



IT Fundamentals

Toegepaste Informatica (Dig-X)

1^{ste} Bachelor

J. Ballet

1^{ste} semester

Design & Technologie

Academiejaar 2014-2015

Voorwoord

Deze cursus hoort bij het opleidingsonderdeel IT Fundamentals, dat aan de Erasmushogeschool Brussel gedoceerd wordt in het eerste jaar Bachelor in de Toegepaste Informatica. De cursus kan evenwel door iedereen gelezen worden die meer wil weten over de werking van een computersysteem. Deze cursus bestaat uit 3 delen:

- Deel 1: Getalrepresentaties, logische poorten en schakelingen: in dit eerste deel worden de verschillende talstelsels bekeken. Kennis van de bestaande talstelsels is enorm belangrijk om bepaalde begrippen in de informatica beter te kunnen begrijpen. In dit deel wordt verder ingegaan op de manier waarop getallen en ook letters in een computer worden opgeslagen. Naast deze basiskennis rond data-opslag worden ook de verschillende logische poorten en schakelingen besproken.
- Deel 2: Computerhardware: het tweede deel is een beschrijving van alle hardware die voorkomt op een computer. Van elke component wordt bekeken waarvoor deze dient, hoe deze is opgebouwd en wat de belangrijkste eigenschappen zijn.
- Deel 3: Software en inleiding tot besturingssystemen: het laatste deel van deze cursus behandelt software. Er wordt bekeken wat een hoge of lage programmeertaal is en hoe de processor omgaat met deze programmeertalen. Tenslotte wordt in dit deel een korte inleiding gegeven over besturingssystemen. Deze besturingssystemen zijn onontbeerlijk om een computer op een nuttige manier te kunnen gebruiken.

Voor de autodidact, die graag meer van computers wil weten en niet zozeer geïnteresseerd is in alle details, volstaan deel 1 en 2. Deel 3 neemt de lezer een stuk dieper mee in de werking van een computer: we duiken letterlijk onder motorkap van een processor om te bekijken wat er gebeurt bij het uitvoeren van een instructie.

Ondanks het feit dat deze cursus met de grootste zorg is samengesteld, kan het zijn dat er toch nog een foutje is ingeslopen. Indien u zo'n fout zou vinden, gelieve de auteur hiervan op de hoogte te brengen via bert.van.rillaer@ehb.be.

Bert Van Rillaer

Inhoudsopgave

I	Getalrepresentaties, logische poorten en schakelingen	1
1	Talstelsels en codesystemen	2
1.1	Talstelsels	2
1.1.1	Grondtallen	3
1.1.2	Het binaire talstelsel	3
1.1.3	Het hexadecimale talstelsel	3
1.1.4	Omzettingen tussen talstelsels	3
1.2	Voorstelling in de computer	7
1.2.1	Bits, bytes en negatieve getallen	7
1.2.2	Bereik	8
1.2.3	Kommagetallen	8
1.3	Bewerkingen op binaire getallen	12
1.3.1	Optellen	12
1.3.2	Aftrekken	12
1.3.3	Vermenigvuldigen	12
1.3.4	Delen	13
1.4	BCD-getallen	13
1.5	Symbolen	14
1.5.1	ASCII	14
1.5.2	EBCDIC	15
1.5.3	Unicode	15
2	Logische poorten en schakelingen	17
2.1	Logische poorten	17
2.1.1	AND- of EN-poort	17
2.1.2	OR- of OF-poort	18
2.1.3	NOT- of NIET-poort	19
2.2	Logische schakelingen	20
2.2.1	De NAND-poort	21
2.2.2	NOR-poort	21
2.2.3	XOR-poort	22
2.3	Rekenregels	22
2.3.1	Basisregels	22
2.3.2	Commutativiteit	23
2.3.3	Associativiteit	23
2.3.4	Distributiviteit	23

