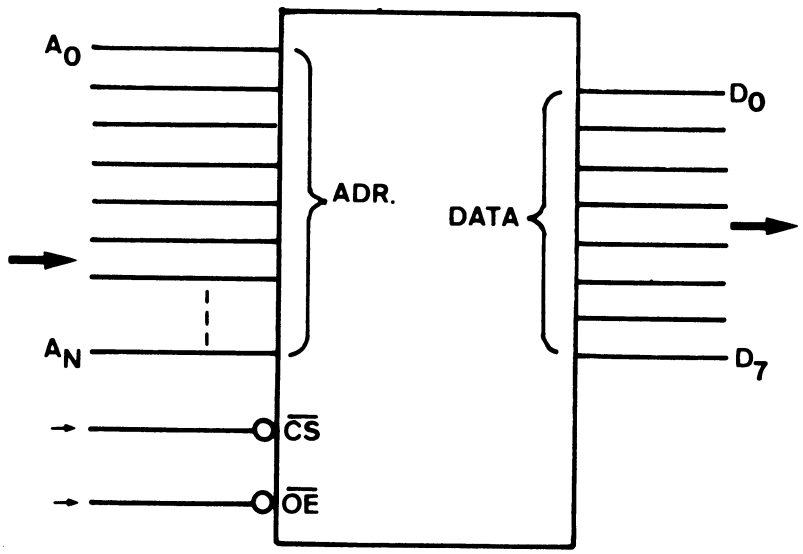
EPROM.

Til enkeltudstyr og mindre serier er EPROM'en enerådende som programhukommelse. Ved programmeringen, som foretages af brugeren på et egnet udstyr, sker der ingen fysiske ændringer, men blot en opladning af de hukommelsesceller (MOS-transistorer) hvis logiske indhold skal ændres.

Hvis det skulle vise sig nødvendigt, kan hukommelsens indhold slettes ved at belyse chippen med en passende ultraviolet lyskilde, hvorefter hukommelsen igen kan programmeres.

Programhukommelsens kontrolsignaler.

Med få undtagelser findes der de samme ben (signaler) på en ROM, PROM og EPROM.



Skitsen viser signalerne på en ROM (PROM, EPROM).

Adresse: Adresseledningernes antal afhænger af hukommelsens størrelse. Typiske størrelser er:

1 k = 1o24 adr.	1o adr. ledninger
2 k = 2o48 adr.	11 adr. ledninger
4 k = 4o96 adr.	12 adr. ledninger

Adresseledningerne forbindes til adressebussen som styres af CPU'en.

Data.

Dataoutputtet leverer det bitmønster, der er lagret på den via adresseledningerne udpegede plads i hukommelsen.

Dataoutputtet er et 3-state output. Det vil sige, at outputtet kan antage tre tilstande, nemlig low, high og off. Den sidste tilstand, hvor outputtet er højimpedant, muliggør at dataoutputtet kan kobles til databussen uden at inteferere med de øverste kredse på bussen, blot der sørges for at kun een kreds er selected af gangen.

Output enable.

Output enable ($\overline{OE}$) indgangen kontrollerer kun 3-state-bufferene i dataudgangene. Det vil sige at hukommelsen kan være chip-selected men stadig holdes 3-stated.

Chip-selected.

Chip select ($\overline{CS}$) indgangen er det overordnede indput på hukommelseskredsen, og styrer adressedecoderen i adresseindgangen samt 3-state bufferene i dataudgangene.

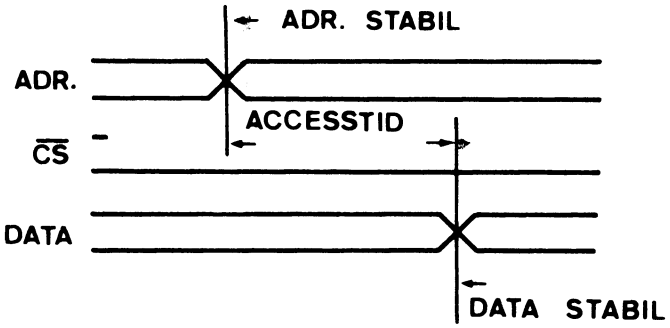
Når $\overline{CS}$ er inaktiv (high) er hukommelsens dataudgange i 3-state. Når $\overline{CS}$ er aktiv (low) er dataudgangene lavimpedante og antager de niveauer, der svarer til indholdet på den adresserede plads.

Ordlængde.

Det almindeligste antal dataledninger på ROM'en, PROM'en og EPROM'en er 8. Det vil sige, at der på hver adresse kan lagres 8 bit, svarende til en byte.

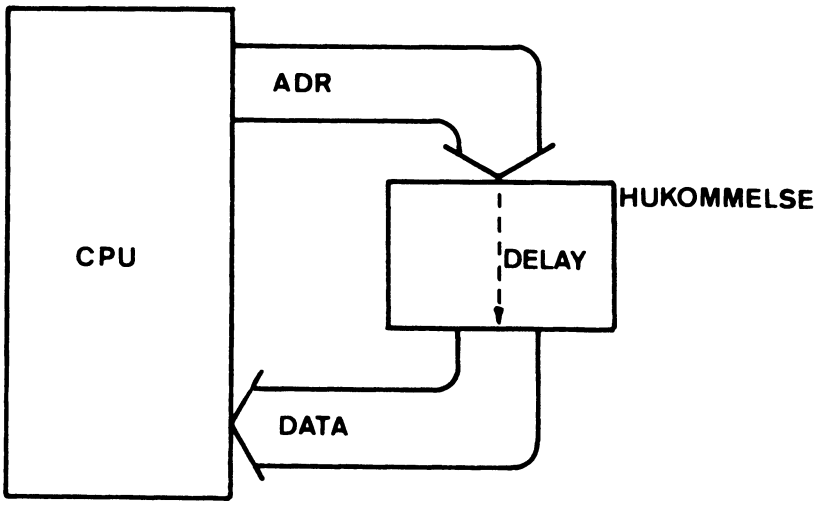
Accestid.

Når hukommelsen skal arbejde sammen med en CPU er hukommelsens accestid en egenskab af stor betydning.
Accestiden (adgangstiden) er den tid, der går fra adressen har stabiliseret sig til dataudgangene leverer stabile data.



Da CS kontrollerer adressedecoderen i kredsen vil det i praksis ofte være forsinkelsen fra CS til data, der skal tages hensyn til.

CPU'en og accestiden.



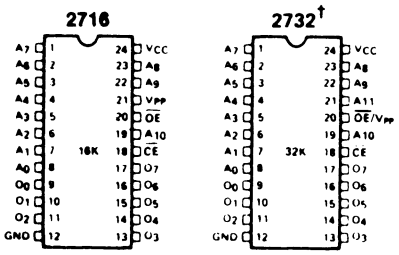
En readcycle har følgende forløb:

1. CPU'en sender adressen ud og starter "nedtællingen".
2. Hukommelsen begynder at finde de ønskede data frem, som den efter en tid (accestiden) får sendt ind til CPU'en.
3. Når CPU'en har "talt ned til nul" tager den de data ind, der findes på dens dataindgange, også selvom hukommelsen egentlig ikke var klar.

   Det betyder, at hukommelsens accestid skal modsvare CPU'ens krav.
   Hvis hukommelsen og dermed accestiden er givet, kan det blive nødvendigt at indskyde en waite-state i hver readcycle.

Data 2716.

PIN CONFIGURATION



†Refer to 2732 data sheet for specifications

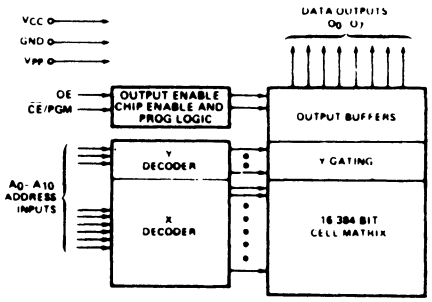
MODE SELECTION

PINS / MODE	CE/PGM (18)	OE (20)	VPP (21)	VCC (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Standby	V_{IH}	Don't Care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IH}	+25	+5	High Z

PIN NAMES

$A_0 - A_{10}$	ADDRESSES
$\overline{CE}/PGM$	CHIP ENABLE/PROGRAM
$\overline{OE}$	OUTPUT ENABLE
$O_0 - O_7$	OUTPUTS

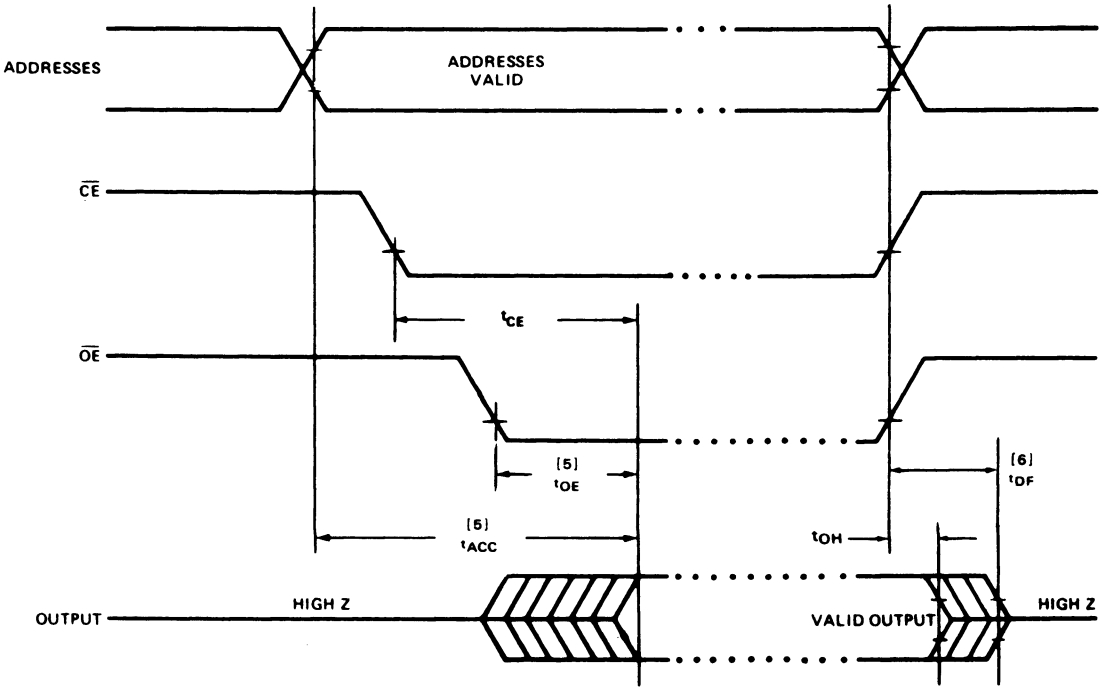
BLOCK DIAGRAM



A.C. Characteristics

Symbol	Parameter	Limits (ns) 2716 Min.	2716 Max.	2716-1 Min.	2716-1 Max.	2716-2 Min.	2716-2 Max.	2716-5 Min.	2716-5 Max.	2716-6 Min.	2716-6 Max.	Test Conditions
t_{ACC}	Address to Output Delay		450		350		390		450		450	$\overline{CE} = \overline{OE} = V_{IL}$
t_{CE}	$\overline{CE}$ to Output Delay		450		350		390		490		650	$\overline{OE} = V_{IL}$
t_{OE}	Output Enable to Output Delay		120		120		120		160		200	$\overline{CE} = V_{IL}$
t_{DF}	Output Enable High to Output Float	0	100	0	100	0	100	0	100	0	100	$\overline{CE} = V_{IL}$
t_{OH}	Output Hold from Addresses, $\overline{CE}$ or $\overline{OE}$ Whichever Occurred First	0		0		0		0		0		$\overline{CE} = \overline{OE} = V_{IL}$

A. C. Waveforms [1]



NOTE:
1. V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.
2. V_{PP} may be connected directly to V_{CC} except during programming. The supply current would then be the sum of I_{CC} and I_{PP1}.
3. Typical values are for $T_A = 25°C$ and nominal supply voltages.
4. This parameter is only sampled and is not 100% tested.
5. OE may be delayed up to $t_{ACC} - t_{OE}$ after the falling edge of CE without impact on t_{ACC}.
6. t_{DF} is specified from OE or CE, whichever occurs first.

Datahukommelsen.

Til den anden type hukommelse anvendes den såkaldte

RAM, (random access memory)

Det betyder "en hukommelse med tilfældig adgang".

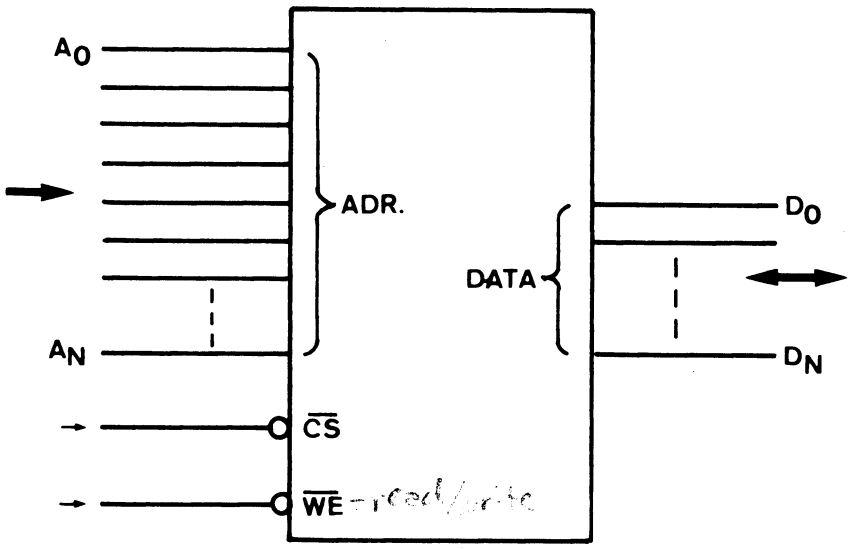
Denne type kaldes også for

Read/write memory, hvilket mere korrekt beskriver denne hukommelses egenskaber.

Teknologi.

Hukommelsescellerne i en RAM (statisk) er opbygget som bistabile flip-flops, hvorfor det er muligt at ændre indholdet i overensstemmelse med de tilførte data.
Til gengæld mistes hukommelsens indhold hvis forsyningsspændingen blot kortvarigt forsvinder.

Datahukommelsens kontrolsignaler.



Adresse. Se programhukommelse.

Chip select. Se programhukommelse.

Data.

Da RAM-hukommelsen er en læse/skrive-hukommelse er dataledningerne to-vejs, styret af write enable ($\overline{WE}$).

Når $\overline{WE}$ er high, går dataledningerne ud af kredsen og den er i read mode når $\overline{CS}$ aktiveres.

Påtrykkes $\overline{WE}$ et low niveau skiftes til write-mode, hvorfor dataledningerne nu er dataindgange, således at det tilførte bitmønster indlæses på den valgte adresse, dog under forudsætning af at $\overline{CS}$ er aktiv.

Write enable.

Write enable indgangen skifter hukommelsen mellem read mode og write mode.
Det er vigtigt, at write enable signalet er timet korrekt i forhold til adressen idet der ellers kan være risiko for utilsigtet at ændre tilfældige adressers indhold.

Ordlængde.

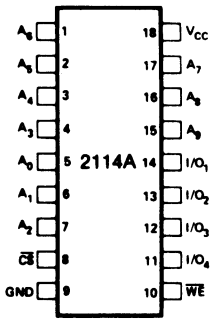
RAM-hukommelser findes med ordlængder på 1 eller 4 bit. Det vil sige at der skal anvendes henholdsvis 8 og 2 kredse i "PARALLEL" for at opnå en ordlængde på 1 byte.

Accestid.

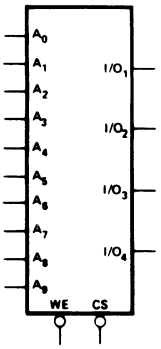
Se programhukommelse.

Data 2114.

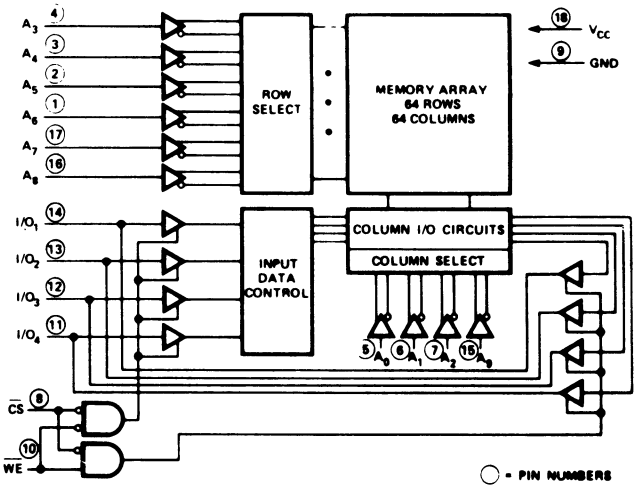
PIN CONFIGURATION



LOGIC SYMBOL



BLOCK DIAGRAM



PIN NAMES

A_0–A_9	ADDRESS INPUTS	V_{CC}	POWER (+5V)
$\overline{WE}$	WRITE ENABLE	GND	GROUND
$\overline{CS}$	CHIP SELECT		
I/O_1–I/O_4	DATA INPUT/OUTPUT		

READ CYCLE [1]

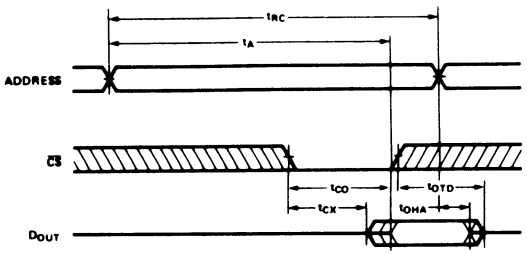
SYMBOL	PARAMETER	2114AL-1 Min.	2114AL-1 Max.	2114AL-2 Min.	2114AL-2 Max.	2114AL-3 Min.	2114AL-3 Max.	2114A-4/L-4 Min.	2114A-4/L-4 Max.	2114A-5 Min.	2114A-5 Max.	UNIT
t_{RC}	Read Cycle Time	100		120		150		200		250		ns
t_A	Access Time		100		120		150		200		250	ns
t_{CO}	Chip Selection to Output Valid		70		70		70		70		85	ns
t_{CX}	Chip Selection to Output Active	10		10		10		10		10		ns
t_{OTD}	Output 3-state from Deselection		30		35		40		50		60	ns
t_{OHA}	Output Hold from Address Change	15		15		15		15		15		ns

WRITE CYCLE [2]

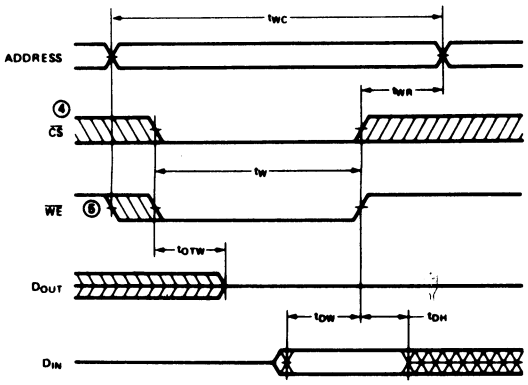
SYMBOL	PARAMETER	2114AL-1 Min.	2114AL-1 Max.	2114AL-2 Min.	2114AL-2 Max.	2114AL-3 Min.	2114AL-3 Max.	2114A-4/L-4 Min.	2114A-4/L-4 Max.	2114A-5 Min.	2114A-5 Max.	UNIT
t_{WC}	Write Cycle Time	100		120		150		200		250		ns
t_W	Write Time	75		75		90		120		135		ns
t_{WR}	Write Release Time	0		0		0		0		0		ns
t_{OTW}	Output 3-state from Write		30		35		40		50		60	ns
t_{DW}	Data to Write Time Overlap	70		70		90		120		135		ns
t_{DH}	Data Hold from Write Time	0		0		0		0		0		ns

WAVEFORMS

READ CYCLE ③



WRITE CYCLE



NOTES:

3. $\overline{WE}$ is high for a Read Cycle.
4. If the $\overline{CS}$ low transition occurs simultaneously with the $\overline{WE}$ low transition, the output buffers remain in a high impedance state.
5. $\overline{WE}$ must be high during all address transitions.

Hukommelsens opbygning.

Hukommelsen er bygget op omkring adressebussen og databussen.

Fordelen ved et bussystem er, at det er nemt at udbygge med flere kredse, idet man blot skal forbinde de tilkomne kredse til adresse- og databus.

3-state.

For at en kreds kan indpasses i et bussystem skal den have 3-state output. Det vil sige at kredsens output kan gøres højimpedanset og dermed kobles fra databussen, idet der kun må være een aktiv kreds af gangen.

Chip-select.

Til at styre de forskellige kredses til- og frakobling tjener chip-select logikken. Denne styres af en del af adressebussen og styrer de forskellige hukommelseskredse via deres chip-select input.

Hukommelsens størrelse.

Det største antal adresser (hukommelsespladser) i en u-computers lager-sektion bestemmes af det antal adresseledninger, der er ført ud af CPU'en (direkte eller multiplexed).

Det almindeligste antal adresseledninger på en 8-bit CPU er 16.

Hukommelsens (mulige) størrelse bliver da:

$$2^{16} = 65536 = 64 \cdot 1024 \rightarrow 64K$$

Opdelingen i "k" er behagelig set i relation til de almindeligste størrelser på hukommelseskredse der er 1, 2 og 4 k.

