4.

Den konventionelle maskine

Indhold:

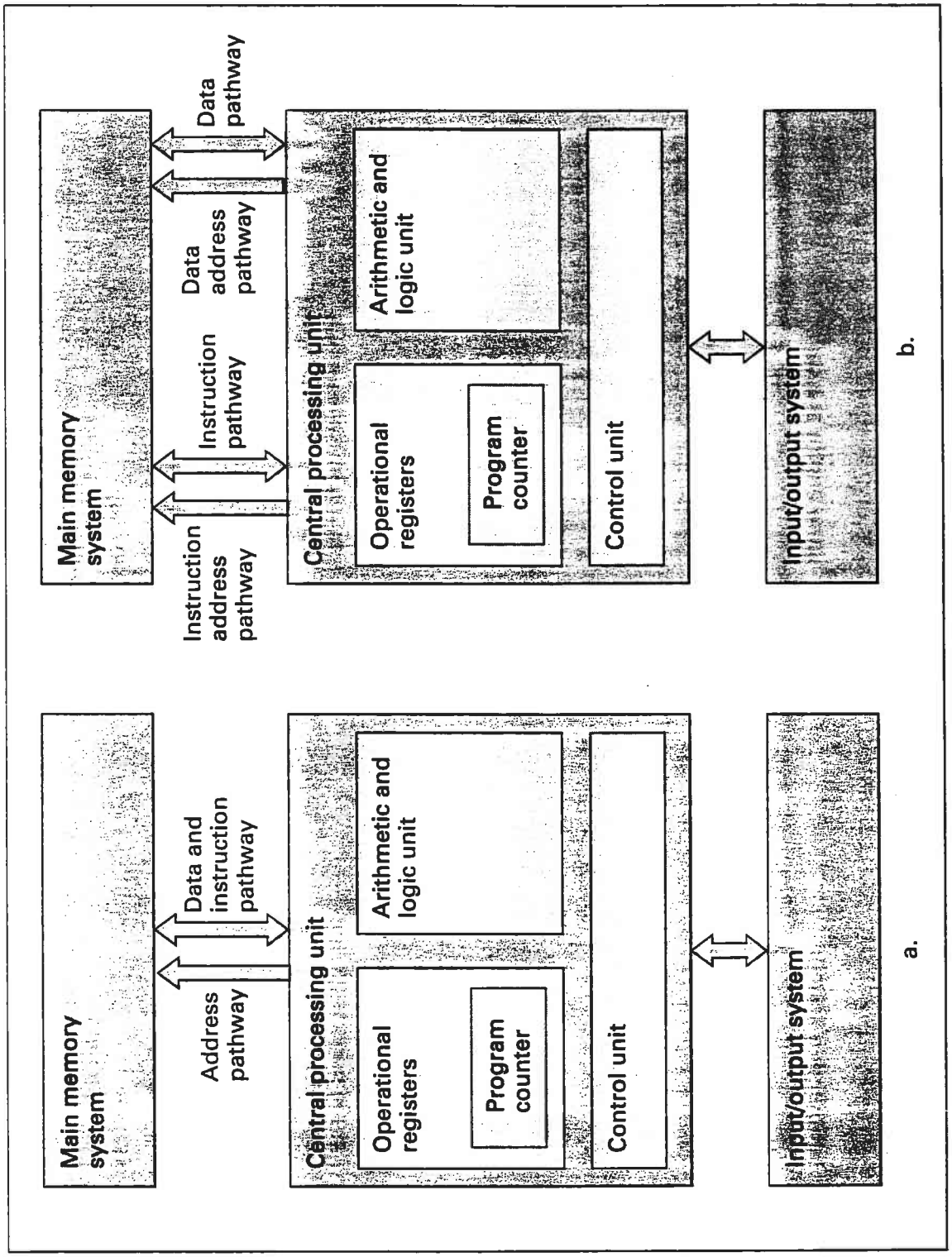
4.1 Von Neuman arkitektur
- a) Diagrammer: Von Neuman maskiner

4.2 Modelmaskinen - simulator
- a) Manual: Modelmaskinen

4.3 Intel 8086 arkitektur
- a) CPU diagram
- b) Oversigt over instruktionssæt
- c) Decodningstabel
- d) Maskinkodeformat
- e) Adresseringsmuligheder
- f) Artikel: Hvordan virker interrupts ?
- g) Anvendelse af 8086-registre

4.1.a

Von Neuman maskiner



Main components of typical von Neumann machines. (a) A conventional von Neumann architecture. (b) A Harvard architecture.

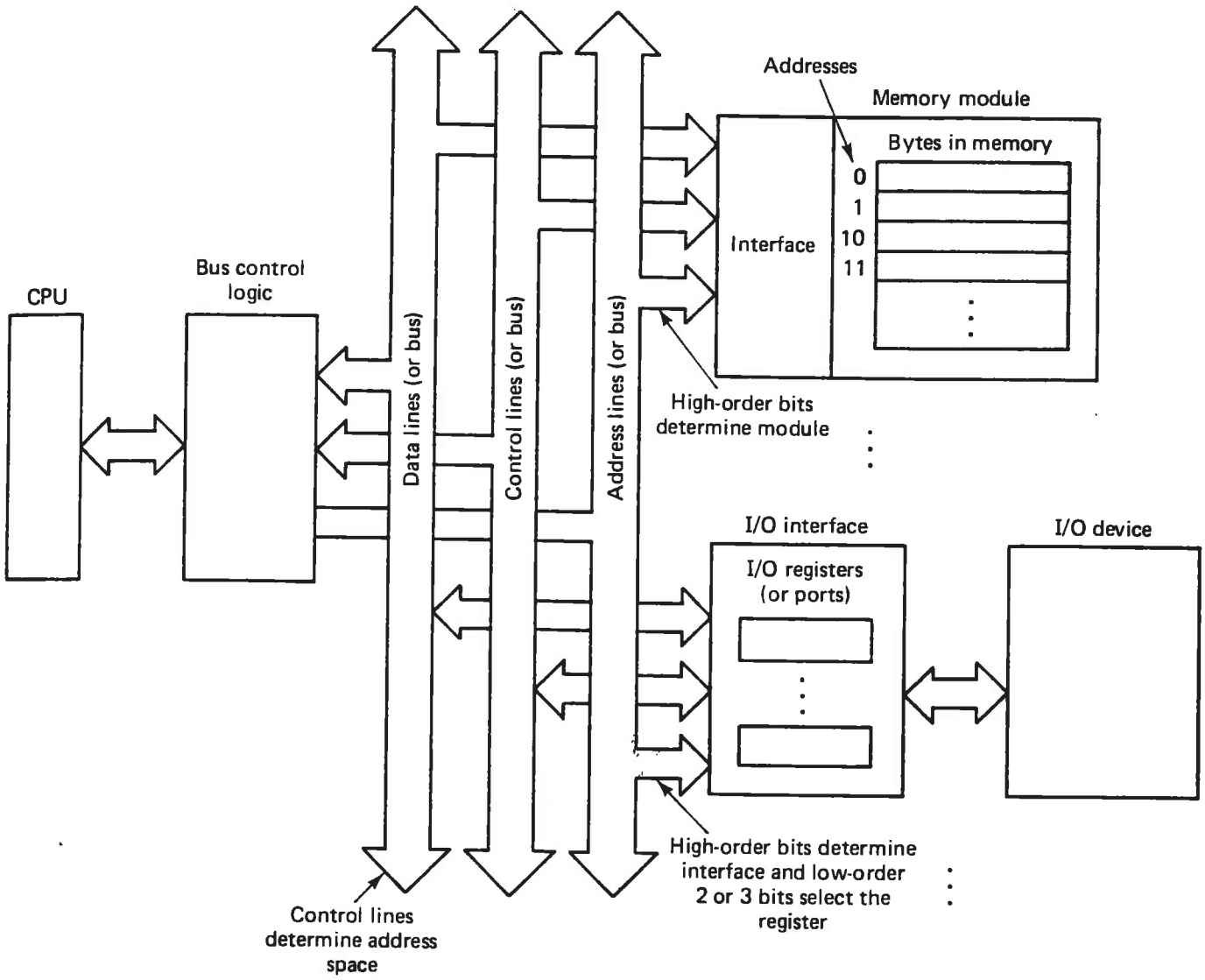
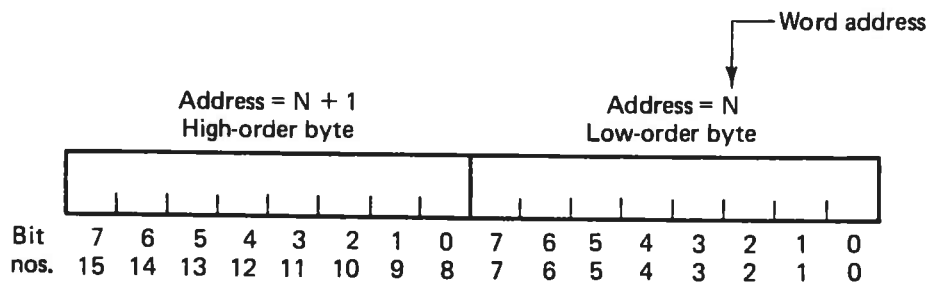


Figure 1-7 Memory and I/O register organization.

The bits are numbered with 0 being assigned to the least significant bit (LSB). In a byte the most significant bit (MSB) is numbered 7 and in a word the MSB is numbered 15. These conventions are summarized in Fig. 1-8.