2.3.5 Absorptie	23
2.3.6 Regels van De Morgan	24
2.3.7 Computationeel compleetheid van NAND en NOR	24
3 Rekenschakelingen	26
3.1 Halve opteller (half adder)	26
3.2 Volledige opteller (full adder)	28
3.3 Intermezzo: maken van logische schakelingen	29
3.3.1 Omzetten naar formulevorm	30
3.3.2 Vereenvoudigen van de formule	30
3.3.3 Omzetten van de formule naar een logische schakeling	31
3.4 Parallel-opteller	31
3.5 Serie-opteller	33
3.6 Simultaan-opteller	33
3.7 Halve aftrekker (half subtractor)	35
3.8 Volledige aftrekker (full subtractor)	35
II Computerhardware	37
4 Computerarchitectuur	38
4.1 von Neumann architectuur	38
4.2 Processor	40
4.3 Werkgeheugen	40
4.4 Permanent geheugen	40
4.5 In- en uitvoerapparaten (randapparatuur)	40
4.6 Chipsets en bussen	40
5 Het werkgeheugen	41
5.1 Geheugenkarakteristieken	42
5.1.1 Geheugencapaciteit	42
5.1.2 Informatiedichtheid	42
5.1.3 Toegankelijkheid	42
5.1.4 Adresseerbaarheid	43
5.1.5 Bestendige en vluchtlige geheugens	43
5.2 Cache-geheugens	43
5.2.1 Wat is cache-geheugen?	43
5.2.2 Locality of reference	44
5.2.3 Level 1 cache geheugen	44
5.2.4 Level 2 cache geheugen	45
5.3 Werkgeheugen en soorten geheugens	45
5.3.1 Wat is werkgeheugen?	45
5.3.2 Statisch RAM	45
5.3.3 Dynamisch RAM	47
5.4 Soorten DRAM geheugens	47
5.4.1 Vergelijking SRAM-DRAM	51
5.4.2 De geheugenpiramide	51

5.4.3	Geheugenmodules	52
5.5	Nieuwigheden en speciale RAMs	53
5.5.1	Nieuwigheden	54
5.5.2	Speciale RAMs	54
5.6	Detectie en foutencontrole	55
5.6.1	Detectie	55
5.6.2	Foutencontrole	56
6	Processoren	57
6.1	Geschiedenis	58
6.1.1	Eerste processoren	58
6.1.2	32 vs 64 bits?	62
6.1.3	Dual-core en quad-core	63
6.2	Bouw van een processor	63
6.2.1	Onderdelen	63
6.2.2	Transistor density	64
6.2.3	Koeling	64
6.2.4	Van 10 µm naar 11nm technologie	65
6.3	Werking	66
6.3.1	Processorcyclus	66
6.3.2	Kloksnelheden	66
6.3.3	Buffers en cachegeheugen	68
6.3.4	Toekomstvoorspelling	68
6.3.5	Instruction sets	68
6.4	Registers	69
7	Chipsets en bussen	70
7.1	Bussen	70
7.1.1	De architectuur	70
7.1.2	Een geïntegreerde memory controller	71
7.1.3	De PCI-bus	72
7.1.4	USB (Universal Serial Bus)	74
7.1.5	AGP (Accelerated Graphics Port)	74
7.1.6	ATA	76
7.2	Chipsets	76
7.2.1	Processor-families	77
7.2.2	DMA (Direct Memory Access)	77
8	Permanente magnetische gegevensopslag	80
8.1	Algemene principes van magnetische geheugens	81
8.1.1	Fysische eigenschappen	81
8.1.2	Dichtheid van (magnetische) materialen	83
8.1.3	Lees- en schrijfkop bij magnetische geheugens	84
8.1.4	Indeling van magnetische schijven	85
8.2	Magnetische tapes (magneetbanden)	87
8.2.1	Geschiedenis	87
8.2.2	Organisatie van gegevens op magneetband	88