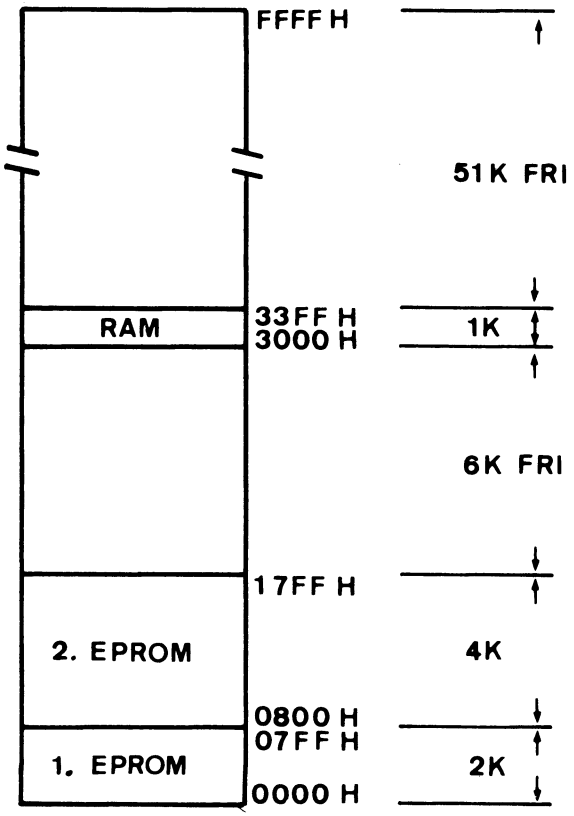
Memorymap.

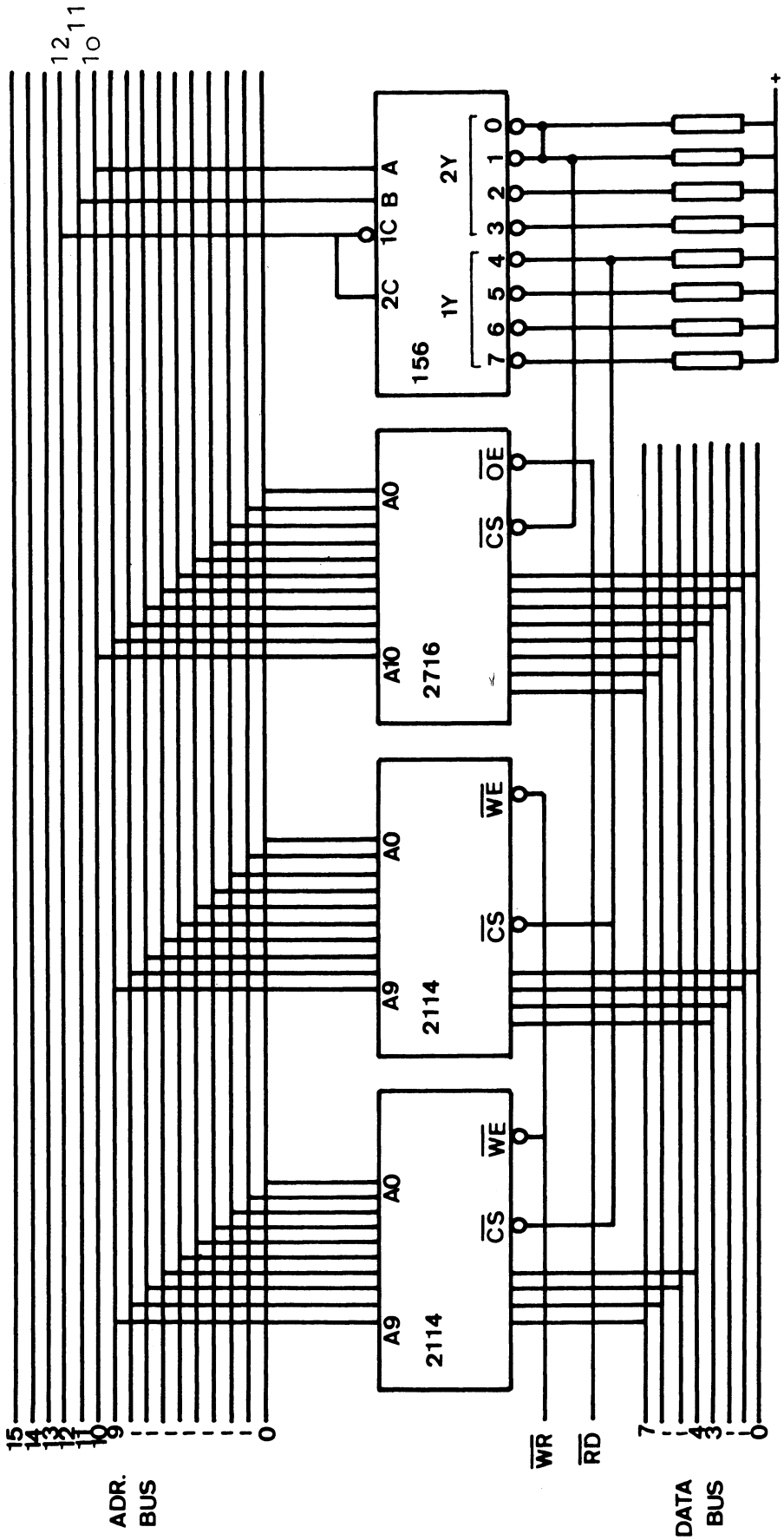
Som et middel til at beskrive hvilke typer kredse (ROM, RAM) en hukommelse indeholder, samt deres adresser anvendes ofte et memorymap.

Eksempel:
En hukommelse indeholder:

1. EPROM, 2716, 2k start-ard. ØØØØH
2. EPROM, 2732, 4k start-ard. Ø8ØØH
3. RAM, 2x2114, 1k start-ard. 3ØØØH

Dette giver følgende memorymap.





Chip-select logik med 1k opdeling.

Chip-select logikken er normalt bygget op omkring en 3 til 8 linier decoder der styres af de højere adresse-ledninger.

Se diagram

Diagrammet viser en hukommelse der består af 1k RAM og 2k EPROM. Som CS-decoder er anvendt en dual 2 til 4 linier decoder med open collektor output. Den kobles som en 3 til 8 linier decoder og styres af A_{10}-A_{12}.

Decoderens udgange dækker følgende adresseområder:

UDG.	ADR.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	fra	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0	0	ØØØØH
0	til	x	x	x	0	0	0	1	1	1	1	1	1	1	1	1	1	Ø3FFH
1	fra	x	x	x	0	0	1	0	-	-	-	-	-	-	-	-	0	Ø4ØØH
1	til	x	x	x	0	0	1	1	-	-	-	-	-	-	-	-	1	Ø7FFH
2	fra	x	x	x	0	1	0	0	-	-	-	-	-	-	-	-	0	Ø8ØØH
2	til	x	x	x	0	1	0	1	-	-	-	-	-	-	-	-	1	ØBFFH
3	fra	x	x	x	0	1	1	0	-	-	-	-	-	-	-	-	0	ØCØØH
3	til	x	x	x	0	1	1	1	-	-	-	-	-	-	-	-	1	ØFFFH
4	fra	x	x	x	1	0	0	0	-	-	-	-	-	-	-	-	0	1ØØØH
4	til	x	x	x	1	0	0	1	-	-	-	-	-	-	-	-	1	13FFH
5	fra	x	x	x	1	0	1	0	-	-	-	-	-	-	-	-	0	14ØØH
5	til	x	x	x	1	0	1	1	-	-	-	-	-	-	-	-	1	17FFH
6	fra	x	x	x	1	1	0	0	-	-	-	-	-	-	-	-	0	18ØØH
6	til	x	x	x	1	1	0	1	-	-	-	-	-	-	-	-	1	1BFFH
7	fra	x	x	x	1	1	1	0	-	-	-	-	-	-	-	-	0	1cØØH
7	til	x	x	x	1	1	1	1	-	-	-	-	-	-	-	-	1	1FFFH

x = ligegyldig.

Som det fremgår af tabellen dækker hver udgang et område på 1k.
Dette passer fint med RAMlagerets størrelse, men da EPROM'en er 2 k stor er det nødvendigt at chip-selecte den fra to udgange. Da der er anvendt en decoder med open collektor output og pull-up modstande kan man blot forbinde to naboudgange.

Sammenholdes diagrammet og tabellen findes følgende adresser på RAM og EPROM:

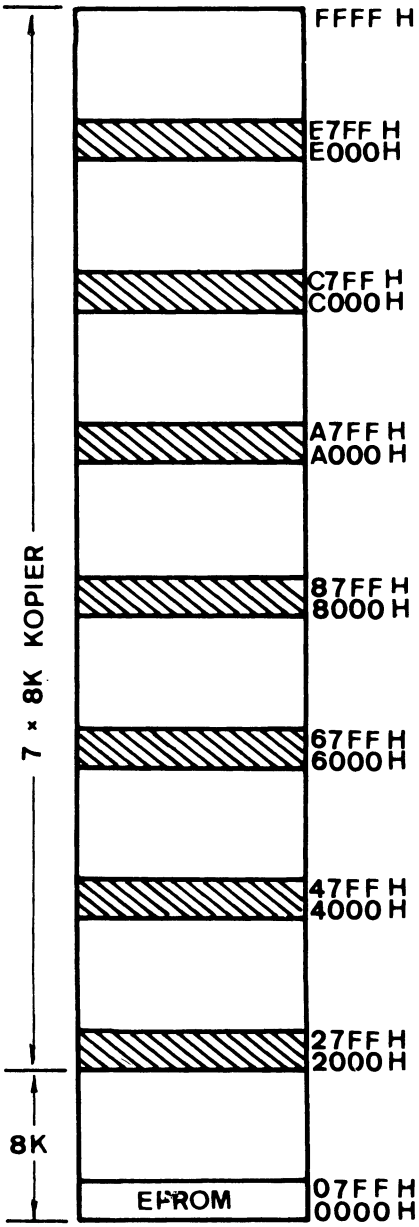
EPROM:	Udg. 0 og 1	ØØØØH - Ø7FFH
RAM:	Udg. 4	1ØØØH - 13FFH

Kopier.

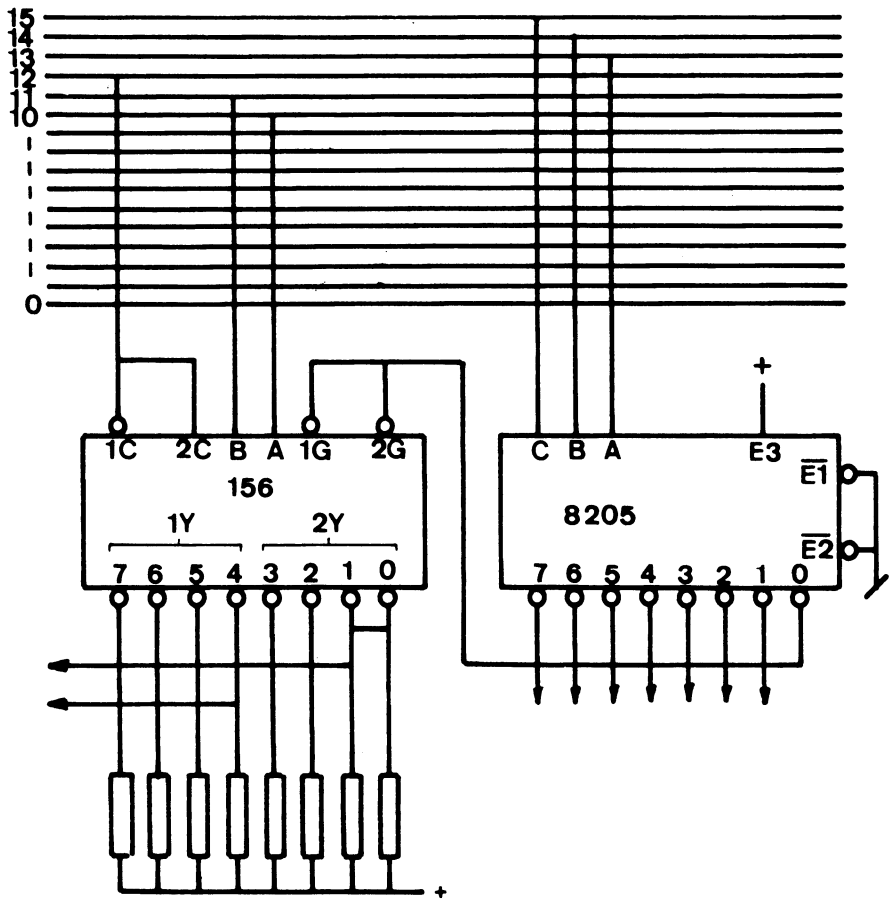
Da de tre øverste bit i adressebussen ikke er ført til decoderen kan de antage en vilkårlig værdi uden at påvirke chip-select.
Set fra CPU'en vil det virke som om den enkelte kreds kan nås på 8 forskellige startadresser.

For eksempel kan laveste adresse i EPROM'en nås på følgende 8 adresser:

ØØØØH, 2ØØØH, 4ØØØH, 6ØØØH, 8ØØØH, AØØØH, CØØØH, EØØØH.



I små systemer, hvor der kun er behov for maksimalt 8k hukommelse gøres der ikke noget for at fjerne kopieffekten. Hvis der er behov for mere end 8k hukommelse må også de tre øverste adresseledninger decodes.
Dette kan gøres ved hjælp af endnu en 3 til 8 linier decoder hvis nul-output anvendes til at enable den første decoder.



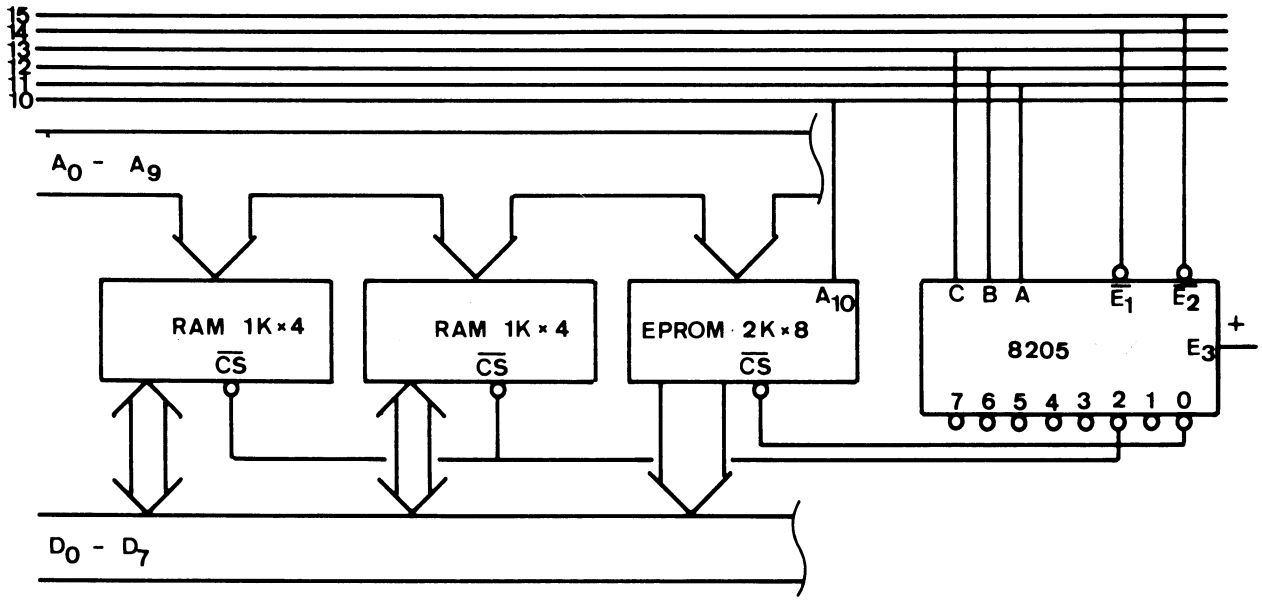
8205's udgange dækker følgende adresseområder.

Udg. no.	Adr. fra	til
0	ØØØØH	1FFFH
1	2ØØØH	3FFFH
2	4ØØØH	5FFFH
3	6ØØØH	7FFFH
4	8ØØØH	9FFFH
5	AØØØH	BFFFH
6	CØØØH	DFFFH
7	EØØØH	FFFFH

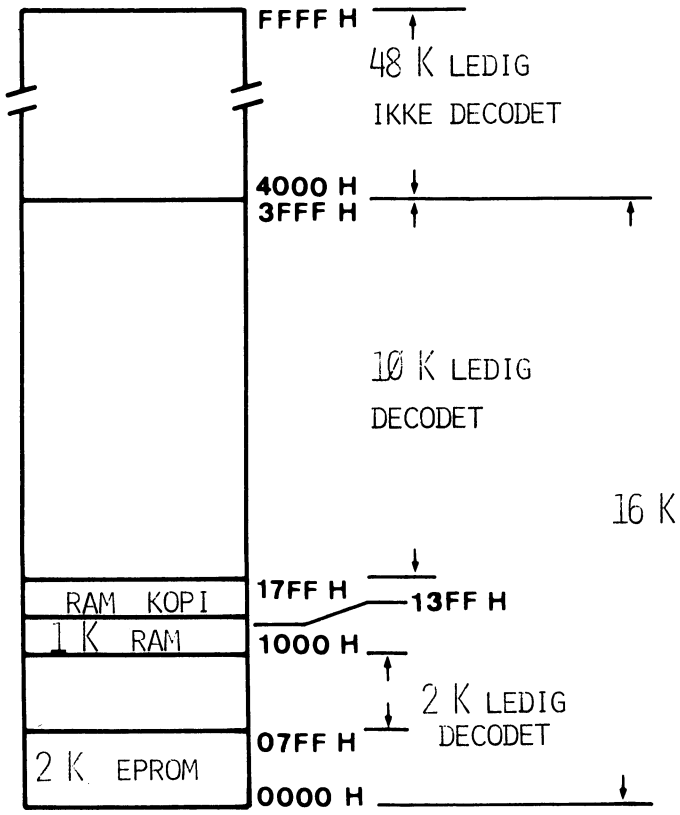
Decodning i 2k afsnit.

I stedet for at decode A_{10}, A_{11} og A_{12} hvorved der chip-selectes i 1k afsnit, kan man i stedet tage A_{11}, A_{12} og A_{13} og selecte i 2k afsnit.

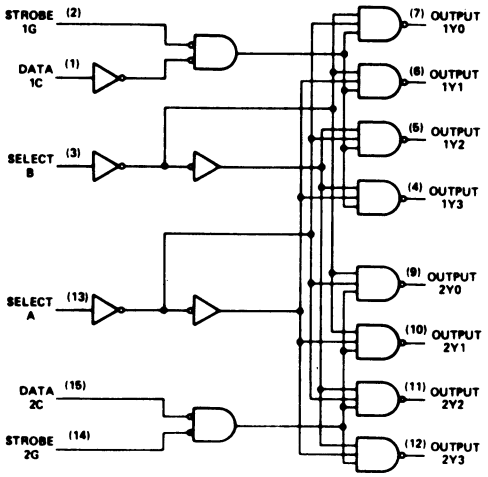
Ved at udnytte decoderens enableindgange kan kopier ud over 16k undgås.



Memorymap.
Da A_{10} hverken decodes eller tilføres RAM-lageret vil dette komme til at dække 2k



74 LS 156. Dual 2 til 4 linier decoder.



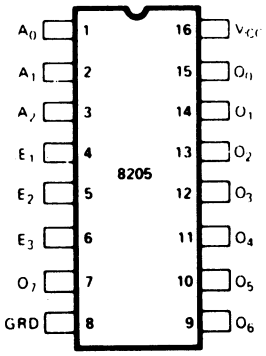
FUNCTION TABLES
2-LINE-TO-4-LINE DECODER
OR 1-LINE-TO-4-LINE DEMULTIPLEXER

INPUTS				OUTPUTS			
SELECT		STROBE	DATA				
B	A	1G	1C	1Y0	1Y1	1Y2	1Y3
X	X	H	X	H	H	H	H
L	L	L	H	L	H	H	H
L	H	L	H	H	L	H	H
H	L	L	H	H	H	L	H
H	H	L	H	H	H	H	L
X	X	X	L	H	H	H	H

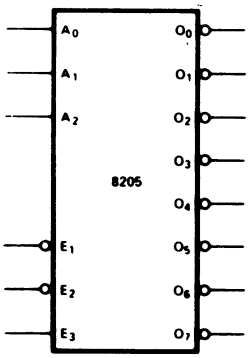
INPUTS				OUTPUTS			
SELECT		STROBE	DATA				
B	A	2G	2C	2Y0	2Y1	2Y2	2Y3
X	X	H	X	H	H	H	H
L	L	L	L	L	H	H	H
L	H	L	L	H	L	H	H
H	L	L	L	H	H	L	H
H	H	L	L	H	H	H	L
X	X	X	H	H	H	H	H

8205/74LS138. 3 til 8 linier decoder.

PIN CONFIGURATION



LOGIC SYMBOL



PIN NAMES

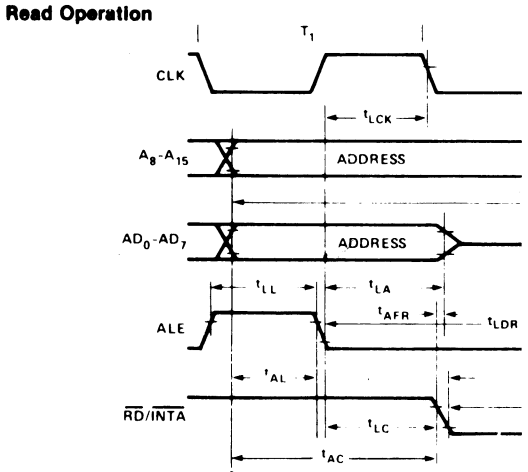
A_0 A_2	ADDRESS INPUTS
$\overline{E_1}$ $\overline{E_3}$	ENABLE INPUTS
$\overline{O_0}$ $\overline{O_7}$	DECODED OUTPUTS

ADDRESS			ENABLE			OUTPUTS							
A_0	A_1	A_2	E_1	E_2	E_3	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

CS-TIMING.