Some computers require that words begin with addresses that are divisible by the number of bytes in a word and produce an alignment error if this rule is not followed. Others, such as the 8086, permit words to begin at any address;

Figure 1-8 Address and bit numbering conventions.



4.2.a

Modelmaskinen

Bjarne Larsen

INDHOLDSFORTEGNELSE

Programmering af datamaskinen. 1

Datamaskinens virkemåde. 1

Modelmaskinens opbygning. 1

Positive og negative tal. 2

Modelmaskinens instruktionssæt. 3
01 LOAD 3
02 ADD 3
03 SUBTRACT 3
04 DIVIDE 3
05 MULTIPLY 3
06 STORE 4
07 JUMP 4
08 TESTZERO 4
09 TESTGREATER 4
99 STOP 4

Eksempel på et program. 4

Programafbrydelser (interrupts) i modelmaskinen. . . 5

Brugervejledning for modelmaskineprogrammet. 5

Opstart af systemet. 5

Funktionsvalg. 6
E - Exit 6
G - Gem program 6
H - Hastighedsændring 7
I - Indsæt i celler 7
L - Læs program ind 8
N - Nulstil lageret 8
S - Startadresse indsættes i P-tæller 8
T - Trinvis udførelse til/fra 9
U - Udfør programmet 9
X - Sætter lyd til/fra 10

Pause menuen 10
F - Fortsæt udførelse 10
A - Afbryd efter udførelse af igangværende ordre 10

Modelmaskinens opbygning og virkemåde

Programmering af datamaskinen.

Ved programmering af datamaskinen er det programmørens opgave at sammensætte et program med instruktioner fra maskinens instruktionssæt. Instruktionerne skal sammensættes på en sådan måde, at de tilsammen og udført i den rigtige rækkefølge vil bevirke, at et givent problem vil blive løst.

Hver enkelt instruktion består af mindst to dele, en operationskode og en adresse. Operationskoden skal fortælle datamaskinen hvilken operation, der skal udføres. Adressen fortæller datamaskinen hvilke data, operationen skal udføres på.

Datamaskinens virkemåde.

Med henblik på at anskueliggøre datamaskinens virkemåde under et programs udførelse er der konstrueret en stærkt forenklet imaginær modelmaskine. Ved hjælp af denne modelmaskine kan man få et overblik over datamaskinens arbejde i de forskellige faser af en instruktions udførelse. Arbejdet i disse faser kan kort beskrives således:

1. Hent næste instruktion på den adresse, som programtælleren peger på. Programtælleren er et specialregister, som altid indeholder adressen på den næste instruktion, der skal udføres.

2. Den fundne instruktion overføres fra det interne lager til processoren, hvor operationskodedelen af instruktionen indsættes i operationskoderegistret og instruktionens adressedel indsættes i adresseregistret.

3. Der adderes 1 til programtælleren, som således kommer til at pege på den næste instruktion i det interne lager.

4. Operationskoden i operationskoderegistret fortolkes.

5. Operationen udføres. Denne fase er forskellig afhængig af den operationskode, som står i operationskoderegistret. Hvad der præcist sker i forbindelse med de enkelte oprationskoder, er omtalt i forbindelse med beskrivelsen af modelmaskinens instruktionssæt.

Modelmaskinens opbygning.

Modelmaskinens centralenhed består at et internt lager og en processor. Det interne lager, som skal indeholde programmet og dets data, er opbygget af 40 celler, som hver kan rumme 8 decimale cifre.

Modelmaskinens opbygning og virkemåde

Processoren består af en styreenhed, en arimetisk/logisk enhed samt et akkumulatorregister, som kan rumme 8 decimale cifre, et oprationskoderegister på 2 decimale cifre, et adresseregister og en programtæller på hver 6 decimale cifre.

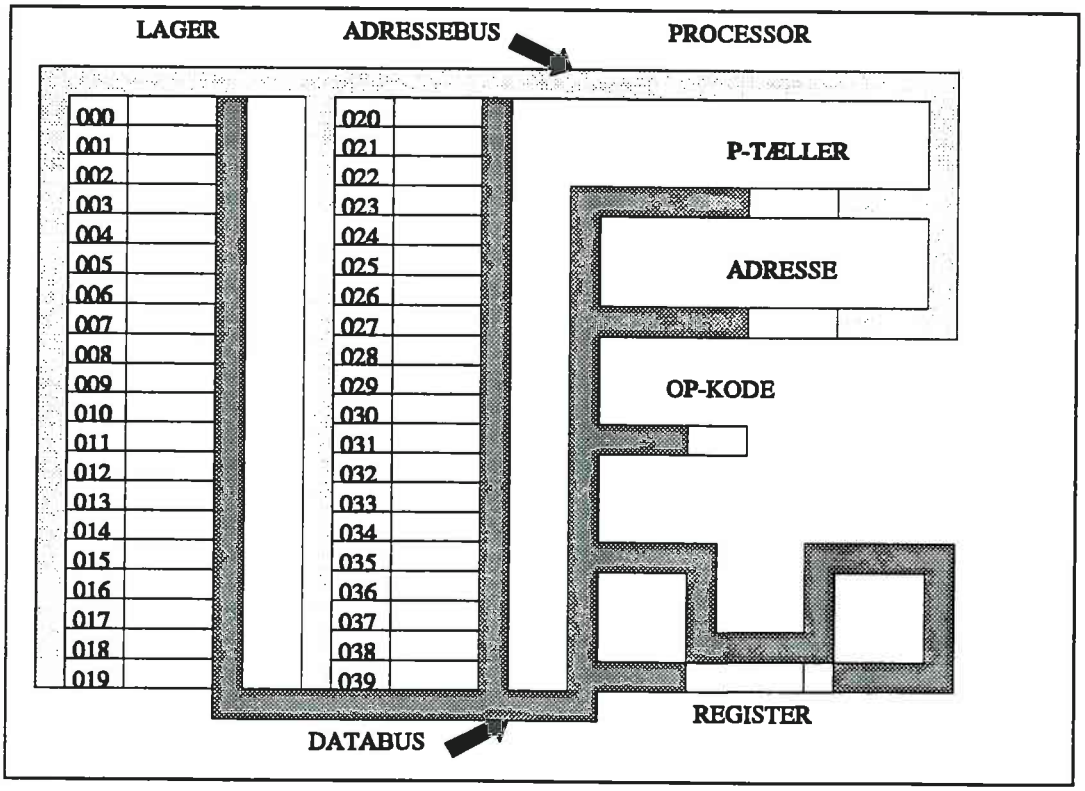
Imellem det interne lager og processoren er der dels en adressebus, som bruges til udpegning af den celle, der ønskes overført til eller fra processoren, dels en databus, hvorigennem indholdet af den udpegede celle i lageret overføres til eller fra processoren.

Positive og negative tal.

Af hensyn til en så forenklet fremstilling som muligt, er der ved konstruktionen af modelmaskinen kun åbnet mulighed for at arbejde med numerisk decimalt indhold i celler og registre. Til gengæld er det muligt at arbejde med såvel positive som negative decimale tal, idet der benyttes sædvanlig 2-komplement repræsentation af negative tal.

I det decimale talsystem betyder dette, at det første ciffer i en celles eller registrets indhold skal være 9, hvis tallet skal være negativt. Eksempelvis vil tallet -423 i en af det interne lagers celler eller akkumulatorregistret være repræsenteret ved tallet 99999577, og tallet -1 vil være repræsenteret ved tallet 99999999.

Principskitse over modelmaskinen.



Modelmaskinens opbygning og virkemåde

Modelmaskinens instruktionssæt.

Inden vi går over til en nærmere beskrivelse af modelmaskinens instruktionssæt, skal vi kort omtale instruktionsformatet. Instruktionsformatet fortæller, hvordan indholdet af en celle er opdelt i en oprationskodedel og en adressedel. Instruktionsformatet er nært forbundet med og afhængig af, hvordan processoren er konstrueret.

I modelmaskinen er der et 2-cifret operationskoderegister og et 6-cifret adresseregister. Det er derfor naturligt at lade modelmaskinens instruktionsformat diktere, at de to første cifre i de celler, som indeholder instruktioner, skal indeholde en to-cifret operationskode, og at de sidste seks cifre skal indeholde adressen på det datafelt, som instruktionen skal virke på.

En datamaskines instruktionssæt er summen af de instruktioner, som maskinen er bygget til at kunne udføre. I det følgende gennemgås de enkelte instruktioner, som modelmaskinen kan udføre. Instruktionerne gennemgås i nummerorden sorteret efter den to-cifret operationskode.

01 LOAD

Med denne instruktion henter man indholdet af den celle, som instruktionens adressedel peger på, til akkumalatorregistret.

02 ADD

Indholdet af den celle, som instruktionens adressedel peger på, hentes til processoren, hvor det adderes til indholdet af akkumulatorregistret. Resultatet af additionen anbringes i registret.

03 SUBTRACT

Indholdet af den celle, som instruktionens adressedel peger på, hentes til processoren, hvor det subtraheres fra indholdet af registret. Resultatet af subtraktionen anbringes i registret.

04 DIVIDE

Indholdet af registret divideres med indholdet af den celle, som instruktionens adressedel peger på. Kvotienten ved divisionen anbringes i registret.

05 MULTIPLY

Indholdet af den celle, som instruktionens adressedel peger på, hentes til processoren, hvor den multipliceres med indholdet af registret. Produktet anbringes i registret.

Modelmaskinens opbygning og virkemåde

06 STORE

Indholdet af registret gemmes i den celle, som instruktionens adressedel peger på.