8.2.3	Tape drives	89
8.2.4	Tapes als backup	89
8.3	Floppy disks (diskettes)	90
8.3.1	Geschiedenis	90
8.3.2	De $5\frac{1}{4}$ inch floppy disk	90
8.3.3	De $3\frac{1}{2}$ inch floppy disk	90
8.3.4	Floppy disk drive	92
8.3.5	Lees- en schrijfkop	93
8.3.6	Toekomst	93
8.4	Harde schijven	93
8.4.1	Geschiedenis	93
8.4.2	Opbouw van een moderne harde schijf	94
8.4.3	Opbouw van de magnetische schijven	95
8.4.4	Luchtcirculatie en koeling	98
8.4.5	Snelheid van harde schijven	100
8.4.6	Evolutie	101
8.5	Geperfectioneerde gegevensopslag: RAID	101
8.5.1	Inleiding	101
8.5.2	RAID concepten	103
8.5.3	RAID niveau's	105
8.5.4	RAID controllers	109
8.5.5	Praktijk	110
9	Optische geheugens en “leesgeheugens”	111
9.1	Optische geheugens	111
9.1.1	CD-ROM	111
9.1.2	CD-R en CD-RW	113
9.1.3	De DVD-RAM	115
9.1.4	DVD-ROM	115
9.1.5	DVD±R en DVD±RW	116
9.1.6	Blu-ray	117
9.2	“Leesgeheugens”	118
9.2.1	Read-only Memory(ROM)	118
9.2.2	Programmable Read-only Memory(PROM)	118
9.2.3	Erasable Programmable Read-only Memory(EPROM)	118
9.2.4	Electronically Erasable Programmable ROM(EEPROM)	118
9.2.5	Flashgeheugens	119
9.2.6	Solid state drives (SSD)	121
10	Poorten	123
10.1	Parallelle vs. seriële communicatie	124
10.2	De parallelle poort	124
10.2.1	Aansluitingen en gebruik	124
10.2.2	Standaardisatie en modes	125
10.3	De seriële poort	125
10.3.1	Aansluitingen en gebruik	125
10.3.2	Asynchrone vs. synchrone communicatie	127

10.4 PS/2 poort	127
10.5 De USB poort	128
10.5.1 Ontwerp	128
10.5.2 Standaardisatie	129
10.5.3 Host controller	130
10.5.4 Apparaatklassen	131
10.5.5 Stroomvoorziening	131
10.5.6 USB 3.0	131
10.6 FireWire (IEEE 1394)	131
10.6.1 Ontwikkeling	132
10.6.2 Werking	132
10.6.3 Standaarden	132
10.6.4 Hot swap gevaa...	133
10.7 eSATA	133
10.8 Bluetooth	134
10.9 VGA-poort	134
11 Randapparatuur	136
11.1 Toetsenbord	136
11.1.1 Geschiedenis	136
11.1.2 Opbouw	137
11.1.3 De microcontroller	138
11.1.4 Interrupts	138
11.2 Muis	140
11.2.1 Geschiedenis	140
11.2.2 Klassieke muis: opbouw en werking	140
11.2.3 Optische muis: opbouw en werking	141
11.2.4 RSI: Repetitive Strain Injury	142
11.3 Scherm	142
11.3.1 Eigenschappen van schermen	143
11.3.2 Geschiedenis	143
11.3.3 Werking van CRT schermen	144
11.3.4 Werking van TFT schermen	144
11.4 Geluid	145
11.5 Netwerken en modems	145
11.6 Scanner	146
11.7 Printer	146
11.8 Stroomvoorziening	147
III Software en inleiding tot besturingssystemen	148
12 Software	149
12.1 Wat is software?	149
12.1.1 Programma's, instructiesets, machinecode	149
12.1.2 Assembler	150
12.1.3 Hogere programmeertalen	150