Som tidligere nævnt, er det chip-select logikkens opgave at sikre, at der kun er en aktiv kreds på databussen af gangen, hvilket også slår til når der kun ses på kredsene i hukommelsen, men ikke nødvendigvis i relation til CPU'en.

Eksempel:



T_{clk} = 32o n sek
T_{AL} = 115 n sek
T_{LA} = 1oo n sek

Her vender databussen ind i CPU'en,
CPU'en sender adrl. på databussen
Her stabiliseres adressen.

Se diagram
Fra adressen er stabiliseret til der er en aktiv chip-select går der ca. 2o n sek. Hvis det er ramlageret der bliver selected (2114) går der yderligere ca. lo nsek. hvorefter dataudgangene går ud af 3-state og bliver aktive.
CPU'en sender samtidig de 8 lave adresser ud på databussen med det resultat at niveauerne bliver forkerte. I værste fald kan det føre til ødelæggelse af en kreds.
For at undgå dette skal hukommelsen forhindres i at gå på databussen inden CPU'en har "Sluppet" den.
Der er flere mulige løsninger på problemet. Har hukommelsen et "outputenable" Input (2716) kan dette styres af et egnet signal, ved 8o85 $\overline{RD}$.

Findes et sådant input ikke kan chip-select forsinkes, for eksempel ved at gate CS-decoderen med et signal der er afledt af $IO/\overline{M}$, $\overline{RD}$ og $\overline{WR}$ (8085).
Den sidste metode kan nødvendiggøre anvendelsen af wait-state.

Parrallel IN/OUT.

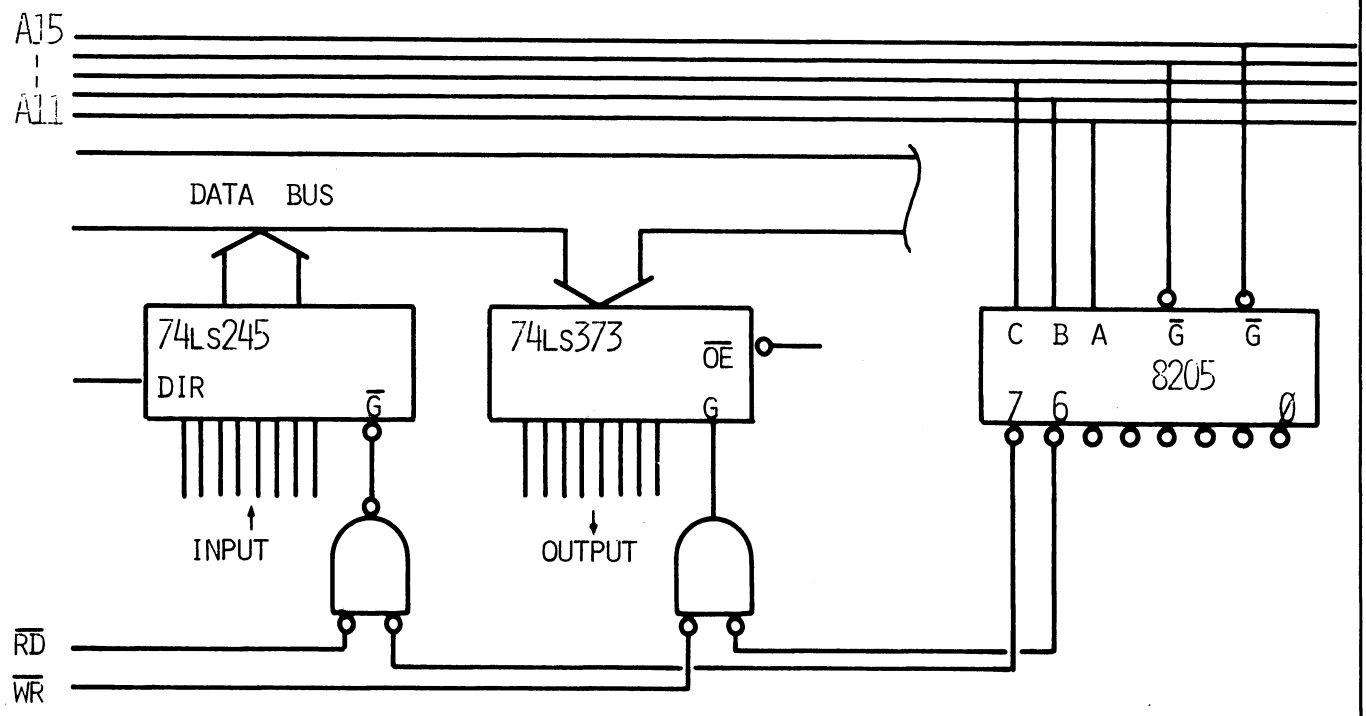
8085A kommunikerer til både hukommelse og porte(I/O) ved hjælp af read- og write-maskincycles. Under en maskincycle sender 8085A adresse og kontrolsignaler samtidig med at den enten sender data på databussen, eller læser data fra databussen. Er CPU'en for eksempel i en read cycle kan den læse fra ROM, RAM, I/O, - eller ingenting. Ligeledes kan en write-cycle ske til RAM, I/O, - eller ingenting.

Adressering af porte.

Der er to måder at adressere porte på i et 8085 system. Hvis IO/$\overline{M}$ signalet fra CPU'en anvendes til at skelne mellem I/O- og hukommelsesread og write cycles kaldes det standard I/O, eller I/O-mapped, I/O.
Hvis IO/$\overline{M}$ ikke anvendes, skelner CPU'en ikke mellem I/O og hukommelse. Dette kaldes memory-mapped I/O.

Memory-mapped I/O.

Ved memory-mapped I/O indpasses portene i hukommelsen sideordnet med de øvrige kredse (RAM og ROM).



I det viste eksempel har inputporten adresse 38ØØH og outputporten adresse 3ØØØH.

Da de to porte chipselectes direkte af chipselectdecoderen kommer de hver især til at dække et område på 2k (A_0-A_{10} er ligegyldige).

Af hensyn til timingen skal chip-select signalerne gates med $\overline{RD}$ og $\overline{WR}$ fra CPU'en.

Ved at udbygge chip-selectlogikken til også at styres af A_0-A_{10} vil hver port kun behøve at beslaglægge en enkelt adresse.

Fordele ved memorymapped I/O.

Fordelene ved memorymapped I/O skal søges i, at de instruktioner, der arbejder på hukommelsen også kan anvendes i forbindelse med portene.

For eksempel:

MOVr,M: Input-port til ethvert register.

MOVM,r: Ethvert register til output-port.

MVI M : Data immediate til output-port.

LDA : Input-port til accumulator.

STA : Accumulator til outputport.

LHLD : Input-porte til H og L reg., 16 bit.

SHLD : H og L reg. til output-porte, 16 bit.

ADD M : Adder inputport til accumulator.
ANA M : And inputport med accumulator.

Som en fordel må også regnes det næsten ubegrænsede antal porte, der kan laves, dog på bekostning af hukommelsesareal.

Ulemper ved memorymapped I/O.

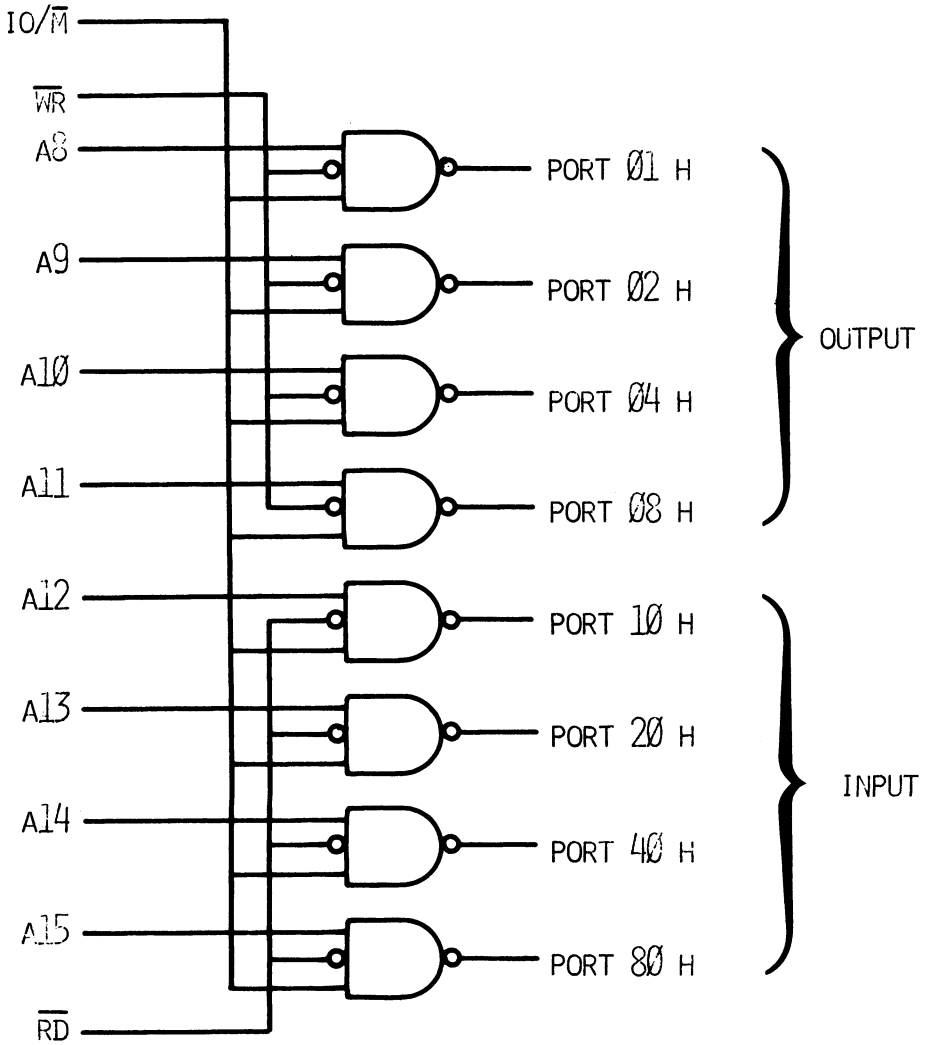
Den indlysende ulempe ved memorymapped I/O er tabet af hukommelsesareal, hvilket specielt gør sig gældende hvis portene kun chip-selectes af de højeste adressebit.

I/O-mapped I/O.

Ved I/O-mapped I/O skelner CPU'en mellem hukommelse og porte ved hjælp af IO/$\overline{M}$-outputtet.
Der er kun to instruktioner, der kan få IO/$\overline{M}$ til at blive high, nemlig IN og OUT.
IN og OUT er to-byte instruktioner, der i anden byte indeholder portens nummer. Det vil sige, at der er mulighed for 256 forskellige porte.
Når CPU'en eksekverer en in eller out instruktion trækkes IO/$\overline{M}$ high, og portnummeret der var indeholdt i instruktionens anden byte sendes ud <u>både</u> på A_{8-15} og multiplexes ud på AD_{0-7}. Iøvrigt svarer forløbet af en I/O maskincycle til en memory-read eller memory-write maskincycle.

Liniær select.

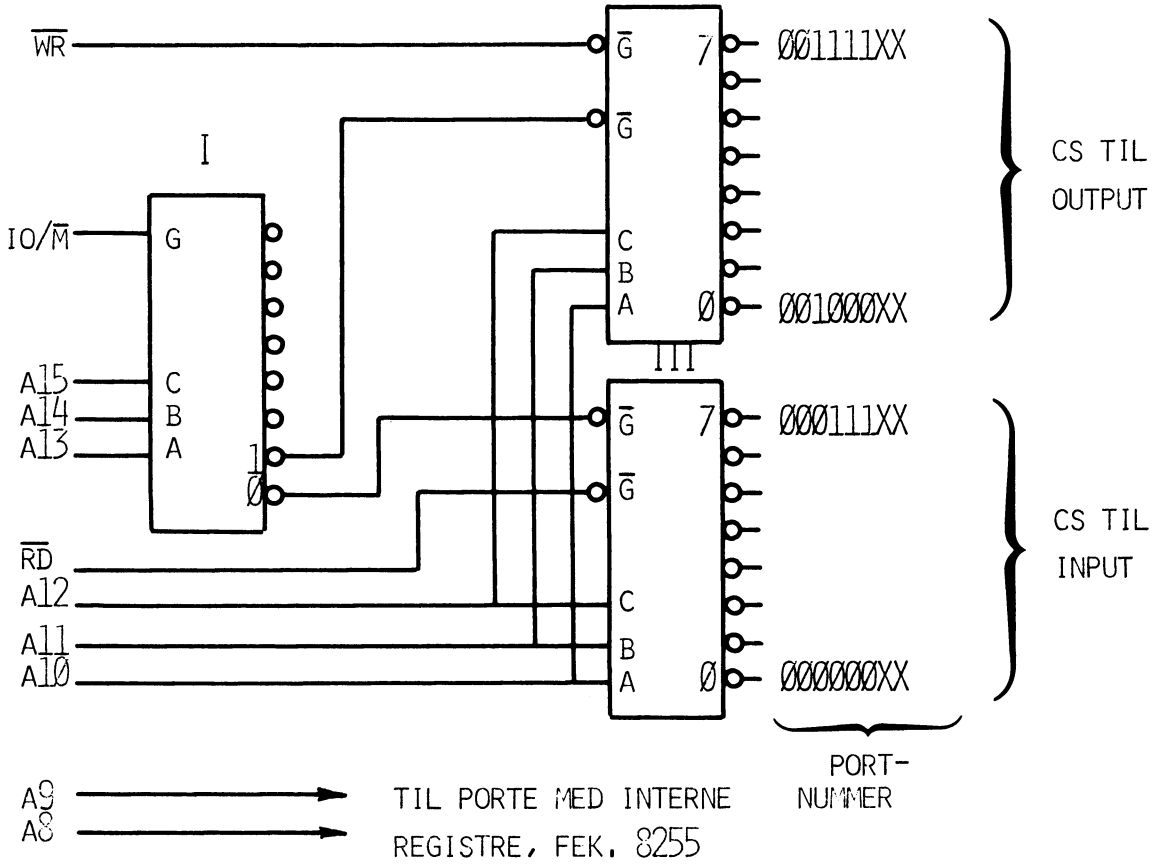
I mindre systemer med mindre end otte porte kan man klare sig med liniær select af portene. Det vil sige at hver Adresseledning vælger en bestemt port.



Portnummer decoder.

Hvis der skal være mere end otte I/O-kredse i systemet anvendes der en decoder til at vælge de enkelte porte. Computerene kommer altså til at indeholde to decodersystemer mellem hvilken der skiftes ved hjælp af IO/$\overline{M}$.

Diagrameksempel.



Decoder I gates ved hjælp af IO/$\overline{M}$, således af decodesystemet kun er aktivt under eksekveringen af en I/O instruktion.

Timing.

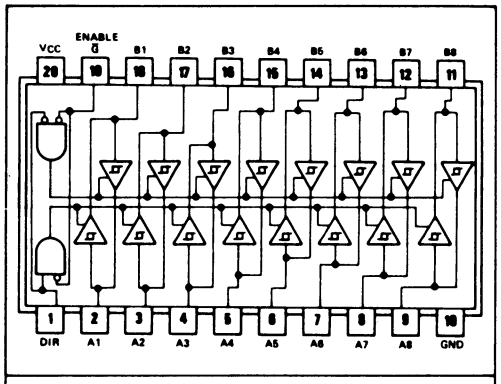
Hvis de anvendte porte er simple latch og buffere kan de sidste decodere i "træet" gates med $\overline{RD}$ og $\overline{WR}$ signalerne fra CPU'en.

Skal decodesystemet derimod anvendes til programmerbare portkredse som f.eks. INTEL 8255A, skal decoderne ikke gates med $\overline{RD}$ og $\overline{WR}$, idet disse signaler skal tilføres 8255A direkte.

Simple portkredse

74LS245 er 8 bidirectionale 3-state buffere med hysterese.Den er anvendelig som inputport.

TYPE	I_{OL} (SINK CURRENT)	I_{OH} (SOURCE CURRENT)
SN54LS245	12 mA	−12 mA
SN74LS245	24 mA	−15 mA



positive logic: see function table

description

These octal bus transceivers are designed for asynchronous two-way communication between data buses. The control function implementation minimizes external timing requirements.

The device allows data transmission from the A bus to the B bus or from the B bus to the A bus depending upon the logic level at the direction control (DIR) input. The enable input ($\overline{G}$) can be used to disable the device so that the buses are effectively isolated.

recommended operating conditions

PARAMETER	SN54LS245			SN74LS245			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−12			−15	mA
Low-level output current, I_{OL}			12			24	mA
Operating free-air temperature, T_A	−55		125	0		70	°C

switching characteristics, V_{CC} = 5 V, T_A = 25°C

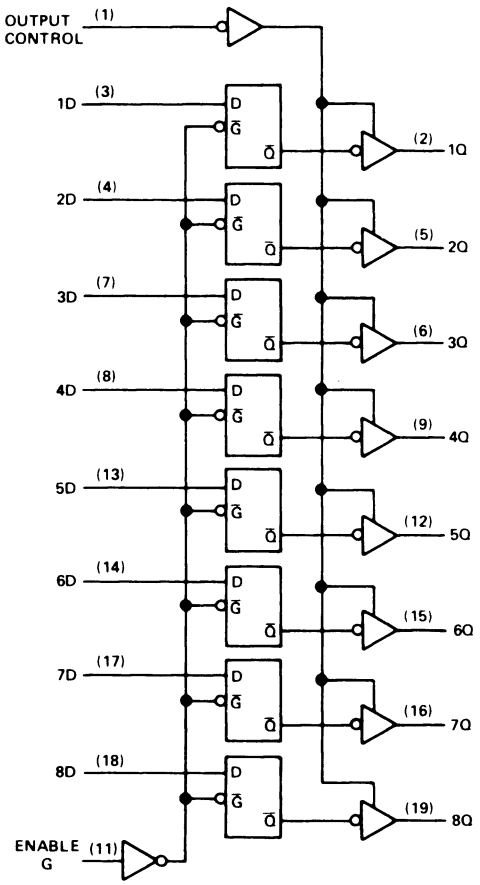
PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Propagation delay time, low-to-high-level output	C_L = 45 pF, R_L = 667 Ω, See Note 2		8	12	ns
t_{PHL}	Propagation delay time, high-to-low-level output	C_L = 45 pF, R_L = 667 Ω, See Note 2		8	12	ns
t_{PZL}	Output enable time to low level	C_L = 45 pF, R_L = 667 Ω, See Note 2		27	40	ns
t_{PZH}	Output enable time to high level	C_L = 45 pF, R_L = 667 Ω, See Note 2		25	40	ns
t_{PLZ}	Output disable time from low level	C_L = 5 pF, R_L = 667 Ω, See Note 2		15	25	ns
t_{PHZ}	Output disable time from high level	C_L = 5 pF, R_L = 667 Ω, See Note 2		15	25	ns

NOTE 2: Load circuit and waveforms are shown on page 3-11.

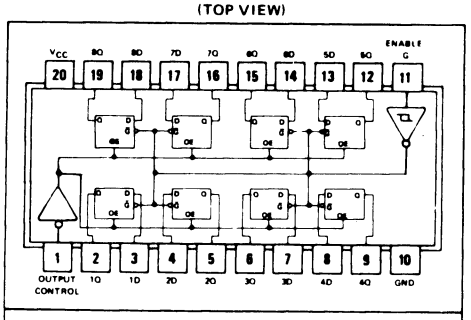
74 LS 373 er 8 latches med 3-state output.

Den kan anvendes som outputport eller inputport med eller uden latch-funktion.

'LS373, 'S373
TRANSPARENT LATCHES



SN54LS373, SN54S373 . . . J PACKAGE
SN74LS373, SN74S373 . . . J OR N PACKAGE
(TOP VIEW)



logic: see function table

'LS373, 'S373
FUNCTION TABLE

OUTPUT CONTROL	ENABLE G	D	OUTPUT
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

description

These 8-bit registers feature totem-pole three-state outputs designed specifically for driving highly-capacitive or relatively low-impedance loads. The high-impedance third state and increased high-logic-level drive provide these registers with the capability of being connected directly to and driving the bus lines in a bus-organized system without need for interface or pull-up components. They are particularly attractive for implementing buffer registers, I/O ports, bidirectional bus drivers, and working registers.

The eight latches of the 'LS373 and 'S373 are transparent D-type latches meaning that while the enable (G) is high the Q outputs will follow the data (D) inputs. When the enable is taken low the output will be latched at the level of the data that was setup.

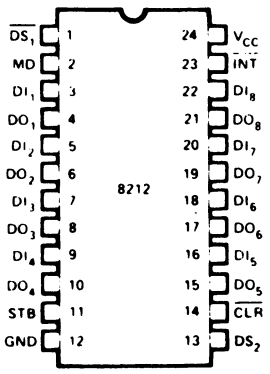
8212 er 8 latches med 3-state output samt interruptlogik.

Den kan anvendes som outputport eller inputport med latchfunktion og service request (interrupt).

The 8212 input/output port consists of an 8-bit latch with 3-state output buffers along with control and device selection logic. Also included is a service request flip-flop for the generation and control of interrupts to the microprocessor.

The device is multimode in nature. It can be used to implement latches, gated buffers or multiplexers. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with this device.