07 JUMP

Med denne og de to næste instruktioner er det muligt at gribe ind i, fra hvilken celle i lageret den næste instruktion skal hentes. Denne instruktion har til formål at bevirke et hop i programmet til den celle, som instruktionens adressedel peger på. I praksis sker dette ved, at indholdet af adresseregistret overføres til programtælleren.

08 TESTZERO

Denne instruktion tester, om indholdet af registret er nul. Hvis udfaldet af testen er sandt (registret er nul), springes næste instruktion over ved at addere 1 til programtælleren. Ellers fortsættes på normal måde med næste instruktion.

09 TESTGREATER

Indholdet af den celle, som instruktionens adressedel peger på, overføres til processoren, hvor det sammenlignes med indholdet af registret. Hvis indholdet af cellen er større end indholdet af registret, er testen sand, og der adderes 1 til programtælleren. Ellers fortsættes på normal måde med næste instruktion.

99 STOP

Udførelsen af denne instruktion bevirker, at programmet stoppes.

Eksempel på et program.

Indholdet af celle 20 og 21 ønskes adderet, og resultatet anbragt i celle 20. Hvis de to tal, som ønskes adderet, for eksempel er 14 og 49, kan en løsning af problemet programmeres på denne måde:

```
000: 01000020     LOAD CELLE 20
001: 02000021     ADD CELLE 21
002: 06000020     STORE CELLE 20
003: 99000000     STOP
 :
 :
020: 00000014
021: 00000049
 :
```

Modelmaskinens opbygning og virkemåde

programafbrydelser (interrupts) i modelmaskinen.

Den normale måde at stoppe et program på er ved at udføre STOP-instruktionen. En sådan afbrydelse af programmet er tilsigtet og planlagt; men der er også mulighed for, at maskinen selv kan afbryde programmet på en mere eller mindre utilsigtet måde. En sådan situation, hvor maskinen selv stopper programmet, kalder man en afbrydelse eller et interrup.

Et interrupt forekommer, når maskinen bliver stillet overfor en uløselig opgave. Modelmaskinen kan afbryde et program som følge af tre forskellige fejl. For det første kan der opstå adresseringsfejl. En adresseringsfejl opstår, hvis en lagerreference enten i programtælleren eller i adresseregistret peger på en celle, som ikke findes i lageret.

For det andet kan der ske afbrydelse på grund af en invalid operationskode. Denne situation opstår, når styreenheden ikke kan genkende operationskoden i operationskoderegistret. En sådan fejl opstår typisk, når en celle som indeholder data, bringes til processoren, og ved et udheld bliver fortolket som om den indeholdt en instruktion.

Den tredje afbrydelse sker, hvis man i en divisionsinstruktion forsøger at dividere med nul, idet division med nul er og bliver umuligt. Programmeringsmæssigt kan man sikre sig imod, at dette problem opstår ved at teste divisor, inden divisionen udføres.

Endelig kan der opstå en fjerde situation, som dog ikke udløser nogen afbrydelse. Situationen kaldes overflow, op opstår, når resultatet af en addition eller multiplikation ikke kan stå i registret. Maskinen markerer, at situationen er opstået, hvorefter de mest betydende cifre i resultatet afskæres, og maskinen fortsætter.

Brugervejledning for modelmaskineprogrammet.

Med henblik på at illustrere principperne bag den foran beskrevne modelmaskines virkemåde, er modelmaskinen blevet programmeret, således at en arbejdende modelmaskine simuleres på en PC-farveskærm som en slags tegnefilm. Ved hjælp af forskellige kommandoer kan man styre simuleringen og på den måde få en klar fornemmelse af maskinens virkemåde i forskellige situationer.

Opstart af systemet.

Systemet kan køres på enhver IBM-kompatibel PC'er med farveskærm. Når maskinen er tændt, og operativsystemet er indlæst, startes modelmaskineprogrammet ved indtastning af kommandoen:

ASM

Brugervejledning

Hvis man ikke råder over en farveskærm, må man anvende den sort-hvide udgave af modelmaskineprogrammet, som startes ved indtastning af kommandoen:

ASMSH

Der er mindre maskinafhængige forskelle i den måde modelmaskinen præsenteres på skærmen; men selve programmets virkemåde er den samme.

Efter opstarten kommer et billede af modelmaskinen frem på skærmen, og samtidig opfordres man til at foretage et funktionsvalg.

Funktionsvalg.

Når man opfordres til at foretage et funktionsvalg, sker det med følgende udskrift i højre side af skærmen:

FUNKTIONSVALG

E - Exit
G - Gem program
H - Hastighedsændring
I - Indsæt i celler
L - Læs program ind
N - Nulstil lagret
S - Startadr. i P-tæller
T - Trinvis udførelse til/fra
U - Udfør programmet
X - Sætter lyd til/fra

INDTAST FUNKTION:

Selve funktionsvalget foretages ved indtastning af det ønskede bogstav på tastaturet. Man kan anvende såvel små som store bogstaver. Funktionen vil blive kaldt straks, når det ønskede bogstav er indtastet. Indtastningen skal altså ikke afsluttes med tryk på returknappen i forbindelse med funktionsvalget.

E - Exit

Bogstavet E indtastes, når man ønsker at afslutte simuleringsprogrammet. Kontrollen gives tilbage til operativsystemet.

G - Gem program

Denne funktion kan med fordel anvendes, når man har indlagt et program i modelmaskinens interne lager, idet funktionen kan bruges til at gemme en kopi af det interne lagers 40 celler på pladelager eller diskette, hvorfra de senere kan indlæses igen. Hvis man ønsker at gemme programmet i sin oprindelige form, bør man gemme det umiddelbart efter, at man har indtastet det, og før en eventuel udførelse påbegyndes, da udførelsen kan bevirke, at nogle af cellernes indhold ændres.

Brugervejledning

Når funktionen er blevet kaldt, udskrives følgende nederst til højre på skærmen:

Programmet gemmes

Indtast navn: [ ]

I det oplyste felt indtastes det navn, som man ønsker at gemme programmet under. Findes der i forvejen et program med dette navn, udskrives der en fejlmeddelse, og programmet gemmes ikke.

H - Hastighedsændring

Det er muligt at ændre den hastighed, hvormed simuleringen af datastrømmene i adresse- og databus foregår. Dette gøres ved at variere den forsinkelsesfaktor, som er skudt ind i simuleringen af datastrømmene, for at 'tegnefilmen' ikke skal køre for hurtigt. Denne forsinkelsesfaktor er som standardværdi blevet tildelt værdien 200 millisekunder mellem hvert nyt billede af datastrømmen. Gøres forsinkelsesfaktoren større, sættes arbejdshastigheden ned. Gøres forsinkelsesfaktoren mindre, øges arbejdshastigheden.

Når funktionen kaldes, udskrives følgende tekst til højre på skærmen:

Ændring af programmets arbejdshastighed under udførelsen - jo større tal der indtastes - jo langsommere

INDTAST FAKTOR: [200]

I det oplyste felt står standardværdien 200, som kan accepteres ved tryk på returknappen. Ønskes en anden værdi, indtastes en værdi mellem 1 og 9999.

I - Indsæt i celler

Denne funktion kaldes, hvis man vil indlægge program og data i det interne lagers celler. Hvis man gør dette umiddelbart efter, at man har startet programmet, er cellerne nulstillet; men hvis der allerede står noget i cellerne, bør man nulstille dem ved kald af nulstil-funktionen (se side 8), inden man begynder at indtaste et nyt indhold.

Når man har kaldt funktionen, skal man svare på følgende spørgsmål, som kommer frem nederst til højre på skærmen:

INDSÆT I CELLE: [99]

CELLEINDHOLD : [ ]

Brugervejledning

Det oplyste felt er det felt, hvor der skal indtastes nummeret på den celle, som skal have tildelt en værdi. Det indtastede cellenummer skal være mellem 0 og 39 eller 99. Indtastning af cellenummer 99 bevirker, at indtastningsfunktionen afbrydes, og der vendes tilbage til funktionsvalget. Cellenummer 99 er det cellenummer, som maskinen selv foreslår, og dette betyder, at indtastningen af celleindhold kan stoppes blot ved at trykke på returknappen. Hvis cellenummeret ligger uden for de acceptable grænser, bipper maskinen.

Når et acceptabelt cellenummer er indtastet, skal cellen i næste linie tildeles et indhold. Det ønskede celleindhold indtastes i det oplyste felt. Indholdet skal være numerisk, hvilket vil sige, at der kun må indtastes cifre.

Skulle man ved et uheld komme til at afbryde indtastningen af celleindhold, før man er helt færdig, eller skulle man få behov for at ændre allerede indtastede cellers indhold, kan man blot kalde indtastningsfunktionen igen.

L - Læs program ind

Denne funktion modsvarer den tidligere beskrevne funktion, hvormed det var muligt at gemme et indtastet program på pladelager eller diskette. Med denne funktion er det muligt at hente et tidligere gemt program og få det lagt ind i modelmaskinens lager - klar til udførelse.

Når funktionen er blevet kaldt, udskrives følgende nederst på skærmen:

Gemt program læses ind

Indtast navn : [ ]

I det oplyste felt indtastes navnet på det program, som man ønsker at indlæse. Hvis det ønskede program ikke findes, udskrives en fejlmeddelse, hvorefter man opfordres til at foretage fornyet funktionskald.