12.2 Registers	152
12.2.1 Breedte van registers	152
12.2.2 Algemene registers	153
12.2.3 Segmentregisters	154
12.2.4 Speciale registers	155
12.3 Geheugensegmentatie	156
12.3.1 Geheugenmodellen	156
12.3.2 Meer over de werking van enkele registers	156
12.4 In- en uitvoer	158
12.4.1 Interrupts	158
12.4.2 Exceptions (uitzonderingen)	160
12.5 RISC of CISC?	160
12.5.1 RISC-processoren	161
12.5.2 CISC-processoren	161
12.5.3 RISC vs. CISC	161
12.5.4 Het post-RISC tijdperk	162
13 Besturingssystemen	163
13.1 Doelen en functies van besturingssystemen	164
13.1.1 Geschiedenis	164
13.1.2 Functies	165
13.2 Process management	166
13.2.1 Processen	166
13.2.2 Procestabel	167
13.2.3 Multiprogramming	168
13.2.4 Interproces communicatie	168
13.2.5 Scheduling	169
13.2.6 Enkele scheduling algoritmen	170
13.2.7 Deadlocks	171
13.3 Memory management	173
13.3.1 Swapping	173
13.3.2 Paging	174
13.3.3 Virtueel geheugen	175
13.4 File management	175
13.4.1 Blokgrootte	175
13.4.2 Allocatie	175
13.4.3 Beheer van vrije ruimte	176
13.4.4 Veiligheid	176

Deel I

Getalrepresentaties, logische poorten en schakelingen

Hoofdstuk 1

Talstelsels en codesystemen

Inhoudsopgave

1.1 Talstelsels	2
1.1.1 Grondtallen	3
1.1.2 Het binaire talstelsel	3
1.1.3 Het hexadecimale talstelsel	3
1.1.4 Omzettingen tussen talstelsels	3
1.2 Voorstelling in de computer	7
1.2.1 Bits, bytes en negatieve getallen	7
1.2.2 Bereik	8
1.2.3 Kommagetallen	8
1.3 Bewerkingen op binaire getallen	12
1.3.1 Optellen	12
1.3.2 Aftrekken	12
1.3.3 Vermenigvuldigen	12
1.3.4 Delen	13
1.4 BCD-getallen	13
1.5 Symbolen	14
1.5.1 ASCII	14
1.5.2 EBCDIC	15
1.5.3 Unicode	15

1.1 Talstelsels

Een computer werkt met enorm veel getallen. Op een of andere manier moeten we deze getallen in een computer kunnen voorstellen. Een computer werkt namelijk met 0-en en 1-en, in het binaire talstelsel.

1.1.1 Grondtallen

Wanneer we rekenen in een bepaald talstelsel, werken we steeds met een bepaald grondtal. In het decimale talstelsel is dat grondtal gelijk aan 10. Dit wil zeggen dat een getal in het decimale talstelsel opgebouwd is uit de machten van 10: 1 (10^0), 10 (10^1), 100 (10^2), 1000 (10^3), ... De cijfers van een talstelstel blijven altijd kleiner dan het grondtal zelf: 0,1,2,3,4,5,6,7,8,9. De waarde van een getal in het decimale talstelsel wordt dan als volgt geschreven:

$$51326 = (5 \times 10^4) + (1 \times 10^3) + (3 \times 10^2) + (2 \times 10^1) + (6 \times 10^0)$$

1.1.2 Het binaire talstelsel

In het binaire of tweetallige talstelsel werken we met grondtal 2, dit wil zeggen dat er twee cijfers zijn: 0 en 1. De machten van 2 zijn hier achtereenvolgens: 1,2,4,8,16,32,... We zouden in feite kunnen spreken over 2-tallen in plaats van 10-tallen en 4-tallen in plaats van 100-tallen. In het binaire talstelsel wordt een getal dus als volgt geschreven:

$$\begin{aligned} 1001 &= (1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\ &= 9_{dec} \end{aligned}$$

Tellen in het binaire talstelsel gaat dan zo: 0,1,10,11,100,101,110,111,1000,1001,1010,1011,...

1.1.3 Het hexadecimale talstelsel

Het hexadecimale talstelsel heeft als grondtal 16. Dit wil zeggen dat we in feite 16 verschillende cijfers moeten hebben in dit talstelsel. Daarom telt men bij dit talstelsel gewoon voort met letters. De opeenvolgende cijfers van het hexadecimale talstelsel zijn:

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F waarbij A tot en met F eigenlijk 10 tot 15 voorstellen.