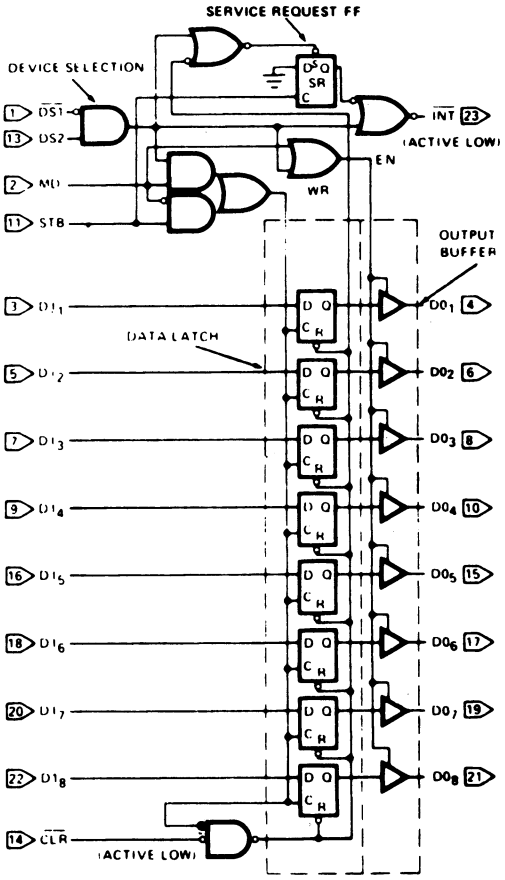
PIN CONFIGURATION



PIN NAMES

DI_1 DI_8	DATA IN
DO_1 DO_8	DATA OUT
$\overline{DS_1}$ DS_2	DEVICE SELECT
MD	MODE
STB	STROBE
$\overline{INT}$	INTERRUPT (ACTIVE LOW)
$\overline{CLR}$	CLEAR (ACTIVE LOW)

LOGIC DIAGRAM

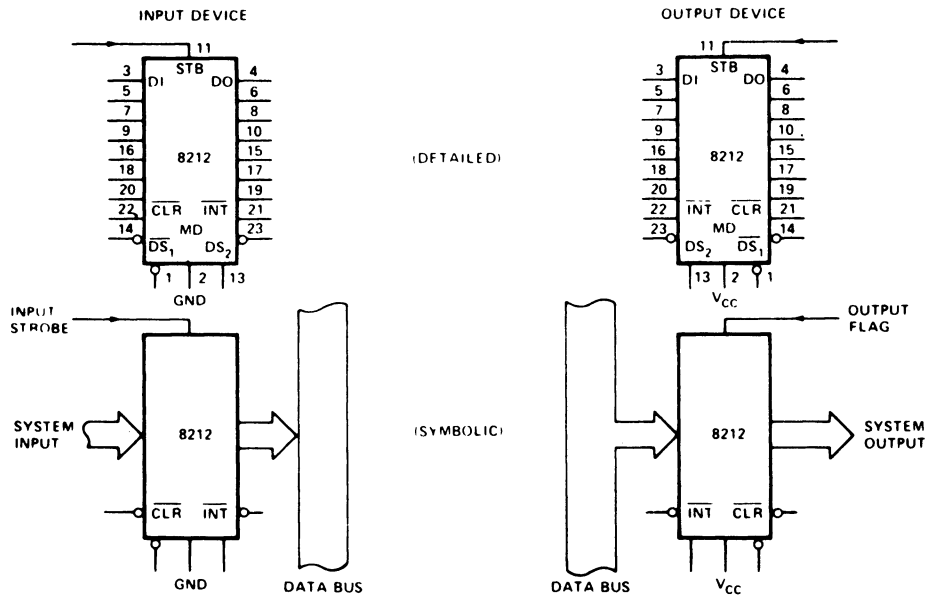


STB	MD	(DS_1 DS_2)	DATA OUT EQUALS
0	0	0	3 STATE
1	0	0	3 STATE
0	1	0	DATA LATCH
1	1	0	DATA LATCH
0	0	1	DATA LATCH
1	0	1	DATA IN
0	1	1	DATA IN
1	1	1	DATA IN

$\overline{CLR}$ - RESETS DATA LATCH
SETS SR FLIP FLOP
(NO EFFECT ON OUTPUT BUFFER)

CLR	($\overline{DS_1}$ DS_2)	STB	*SR	INT
0	0	0	1	1
0	1	0	1	0
1	1	↗	0	0
1	1	0	1	0
1	0	0	1	1
1	1	↗	1	0

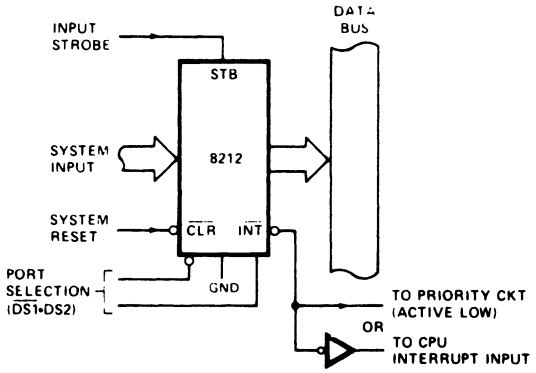
*INTERNAL SR FLIP FLOP

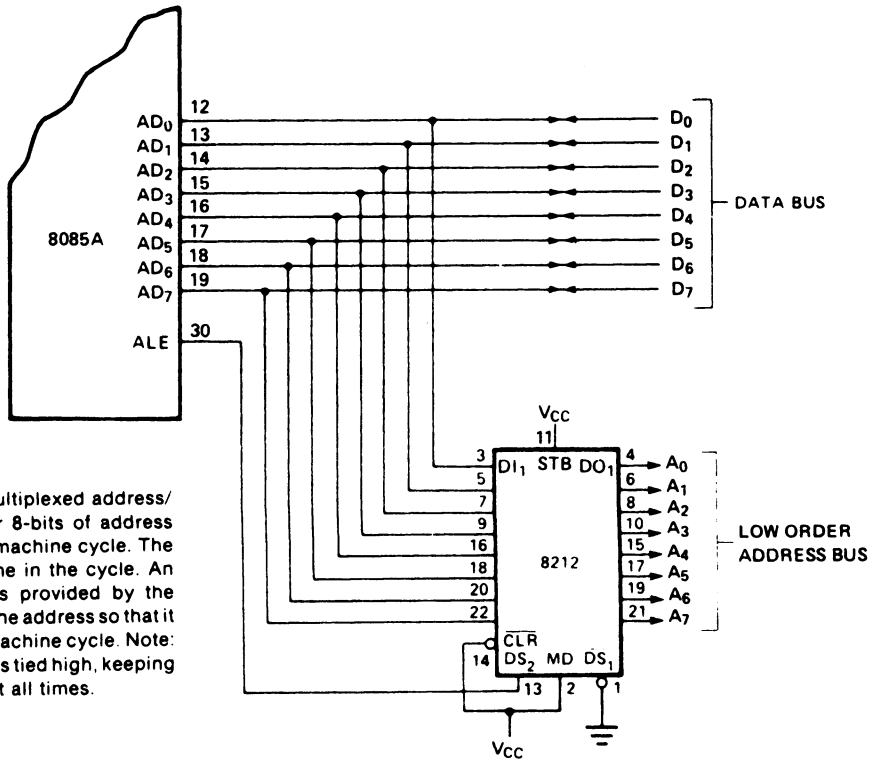


Interrupting Input Port

This use of an 8212 is that of a system input port that accepts a strobe from the system input source, which in turn clears the service request flip-flop and interrupts the processor. The processor then goes through a service routine, identifies the port, and causes the device selection logic to go true — enabling the system input data onto the data bus.

INTERRUPTING INPUT PORT





8085A Low-Order Address Latch

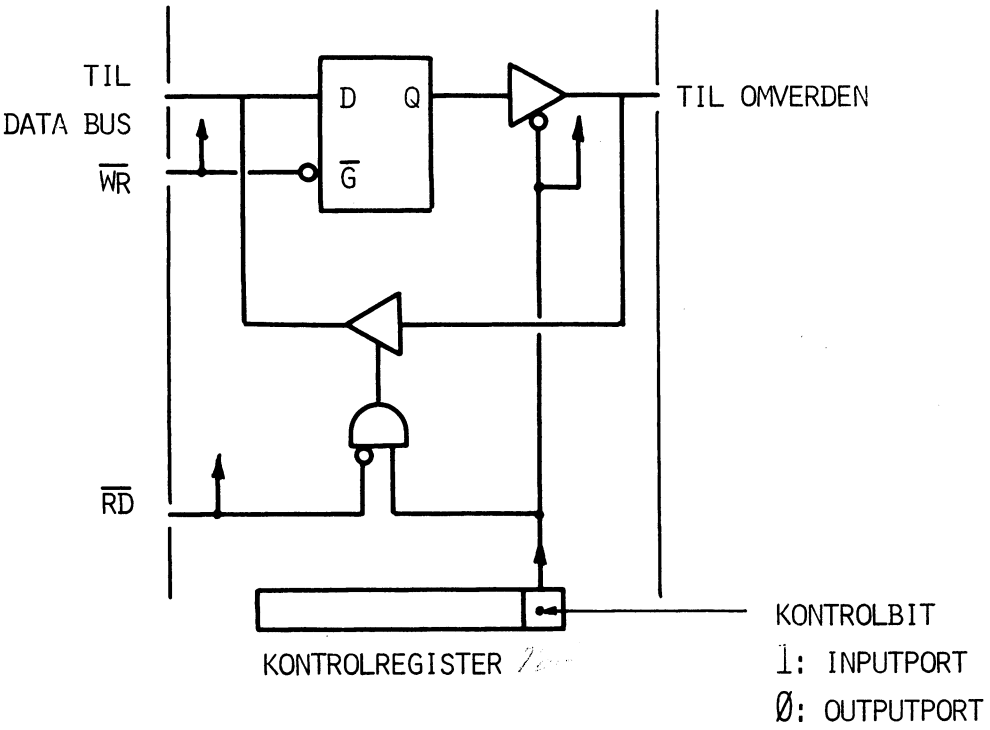
The 8085A microprocessor uses a multiplexed address/data bus that contains the low order 8-bits of address information during the first part of a machine cycle. The same bus contains data at a later time in the cycle. An address latch enable (ALE) signal is provided by the 8085A to be used by the 8212 to latch the address so that it may be available through the whole machine cycle. Note: In this configuration, the MODE input is tied high, keeping the 8212's output buffers turned on at all times.

Programmerbare I/O-kredse.

I princippet er en programmerbar I/O-kreds en kombination af en inputbuffer og en outputlatch.

Hvilken af de to, der skal indkobles bestemmes via et kontrolregister, hvis indhold kan ændres af programmet.

Dette betyder, at et givet ben på kredsen, under programkørslen, kan skiftes fra at være et output til at være et input og omvendt.



Ud over de simple portfunktioner indeholder disse avancerede portkredse ofte handshakelogik og interruptlogik. Andre omfatter timerne og skifteregistre til parallel/serie konvertering og andre igen er kombinationer af porte og hukommelse (RAM eller ROM).

Intel 8255 A.

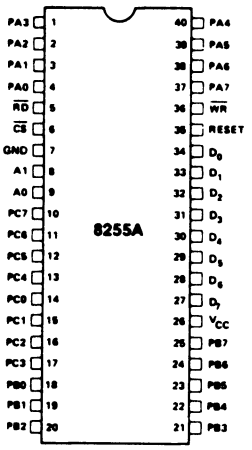
8255A er en programmable peripheral interface-kreds, PPI.

8255A kan arbejde i 3 modes, hvoraf kun mode ø vil blive beskrevet her.

The 8255A is a general purpose programmable I/O device designed for use with both the 8008 and 8080 microprocessors. It has 24 I/O pins which may be individually programmed in two groups of twelve and used in three major modes of operation. In the first mode (Mode 0), each group of twelve I/O pins may be programmed in sets of 4 to be input or output. In Mode 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining four pins three are used for handshaking and interrupt control signals. The third mode of operation (Mode 2) is a Bi-directional Bus mode which uses 8 lines for a bi-directional bus, and five lines, borrowing one from the other group, for handshaking.

Other features of the 8255A include bit set and reset capability and the ability to source 1 mA of current at 1.5 volts. This allows darlington transistors to be directly driven for applications such as printers and high voltage displays.

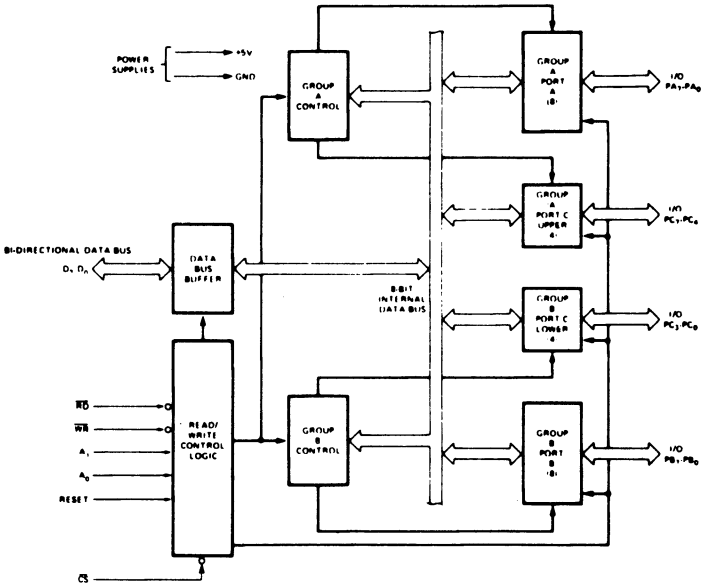
PIN CONFIGURATION



PIN NAMES

D_7–D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	Ø VOLTS

8255A BLOCK DIAGRAM



Read/write controllogikken varetager al overførsel af data og kontrolsignaler, såvel internt som externt. Ved hjælp af A_0 og A_1 vælges de enkelte registre og porte i 8255A.

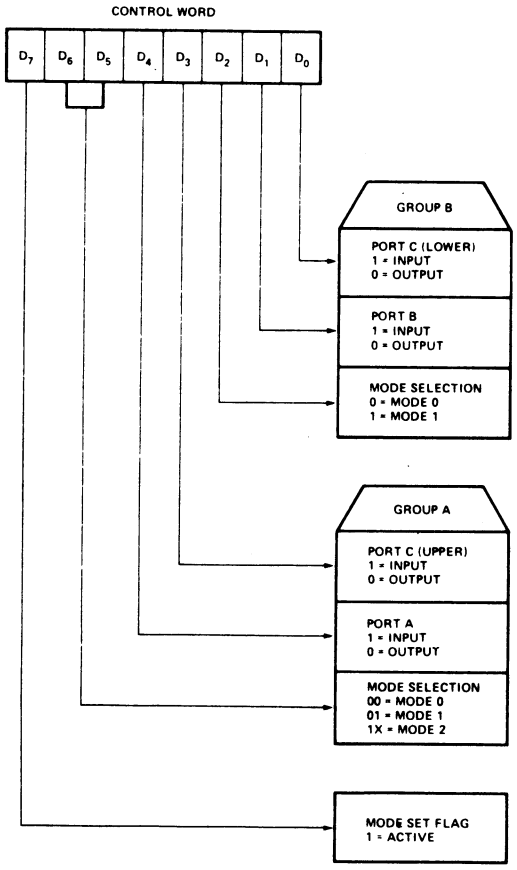
Group A-og group B - control er kredsens kontrolregister i hvilket informationen om mode og dataretning findes.

8255 BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	INPUT OPERATION (READ)
0	0	0	1	0	PORT A → DATA BUS
0	1	0	1	0	PORT B → DATA BUS
1	0	0	1	0	PORT C → DATA BUS
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	DATA BUS → PORT A
0	1	1	0	0	DATA BUS → PORT B
1	0	1	0	0	DATA BUS → PORT C
1	1	1	0	0	DATA BUS → CONTROL
					DISABLE FUNCTION
X	X	X	X	1	DATA BUS → 3-STATE
1	1	0	1	0	ILLEGAL CONDITION
X	X	1	1	0	DATA BUS → 3-STATE

Control reg.

Controlregistret nås ved at skrive til adresse xxx3H.



Eksempel.

Følgende ønskes:

Mode ø, port A som output, port B som input og port C som output.

Controlord = løøøøøløø.

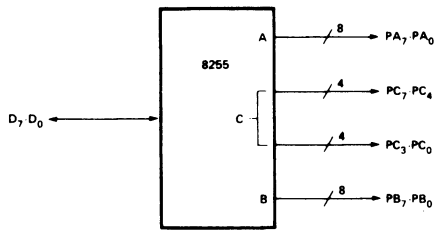
Porten kan sættes op i 16 forskellige kombinationer i mode ø.

MODE 0 PORT DEFINITION CHART

A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

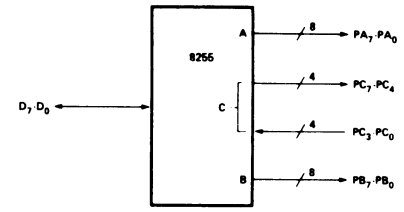
CONTROL WORD #0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	0	0



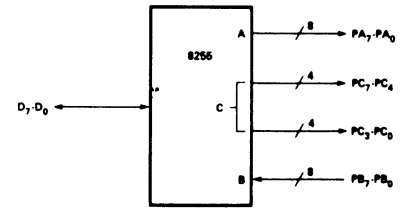
CONTROL WORD #1

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	0	1



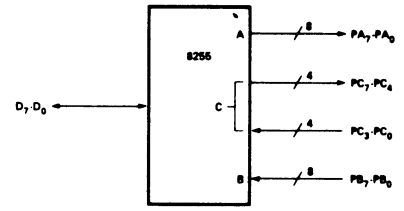
CONTROL WORD #2

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	1	0



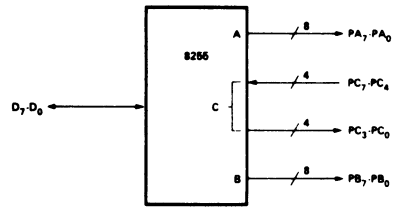
CONTROL WORD #3

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	1	1



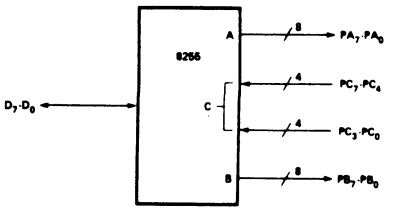
CONTROL WORD #4

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	0



CONTROL WORD #5

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	1



CONTROL WORD #6

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	0

CONTROL WORD #7

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	1

CONTROL WORD #8

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	0

CONTROL WORD #9

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	1

CONTROL WORD #10

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	0

CONTROL WORD #11

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	1

CONTROL WORD #12

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	0	0

CONTROL WORD #13

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	0	1

CONTROL WORD #14

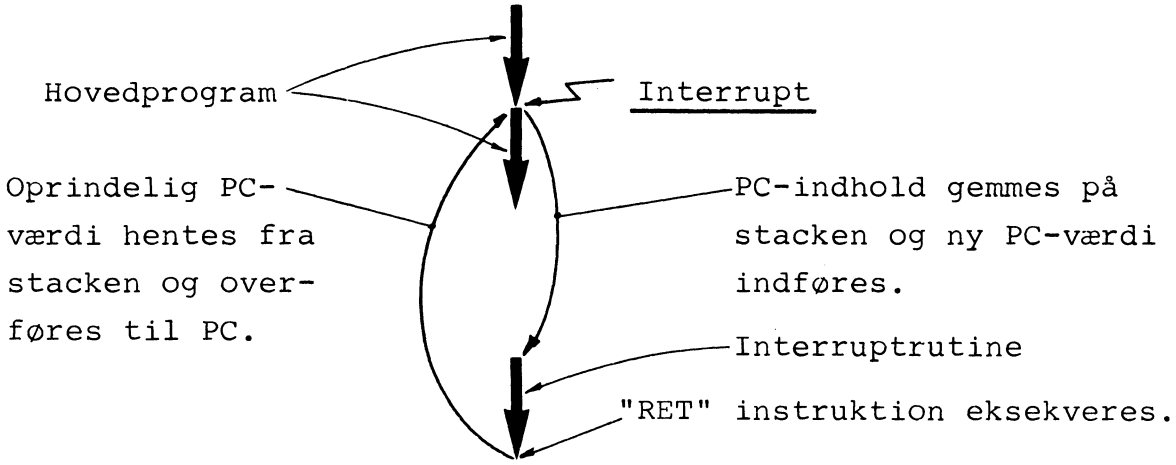
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	1	0

CONTROL WORD #15

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	1	1

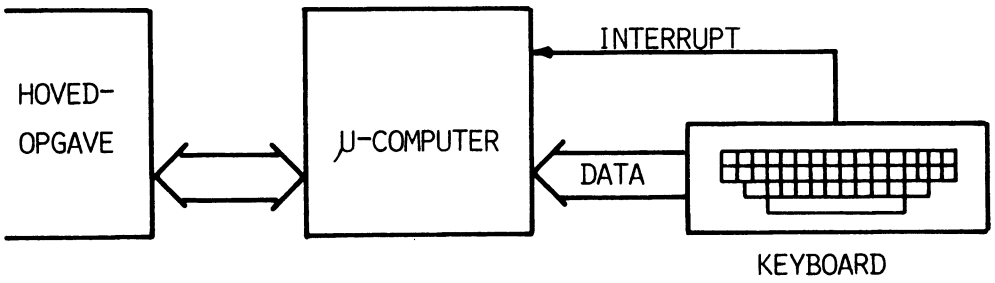
Interrupt, princip.

At interrupte en computer vil sige at afbryde afviklingen af det igangværende program. Efter afbrydelsen gemmes programcounterens indhold i en del af ramlageret, der kaldes stacken, hvorefter PC's indhold skiftes ud med en ny værdi, der peger på begyndelsen af et andet program. CPU'en vil derefter fortsætte med at eksekvere dette. Det program CPU'en hopper til ved en interrupt kaldes ofte interruptrutinen.
Interruptrutinen afsluttes med en RET (return) instruktion. Ved dennes eksekvering udskiftes PC's indhold med den oprindelige værdi, der har været gemt på stacken. Herefter fortsætter CPU'en på det sted i det oprindelige program, hvor den blev afbrudt.



Eksempel på anvendelse.