N - Nulstil lageret

Denne funktion kaldes, når modelmaskinens interne lager ønskes nulstillet. Dette behov opstår hovedsageligt, når et nyt program skal indtastes. Man skal være opmærksom på, at alle celler nulstilles. Hvis det kun er enkelte celler, som ønskes nulstillet, skal man anvende den normale indtastningsfunktion, hvormed man også kan indsætte nuller i enkelte celler.

S - Startadresse indsættes i P-tæller

Inden et program, som ligger i modelmaskinens interne lager, kan udføres, er det nødvendigt at indsætte programmets startadresse i programtælleren. Programmets startadresse er nummeret på den celle i det interne lager, hvor programmets første instruktion findes.

Brugervejledning

Når funktionen kaldes, udskrives følgende tekst nederst til højre på skærmen:

Programmets startadresse
indsættes i P-tæller

INDTAST P-TÆLLER: [0]

I det oplyste felt indtastes nummeret på den celle, hvor programmets første instruktion findes. Den foreslåede standardværdi for feltet er 0, som kan accepteres ved blot at trykke på returknappen. Samtidig med at P-tælleren bliver tildelt en værdi, nulstilles de øvrige registre.

T - Trinvis udførelse til/fra

Af pædagogiske årsager kan det være overordentligt hensigtsmæssigt at lade maskinen stoppe efter udførelse af hver enkelt instruktion. Dette kan lade sig gøre ved at slå faciliteten "trinvis udførelse" til ved tryk på T-knappen på tastaturet. Næste tryk på T-knappen slår trinvis udførelse fra igen og så fremdeles. Når trinvis udførelse er slået til, skal man kalde udfør-instruktionen hver gang, man ønsker næste instruktion udført. Standardværdien for trinvis udførelse ved opstart af programmet er "fra".

U - Udfør programmet

Når man ønsker at starte udførelsen af programmet, eller hvis faciliteten trinvis udførelse er slået til, og man ønsker at udføre den næste instruktion, kaldes udfør-funktionen. Det er i denne funktion, at datamaskinens funktion under udførelse af programmet simuleres. Når udfør-funktionen kaldes udskrives følgende tekst til højre på skærmen:

Programmet udføres

P - Pause i udførelsen

Nedenunder denne tekst, som er fast under hele udførelsen, fortæller skiftende tekster om, hvad maskinen er i gang med netop nu. Med hensyn til nærmere beskrivelse af modelmaskinens arbejdsmåde henvises til den tidligere gennemgåede og mere uddybende beskrivelse af modelmaskinens funktion og instruktionssæt.

Den ovenfor omtalte faste tekst, som står på skærmen under hele udførelsen hentyder til, at det er muligt midlertidigt at afbryde programudførelsen. Dette gøres ved at holde P-tasten nedtrykket, indtil en særlig "Pause menu" er kommet frem på skærmens højre side. I forbindelse med pause menuen har man mulighed for at ændre forskellige parametre, som har relation til udførelsen af programmet, såsom hastighed, trinvis udførelse og lyd. Når man har rettet parametrene, kan programudførelsen genoptages. Hele pause menuen er nærmere beskrevet i et senere afsnit(se side 10).

Brugervejledning

X - Sætter lyd til/fra

Som omtalt i forbindelse med udfør-funktionen fremkommer der under programmets udførelse tekster på højre side af skærmen, som beskriver, hvad modelmaskinen er i gang med netop nu. I forbindelse hermed er det muligt, at få maskinen til at udsende en bip-lyd hver gang der fremkommer en ny tekst. Dette kan være en hjælp, hvis man vil prøve at følge med i, hvad der foregår, ved at læse teksterne efterhånden som de kommer frem på skærmen. Bip-lyden slås til ved at trykke på X-tasten. Bip-lyden slås fra ved at trykke på X-tasten igen og så fremdeles. Som standardværdi ved opstart af programmet er bip-lyden slået fra.

Pause menuen

Som omtalt i forbindelse med udfør-funktionen er det muligt at foretage en midlertid afbrydelse af programudførelsen ved at hold P-tasten nedtrykket indtil nedenstående tekst kommer frem i højre side af skærmen:

UDFØRELSEN ER STOPPE

F - Fortsæt udførelse
H - Hastighedsændring
X - Sætter lyd til/fra
A - Afbryd efter udfø-
    relsen af igangvæ-
    rende ordre
T - Trinvis udf. til/fra

Nedenfor gennemgås de to funktioner, som ikke er kendt fra det allerede gennemgåede normale funktionsvalg.

F - Fortsæt udførelse

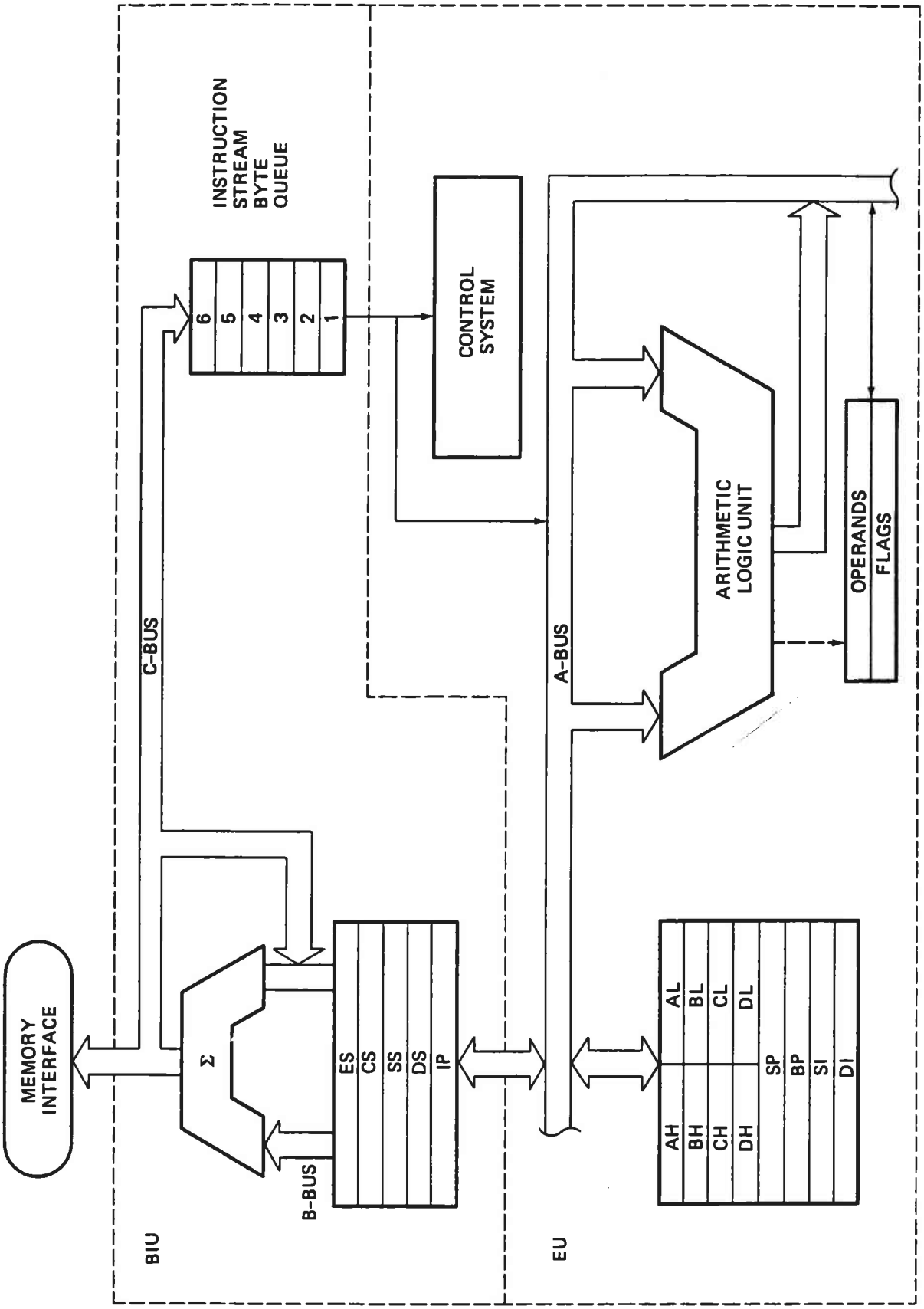
Når man har foretaget de ønskede ændringer af parametrene, kan man genoptage den afbrudte udførelse ved at trykke på F-tasten.

A - Afbryd efter udførelse af igangværende ordre

Den funktion kan anvendes i situationer, hvor man under programudførelsen opdager fejl, som bevirker, at resten af programudførelsen mister sin værdi. I stedet for at skulle sidde og vente på den bitre ende, kan man ved kald af denne funktion få maskinen til at stoppe programudførelsen, når den igangværende instruktion er færdig. Man er altså nødt til at trykke på F-tasten for at genoptage udførelsen, før programmet vil blive afbrudt.

4.3.a

8086 CPU arkitektur



8086 internal block diagram. (*Intel Corp.*)

Instruction Set for the Intel 8088

The Intel 8088 Instruction Set is explained in detail in this section. The instruction set for the Intel 8088 is the same as the instruction set for the Intel 8086.

Instruction Set Groupings

The 8086/8088 instructions can be grouped according to the purpose of the instruction. The six general classifications of instructions are

Data transfer
Arithmetic
Bit manipulation
String manipulation
Control transfer
Flag and processor control

Table 8088.1 shows how the individual instructions within these categories are grouped.