Om verwarring te vermijden, geven we steeds aan in welk talstelsel we werken. Bijvoorbeeld: 10101_{bin} , $14A6_{hex}$, 16543_{dec} .

Ook een hexadecimaal getal kan nu weer worden uitgeschreven in termen van het grondtal:

$$\begin{aligned} 3A7C_{hex} &= (3 \times 16^3) + (A \times 16^2) + (7 \times 16^1) + (C \times 16^0) \\ &= (3 \times 4096) + (10 \times 256) + (7 \times 16) + (12 \times 1) \\ &= 14972_{dec} \end{aligned}$$

Hexadecimale getallen worden dikwijls gebruikt om een binair getal verkort weer te geven. Denk bijvoorbeeld aan MAC-adressen. De hexadecimale schrijfwijze is veel makkelijker te lezen door mensen en bovendien gemakkelijk om te zetten naar de binaire tegenhanger (zie verder). Tabel 1.1 geeft een idee van de overeenkomst tussen de decimale, hexadecimale en binaire schrijfwijze. In de toekomst zullen we nog veel geconfronteerd worden met hexadecimale getallen, aangezien IPv6 (de opvolger van de huidige IP-adresseringswijze) voorschrijft om de adressen in hexadecimale vorm te noteren.

1.1.4 Omzettingen tussen talstelsels

We zullen nu een aantal omzettingen tussen de verschillende talstelsels nader bekijken.

Decimaal	Binair	Hexadecimaal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Tabel 1.1: Overeenkomstige getallen in verschillende talstelsels

Binair - Decimaal

De omzetting van een binair getal naar de decimale voorstelling gebeurt door het optellen van de r -de machten van 2, waar een 1 staat in de binaire voorstelling:

$$\begin{aligned}
 1001010 &= (1 \times 2^6) + (0 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) \\
 &= 2^6 + 0 + 0 + 2^3 + 0 + 2^1 + 0 \\
 &= 64 + 0 + 0 + 8 + 0 + 2 + 0 \\
 &= 74
 \end{aligned}$$

De omzetting van decimaal naar binair is iets lastiger. We volgen hiervoor deze methode:

- Zoek de hoogste macht van 2 (1,2,4,8,16,32,64,...) die in het getal past
- Trek dit getal af van het om te rekenen getal. Schrijf een 1 helemaal links in de binaire schrijfwijze.
- Overloop vanaf de hierboven gevonden macht alle kleinere machten van 2 af, indien een macht in het restgetal past, trek het hiervan af en schrijf een 1 in de binaire schrijfwijze. Indien een macht niet in het getal past, schrijf een 0 in de binaire schrijfwijze. Stop pas als je aan de kleinste macht van 2 komt: 1!

Voorbeeld:

Omzetting van 165_{dec} naar binair: 128 is de grootste macht van 2 die in het getal geraakt. We schrijven een 1 in de binaire voorstelling en trekken het van 165 af:

Binaire voorstelling: 1... Restgetal: 37

De volgende macht van 2 is 64. Geraakt niet in het restgetal dus schrijven we een 0:

Binaire voorstelling: 10... Restgetal: 37

De volgende macht van 2 is 32. Geraakt in het restgetal dus:

Binaire voorstelling: 101... Restgetal: 5

De volgende macht van 2 is 16. Geraakt niet in het restgetal dus:

Binaire voorstelling: 1010... Restgetal: 5

De volgende macht van 2 is 8. Geraakt niet in het restgetal dus:

Binaire voorstelling: 10100... Restgetal: 5

De volgende macht van 2 is 4. Geraakt in het restgetal dus:

Binaire voorstelling: 101001... Restgetal: 1

De volgende macht van 2 is 2. Geraakt niet in het restgetal dus:

Binaire voorstelling: 1010010... Restgetal: 1

De volgende macht van 2 is 1. Geraakt in het restgetal dus:

Binaire voorstelling: 10100101 Restgetal: 0

We mogen nu stoppen aangezien we aanbeland zijn bij de kleinste macht van 2. We kunnen dus stellen dat $165_{dec} = 10100101_{bin}$.