Figuren viser en µ-computer, der ud over sin hovedopgave skal modtage karakterer fra et keyboard (KB). Ved at lade KB generere en interrupt når der foretages en indtastning, behøver computeren ikke beskæftige sig med keyboardet medmindre der foretages en indtastning. Derved bliver der mere tid til computerens egentlige opgaver, idet betjeningen af KB vil beslaglægge et minimum af den samlede computertid.



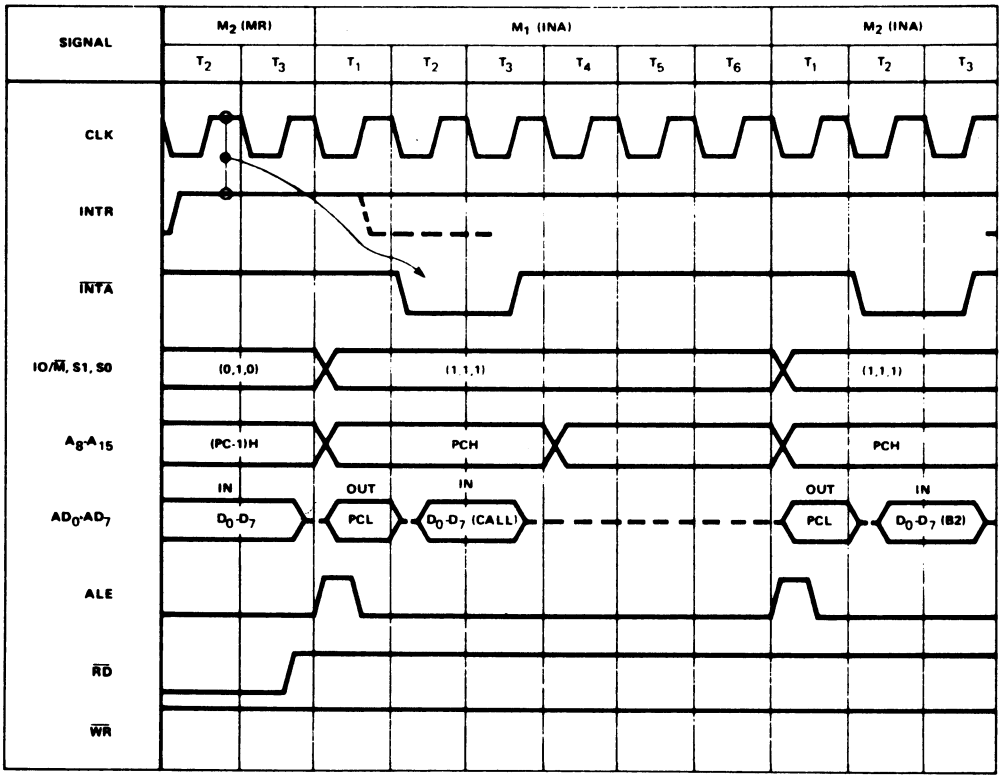
8085's interrupts.

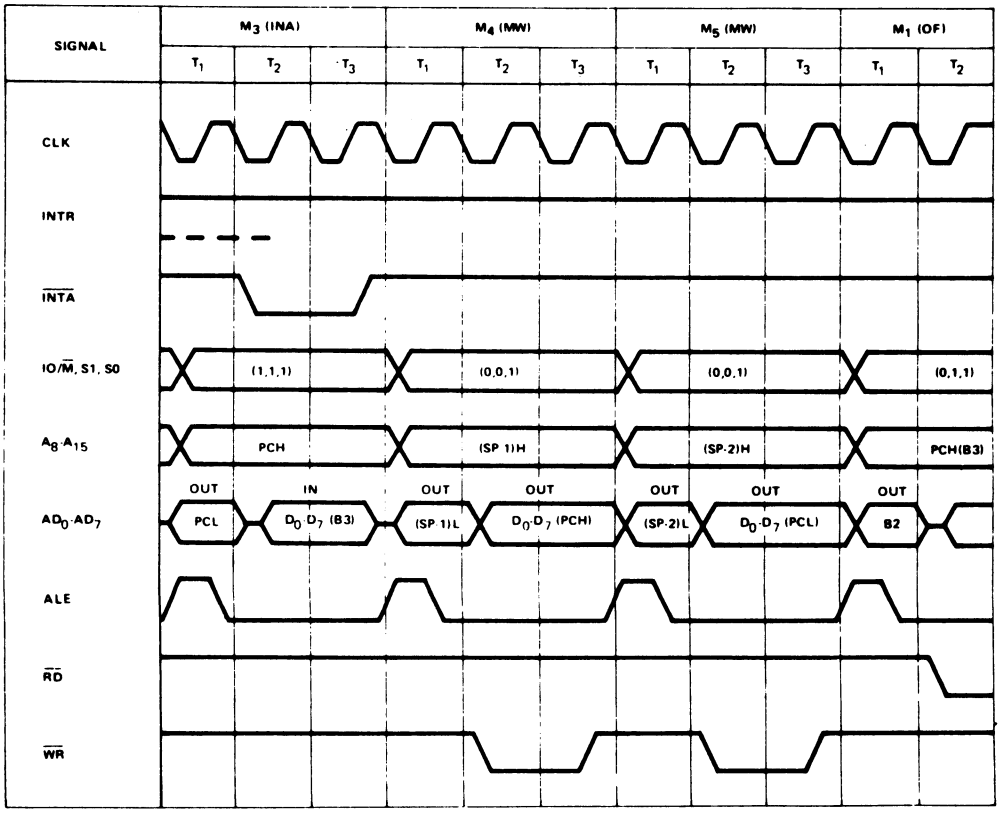
INTR:

De fem interrupts på 8085 er af tre typer. INTR svarer til 8085's INT funktion, det vil sige at det kan maskes ved hjælp af en EI-instruktion (interrupt enable), eller en DI-instruktion (disable interrupt), og CPU'en henter en RST instruktion, der samtidigt skal tilføres databussen.
Dette bevirker et hop til en ud af otte faste hukommelsespladser, hvor programeksekveringen fortsætter.

INTR kan også kontrolleres ved hjælp af 8259, der er en programmerbar interruptcontroler, som kan generere en CALL-instruktion i stedet for RST, hvorved det er muligt at hoppe til en subrutine på en vilkårlig adresse inden for hukommelsen.

INTERRUPT ACKNOWLEDGE MACHINE CYCLES (WITH CALL INSTRUCTION IN RESPONSE TO INTR)



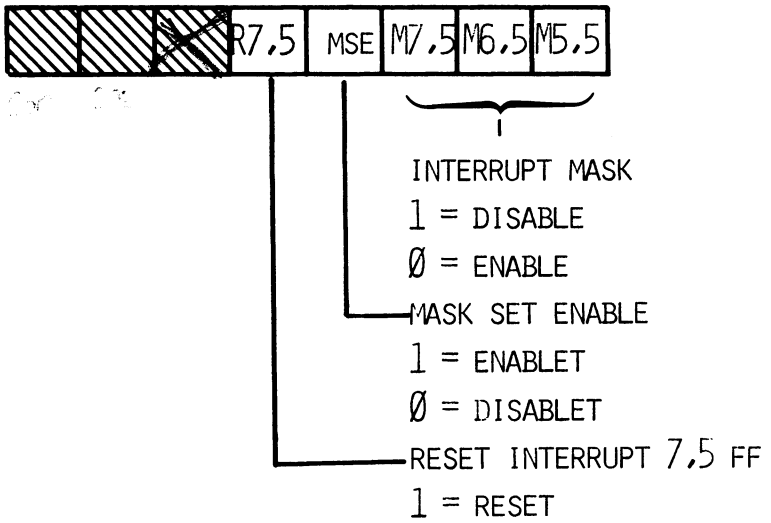


RST 5.5, RST 6.5, RST 7.5.

RST 5.5, RST 6.5 og RST 7.5 er forskellige fra INTR ved at de kan maskes ved hjælp af en SIM-instruktion som enabler eller disabler disse interrupts på basis af data i accumulatoren inden SIM-instruktionen. Den øjeblikkelige status af interruptmaskerne kan læses ved hjælp af en RIM-instruktion.

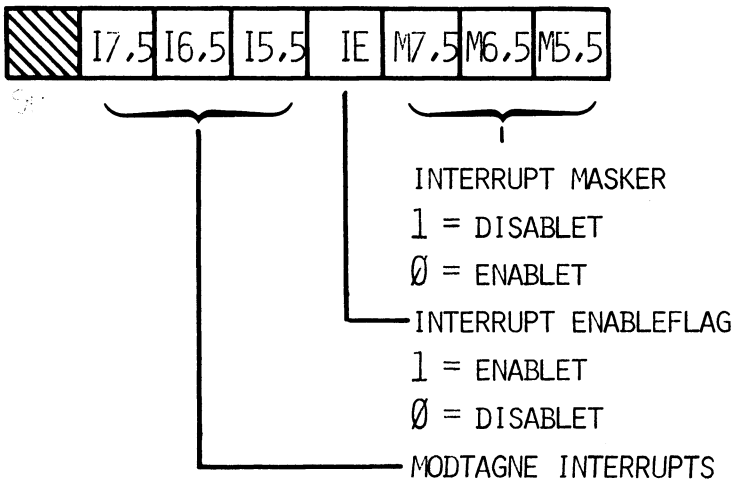
SIM - SET INTERRUPT MASK

ACCUMULATORINDHOLD INDEN SIM



RIM - READ INTERRUPT MASK

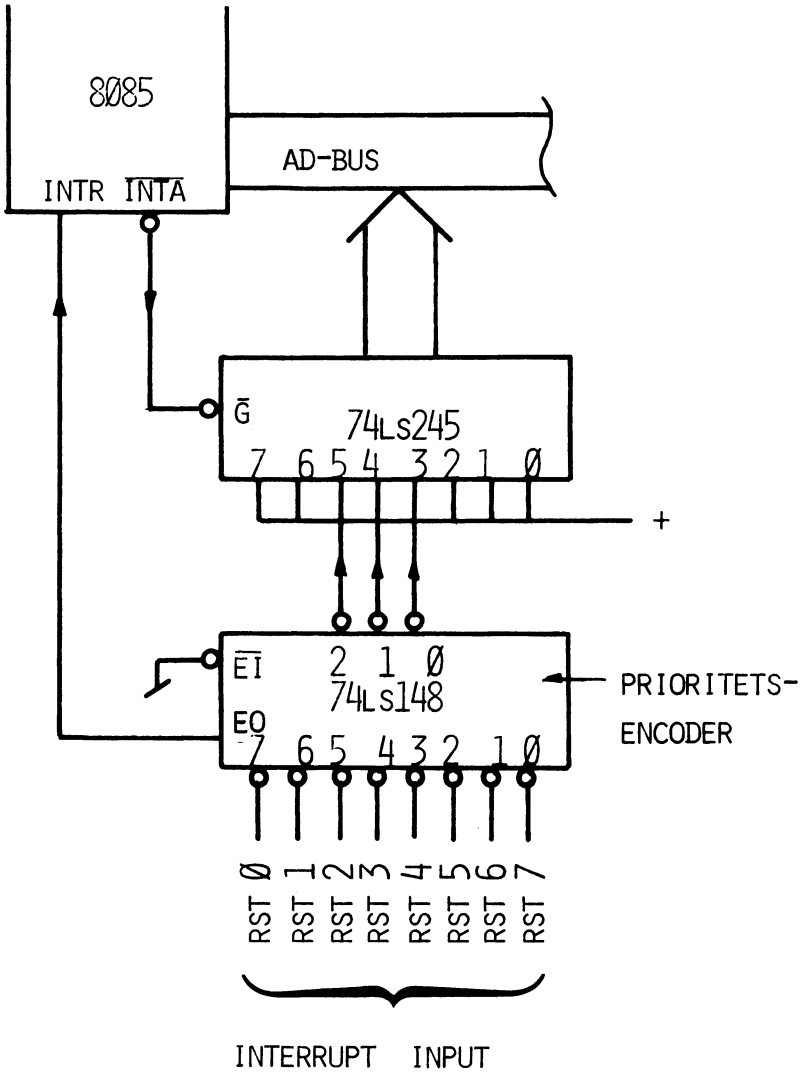
ACCUMULATORINDHOLD EFTER RIM



RST 5.5, RST 6.5 og RST 7.5 enables og disables også ved hjælp af EI- og DI-instruktionerne.

INTA:

Interrupt acknowledge er CPU'ens svarsignal på en akcepteret INTR.
INTA har samme timing som RD og genereres i stedet for denne i en interrupt acknowledge maskincycle.



Diagrammet viser et logisk kredsløb til håndtering af prioriteret RST.
74LS148 er en prioritetsencoder med højeste prioritet på input 7.
Når en af inputtene arkiveres (aktiv low) går EO outputtet high og leverer et interruptinput til CPU'en.
Encoderens output skal lægges ind som bit 3, 4 og 5 i RST-instruktionen og de resterende bit skal være logisk 1.

Sammensætningen af RST-instruktionen sker i en 74LS245 portkreds, hvis enableinput styres af $\overline{INTA}$.

INTR, RST 5.5 og RST 6.5 er niveaufølsomme input, hvilket betyder at de vil blive opfattet af CPU'en hvis de holdes på high-niveau. RST 7.5 er kantrigget ved hjælp af en intern flip-flop der settes når der kommer en stigende flanke på på RST 7.5 inputtet. Signalet behøver herefter ikke holdes højt, idet flip-floppen vil forblive sat indtil den bliver clearet af en af følgende hændelser.

1) CPU'en reagerer på interrupten og sender et internt resetsignal til RST 7.5 flip-floppen.

2) CPU'en modtager et RESET IN signal inden den når at reagere på RST 7.5 interrupten.

3) CPU'en eksekverer en SIM-instruktion med bit 4 i accumulatoren sat til 1.

TRAP-interrupt.

Den tredie type hardware interrupt er TRAP. Dette input kan ikke disables på nogen måde.
Modtagelsen af en stigende flanke på TRAP-inputtet vil trigge CPU'ens interruptsekvens, blot skal TRAP-inputtet holdes højt indtil CPU'en har akcepteret inputtet.

Interrupternes samplingtidspunkt.

Samplingen af alle interruptinput finder sted på den faldende clockflanke en clockperiode før slutningen af den instruktion i hvilken der opstod et aktivt interruptinput.
For at blive aksepteret skal der være en gyldig interrupt mindst 160 nsek inden samplingstidspunktet.
Det betyder, at INTR, TRAP, RST 5.5 og RST 6.5 skal holdes high i mindst 17 clockperioder plus 100 nsek, idet det antages at interrupten kom "et hårsbredde" for sent til at blive aksepteret ved en given instruktion, og den næste instruktion er en 18 states CALL. Dette forudsætter iøvrigt, at der ikke er nogen wait-states.

Interrupternes prioritet.

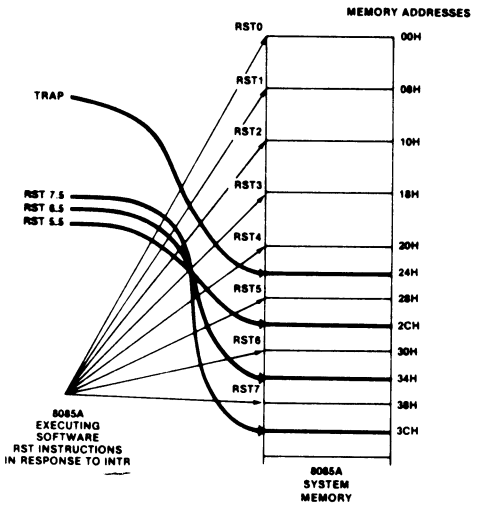
Interruptinputtenes karakteristikker og indbyrdes prioritet er vist i følgende tabel.

Navn	Prioritet	Hop adr.(1)	Trigges af
TRAP	1.	ØØ24H	
RST 7.5	2.	ØØ3cH	
RST 6.5	3.	ØØ34H	
RST 5.5	4.	ØØ2cH	
INTR	5.	(2)	

(1) Programcounterens indhold skubbes på stacken inden CPU'en barncher til den angivne adr.

(2) Afhænger af den tilførte instruktion.

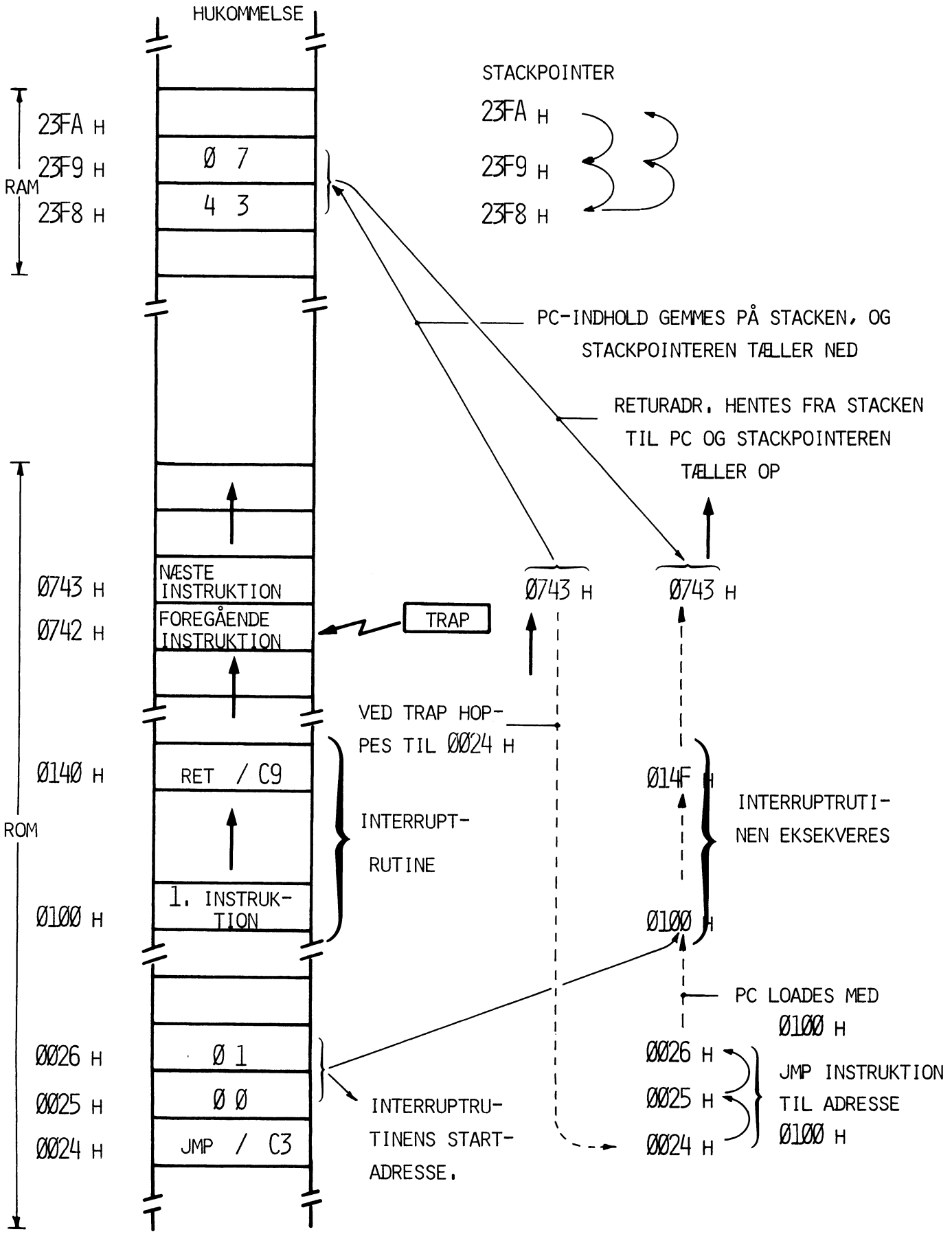
Handware og software RST HOP adresser.



Eksempel på forløb af et TRAP interrupt.

Se tegning.

Under eksekveringen af hovedprogrammet er CPU'en nået til adr. ø742H, da der modtages et trap interrupt.
Når interruptet er erkendt af CPU'en, fortsættes med en stackoperation hvorunder programcounterens indhold lagres på to på hinanden følgende pladser i stacken.
Derefter udskiftes programcounterens indhold med øø24H og CPU'en går ind i en M1 maskincycle.
På adr. øø24H, øø25H og øø26H ligger der en JMP-instruktion til adr. øløøH der er interruptrutinens startadresse.
Som den sidste instruktion i interruptrutinen findes en RET (return).
Denne bevirker at CPU'en foretager en stackoperation hvorunder indholdet af de pladser stackpointeren peger på overføres til programcounteren.
Programcounteren indeholder nu adressen på den instruktion i hovedprogrammet, der ligger efter det sted hvor interrupten opstod. CPU'en går nu ind i en M1 - maskincycle og fortsætter eksekveringen af hovedprogrammet.



RS 232c.

RS 232c er en standard der beskriver hvorledes en modem og en terminal eller computer kan kobles sammen, når der anvendes seriel transmission. Med små afvigelser fra standarden anvendes den ofte ved sammenkobling af en printer eller skærmterminal og en computer. Computeren vil i dette sammenhæng benævnes som data set og printer eller skærmterminal som data terminal.