Table 8088.1
Instruction Set Groupings

Instruction	*Meaning*
Data Transfer	
IN	Input from port
LAHF	Load AH register with flags
LDS	Load DS register
LEA	Load effective address
LES	Load ES register
MOV	Move
OUT	Output to port
POP	Remove data from stack
POPF	Remove flags from stack
PUSH	Place data on stack
PUSHF	Place flags on stack
SAHF	Store AH into flag register
XCHG	Exchange
XLAT	Translate

Table 8088.1—cont.

Instruction	*Meaning*
Arithmetic	
AAA	ASCII adjust for addition
AAD	ASCII adjust for division
AAM	ASCII adjust for multiplication
AAS	ASCII adjust for subtraction
ADC	Add with carry
ADD	Add
CBW	Convert byte to word
CMP	Compare
CWD	Convert word to doubleword
DAA	Decimal adjust for addition
DAS	Decimal adjust for subtraction
DEC	Decrement
DIV	Divide
IDIV	Integer divide
IMUL	Integer multiply
INC	Increment
MUL	Multiply
NEG	Negate
SBB	Subtract with carry
SUB	Subtract
Bit Manipulation	
AND	Logical AND on bits
NOT	Logical NOT on bits
OR	Logical OR on bits
RCL	Rotate left through carry
RCR	Rotate right through carry
ROL	Rotate left
ROR	Rotate right
SAL	Arithmetic shift left
SAR	Arithmetic shift right
SHL	Shift left
SHR	Shift right
TEST	Test bits
XOR	Logical exclusive-or on bits

Table 8088.1—cont.

Instruction	*Meaning*
String Manipulation	
CMPSB	Compare strings byte for byte
CMPSW	Compare strings word for word
LODSB	Load a byte from string into AL
LODSW	Load a word from string into AX
MOVSB	Move string byte-by-byte
MOVSW	Move string word-by-word
REP	Repeat
REPE	Repeat if equal
REPNE	Repeat if not equal
REPNZ	Repeat if not zero
REPZ	Repeat if zero
SCASB	Scan string for byte
SCASW	Scan string for word
STOSB	Store byte in AL at string
STOSW	Store word in AX at string
Control Transfer	
CALL	Perform subroutine
INT	Software interrupt
INTO	Interrupt on overflow
IRET	Return from interrupt
JA	Jump if above
JAE	Jump if above or equal
JB	Jump if below
JBE	Jump if below or equal
JC	Jump on carry
JCXZ	Jump if CX=0
JE	Jump if equal
JG	Jump if greater
JGE	Jump if greater or equal
JL	Jump if less than
JLE	Jump if less than or equal
JMP	Jump
JNA	Jump if not above
JNAE	Jump if not above or equal
JNB	Jump if not below
JNBE	Jump if not below or equal
JNC	Jump on no carry

Table 8088.1—cont.

Instruction	*Meaning*
JNE	Jump if not equal
JNG	Jump if not greater than
JNGE	Jump if not greater than or equal
JNL	Jump if not less than
JNLE	Jump if not less than or equal
JNO	Jump on no overflow
JNP	Jump on no parity
JNS	Jump on not sign
JNZ	Jump on not zero
JO	Jump on overflow
JP	Jump on parity
JPE	Jump on parity even
JPO	Jump on parity odd
JS	Jump on sign
JZ	Jump on zero
LOOP	Loop
LOOPE	Loop while equal
LOOPNE	Loop while not equal
LOOPNZ	Loop while not zero
LOOPZ	Loop while zero
RET	Return from subroutine
Flag and Processor Control	
CLC	Clear carry flag
CLD	Clear direction flag
CLI	Clear interrupt flag
CMC	Complement carry flag
ESC	Escape
HLT	Halt
LOCK	Lock bus
NOP	No operation
STC	Set carry flag
STD	Set direction flag
STI	Set interrupt flag
WAIT	Wait

Dekodningstabel for 8086-maskininstruktioner

Hi \ Lo	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADD b,f,r/m	ADD w,f,r/m	ADD b,t,r/m	ADD w,t,r/m	ADD b,ia	ADD w,ia	PUSH ES	POP ES	OR b,f,r/m	OR w,f,r/m	OR b,t,r/m	OR w,t,r/m	OR b,i	OR w,i	PUSH CS	
1	ADC b,f,r/m	ADC w,f,r/m	ADC b,t,r/m	ADC w,t,r/m	ADC b,i	ADC w,i	PUSH SS	POP SS	SBB b,f,r/m	SBB w,f,r/m	SBB b,t,r/m	SBB w,t,r/m	SBB b,i	SBB w,i	PUSH DS	POP DS
2	AND b,f,r/m	AND w,f,r/m	AND b,t,r/m	AND w,t,r/m	AND b,i	AND w,i	SEG =ES	DAA	SUB b,f,r/m	SUB w,f,r/m	SUB b,t,r/m	SUB w,t,r/m	SUB b,i	SUB w,i	SEG =CS	DAS
3	XOR b,f,r/m	XOR w,f,r/m	XOR b,t,r/m	XOR w,t,r/m	XOR b,i	XOR w,i	SEG =SS	AAA	CMP b,f,r/m	CMP w,f,r/m	CMP b,t,r/m	CMP w,t,r/m	CMP b,i	CMP w,i	SEG =DS	AAS
4	INC AX	INC CX	INC DX	INC BX	INC SP	INC BP	INC SI	INC DI	DEC AX	DEC CX	DEC DX	DEC BX	DEC SP	DEC BP	DEC SI	DEC DI
5	PUSH AX	PUSH CX	PUSH DX	PUSH BX	PUSH SP	PUSH BP	PUSH SI	PUSH DI	POP AX	POP CX	POP DX	POP BX	POP SP	POP BP	POP SI	POP DI
6	PUSHA	POPA	BOUND w,f,r/m						PUSH w,i	IMUL w,i	PUSH b,i	IMUL b,i	INS b	INS w	OUTS b	OUTS w
7	JO	JNO	JB/ JNAE	JNB/ JAE	JE/ JZ	JNE/ JNZ	JBE/ JNA	JNBE/ JA	JS	JNS	JP/ JPE	JNP/ JPO	JL/ JNGE	JNL/ JGE	JLE/ JNG	JNLE/ JG
8	Immed b,r/m	Immed w,r/m	Immed b,r/m	Immed is,r/m	TEST b,r/m	TEST w,r/m	XCHG b,r/m	XCHG w,r/m	MOV b,f,r/m	MOV w,f,r/m	MOV b,t,r/m	MOV w,t,r/m	MOV sr,f,r/m	LEA	MOV sr,t,r/m	POP r/m
9	XCHG AX	XCHG CX	XCHG DX	XCHG BX	XCHG SP	XCHG BP	XCHG SI	XCHG DI	CBW	CWD	CALL l,d	WAIT	PUSHF	POPF	SAHF	LAHF
A	MOV m→AL	MOV m→AX	MOV AL→m	MOV AX→m	MOVS	MOVS	CMPS	CMPS	TEST b,i,a	TEST w,i,a	STOS	STOS	LODS	LODS	SCAS	SCAS
B	MOV i→AL	MOV i→CL	MOV i→DL	MOV i→BL	MOV i→AH	MOV i→CH	MOV i→DH	MOV i→BH	MOV i→AX	MOV i→CX	MOV i→DX	MOV i→BX	MOV i→SP	MOV i→BP	MOV i→SI	MOV i→DL
C	Shift b,i	Shift w,i	RET. (i+SP)	RET	LES	LDS	MOV b,i,r/m	MOV w,i,r/m	ENTER	LEAVE	RET. l.(i+SP)	RET l	INT Type 3	INT (Any)	INTO	IRET
D	Shift b	Shift w	Shift b,v	Shift w,v	AAM	AAD		XLAT	ESC 0	ESC 1	ESC 2	ESC 3	ESC 4	ESC 5	ESC 6	ESC 7
E	LOOPNZ/ LOOPNE	LOOPZ/ LOOPE	LOOP	JCXZ	IN b	IN w	OUT b	OUT w	CALL d	JMP d	JMP l,d	JMP si,d	IN v,b	IN v,w	OUT v,b	OUT v,w
F	LOCK		REP	REP Z	HLT	CMC	Grp 1 b,r/m	Grp 1 w,r/m	CLC	STC	CLI	STI	CLD	STD	Grp 2 b,r/m	Grp 2 w,r/m

where:

mod□r/m	000	001	010	011	100	101	110	111
Immed	ADD	OR	ADC	SBB	AND	SUB	XOR	CMP
Shift	ROL	ROR	RCL	RCR	SHL/SAL	SHR	—	SAR
Grp 1	TEST	—	NOT	NEG	MUL	IMUL	DIV	IDIV
Grp 2	INC	DEC	CALL id	CALL l,id	JMP id	JMP l,id	PUSH	—

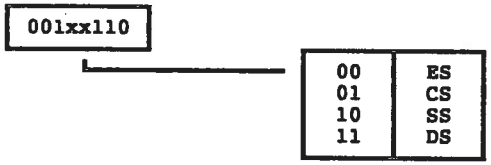
b = byte operation
d = direct
f = from CPU reg
i = immediate
ia = immed. to accum.
id = indirect
is = immed. byte, sign ext.
l = long ie. intersegment
m = memory
r/m = EA is second byte
si = short intrasegment
sr = segment register
t = to CPU reg
v = variable
w = word operation
z = zero

OBJECT KODE FORMAT.

8088/8086 Object koderne er sammensat af forskellige dele, alt efter instruktions typen. De instruktioner, der ikke kræver operander, fylder som regel 1 byte, hvorimod instruktioner med operander, kan fylde fra 2 til 7 bytes.