Hexadecimaal - Decimaal

De omzetting van hexadecimaal naar decimaal werd reeds eerder vermeld (zie 1.1.3). De omzetting van decimaal naar hexadecimaal is iets lastiger. Hiervoor gebruiken we een omzettingstabell (tabel 1.2).

- Neem het grootste getal uit de tabel dat in het decimale getal past. Trek dit af van het oorspronkelijke getal en schrijf het overeenkomstige hexadecimale cijfer op.
- Zoek nu in de volgende (naar rechts toe) kolom en neem het grootste getal dat nog in het restgetal past. Trek dit weer af van het restgetal, en schrijf het overeenkomstige cijfer in de hexadecimale schrijfwijze. Blijf dit herhalen tot de laatste kolom.

Voorbeeld: Voor het omzetten van het decimale getal 125483 naar het hexadecimale talstelsel maken we gebruik van bovenstaande redenering. In kolom 5 zien we dat 65536 het grootste getal is dat nog in het oorspronkelijke getal past. We trekken het dus af van 125483 en schrijven een 1 in de hexadecimale schrijfwijze. In kolom 4 zien we dat 57344 het grootste getal is dat in het restgetal 59947 (125483-65536) past. We schrijven dus een E en gaan zo door... We bekomen uiteindelijk 1EA2B. Ga dit zelf na! Tabel 1.3 toont de geselecteerde getallen (vet gedrukt).

Hexadecimaal - Binair

Tenslotte komen we tot een van de meest eenvoudige omzettingen: binair naar hexadecimaal en omgekeerd. Het feit dat deze omzettingen betrekkelijk eenvoudig verlopen ligt aan het

8	7	6	5	4	3	2	1
0	0	0	0	0	0	0	0
1	268435456	1	16777216	1	1048576	1	65536
2	536870912	2	33554432	2	2097152	2	131072
3	805306368	3	50331648	3	3145728	3	196608
4	1073741824	4	67108864	4	4194304	4	262144
5	1342177280	5	83886080	5	5242880	5	327680
6	1610612736	6	100663296	6	6291456	6	393216
7	1879048192	7	117440512	7	7340032	7	458752
8	2147483648	8	134217728	8	8388608	8	524288
9	2415919104	9	150994944	9	9437184	9	589824
A	2684354560	A	167772160	A	10485760	A	655360
B	2952790016	B	184549376	B	11534336	B	720896
C	3221225472	C	201326592	C	12582912	C	786432
D	3489660928	D	218103808	D	13631488	D	851968
E	3758096384	E	234881024	E	14680064	E	917504
F	4026531840	F	251658240	F	15728640	F	983040

Tabel 1.2: Omzettingstabel decimaal - hexadecimaal

8	7	6	5	4	3	2	1
0	0	0	0	0	0	0	0
1	268435456	1	16777216	1	1048576	1	65536
2	536870912	2	33554432	2	2097152	2	131072
3	805306368	3	50331648	3	3145728	3	196608
4	1073741824	4	67108864	4	4194304	4	262144
5	1342177280	5	83886080	5	5242880	5	327680
6	1610612736	6	100663296	6	6291456	6	393216
7	1879048192	7	117440512	7	7340032	7	458752
8	2147483648	8	134217728	8	8388608	8	524288
9	2415919104	9	150994944	9	9437184	9	589824
A	2684354560	A	167772160	A	10485760	A	655360
B	2952790016	B	184549376	B	11534336	B	720896
C	3221225472	C	201326592	C	12582912	C	786432
D	3489660928	D	218103808	D	13631488	D	851968
E	3758096384	E	234881024	E	14680064	E	917504
F	4026531840	F	251658240	F	15728640	F	983040

Tabel 1.3: Voorbeeld omzetting decimaal - hexadecimaal

grondtal: 2 voor binair en 16 voor hexadecimaal. Aangezien $2^4 = 16$ kunnen we hieruit afleiden dat elk cijfer uit het hexadecimale talstelsel zal worden voorgesteld door 4 bits. Zo ook werkt de omzetting:

- Zet het meest rechts cijfer uit het hexadecimale getal om naar binaire voorstelling. Dit zijn de 4 meest rechtse bits van de binaire voorstelling.
- Zet het volgende cijfer uit het hexadecimale getal om naar binaire voorstelling. Dit zijn de volgende 4 bits van de binaire voorstelling.
- Herhaal de vorige stap tot alle cijfers zijn omgezet.