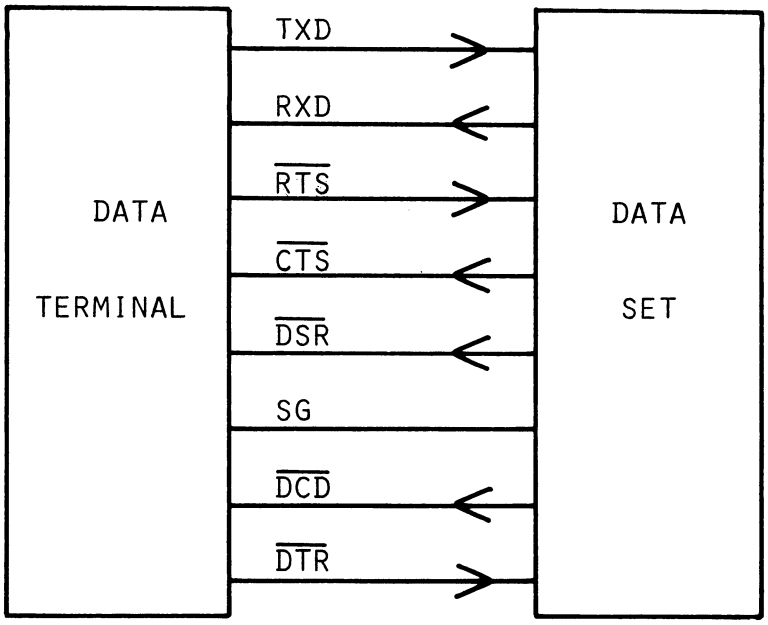
Standarden RS232c vil her kun blive beskrevet i det omfang, det er nødvendigt for at sammenkoble en printer eller skærmterminal og computer der arbejder i simplex eller half duplex.

Følgende signaler indgår i sammenkoblingen:

TxD	transmitted data
RxD	received data
RTS	request to send
CTS	clear to send
DSR	data set ready
SG	signal ground
DCD	data carrier detect
DTR	data terminal ready

På linierne anvendes følgende logiske niveauer

+3V til +12V = "0"
-3V til -12V = "1"

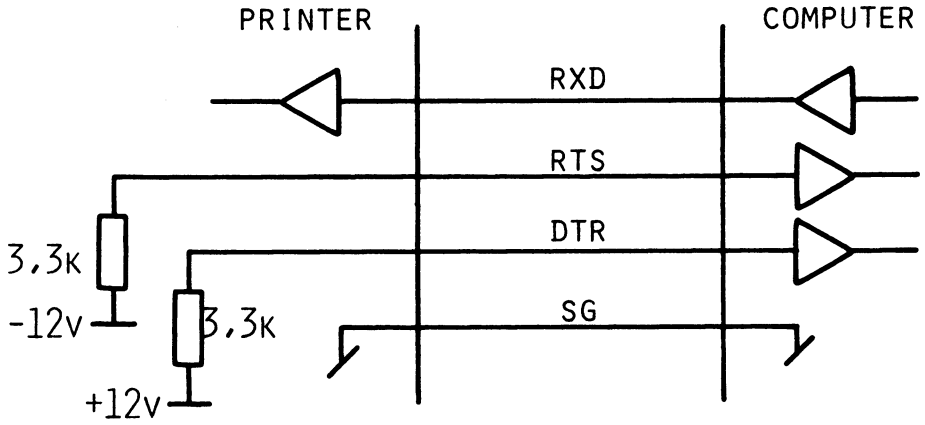


$\overline{DTR}$ og $\overline{DSR}$ er de overordnede signaler, med logisk "o" er henholdsvis data terminal og data set klar d.v.s. der er forsyningsspænding på, data terminalen kan modtage data, og data set'et kan sende.
$\overline{DCD}$ er et specielt modem signal, der indikerer at data carrien på telefonlinien har rigtig frekvens og amplitude, det signal forbindes i data set'et til logisk "o".
I hvilestilling står dataterminalen klar til at modtage data på RxD, og data set'et (computeren) kan sende data via RxD til data terminalen.
Dette kan fortsættes indtil data terminalen ikke længere vil modtage data ($\overline{DTR}$ = "1").
Når data terminalen ønsker at sende til data set'et anmodes om lov til at sende ved at sende $\overline{RTS}$ = "o" til data set'et, når data set'et er klar til at modtage, sendes $\overline{CTS}$ = "o" til data terminalen, hvorefter terminalen kan sende data via TxD. Når terminalen ikke længere ønsker at sende sættes $\overline{RTS}$ = "1", data set'et vil med $\overline{CTS}$ indikerer om set'et er klar til at modtage fra terminalen.

Sammenkobling af printer og computer.

Forestiller vi os, at en printer skal skrive de data en computer afleverer, betragtes printer som data terminal og computer som dataset.
En simpel printer har ingen kontrolsignaler, derfor forbindes printeren så den melder klar, når der er forsyningsspænding tilsluttet printeren.

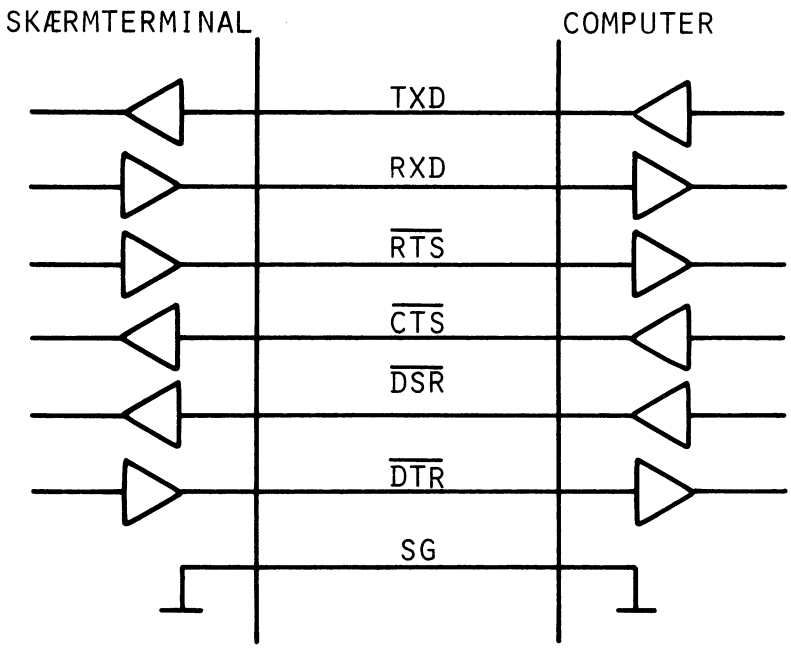
$\overline{DTR}$ = "o" angiver at printer er tændt..



Da printeren ikke afleverer data til computeren, så skal $\overline{RTS}$ forbindes til logisk "1".

Sammenkobling af skærmterminal og computer

En skærmterminal kan både sende og modtage data, derfor skal anvendes flere af kontrolsignalerne. Computeren sender data på RxD og skærmterminalen på TxD.



$\overline{DTR}$ fortæller computeren at skærmterminalen er klar til at modtage eller sende data.

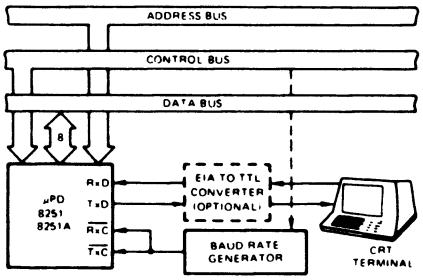
$\overline{DSR}$ fortæller skærmterminalen at computeren er klar til at modtage et RTS signal eller aflevere data til skærmterminalen.

$\overline{RTS}$ er et signal fra terminalen der anmoder computeren om tilladelse til at sende.

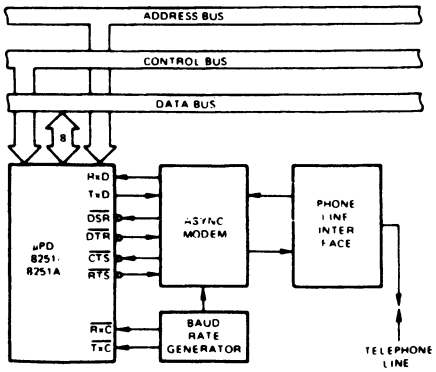
$\overline{CTS}$ er computerens svar på RTS når computeren er klar til at modtage.

USART

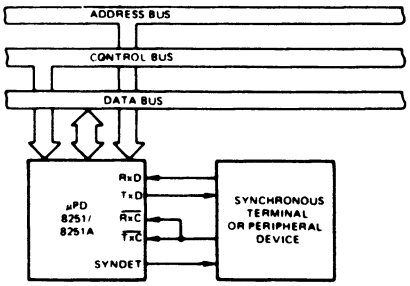
En USART er et integreret kredsløb der omsætter parallelle data til serielle data og omvendt.
USART står for Universal Synchronous/Asynchronous Receiver/Transmitter d.v.s. kredsløbet kan programmeres til at sende/modtage synkron og asynkrone serielle data.
Kredsløbet kan også arbejde med kontrolsignaler som beskrevet under RS232 dog ofte med signalniveauer på o-5v (TTL niveauer).
En 8251A er en USART der er programmerbar og kan anvendes ved kommunikation mellem en computer og en skærmterminal, synkron og asynkron modem's eller anden perifer enhed der arbejder med RS232 signaler.



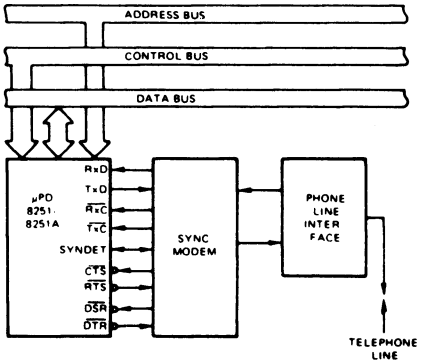
ASYNCHRONOUS SERIAL INTERFACE TO CRT TERMINAL, DC to 9600 BAUD



ASYNCHRONOUS INTERFACE TO TELEPHONE LINES

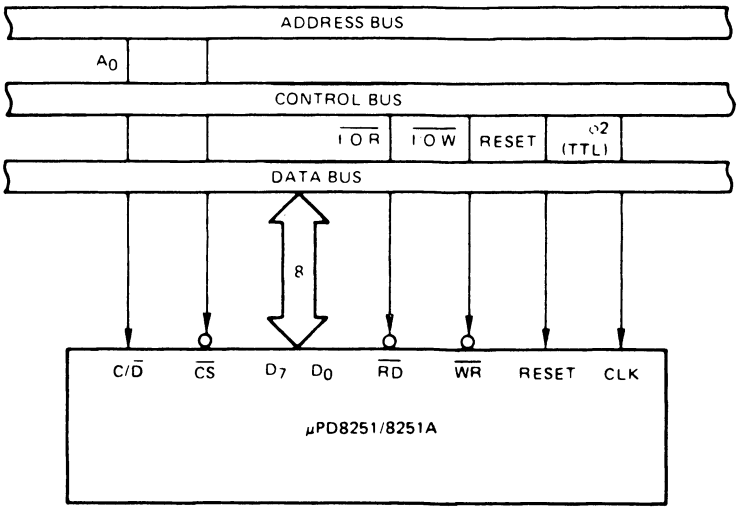


SYNCHRONOUS INTERFACE TO TERMINAL OR PERIPHERAL DEVICE



SYNCHRONOUS INTERFACE TO TELEPHONE LINES

Kredsløbet forbindes til computerens data bus med data I/o linierne. $\overline{RD}$, $\overline{WR}$, RESET og Clk forbindes til kontrolbussen, C/$\overline{D}$ forbindes til A_0 og $\overline{CS}$ til et chip-selectsignal.

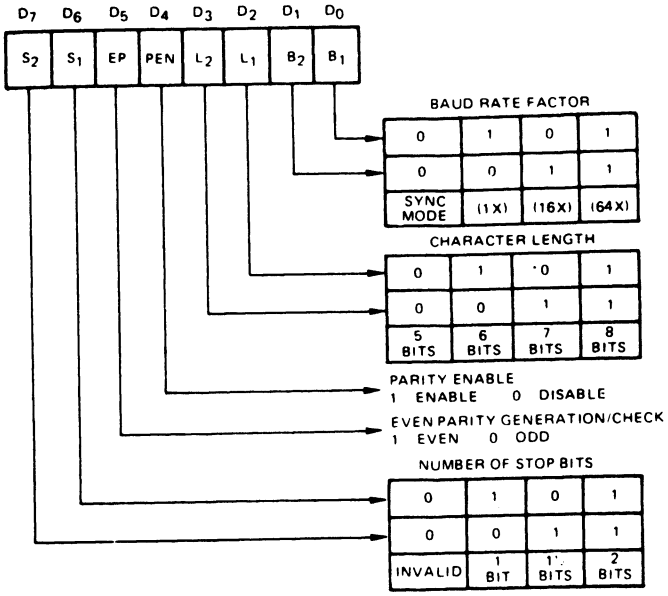


C/$\overline{D}$ bestemmer om data der læses/skrives er i control eller data mode.

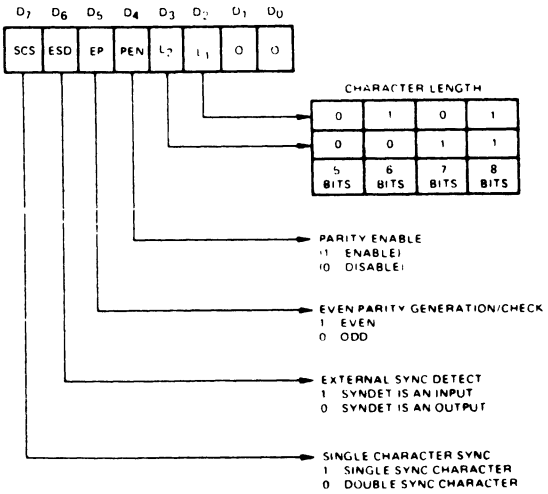
Mode control ord.

Efter reset, enten hardware eller software reset, forventer 8251A at modtage et mode control ord, der programmerer sync/async mode, karakterlængde, paritet kontrol, antallet af stopbit ved async mode, karakter længde, paritet kontrol, antallet af stopbit ved async mode eller antal sync karakterer ved sync mode.

MODE INSTRUCTION
FORMAT
ASYNKRON MODE



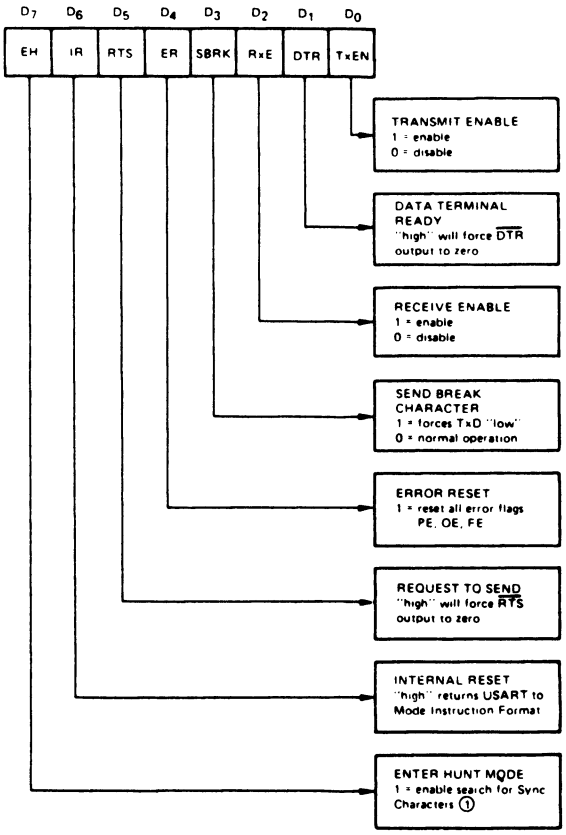
MODE INSTRUCTION
FORMAT
SYNKRON MODE



Async mode.

I async mode efterfølges mode control ordet med et kommando ord der skal resette diverse flag i status ordet der kan læses når C/$\bar{D}$ = 1. Control ordet kan også modtage og sende enable bit.

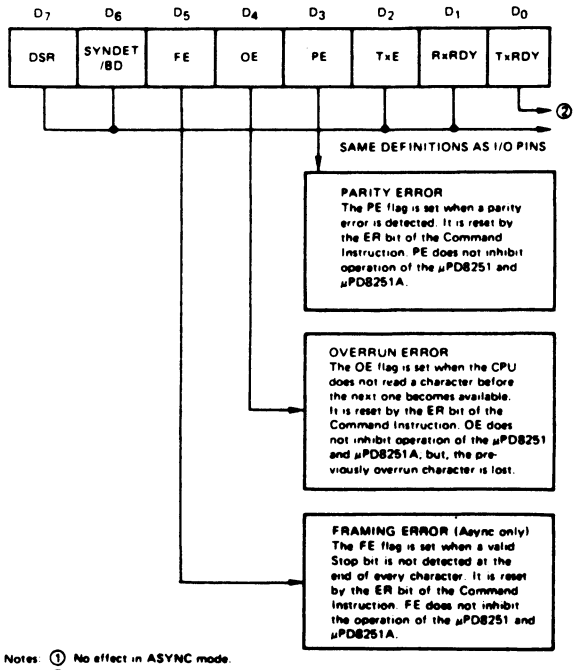
COMMAND INSTRUCTION FORMAT



(1) anvendes ikke i async mode.

Et ord, der skal sendes, indlæses i data mode C/$\overline{D}$=o og IC'en sætter selv start, paritet og stop bit på ordet. Når hardware signalet $\overline{CTS}$ er low og TxEN bitet i command ordet er sat til 1 sendes ordet. Når ordet er fjernet fra data bus bufferen og senderen er klar til næste ord sættes TxRDY i status ordet lig 1.
Når der skal modtages skal DTR, RxE og ER sættes, hvorefter modtageren er klar. Efter modtagelse af et ord vil RxRDY benet/bit blive høj, processoren skal nu læse det modtagne ord, hvorved RxRDY automatisk resettes. Dersom ordet ikke læses inden det næste ord er modtaget vil OE (overflow error) blive sat i status ordet.

STATUS READ FORMAT



Notes: ① No effect in ASYNC mode.
② TxRDY status bit is not totally equivalent to the TxRDY output pin, the relationship is as follows:
TxRDY status bit = DB Buffer Empty
TxRDY (pin 15) = DB Buffer Empty • $\overline{CTS}$ • TxEn

Syncron mode.

Efter mode control ordet, der definerer synchron mode, skal synkronisering karaktererne programmeres, derefter vil efterfølgende control ord blive opfattet som kommando ord.
Betingelsen for at sende, er at $\overline{CTS}$ er low og TxEN bit'et er high, derefter skal data indlæses hver gang TxRDY bit'et er high. Dersom der ikke er indlæst data inden senderen er færdig med sending af sidste data ord vil sync karaktererne blive udsendt.
Ved syncron modtagelse kan karaktersynkronisering enten være intern eller extern. Ved extern skal SYNDET holdes high en hel clock periode på RxC, hvorefter modtageren er synkroniseret.
I intern mode startes modtageren i en søgestilling (HUNT mode) ved at sætte EH bit'et, og RxE bit'et vil modtageren søge efter SYNC karaktererne, når de er fundet gåt SYNDET ben/bit'en high, dette fortæller processoren, at modtageren har fundet SYNC karaktererne.

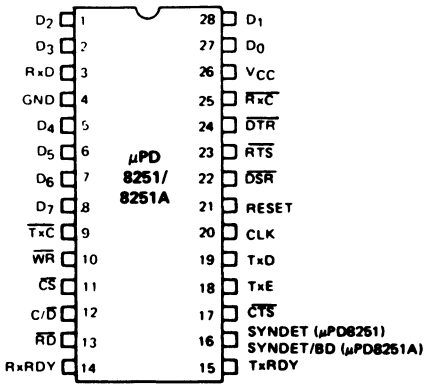
SYNDET bliver automatisk reset ved læsning af statusordet. Efter synkroniseringen fortæller modtageren med RxRDY hver gang der er en karakter klar til processoren.
Paritets error og overflow error bit'ene indikerer også fejl i syncmode, hvorimod Frame error (FE) ingen virkning har i sync mode.

Clocksignal.

Clocksignalerne skal forbindes til TxC for senderen og RxC for modtageren. I sync mode angiver den frekvens band rate'n direkte, i async mode kan denne frekvens deles med 1, 16 eller 64, dette angives med to bit i mode ordet.
CLK signalet skal være mindst 3o gange hurtigere end RxC og TxC signalerne.

Ben forbindelser.