Nedenstående er generelt for alle instruktionerne, men der findes instruktioner, der har sit eget specielle format.

Segment override byte.
Dette er den først byte, der kan være i en instruktion. Den fortæller at man ønsker at benytte et andet segment register end hvad default angiver. Den kan kun forekomme sammen med instruktioner, med memory adressering.



Instruktioners object kode.
Denne byte fortæller, hvilken instruktion, der er tale om. I denne byte kan der forekomme nogle bits, der har betydning for instruktionens format. Disse kan bla. være:

D. (er placeret som bit nummer 1).
Denne bit angiver, hvad der er source og hvad der er destination (højre og venstre operand).

D = 1:
Så angives destinationen af REG-feltet og source af MOD og R/M feltet i den efterfølgende MODRM byte.

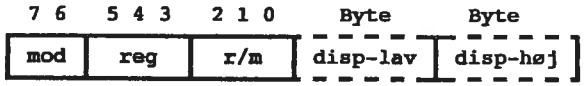
D = 0:
Så angives destinationen af MOD og R/M felterne og source angives af REG feltet, i den efterfølgende MODRM byte.

W. (er placeret som bit nummer 0).
Denne bit angiver om operanderne er bytes eller words.

W = 0: Er operanderne BYTES.
W = 1: Er operanderne WORDS.

MORDM byten/bytes.
Denne (fylder 1,2 eller 3 bytes) benyttes til at specificere operanden/operanderne i instruktionen. Hvordan denne opfattes er lidt forskellig fra instruktion til instruktion.

Den først byte (MODRM), ser generelt sådan ud:



MOD feltet angiver om MODRM efterfølges af et displacement til instruktionen (samt om det er et 8 eller 16 bits displacement), eller om r/m feltet skal opfattes som et "REG" felt.

Mod	Displacement.
00	Displacement findes ikke *
01	Der findes et 8 bits displacement
10	Der findes et 16 bits displacement
11	R/M feltet opfattes som et "REG" felt.

NB: Hvis MOD = 00 og R/M feltet er 110, findes et 16 bit displacement, der opfattes som EA (addr-lav.addr-høj).

REG feltet angiver hvilket register, der skal indgå som en operand. Bemærk at R/M feltet også kan opfattes som et REG felt (hvis MOD = 11).

REG	W = 1	W = 0
000	AX	AL
001	CX	CL
010	DX	DL
011	BX	BL
100	SP	AH
101	BP	CH
110	SI	DH
111	DI	BH

NB: I visse instruktioner, er REG feltet i selve object kode byten, f.eks POP og PUSH instruktionerne. Betydningen er dog den samme som i skemaet.

R/M feltet angiver sammen med MOD feltet, hvilken memory adresseringsform, der eventuelt skal benyttes (MOD feltet <> 11). Nedenstående skema viser hvilke kombinationer af MOD og R/M feltet, der kan forekomme.

	MOD=00	MOD = 01	MOD = 10	MOD=11	
R/M	Offset	Offset	Offset	W=1	W=0
000	BX + SI	BX + SI + D8	BX + SI + D16	AX	AL
001	BX + DI	BX + DI + D8	BX + DI + D16	CX	CL
010	BP + SI	BP + SI + D8	BP + SI + D16	DX	DL
011	BP + DI	BP + DI + D8	BP + DI + D16	BX	BL
100	SI	SI + D8	SI + D16	SP	AH
101	DI	DI + D8	DI + D16	BP	CH
110	Note 1.	BP + D8	BP + D16	SI	DH
111	BX	BX + D8	BX + D16	DI	BH

Note 1: Hvis Mod er 00 og R/M er 110, efterfølges MODRM byten af to bytes, der angiver et offset.

Eksempel 1:
Object kode for MOV DX,[BX+1]

Formatet er MOV reg,mem, og object koden for dette format er:

100010dw	mod reg r/m	disp8/disp16

For ovenstående instruktion, vil object kode være:

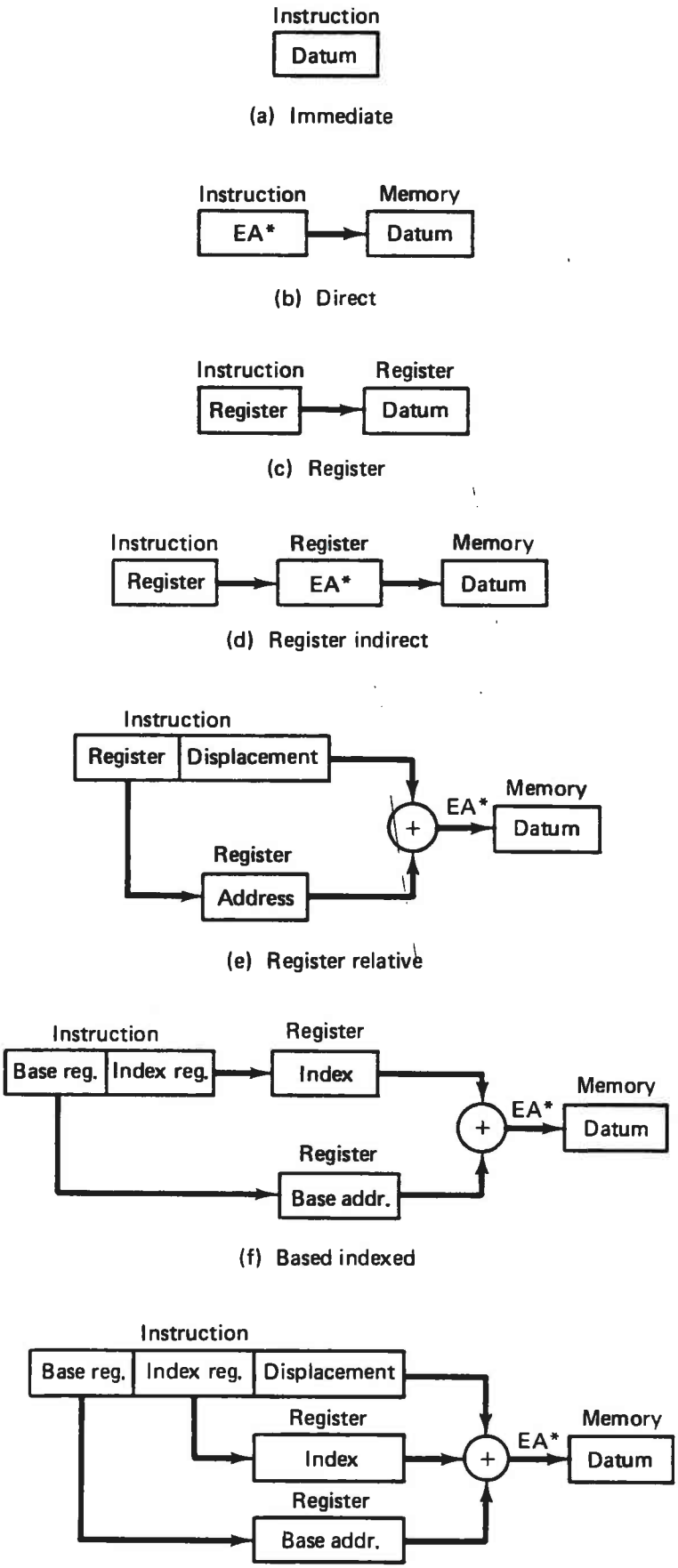
10001011	01 010 111	00000001

= 8BH 57H 01H.

- 8 bits displacement.
- Når MOD = 01, betyder 111 i R/M, BX+disp8.
- 010 i REG angiver DX.
- MOD = 01 angiver disp8.
- W bit = 1, operanderne er words.
- D bit = 1 angiver at REG angiver destinationen og MOD og R/M angiver sourcen.

8086 Architecture

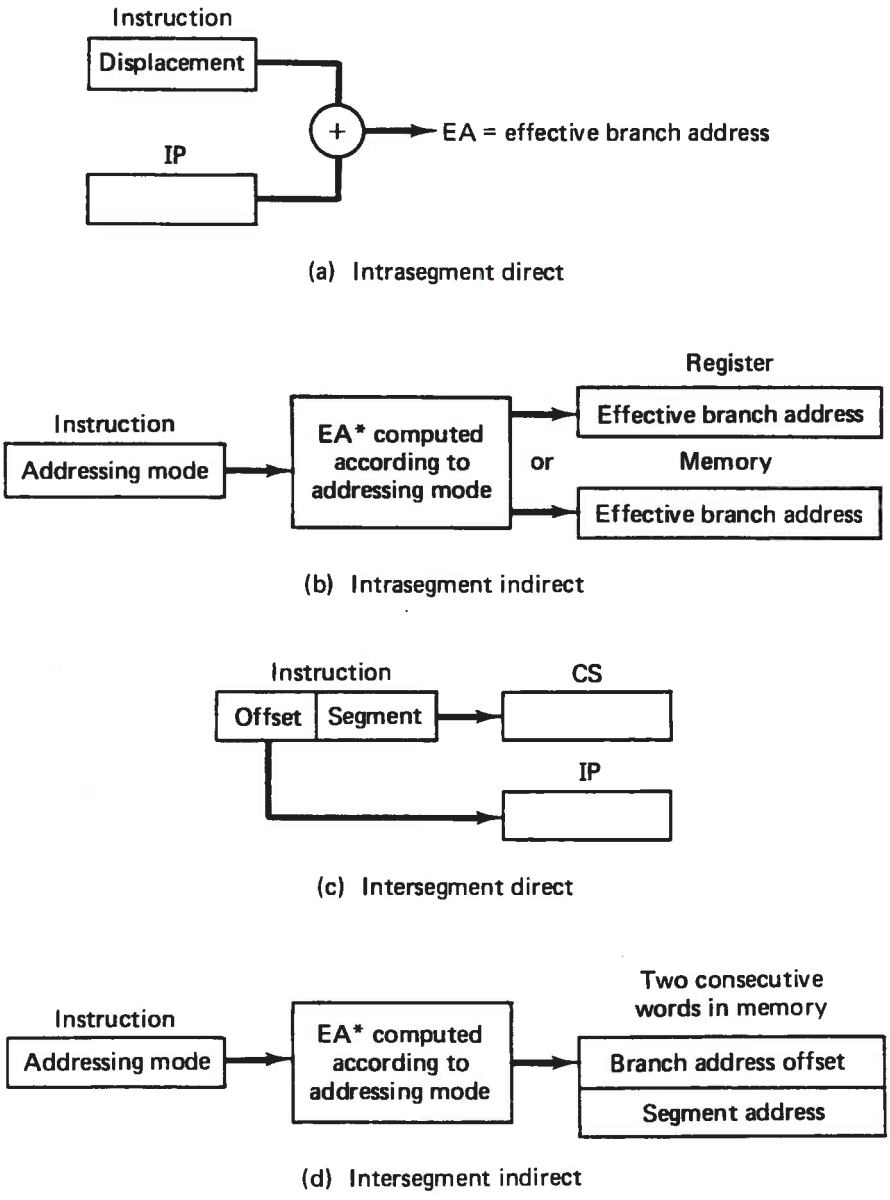
Data-related addressing modes.