Voorbeeld:

Omzetting van $A2B_{hex}$ naar het binaire talstelsel. We zetten B om naar het binaire talstelsel, dit geeft ons 1011 (zie tabel 1.1). Het binaire getal wordt dus ...1011 en we moeten A2 nog omzetten. 2 binair geeft 0010 dus verkrijgen we ...00101011 en A nog om te zetten. A in het binaire stelsel is gelijk aan 1010 dus is het uiteindelijke binaire getal 101000101011_{bin} . De omzetting is voltooid.

De omzetting van binair naar hexadecimaal verloopt analoog:

- Eerst worden de 4 meest rechtse bits omgezet naar een hexadecimaal cijfer. Dit is het meest rechtse cijfer van het hexadecimale getal.
- De volgende 4 bits worden omgezet naar een hexadecimaal cijfer. Herhaal dit totdat alle bits op zijn.

Volgende voorstelling maakt de omzetting nog duidelijker:

$\underbrace{1010}_{A} \underbrace{0010}_{2} \underbrace{1011}_{B}$

De wiskundige relatie tussen deze twee talstelsels impliceert ook dat elk getal in hexadecimale voorstelling 4 keer minder cijfers nodig heeft dan in binaire voorstelling. Dit geeft meteen de kracht weer van hexadecimale voorstellingen. Het typische voorbeeld is een MAC adres van een netwerkkaart. Een MAC adres bestaat uit 48 bits, maar in hexadecimale schrijfwijze hebben we slechts 12 cijfers nodig om het te kunnen voorstellen. Dit is dus veel eenvoudiger om te gebruiken (en bv. telefonisch door te geven).

1.2 Voorstelling in de computer

1.2.1 Bits, bytes en negatieve getallen

In een computer is de adresseerbare eenheid meestal niet een individuele bit, maar een reeks van bits. We spreken in dit verband van een nibble (4 bits), een byte (8 bits), een woord (16 bits) en een dubbelwoord (32 bits). Meestal worden ook negatieve getallen voorgesteld in een computer, dus moeten de beschikbare bits verdeeld worden tussen de negatieve en positieve getallen (en 0). De linkerbit doet dan dienst als tekenbit. Indien de meest linkerbit een 1 is, spreken we over een negatief getal.

Voor negatieve getallen wordt meestal de two's complement voorstelling gebruikt. Deze 2's complement werkt volgens het volgende principe:

- Als de tekenbit 0 is, rekenen we het binaire getal gewoon om naar decimaal om de waarde ervan te kennen.
- Als de tekenbit 1 is, hebben we echter te maken met een negatief getal. In dat geval volgen we de volgende procedure om te weten welk getal er wordt voorgesteld:
 - Alle bits inverteren (0 wordt 1, 1 wordt 0)
 - 1 optellen bij het binaire getal

- Omzetten naar decimale talstelsel
- Het gevonden getal is de overeenkomende positieve waarde van het getal (dus nu nog een minteken ervoor zetten).

We bespreken nu enkele voorbeelden om het geheel te verduidelijken.

Voorbeeld: 1000 0101 is een byte in 2's complement. Indien we willen weten welk getal hier wordt voorgesteld, inverteren we eerst alle bits: 0111 1010. We tellen er nu 1 bij op: 0111 1011. Dit zetten we om naar decimale voorstelling: 123. Het voorgestelde getal is dus -123.