PIN CONFIGURATION

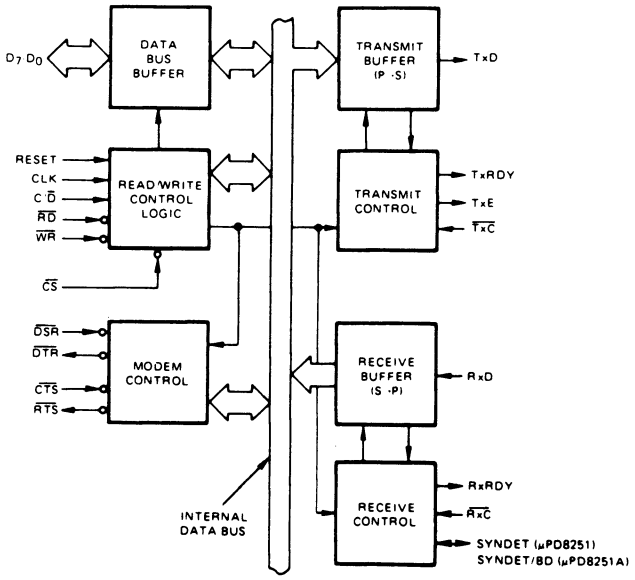


PIN NAMES

D_7-D_0	Data Bus (8 bits)
C/D	Control or Data is to be Written or Read
RD	Read Data Command
WR	Write Data or Control Command
CS	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
TxC	Transmitter Clock (TTL)
TxD	Transmitter Data
RxC	Receiver Clock (TTL)
RxD	Receiver Data
RxRDY	Receiver Ready (has character for 8080)
TxRDY	Transmitter Ready (ready for char. from 8080)
DSR	Data Set Ready
DTR	Data Terminal Ready
SYNDET	Sync Detect
SYNDET/BD	Sync Detect/Break Detect
RTS	Request to Send Data
CTS	Clear to Send Data
TxE	Transmitter Empty
V_{CC}	+5 Volt Supply
GND	Ground

Blokdiagram

BLOCK DIAGRAM



```
MACRO-80 3.36   17-Mar-80       PAGE    1

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;                                  ;
                                ;          ØVEPROM 1               ;
                                ;                                  ;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
2400                            STACKT  EQU     2400H
2000                            RAMLA   EQU     2000H
                                ;
                                ;
                                        .PHASE 00
0000    31 2400                 BEGYND: LXI     SP,STACKT
0003    C3 0100'                        JMP     START
                                        .DEPHASE
                                        ORG     100H
0100'   3A 0200'                START:  LDA     TABEL
0103'   3A 0201'                        LDA     TABEL+1
0106'   3A 0202'                        LDA     TABEL+2
0109'   3A 0203'                        LDA     TABEL+3
010C'   00                              NOP
010D'   00                              NOP
010E'   00                              NOP
010F'   00                              NOP
0110'   3A 2000                 START2: LDA     RAMLA
0113'   3A 2001                         LDA     RAMLA+1
0116'   3A 2002                         LDA     RAMLA+2
0119'   3A 2003                         LDA     RAMLA+3
011C'   00                              NOP
011D'   00                              NOP
011E'   00                              NOP
011F'   00                              NOP
0120'   3A 0200'                START3: LDA     TABEL
0123'   32 2000                         STA     RAMLA
0126'   3A 0201'                        LDA     TABEL+1
0129'   32 2001                         STA     RAMLA+1
012C'   00                              NOP
012D'   00                              NOP
012E'   00                              NOP
012F'   00                              NOP
0130'   3A 2000                 START4: LDA     RAMLA
0133'   3A 2001                         LDA     RAMLA+1
0136'   3A 2002                         LDA     RAMLA+2
0139'   3A 2003                         LDA     RAMLA+3
013C'   00                              NOP
013D'   00                              NOP
013E'   00                              NOP
013F'   00                              NOP
```

```
MACRO-80 3.36   17-Mar-80       PAGE    1-1

47      0140'   00                      START5: NOP
48      0141'   C3 0140'                        JMP     START5
49                                      ;
50                                      ;
51                                      ;
52                                              ORG     200H
53      0200'   03 02 01 00             TABEL:  DB      3,2,1,0
54                                              END     BEGYND
```

```
        MACRO-80 3.36   17-Mar-80       PAGE    S


Macros:

Symbols:
BEGYND  0000    RAMLA   2000    STACKT  2400    START   0100'
START2  0110'   START3  0120'   START4  0130'   START5  0140'
TABEL   0200'


No  Fatal error(s)
```

BEGYND	11#	54									
RAMLA	7#	23	24	25	26	32	34	39	40	41	42
STACKT	6#	11									
START	12	15#									
START2	23#										
START3	31#										
START4	39#										
START5	47#	48									
TABEL	15	16	17	18	31	33	53#				

```
MACRO-80 3.36                   PAGE    1


                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;                                  ;
                                ;         ØVEPROM2                 ;
                                ;                                  ;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;
                                ;
  2400                          STACKT  EQU     2400H
  1800                          RVEC    EQU     1800H
  1802                          DISP1   EQU     1802H
  1803                          DISP2   EQU     1803H
  0001                          INPUT   EQU     1
                                ;
                                ;
  0000'   31 2400                       LXI     SP,STACKT
  0003'   2A 1800                       LHLD    RVEC
  0006'   E9                            PCHL
                                ;
                                ;
                                        ORG     3CH
  003C'   F5                            PUSH    PSW
  003D'   C3 0111'                      JMP     WHRITE
                                ;
                                ;
                                        ORG     100H
  0100'   3E 0B                 READ:   MVI     A,0BH
  0102'   30                            SIM
  0103'   FB                            EI
  0104'   2A 1800               READ1:  LHLD    RVEC
  0107'   7E                            MOV     A,M
  0108'   22 1803                       SHLD    DISP2
  010B'   32 1802                       STA     DISP1
  010E'   C3 0104'                      JMP     READ1
  0111'   2A 1800               WHRITE: LHLD    RVEC
  0114'   DB 01                         IN      INPUT
  0116'   77                            MOV     M,A
  0117'   F1                            POP     PSW
  0118'   FB                            EI
  0119'   C9                            RET
                                ;
                                ;
                                        ORG     200H
  0200'   F3                    READ2:  DI
  0201'   2A 1800                       LHLD    RVEC
  0204'   7E                            MOV     A,M
  0205'   22 1803                       SHLD    DISP2
```

```
MACRO-80 3.36                    PAGE    1-1

47      0208'   32 1802                  STA     DISP1
48      020B'   C3 0200'                 JMP     READ2
49                                       END
```

```
        MACRO-80 3.36                    PAGE    S


Macros:

Symbols:
DISP1   1802    DISP2   1803    INPUT   0001    READ    0100'
READ1   0104'   READ2   0200'   RVEC    1800    STACKT  2400
WHRITE  0111'


No  Fatal error(s)
```

DISP1	10#	32	47		
DISP2	11#	31	46		
INPUT	12#	35			
READ	26#				
READ1	29#	33			
READ2	43#	48			
RVEC	9#	16	29	34	44
STACKT	8#	15			
WHRITE	22	34#			

```
MACRO-80 3.36                    PAGE    1

                                 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                 ;                                  ;
                                 ;          MONITOR PROM TIL        ;
                                 ;             TRÆNER               ;
                                 ;            (MON 2)               ;
                                 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

1800                             RVEC    EQU     1800H
1802                             DISP1   EQU     1802H
1803                             DISP2   EQU     1803H
23F9                             STACKT  EQU     23F9H
23F9                             IRQV    EQU     23F9H
23FB                             KTAST   EQU     23FBH
23FC                             AMODE   EQU     23FCH
23FD                             DATA    EQU     23FDH
23FE                             ADDR    EQU     23FEH
3800                             PA      EQU     3800H
3802                             PC      EQU     3802H
3803                             KONTR   EQU     3803H

                                         .PHASE  00

0000    31 23F9                  RST0:   LXI     SP,STACKT
0003    C3 0100                          JMP     INIT
                                         .DEPHASE

                                         .PHASE  24H
0024    C3 023B                  TRAP:   JMP     TRAP2
                                         .DEPHASE

                                         .PHASE  38H
0038    C3 023D                  RST7:   JMP     RST7A
003B    00                               NOP
003C    2A 23F9                  RST7.5: LHLD    IRQV
003F    E9                               PCHL
                                         .DEPHASE

0011'                                    ASEG
                                         ORG     100H
0100    21 5DBB                  INIT:   LXI     H,5DBBH ;SKRIV PA DISPLAY
0103    22 1803                          SHLD    DISP2
0106    3E 85                            MVI     A,85H
0108    32 1802                          STA     DISP1
010B    21 0000                          LXI     H,0     ;NULSTIL ADDR/DATA
010E    22 23FE                          SHLD    ADDR
0111    22 23FC                          SHLD    AMODE   ;OG AMODE
```

```
MACRO-80 3.36   17-Mar-80      PAGE    1-1

0114    3E 9A                          MVI     A,9AH   ;INIT PPI
0116    32 3803                        STA     KONTR
0119    CD 0167                 START: CALL     KSCAN
011C    DA 012F                        JC      STARTI
011F    0F                             RRC
0120    DA 01E8                        JC      SETA
0123    0F                             RRC
0124    DA 01EF                        JC      SETD
0127    0F                             RRC
0128    DA 01F7                        JC      SETP
012B    0F                             RRC
012C    DA 020F                        JC      SETM
012F    CD 0183                 STARTI: CALL    TSCAN
0132    DA 0119                        JC      START
0135    47                             MOV     B,A
0136    3A 23FC                        LDA     AMODE
0139    B7                             ORA     A
013A    F2 014D                        JP      ADIN
013D    21 23FD                        LXI     H,DATA
0140    7E                             MOV     A,M
0141    87                             ADD     A
0142    87                             ADD     A
0143    87                             ADD     A
0144    87                             ADD     A
0145    B0                             ORA     B
0146    77                             MOV     M,A
0147    32 1802                        STA     DISP1
014A    C3 01DF                        JMP     SLUTI
014D    2A 23FE                 ADIN:  LHLD    ADDR
                                       REPT    4
                                       DAD     H
                                       ENDM
0150    29                      +      DAD     H
0151    29                      +      DAD     H
0152    29                      +      DAD     H
0153    29                      +      DAD     H
0154    7D                             MOV     A,L
0155    B0                             ORA     B
0156    6F                             MOV     L,A
0157    22 23FE                        SHLD    ADDR
015A    22 1803                        SHLD    DISP2
015D    7E                             MOV     A,M
015E    32 23FD                        STA     DATA
0161    32 1802                        STA     DISP1
0164    C3 01DF                        JMP     SLUTI
```

```
MACRO-80 3.36   17-Mar-80      PAGE    1-2


                                
0167    21 23FB                 KSCAN:  LXI     H,KTAST
016A    3A 3800                         LDA     PA
016D    2F                              CMA
016E    E6 0F                           ANI     0FH
0170    CA 0181                         JZ      KSCANI
0173    77                              MOV     M,A
0174    CD 01C2                         CALL    PREL
0177    3A 3800                         LDA     PA
017A    2F                              CMA
017B    E6 0F                           ANI     0FH
017D    BE                              CMP     M
017E    CA 0182                         JZ      KSCANU
0181    37                      KSCANI: STC
0182    C9                      KSCANU: RET


0183    3E 9A                   TSCAN:  MVI     A,9AH
0185    32 3803                         STA     KONTR
0188    AF                              XRA     A
0189    32 3802                         STA     PC
018C    3A 3802                 TSCANA: LDA     PC
018F    FE F0                           CPI     0F0H
0191    CA 01C0                         JZ      TSCANO
0194    CD 01C2                         CALL    PREL
0197    3A 3802                         LDA     PC
019A    FE F0                           CPI     0F0H
019C    CA 01C0                         JZ      TSCANO
019F    47                              MOV     B,A
01A0    3E 93                           MVI     A,93H   ;VEND C0-3 OG C4-7
01A2    32 3803                         STA     KONTR
01A5    78                              MOV     A,B
01A6    32 3802                         STA     PC
01A9    3A 3802                         LDA     PC
01AC    21 0236                         LXI     H,TABEL+15
01AF    0E 0F                           MVI     C,15
01B1    BE                      TSCANB: CMP     M
01B2    CA 01BD                         JZ      TSCANC
01B5    2B                              DCX     H
01B6    0D                              DCR     C
01B7    F2 01B1                         JP      TSCANB
01BA    C3 0183                         JMP     TSCAN
01BD    79                      TSCANC: MOV     A,C
01BE    B7                              ORA     A
01BF    C9                              RET
01C0    37                      TSCANO: STC
```

```
MACRO-80 3.36   17-Mar-80       PAGE    1-3


139     01C1    C9                      RET
140
141
142     01C2    E5              PREL:   PUSH    H
143     01C3    D5                      PUSH    D
144     01C4    21 0078                 LXI     H,120
145     01C7    11 FFFF                 LXI     D,0FFFFH
146     01CA    19              PRELA:  DAD     D
147     01CB    DA 01CA                 JC      PRELA
148     01CE    D1                      POP     D
149     01CF    E1                      POP     H
150     01D0    C9                      RET
151
152
153     01D1    CD 0167         SLUT:   CALL    KSCAN
154     01D4    DA 0119                 JC      START
155     01D7    FE 03                   CPI     3
156     01D9    CA 0237                 JZ      GOOFF
157     01DC    C3 01D1                 JMP     SLUT
158     01DF    CD 0183         SLUTI:  CALL    TSCAN
159     01E2    D2 01DF                 JNC     SLUTI
160     01E5    C3 0119                 JMP     START
161
162
163     01E8    AF              SETA:   XRA     A
164     01E9    32 23FC                 STA     AMODE
165     01EC    C3 01D1                 JMP     SLUT
166     01EF    3E 80           SETD:   MVI     A,80H
167     01F1    32 23FC                 STA     AMODE
168     01F4    C3 01D1                 JMP     SLUT
169     01F7    2A 23FE         SETP:   LHLD    ADDR
170     01FA    3A 23FD                 LDA     DATA
171     01FD    77                      MOV     M,A
172     01FE    23                      INX     H
173     01FF    22 23FE                 SHLD    ADDR
174     0202    22 1803                 SHLD    DISP2
175     0205    7E                      MOV     A,M
176     0206    32 1802                 STA     DISP1
177     0209    32 23FD                 STA     DATA
178     020C    C3 01D1                 JMP     SLUT
179     020F    2A 23FE         SETM:   LHLD    ADDR
180     0212    3A 23FD                 LDA     DATA
181     0215    77                      MOV     M,A
182     0216    2B                      DCX     H
183     0217    22 23FE                 SHLD    ADDR
184     021A    22 1803                 SHLD    DISP2
```

```
        MACRO-80 3.36   17-Mar-80       PAGE    1-4

185     021D    7E                              MOV     A,M
186     021E    32 1802                         STA     DISP1
187     0221    32 23FD                         STA     DATA
188     0224    C3 01D1                         JMP     SLUT
189
190
191
192     0227    E7 D7 B7 77             TABEL:  DB      0E7H,0D7H,0B7H,77H
193     022B    EB DB BB 7B                     DB      0EBH,0DBH,0BBH,7BH
194     022F    ED DD BD 7D                     DB      0EDH,0DDH,0BDH,7DH
195     0233    EE DE BE 7E                     DB      0EEH,0DEH,0BEH,7EH
196
197     0237    2A 23FE                 GOOFF:  LHLD    ADDR
198     023A    E9                              PCHL
199
200     023B    00                      TRAP2:  NOP
201     023C    C9                              RET
202
203     023D    00                      RST7A:  NOP
204     023E    00                              NOP
205     023F    C9                              RET
206
207                                             END     RST0
```

```
MACRO-80 3.36   17-Mar-80      PAGE    S


Macros:

Symbols:
ADDR    23FE    ADIN    014D    AMODE   23FC    DATA    23FD
DISP1   1802    DISP2   1803    GOOFF   0237    INIT    0100
IRQV    23F9    KONTR   3803    KSCAN   0167    KSCANI  0181
KSCANU  0182    KTAST   23FB    PA      3800    PC      3802
PREL    01C2    PRELA   01CA    RST7    0038    RST7.5  003C
RST7A   023D    RST0    0000    RVEC    1800    SETA    01E8
SETD    01EF    SETM    020F    SETP    01F7    SLUT    01D1
SLUTI   01DF    STACKT  23F9    START   0119    STARTI  012F
TABEL   0227    TRAP    0024    TRAP2   023B    TSCAN   0183
TSCANA  018C    TSCANB  01B1    TSCANC  01BD    TSCAND  01C0


No  Fatal error(s)
```

ADDR	16#	45	75	86	169	173	179	183	197
ADIN	64	75#							
AMODE	14#	46	62	164	167				
DATA	15#	65	89	170	177	180	187		
DISP1	9#	43	73	90	176	186			
DISP2	10#	41	87	174	184				
GOOFF	156	197#							
INIT	24	40#							
IRQV	12#	34							
KONTR	19#	48	111	123					
KSCAN	49	94#	153						
KSCANI	98	106#							
KSCANU	105	107#							
KTAST	13#	94							
PA	17#	95	101						
PC	18#	113	114	118	125	126			
PREL	100	117	142#						
PRELA	146#	147							
RST7	32#								
RST7.5	34#								
RST7A	32	203#							
RSTO	23#	207							
RVEC	8#								
SETA	52	163#							
SETD	54	166#							
SETM	58	179#							
SETP	56	169#							
SLUT	153#	157	165	168	178	188			
SLUTI	74	91	158#	159					
STACKT	11#	23							
START	49#	60	154	160					
STARTI	50	59#							
TABEL	127	192#							
TRAP	28#								
TRAP2	28	200#							
TSCAN	59	110#	134	158					
TSCANA	114#								
TSCANB	129#	133							
TSCANC	130	135#							
TSCAND	116	120	138#						

Formål: Analyse af 8o85 restart.

1: Forbind logikanalysator (LSA),

POD 0 : AD Ø - AD 7

POD 1 : A 8 - A 15

POD 2 : bit Ø = SØ

" 1 = S1

" 2 = IO/$\overline{M}$

" 3 = ALE

" 4 = $\overline{RD}$

CK Ø = CLK OUT på 8o85

Q Ø = RESET OUT på 8o85

2: Tilfør et aktivt LOW-signal til $\overline{RESIN}$ på 8o85.

3: LSA setup:

CK Ø : pos. flanke

Q Ø : Ø

Triggerord: ØØØØH

Delay : 5 States

Data : States

4: Gennemførelse:

A) Hold $\overline{RESIN}$ på 8o85 LOW

B) Start LSA

C) Slip $\overline{RESIN}$

5: Analyse:

A) Bestem typen af maskincycle.

B) Kontroller T-states i relation til den fundne maskincycle.

Formål: Analyse af OP-code fetch og eksekvering.

1: Forbind LSA:

POD 0 : AD Ø - AD 7

POD 1 : A 8 - A 15

POD 2 : bit Ø = SØ

" 1 = S1

" 2 = IO/$\overline{M}$

" 3 = ALE

" 4 = RD

CK Ø = CLK OUT på 8o85

Q Ø = RESET OUT på 8o85

2: Tilfør et aktivt LOW-signal til $\overline{RESIN}$ på 8o85.

3: Forbind $\overline{RD}$ til $\overline{G}$ på port SADRL.

4: Tilfør porten maskinkoden for "NOP".

5: LSA setup:

CK Ø : pos. flanke

Q Ø : Ø

Triggerord: ØØØØH

Delay : 126 states

Data : States

6: Gennemførelse:

A) Hold $\overline{RESIN}$ på 8o85 LOW.

B) Start LSA

C) Slip $\overline{RESIN}$

7: Analyse:

A) Bestem typerne af maskincycler.

B) Kontroller T-states i relation til den/de fundne maskincycler og aktuelle OP-code.

8: Ændring af LSA setup:
Clockqualify med RESET OUT og ALE.
Foretag opsamling af data som før.
Hvilke data er nu samlet op?

9) Gentag punkt 4, 5, 6, 7 og 8 med maskinkoderne for "ANI", "IN", "OUT" og "DAD".

Formål: Øvelsen belyser samspillet mellem CPU'en og hukommelsen.

ØVEPROM 1

1: Forbind den nødvendige decodningslogik for tilslutning af EPROM'en på adresse 0000H.

2: Forbind LSA:

POD 0 : AD 0 - AD 7 } som øvelse 1
POD 1 : A 8 - A 15 }
POD 2 : CK 0 = ALE lav om / bit 7
CK 1 = $\overline{MW}$ + $\overline{MR}$ " / bit 6
bit 0= S0 } som øvelse 1
" 1= S1 }
" 2= IO/$\overline{M}$ }

3: LSA setup:

CK 0 : neg. flanke af ALE = 0 data
CK 1 : ~~pos.~~ neg. flanke af $\overline{MW}$ og $\overline{MR}$ = 0 mem
Trig,CK0 : 0000H adr.
Delay : 126 States
Data : States

4: Gennemførelse:

Start LSA og reset 8085.

5: Blank ikke tilsluttede og dublerede kanaler på LSA.

6: Analyser de opsamlede data, og understreg de data der er "hentet" i EPROM'en.

7: Nyt LSA setup: IO/$\overline{M}$ S1 S0

Trig,CK0 : XX011 XX XX
Data : Triggers OF! (OPKODE FETCH)
Øvrige : Uændret

8: Foretag nu dataopsamling.

Hvilke data er nu samlet op? hvorfor?

Formål: Øvelsen belyser samspillet mellem CPU'en og hukommelsen.

1: Forbind de nødvendige signaler til RAM-hukommelsen således at startadressen bliver 2000H.

2: LSA forbindelse og setup som punkt 2 og 3 i øvelse 3.

3: Gennemførelse:

A) Tryk på reset tasten

B) Tænd først nu for øvelsespanelet

C) Start LSA

D) Slip reset tasten

4: Analyse:

Kontroller datatransporten til og fra RAM-hukommelsen.

Formål: At følge en programafvikling ved hjælp af logikanalysator og programudskrift.

1: Forbind SADRH, SADRL, INPUT-PORT, OUTPUT-PORT og DISPLAY i henhold til nedenstående memory- og I/O-map.

Memory-map

	adr.
Venstre display	1804H
Midders. "	1803H
Højre "	1802H
SADRH	1801H
SADRL	1800H

I/O-map

	port Nr.
input	00000001B
output	00000010B

2: EPROM 2 monteres (EPROM 1 afleveres)

3: Start eksekveringen af READ-programmet.

4: Sammenhold de opsamlede data med programudskriften.
Kører programmet?

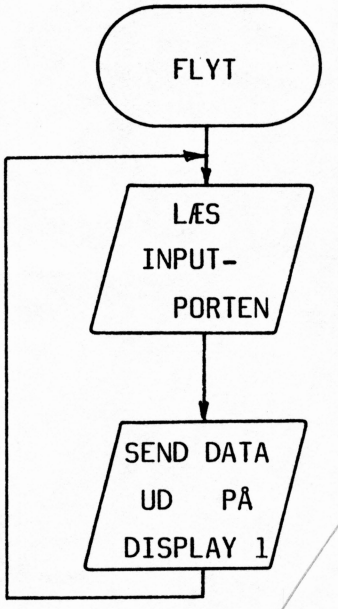
Formål: At udvikle og afprøve simple programmer på øvelsespanelet.

Monitor 503085

1: Forbind RST 7.5 til en aktiv HIGH puls.

2: Forbind HEX-kodeomskifterens lave byte til INPUT-porten.

3: Skriv, indtast og prøvekør et program der kontinuert overfører data fra INPUT-porten til det højre display. Programmet indlæses fra adr. 2000H.



Set up:
Som måleøvelse 3
adr. 2000H
DLY 0100 S

4: Følg programeksekveringen med LSA.