*EA is added to 16_{10} times the contents of the appropriate segment register.

8086 Architecture

Branch-related addressing modes.



*EA is added to 16_{10} times the contents of the appropriate segment register.

DOS-KASSEN

Hvordan virker interrupts?

Interrupts er uundværlige - men irriterende.

Af Klaus Møllgaard

Med den hastigt stigende udbredelse af "add-on boards" (blandt andet multimedia-hardware) til opgradering af PC'er vil et stigende antal brugere løbe ind i både interrupt-kanal, DMA-kanal og adresse-konflikter.

Interrupts er væsentlige for, at en PC overhovedet kan fungere, og det følgende er ment som en kort indføring i interrupt-funktionen, de problemer interrupt-konflikter kan give for PC-brugeren og nogle retningslinier for, hvordan man undgår interrupt-konflikter.

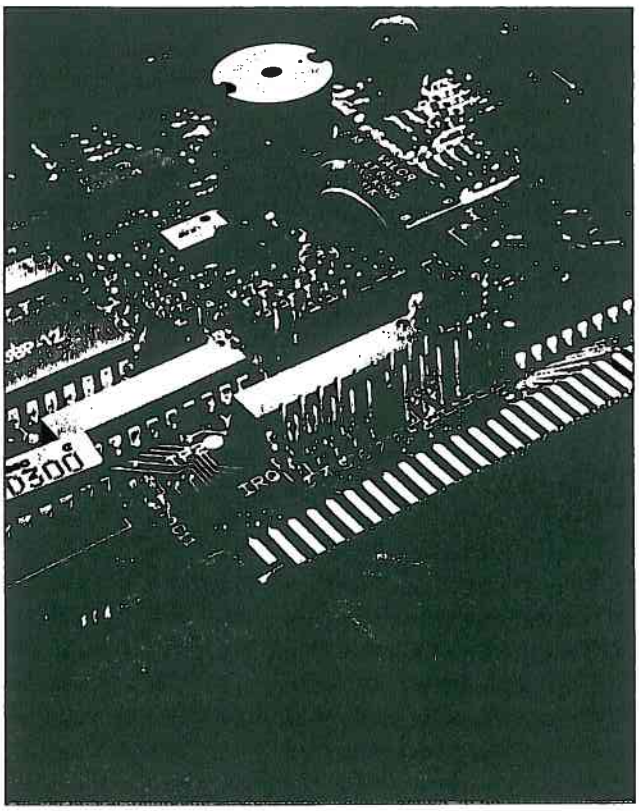


Foto: Morten Holtum Nielsen

Valg af IRQ-kanal kan foretages ved at jongle-re med jumper-indstillingen på kortet. I visse tilfælde kan IRQ-kanalen ændres med et program.

Hvordan håndterer vi afbrydelser?

Interrupt betyder afbrydelse og for at forstå, hvad et interrupt er, er det vigtigt at forstå, hvordan PC'en håndterer afbrydelser. Hverdagen er fyldt med uregelmæssige, ikke forudsigelige afbrydelser, og hvordan håndterer vi selv dem? Søndag morgen - koncentreret om avislæsning - telefonen ringer, samtidig med at det banker på døren, og den mindste skriger på et bleskift. Tre samtidige afbrydelser i det, der var hovedopgaven: at læse avisen. Afbrydelserne skal først prioriteres efter betydning og dernæst klares én for én, før avislæsningen kan genoptages.

PC'en håndterer uregelmæssige eller asynkrone afbrydelser på samme måde - blot som oftest noget hurtigere og med en lidt mere rigid prioritering. En asynkron afbrydelse kan være et tryk på tastaturet, en bevægelse af musen, søgning på disken, opdatering af skærmen og så videre. Hvis PC'en regelmæssigt skulle ud og checke samtlige I/O-enheder for, om de muligvis havde noget, de skulle have hjælp til, ville CPU-regnekraften nedsættes betragteligt.

Interrupt request-kanaler

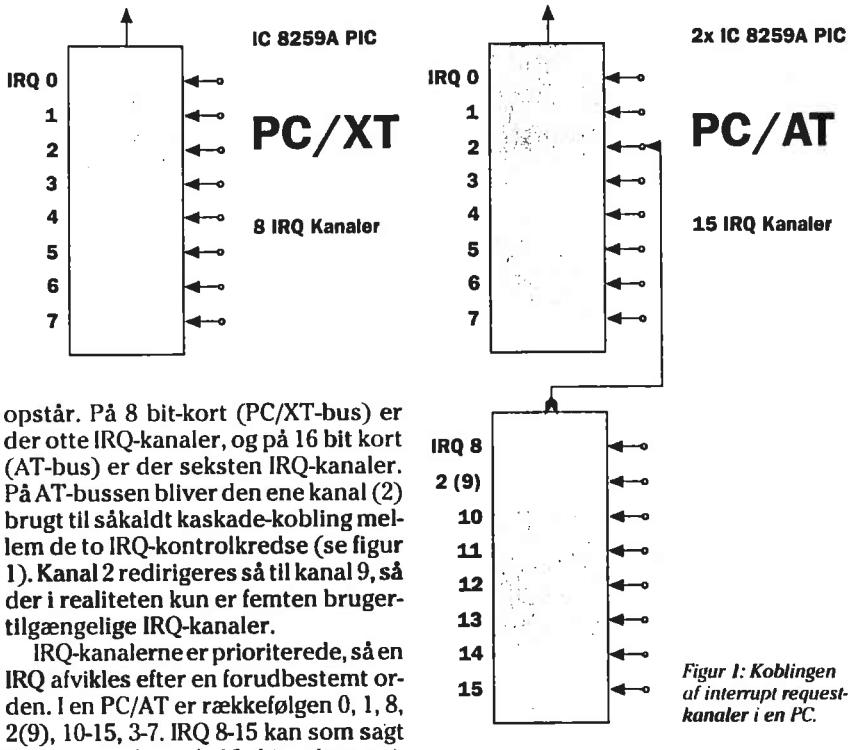
Computeren har derfor indbygget en række interrupt-kanaler, der kan tildeles de forskellige perifere enheder til at håndtere sådanne asynkrone afbrydelser. Her kan de så rejse en interrupt request (IRQ) - en anmodning om afbrydelse af og hjælp fra CPU'en. Computerens CPU udfører sine tildelte opgaver og instruktioner på en skemalagt og sekventiel måde indtil en IRQ dukker op. CPU'en stopper, hvad den er i gang med, behandler (og udfører) interrupt'en og genoptager herefter sin sekventielle afvikling af de tidligere instruktioner.

Interrupt-problemerne på dagens computere udspringer (endnu engang) af IBM's manglende fremsyn med hensyn til udviklingen inden for den personlige computer (det er altid nemt at være bagklog). IBM "glemte" nemlig at lære PC'en at dele IRQ-kanaler på ISA-bussen. Utallige er de timer, PC-ejere har brugt på at løse problemer, der i sidste ende har vist sig at ligge i en interrupt-konflikt; for eksempel mellem de serielle porte, modem og en serielt tilsluttet mus på IRQ 3 eller 4. Eller VGA/netkort-konflikt på IRQ 2, der fungerer nogenlunde på samme IRQ, indtil man kalder Windows. EISA- og MCA-bus systemkort tillader til en vis grad deling af IRQ mellem flere funktioner og overkommer dermed en af ISA-buskonstruktionens store bagdele.

Soft- og hardware IRQs

Der er i alt 256 IRQ-kanaler i en PC. Alle IRQ-kanaler kan adresseres via software, men de første 8/16 (IRQ 0-15) er også addressérbare direkte via ISA-bussen via tilsluttet hardware, og det er som regel her, problemerne ▷

DOS-KASSEN



Figur 1: Koblingen af interrupt request-kanaler i en PC.

opstår. På 8 bit-kort (PC/XT-bus) er der otte IRQ-kanaler, og på 16 bit kort (AT-bus) er der seksten IRQ-kanaler. På AT-bussen bliver den ene kanal (2) brugt til såkaldt kaskade-kobling mellem de to IRQ-kontrolkredse (se figur 1). Kanal 2 redirigeres så til kanal 9, så der i realiteten kun er femten brugertilgængelige IRQ-kanaler.