5: Skriv, indtast og prøvekør et program der kontinuerligt udfører en logisk AND-funktion mellem bit 0, 1, 2 og 3 i INPUT-porten, og viser resultatet på bit 0 i OUTPUT-porten.

6: Følg programeksekveringen med LSA, både når AND-funktionen er opfyldt og når den ikke er opfyldt.

Formål: Måling med oscilloscop på 8o85's kontrolsignaler.

1: Indtast programmet fra øvelse 6 punkt 3.

2: Trig oscilloscopet på adresse 2000H.

3: Iagttag og tegn følgende signaler korrekt i forhold til hinanden: $\overline{RD}$, $\overline{WR}$, IO/$\overline{M}$, AD 0 og A 0.

4: Marker på AD 0, hvor der er adresser, og hvor der er data på bussen.

LSA:

Word: in

Delay: ud

Trig word: 2000

Formål: Øvelsen belyser funktionen af en programmerbar portkreds.

1: Monter PPI kredsen 8255, så dens startadresse bliver 0800H

2: Gennemførelse:

A) Load kontrolregistret med et kontrolord der programmerer 8255 til: MODE 0, A=IND, B=UD, C=IND.

B) Forbind LED-indikatorer til de fire lave bit i port-B Læs forskellige bitmønstre ud til port-B og se reaktionen.

C) Forbind fire niveauomskiftere til de fire lave bit i port-A. Læs port-A med forskellige bitmønstre.

3: Skriv et program der programmerer 8255 som i punkt 2, og derefter kontinuerligt overfører inputtet på port-A til LED-indikatorerne på port-B.

Port A = 0800H
B = 0801H
C = 0802H
Control = 0803H

3E 99
32 03 08
3A 00 08
32 01 08
C3 03 20

Formål: At belyse 8o85's interruptfaciliteter.

TRAP interrupt

1: LSA setup:
Trig,CKØ : ØØ24H

2: Tilfør TRAP inputtet på 8o85 et signal og analyser de opsamlede data.

INTR interrupt

3: Tilfør SADRL et enablesignal ($\overline{G}$) fra $\overline{INTA}$ på 8o85.

4: Tilfør SADRL koden for RST 7.

5: LSA setup:
Trig,CKØ : Hop-adresse for RST 7.

6: Skriv, indtast og eksekver et program der enabler interrupt.

7: Tilfør INTR inputtet et signal, og analyser de opsamlede data.

fra hvor??

1: Sæt register D til 6 (D) = 6 6=0006H

2: Sæt register E til -5 (E) =-5 (sign bit)

0÷5 → E

3: Sæt register A til indholdet af register B (A) = (B)

MOV 78

4: Adder register B og C og anbring resultatet i D (D) = (B) + (C)

B→A; MOV 78 -- A=A+C; ADD 81 -- MOV 57

5: Anbring indholdet af adresse 2000H i register B (B) = (2000H)

mov reg,mem ~ mov mem,reg 46.0020

6: Anbring indholdet af adresse 2050H i register A, og multiplicer med -1. (A) = -(2050H)

7: Sæt indholdet af adresse 2030H til den negative værdi af adresse 2040H. (2030H) = -(2040H)

8: Beregn differencen mellem indholdet af adresserne 2055H og 2056H og anbring resultatet i adresse 2057H. (2057H) = (2055H) - (2056H)

9: Ombyt indholdet af adresserne 2040H og 2045H. (2040H)←→(2045H)

10: Sæt register A til 0. (A) = 0

11: Skriv et program der incrementerer L registret, og hvis dette bliver 00, incrementerer H registret.

12: Læg værdierne 02H, 03H og 04H i adresserne 2010H og fremefter.

13: Incrementer værdien i adresse 2050H.

14: Nulstil indholdet i adresserne 2030H til 2070H.

15: Flyt indholdet i adresserne 2020H - 202F til 2000H - 200FH.

16: Skriv et program der læser fire omskiftere, og tænder LED 1 ved et ulige antal omskiftere der er i stilling 1, og LED 0 ved et lige antal omskiftere der er i stilling 1.

17: Læs bit 4 - 7 på input-porten, og anbring resultatet på bit 0 - 3 på output-porten.

18: Skriv et program der læser vippeomskifterene, og udlæser niveauerne på lysdioderne med bit 2 konstant tændt.

19: Skriv et tidsforsinkelsesprogram som kan bruges generelt. Programmet skal forsinke et antal 4,5 msek. , hvor antallet bestemmes af indholdet i B-registret.

2o: Som 19, men med 2o msek.

21: Skriv et program, der kontinuerligt tænder lysdioderne 2o msek. efter tur. (Ringtæller).

22: Skriv et program, der læser en byte på thumbweelomskifterene og adderer de to nipler og viser resultatet på displayet.

23: Som 22, men med BCD-tal. Indtastes et ikke BCD-tal skal displayet vise EE.

24: Skriv et program hvor stillingen af omskifter 3 bestemmer hvilken tabel der anvendes til at omsætte værdien af omskifterene 0 - 2 til lysdioderne.

Omsk. nummer: (2)(1)(0)	(3)= 1	(3)= 0
0 0 0	0 1 1 0	1 0 0 0
0 0 1	0 1 1 1	0 1 1 0
0 1 0	1 1 1 1	1 1 1 1
0 1 1	0 0 0 0	0 0 0 0
1 0 0	1 0 1 1	1 1 0 0
1 0 1	0 0 0 0	1 0 0 0
1 1 0	0 1 0 1	0 1 1 0
1 1 1	0 1 1 0	0 1 0 1

25: I adresserne 2200H til 220FH er placeret 16 vilkårlige talværdier. Skriv et program, der sorterer disse værdier efter numerisk værdi, ider det mindste placeres på adresse 2200H, og det største på adresse 220FH.

26: I adresse 2280H til 229FH er placeret 32 vilkårlige talværdier. Skriv et program der undersøger hvilke værdier der er representeret mere end to gange. Disse værdier skal udlæses på displayet i minimum 1 sek.

27: Skriv et program, der kalder den underrutine i monitoren der læser keyboardet, og send den indtastede karakter som asynkron seriel transmission ud af SOD-outputtet på 8085. Der skal sendes med en baud-rate på 110, og være 2 stop bit.

28: Skriv et program, der kan modtage serielle asynkrone data via SID-inputtet på 8085 og vise de modtagne data på displayet. Baud-rate og stopbit som i opgave 27.

OP-koder

MASKINKODER OG MNEONIC

DATA TRANSFER GROUP

Move

MOV	A,A	7F
	A,B	78
	A,C	79
	A,D	7A
	A,E	7B
	A,H	7C
	A,L	7D
	A,M	7E
MOV	B,A	47
	B,B	40
	B,C	41
	B,D	42
	B,E	43
	B,H	44
	B,L	45
	B,M	46
MOV	C,A	4F
	C,B	48
	C,C	49
	C,D	4A
	C,E	4B
	C,H	4C
	C,L	4D
	C,M	4E
MOV	D,A	57
	D,B	50
	D,C	51
	D,D	52
	D,E	53
	D,H	54
	D,L	55
	D,M	56

Move (cont)

MOV	E,A	5F
	E,B	58
	E,C	59
	E,D	5A
	E,E	5B
	E,H	5C
	E,L	5D
	E,M	5E
MOV	H,A	67
	H,B	60
	H,C	61
	H,D	62
	H,E	63
	H,H	64
	H,L	65
	H,M	66
MOV	L,A	6F
	L,B	68
	L,C	69
	L,D	6A
	L,E	6B
	L,H	6C
	L,L	6D
	L,M	6E
MOV	M,A	77
	M,B	70
	M,C	71
	M,D	72
	M,E	73
	M,H	74
	M,L	75

XCHG EB

byt om

Move Immediate

MVI	A, byte	3E
	B, byte	06
	C, byte	0E
	D, byte	16
	E, byte	1E
	H, byte	26
	L, byte	2E
	M, byte	36

Load Immediate

LXI	B, dble	01
	D, dble	11
	H, dble	21
	SP, dble	31

dobbl. reg.

Load Store

LDAX B	0A
LDAX D	1A
LHLD adr	2A
LDA adr	3A
STAX B	02
STAX D	12
SHLD adr	22
STA adr	32

byte = constant, or logical/arithmetic expression that evaluates to an 8-bit data quantity. (Second byte of 2-byte instructions).

dble = constant, or logical/arithmetic expression that evaluates to a 16-bit data quantity. (Second and Third bytes of 3-byte instructions).

adr = 16-bit address (Second and Third bytes of 3-byte instructions).

* = all flags (C, Z, S, P, AC) affected.

** = all flags except CARRY affected; (exception: INX and DCX affect no flags).

† = only CARRY affected.

ARITHMETIC AND LOGICAL GROUP

Add*

A-reg. altid den ene part

ADD	A	87
	B	80
	C	81
	D	82
	E	83
	H	84
	L	85
	M	86
ADC	A	8F
	B	88
	C	89
	D	8A
	E	8B
	H	8C
	L	8D
	M	8E

Subtract*

SUB	A	97
	B	90
	C	91
	D	92
	E	93
	H	94
	L	95
	M	96
SBB	A	9F
	B	98
	C	99
	D	9A
	E	9B
	H	9C
	L	9D
	M	9E

Double Add †

DAD	B	09
	D	19
	H	29
	SP	39

Increment = +1**

INR	A	3C
	B	04
	C	0C
	D	14
	E	1C
	H	24
	L	2C
	M	34
INX	B	03
	D	13
	H	23
	SP	33

Decrement = ÷ 1**

DCR	A	3D
	B	05
	C	0D
	D	15
	E	1D
	H	25
	L	2D
	M	35
DCX	B	0B
	D	1B
	H	2B
	SP	3B

dobbl. reg.

Specials

DAA*	27
CMA	2F
STC†	37
CMC†	3F

Rotate †

RLC	07
RRC	0F
RAL	17
RAR	1F

Logical*

ANA	A	A7
	B	A0
	C	A1
	D	A2
	E	A3
	H	A4
	L	A5
	M	A6
XRA	A	AF
	B	A8
	C	A9
	D	AA
	E	AB
	H	AC
	L	AD
	M	AE
ORA	A	B7
	B	B0
	C	B1
	D	B2
	E	B3
	H	B4
	L	B5
	M	B6
CMP	A	BF
	B	B8
	C	B9
	D	BA
	E	BB
	H	BC
	L	BD
	M	BE

and / xor / or / compare

Arith & Logical Immediate

ADI byte	C6
ACI byte	CE
SUI byte	D6
SBI byte	DE
ANI byte	E6
XRI byte	EE
ORI byte	F6
CPI byte	FE

00	NOP		2B	DCX	H	56	MOV	D,M	81	ADD	C
01	LXI	B,dble	2C	INR	L	57	MOV	D,A	82	ADD	D
02	STAX	B	2D	DCR	L	58	MOV	E,B	83	ADD	E
03	INX	B	2E	MVI	L,byte	59	MOV	E,C	84	ADD	H
04	INR	B	2F	CMA		5A	MOV	E,D	85	ADD	L
05	DCR	B	30	SIM*		5B	MOV	E,E	86	ADD	M
06	MVI	B,byte	31	LXI	SP,dble	5C	MOV	E,H	87	ADD	A
07	RLC		32	STA	adr	5D	MOV	E,L	88	ADC	B
08	---		33	INX	SP	5E	MOV	E,M	89	ADC	C
09	DAD	B	34	INR	M	5F	MOV	E,A	8A	ADC	D
0A	LDAX	B	35	DCR	M	60	MOV	H,B	8B	ADC	E
0B	DCX	B	36	MVI	M,byte	61	MOV	H,C	8C	ADC	H
0C	INR	C	37	STC		62	MOV	H,D	8D	ADC	L
0D	DCR	C	38	---		63	MOV	H,E	8E	ADC	M
0E	MVI	C,byte	39	DAD	SP	64	MOV	H,H	8F	ADC	A
0F	RRC		3A	LDA	adr	65	MOV	H,L	90	SUB	B
10	---		3B	DCX	SP	66	MOV	H,M	91	SUB	C
11	LXI	D,dble	3C	INR	A	67	MOV	H,A	92	SUB	D
12	STAX	D	3D	DCR	A	68	MOV	L,B	93	SUB	E
13	INX	D	3E	MVI	A,byte	69	MOV	L,C	94	SUB	H
14	INR	D	3F	CMC		6A	MOV	L,D	95	SUB	L
15	DCR	D	40	MOV	B,B	6B	MOV	L,E	96	SUB	M
16	MVI	D,byte	41	MOV	B,C	6C	MOV	L,H	97	SUB	A
17	RAL		42	MOV	B,D	6D	MOV	L,L	98	SBB	B
18	---		43	MOV	B,E	6E	MOV	L,M	99	SBB	C
19	DAD	D	44	MOV	B,H	6F	MOV	L,A	9A	SBB	D
1A	LDAX	D	45	MOV	B,L	70	MOV	M,B	9B	SBB	E
1B	DCX	D	46	MOV	B,M	71	MOV	M,C	9C	SBB	H
1C	INR	E	47	MOV	B,A	72	MOV	M,D	9D	SBB	L
1D	DCR	E	48	MOV	C,B	73	MOV	M,E	9E	SBB	M
1E	MVI	E,byte	49	MOV	C,C	74	MOV	M,H	9F	SBB	A
1F	RAR		4A	MOV	C,D	75	MOV	M,L	A0	ANA	B
20	RIM*		4B	MOV	C,E	76	HLT		A1	ANA	C
21	LXI	H,dble	4C	MOV	C,H	77	MOV	M,A	A2	ANA	D
22	SHLD	adr	4D	MOV	C,L	78	MOV	A,B	A3	ANA	E
23	INX	H	4E	MOV	C,M	79	MOV	A,C	A4	ANA	H
24	INR	H	4F	MOV	C,A	7A	MOV	A,D	A5	ANA	L
25	DCR	H	50	MOV	D,B	7B	MOV	A,E	A6	ANA	M
26	MVI	H,byte	51	MOV	D,C	7C	MOV	A,H	A7	ANA	A
27	DAA		52	MOV	D,D	7D	MOV	A,L	A8	XRA	B
28	---		53	MOV	D,E	7E	MOV	A,M	A9	XRA	C
29	DAD	H	54	MOV	D,H	7F	MOV	A,A	AA	XRA	D
2A	LHLD	adr	55	MOV	D,L	80	ADD	B	AB	XRA	E

AC	XRA	H	D7	RST	2			
AD	XRA	L	D8	RC				
AE	XRA	M	D9	---				
AF	XRA	A	DA	JC	adr			
B0	ORA	B	DB	IN	byte			
B1	ORA	C	DC	CC	adr			
B2	ORA	D	DD	---				
B3	ORA	E	DE	SBI	byte			
B4	ORA	H	DF	RST	3			
B5	ORA	L	E0	RPO				
B6	ORA	M	E1	POP	H			
B7	ORA	A	E2	JPO	adr			
B8	CMP	B	E3	XTHL				
B9	CMP	C	E4	CPO	adr			
BA	CMP	D	E5	PUSH	H			
BB	CMP	E	E6	ANI	byte			
BC	CMP	H	E7	RST	4			
BD	CMP	L	E8	RPE				
BE	CMP	M	E9	PCHL				
BF	CMP	A	EA	JPE	adr			
C0	RNZ		EB	XCHG				
C1	POP	B	EC	CPE	adr			
C2	JNZ	adr	ED	---				
C3	JMP	adr	EE	XRI	byte			
C4	CNZ	adr	EF	RST	5			
C5	PUSH	B	F0	RP				
C6	ADI	byte	F1	POP	PSW			
C7	RST	0	F2	JP	adr			
C8	RZ		F3	DI				
C9	RET		F4	CP	adr			
CA	JZ	adr	F5	PUSH	PSW			
CB	---		F6	ORI	byte			
CC	CZ	adr	F7	RST	6			
CD	CALL	adr	F8	RM				
CE	ACI	byte	F9	SPHL				
CF	RST	1	FA	JM	adr			
D0	RNC		FB	EI				
D1	POP	D	FC	CM	adr			
D2	JNC	adr	FD	---				
D3	OUT	byte	FE	CPI	byte			
D4	CNC	adr	FF	RST	7			
D5	PUSH	D						
D6	SUI	byte						

*8085 Only

INTERNAL REGISTER ORGANIZATION

A Reg. (8)

B Reg. (8)	C Reg. (8)
D Reg. (8)	E Reg. (8)
H Reg. (8)	L Reg. (8)
Program Counter (16)	
Stack Pointer (16)	

FLAG BYTE

D_7							D_0
S	Z	X	AC	X	P	X	C

CARRY
PARITY
AUX CARRY
ZERO
SIGN

X UNDEFINED

resultat i Akumulator efter sidste ALU opr.

BRANCH CONTROL INSTRUCTIONS

Flag Condition	Jump		Call		Return	
Zero True	JZ	CA	CZ	CC	RZ	C8
Zero False	JNZ	C2	CNZ	C4	RNZ	C0
Carry True	JC	DA	CC	DC	RC	D8
Carry False	JNC	D2	CNC	D4	RNC	D0
Sign Positive	JP	F2	CP	F4	RP	F0
Sign Negative	JM	FA	CM	FC	RM	F8
Parity Even	JPE	EA	CPE	EC	RPE	E8
Parity Odd	JPO	E2	CPO	E4	RPO	E0
Unconditional	JMP	C3	CALL	CD	RET	C9

REGISTER-PAIR ORGANIZATION

PSW processor status word

A (8)	FLAGS (8)

NOTE: Leftmost Byte is high-order byte for arithmetic operations and addressing. Left byte is pushed on stack first. Right byte is popped first.

B (B/C) (16)
D (D/E) (16)
H (H/L) (16)
Prog. Ctr. (16)
Stack Ptr. (16)

ACCUMULATOR OPERATIONS

	Code	Function
XRA A	AF	Clear A and Clear Carry
ORA A	B7	Clear Carry
CMC	3F	Complement Carry
CMA	2F	Complement Accumulator
STC	37	Set Carry
RLC	07	Rotate Left
RRC	0F	Rotate Right
RAL	17	Rotate Left Thru Carry
RAR	1F	Rotate Right Thru Carry
DAA	27	Decimal Adjust Accum

REGISTER PAIR AND STACK OPERATIONS

	PSW (A/F)	Register Pair B (B/C)	D (D/E)	H (H/L)	SP	PC	Function
INX		03	13	23	33		Increment Register Pair
DCX		0B	1B	2B	3B		Decrement Register Pair
LDAX		0A	1A	7E(1)			Load A Indirect (Reg. Pair holds Adrs)
STAX		02	12	77(2)			Store A Indirect (Reg. Pair holds Adrs)
LHLD				2A			Load H/L Direct (Bytes 2 and 3 hold Adrs)
SHLD				22			Store H/L Direct (Bytes 2 and 3 hold Adrs)
LXI		01	11	21	31	C3(3)	Load Reg. Pair Immediate (Bytes 2 and 3 hold immediate data)
PCHL						E9	Load PC with H/L (Branch to Adrs in H/L)
XCHG			EB				Exchange Reg. Pairs D/E and H/L
DAD		09	19	29	39		Add Reg. Pair to H/L
PUSH	F5	C5	D5	E5			Push Reg. Pair on Stack sætund på stack
POP	F1	C1	D1	E1			Pop Reg. Pair off Stack hent fra stack
XTHL				E3			Exchange H/L with Top of Stack
SPHL					F9		Load SP with H/L

Notes: 1. This is MOV A,M. 2. This is MOV M,A. 3. This is JMP.

RESTART TABLE

Name	Code	Restart Address
RST 0	C7	0000_{16}
RST 1	CF	0008_{16}
RST 2	D7	0010_{16}
RST 3	DF	0018_{16}
RST 4	E7	0020_{16}
TRAP	Hardware* Function	0024_{16}
RST 5	EF	0028_{16}
RST 5.5	Hardware* Function	$002C_{16}$
RST 6	F7	0030_{16}
RST 6.5	Hardware* Function	0034_{16}
RST 7	FF	0038_{16}
RST 7.5	Hardware* Function	$003C_{16}$

*NOTE: The hardware functions refer to the on-chip Interrupt feature of the 8085 only.

USE OF THE A REGISTER BY RIM AND SIM INSTRUCTIONS (8085 ONLY)

A REGISTER AFTER EXECUTING RIM

D_7							D_0
SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5

INTERRUPT MASKS
INTERRUPT ENABLE FLAG
INTERRUPTS PENDING
SERIAL INPUT DATA

A REGISTER BEFORE EXECUTING SIM

D_7							D_0
SOD	SOE	X	R7.5	MSE	M7.5	M6.5	M5.5

RST.5.5 MASK
RST 6.5 MASK
RST 7.5 MASK
MASK SET ENABLE
RESET RST 7.5
UNDEFINED
SOD ENABLE
SERIAL OUTPUT DATA

BRANCH CONTROL GROUP

Jump

JMP adr	C3
JNZ adr	C2
JZ adr	CA
JNC adr	D2
JC adr	DA
JPO adr	E2
JPE adr	EA
JP adr	F2
JM adr	FA
PCHL	E9

Call

CALL adr	CD
CNZ adr	C4
CZ adr	CC
CNC adr	D4
CC adr	DC
CPO adr	E4
CPE adr	EC
CP adr	F4
CM adr	FC

Return

RET	C9
RNZ	C0
RZ	C8
RNC	D0
RC	D8
RPO	E0
RPE	E8
RP	F0
RM	F8

Restart

RST	0	C7
	1	CF
	2	D7
	3	DF
	4	E7
	5	EF
	6	F7
	7	FF

I/O AND MACHINE CONTROL

Stack Ops

PUSH	B	C5
	D	D5
	H	E5
	PSW	F5
POP	B	C1
	D	D1
	H	E1
	PSW*	F1
XTHL		E3
SPHL		F9

Input Output

OUT byte	D3
IN byte	DB

Control

DI	F3
EI	FB
NOP	00
HLT	76

New Instructions (8085 Only)

RIM	20
SIM	30