IRQ-kanalerne er prioriterede, så en IRQ afvikles efter en forudbestemt orden. I en PC/AT er rækkefølgen 0, 1, 8, 2(9), 10-15, 3-7. IRQ 8-15 kan som sagt kun anvendes af 16 bit ekspansionsslot/kort, hvilket giver en stor belastning af de første otte kanaler. Her ligger både COM- og LPT-porte, floppydisk-controller, tastatur og systemtimer, mens harddisk-controller og FPU er flyttet højere op (se tabel 1). For at opnå fuld kompatibilitet med gamle maskiner, har mange producenter indtil nu valgt at lave kort i 8 bit-udgaver med deraf følgende potentielle konflikter kort imellem og mellem kort og kommunikationsporte. Det er ved at ændre sig i takt med, at 8 bit-busmaskiner er på vej ud.

Indstilling af en IRQ

På et kort kan IRQ-kanalvalg enten foregå via en fysisk justering med en jumper, der flyttes til den ønskede indstilling eller via et program, der læser den valgte kanal ind i en lille hukommelse på kortet. Jumperindstillingen er måske den mest overskuelige og den "ældste" måde at gøre det på. Den har den ulempe, at man skal splitte computeren ad for at ændre (eller kontrollere) indstillingen. Den nyere software-kontrollerede indstilling (af hardware interrupt) er meget nemmere og vinder mere og mere indpas på markedet. Her kalder man et lille program, der som regel er i stand til at fortælle brugeren, hvilke IRQ-kanaler, der er ledige til den nye enhed inden valget foretages.

Hvis man vælger en IRQ, der allerede er besat af en anden enhed, kan der opstå flere forskellige problemer, afhængig af hvilke to enheder, der konflikter med hinanden. Som regel giver konflikter sig udtryk i, at computeren låser uhjælpeligt fast og må boot'es om. Hvis computeren låser allerede under opstarten, kan man prøve at starte fra diskette og så finde problemet - ellers kan det være nødvendigt at fjerne de installerede kort et efter et; mindst betydende kort fjernes først - selvfølgelig. Andre årsager til problemer kan være BIOS I/O adresse-konflikt, memory adresse-konflikt eller sjældnere DMA kanal-konflikt (Direct Memory Access).

Husk at ...

Periferi-kort kommer med en standard indstilling af jumpere og DIP-switches, så de umiddelbart kan anvendes i de fleste computere uden problemer. Visse programmer, der understøtter fx lydkort, kan selv detektere de adresser og kanaler, som kortene anvender, mens andre programmer skal have informationen ved installation eller programstart. Det er derfor en god ide ved installation af ekstra kort at notere de forskellige indstillinger, så du ikke skal splitte maskinen ad hver gang, du bliver spurgt.

Med det stigende antal "add-on boards" (lyd, video, scannere, CD-ROM, osv.) vil man før eller siden løbe ind i problemer eller overvejelser med hensyn til fordelingen af IRQ-kanaler - især med for mange 8 bit-boards, hvor 16 bit-boards giver adgang til mange flere ikke reserverede kanaler. Betal hellere lidt mere for 16 bit-udgaver af kort; den investering betaler sig af flere grunde i længden. For alle 16 bit-kort bør du vælge IRQ i den høje ende, så 8 bit-kortene kan bruge de andre.

Du kan godt tildele et kort en interrupt fra en kommunikationsport, du ikke anvender (for eksempel LPT2 eller COM2), men hvis du har tre eller fire serielle eller parallelle porte eller modem, så husk at tage højde for, at port 1 og 3 samt port 2 og 4 som regel er sat op til at dele IRQ-kanal. Brug ikke IRQ 2 til noget, hvis du har VGA-kort eller netkort, det *kan* give problemer. □

Tabel 1

IRQ kanaler på PC/AT

	Fast anvendelse	Normal anvendelse
0	System-timer	
1	Tastatur	
2 (9)		Generel (VGA/netkort)
3		Generel (COM2/COM4)
4		Generel (COM1/COM3)
5		Generel (LPT2/LPT4)
6		Floppy controller
7		Generel (LPT1/LPT3)
8	Real time clock	
10		Generel
11		Generel
12		Generel
13	Coprocessor (FPU)	
14		Harddisk-controller
15		Generel

4.3.g

Anvendelse af 8086-registrene

Arbejdsregistrene:

AX **(Accumulator-register)**
Anvendes som allround arbejdsregister.
Kan opdelses i to "halve" registre (Ah og Al), der kan benyttes som selvstændige 8-bits registre.
Multiplikation og division kan kun udføres med dette register som "1. operant", idet DX-registret dog medtages ved 16-bit beregninger, således at DX-AX tilsammen fungerer som et 32-bits register.
IO-instruktioner knytter sig også til dette register.
Instruktionen CVB anvendes til at udvide værdi (med fortegn) i AL til 16-bit i AX. CWD anvendes til at udvide værdi (med fortegn) i AX til 32-bit i registrer-paret DXAX.

BX **(Base-register)**
Registret kan anvendes til memory-adressering (evt. sammen med SI- eller DI-registret).
Registret kan ligesom AX benyttes som to 8-bits registre.
Registret kan anvendes som ekstra arbejdsregister (register-variabel).

CX **(Counter-register)**
Registret anvendes som counter-register i forbindelse med LOOP-instruktionerne samt nogle af streng-instruktionerne. I disse instruktioner kan man ikke selv angive hvilke registre, der skal anvendes.
I forbindelse med shift-instruktionerne anvendes CL-registret til, at angive hvor mange "pladser" der skal shiftes. Det er heller ikke her muligt, at angive et andet register.
Registret kan ligesom AX benyttes som to 8-bits registre.
Registret kan anvendes som ekstra arbejdsregister (register-variabel).

DX **(Destination-register)**
Registret anvendes sammen med AX i forbindelse med multiplikation og division.
I forbindelse med en række funktionskald (Software-interrupt) anvendes registret sammen med DS (Datasegment) som startadresse (pointer) til memory-buffer.
Registret kan ligesom AX benyttes som to 8-bits registre.
Registret kan anvendes som ekstra arbejdsregister (register-variabel).

Adresse-/pointer-registre.

BP **(Base-pointer)**
Registret anvendes primært til memory-adressering på stakken (evt. sammen med SI- eller DI-registret). **Når BP-registret anvendes vil SS registret blive anvendt som segmentregister** (også samnen med SI/DI), hvis der ikke bliver angivet et segment-register, imodsætning til BX-, SI- , DI-registrene der anvender DS registret som segmentregister.
Registret kan **kun** benyttes som 16-bit register.
Registret kan anvendes som et ekstra arbejdsregister (register-variabel).

SI **(Source-index)** **(source = kilde/afsender)**
Registret anvendes primært til memory-adressering.
Registret anvendes i forbindelse med streng-instruktionerne sammen med DS-segmentregistret. I disse instruktioner kan man ikke selv angive hvilke registre, der skal anvendes.
Registret kan **kun** benyttes som 16-bit register.
Registret kan anvendes som et ekstra arbejdsregister (register-variabel).

DI **(Destination-index)** **(destination = modtager)**
Registret anvendes primært til memory-adressering.
Registret anvendes i forbindelse med streng-instruktionerne sammen med ES-segmentregistret. I disse instruktioner kan man ikke selv angive hvilke registre, der skal anvendes.
Registret kan **kun** benyttes som 16-bit register.
Registret kan anvendes som et ekstra arbejdsregister (register-variabel).

Specielle adresse-/pointer-registre

SP **(Stak-pointer)**
Registret anvendes sammen med SS-segmentregistret som pointer til staktoppen.
Registret rettes "automatisk" ved anvendes af instriktionerne PUSH, POP, CALL, RET, INT og IRET.
Registeret må ikke benyttes som arbejdsregister.

IP **(Instruktions-pointer)**
Registret anvendes til udpegning af instruktioner sammen med CS-segmentregistret.
Registret "flyttes" automatisk frem til næste instruktion, idet der kan foretages justereing med jump-instruktioner, call, ret m.fl.
Registret må ikke benyttes som arbejdsregister.

Segmentadresse-registre

CS **(Code-segment)**
Registret anvendes til udpegning af instruktioner sammen med IP-segmentregistret.

DS **(Data-segment)**
Registret anvendes til udpegning af data.

SS **(Stak-segment)**
Registret anvendes til udpegning af staktop sammen med SP-registret.

ES **(Ekstra-segment)**
Registret kan anvendes anvendes til udpegning af data, idet der så normalt anvendes segment-overwrite.
Registret anvendes sammen med DI-registeret i forbindelse med nogle af streng-instruktionerne.

Flag-register

F **(Flag)**
Registret inderholder dels resultats-flag og dels kontrol-flag.
Resultatsflagene sættes i forbindelse med udførslen af aritmetiske og logiske instruktioner. Flagene kan efterfølgende aftestes med betingede jumpinstruktioner. Carry-flaget kan evt. sættes on/off med STC og CLS.
Kontrolflagene sættes med specielle instruktioner og anvendes til styring af cpu-ens arbejdsmåde.
Der er f.eks. kontrolflag til at lukke og åbne for interruptsystemet.
I forbindelse med streng-instruktionerne findes et kontrolflag, som styrer den "retning" der arbejdes med (om adresser tælles op eller de tælles ned).