Voorbeeld: 0110 1010 is een byte in 2's complement. Hier is de eerste bit 0 dus hebben we een positief getal. Gewoon omzetten naar decimaal dus, dit geeft ons 106.

Waarom zouden we 2's complement voorstelling gebruiken? Het antwoord is eenvoudig: computers kunnen heel eenvoudig rekenen met 2's complement notatie. Het aftrekken van twee binaire getallen komt nu overeen met het optellen van het eerste getal met de negatieve voorstelling van het tweede getal.

1.2.2 Bereik

We kunnen nu spreken over een bereik van een binaire voorstelling: dit is het aantal mogelijke getallen die kunnen worden voorgesteld, gegeven een aantal bits voor deze voorstelling.

Voor een **byte**: grootste positief getal: 127 (0111 1111)
grootste negatief getal: -128 (1000 0000)

Voor een **woord**: grootste positief getal: 32767 (0111 1111 1111 1111)
grootste negatief getal: -32768 (1000 0000 0000 0000)

Voor een **dubbelwoord**: grootste positief getal: 2147483647
grootste negatief getal: -2147483648

1.2.3 Kommagetallen

Floating point en IEEE

Ook niet-gehele getallen moeten worden voorgesteld, die in computertermen ook wel floating point getallen worden genoemd. Hiervoor wordt meestal de notatie gebruikt, die door het IEEE¹ werd vastgelegd. Vóór 1985 gebruikten computerfabrikanten een eigen voorstelling van komaaggetallen in de computer. Vanaf de introductie van de IEEE standaard schakelden de fabrikanten stapsgewijs over naar deze gestandaardiseerde manier van getalrepresentatie. Dit

¹IEEE staat voor Institute of Electrical and Electronics Engineers en is een non-profit organisatie die meer dan 375000 leden (waarvan 60000 studenten) over 160 landen groepeert. De voornaamste activiteiten van IEEE zijn: het publiceren van een aantal journals over elektrotechnisch, computer en controle technologie, het organiseren van meer dan 850 conferenties en het definiëren van standaarden (tot nu toe bestaan er ongeveer 1000 standaarden en zijn er 1300 in de maak).

heeft als voordeel dat bij het uitwisselen van data er geen conversies hoeven te gebeuren. Bij de IEEE notatie wordt vertrokken van de wetenschappelijke notatie van een kommagetal:

$$\text{getal} = \text{mantisse} \times \text{grondtal}^{\text{exponent}} \quad (1.1)$$

of met afkortingen:

$$\text{getal} = m \times g^e \quad (1.2)$$

Voorbeeld:

$$5 = 1.01000_{\text{bin}} \times 2^2$$

$$2 = 1.00000_{\text{bin}} \times 2^1$$

$$-1 = -1.00000_{\text{bin}} \times 2^0$$

We moeten voor elk getal dus drie zaken bijhouden:

1. De tekenbit (negatief/positief)
2. De mantisse
3. De exponent

Hierbij werken we steeds met genormaliseerde weergave:

$$\frac{1}{g} \leq \|m\| < 1 \text{ of } m = 0 \quad (1.3)$$

De genormaliseerde weergaveformule, toegepast op het binaire talstel geeft ons:

$$\frac{1}{2} \leq \|m\| < 1 \text{ of } m = 0 \quad (1.4)$$

Dit wil zeggen dat de mantisse groter of gelijk aan 1/2 is en kleiner dan 1 is. In binaire vorm vertaalt dit zich in het feit dat de mantisse groter of gelijk aan 0.1 is en kleiner is dan 1.0. We kunnen dus besluiten dat de mantisse in binaire vorm altijd begint met 0.1. Let op: een kommagetal in binaire weergave is niet helemaal hetzelfde als wat je verwacht:

$$\begin{aligned} 1.101 &= 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ &= 1 \times 1 + 1 \times \frac{1}{2} + 0 \times \frac{1}{4} + 1 \times \frac{1}{8} \\ &= 1.625 \end{aligned}$$

IEEE legt vast hoeveel bits worden gebruikt voor de mantisse en de exponent en de manier waarop de tekenbit wordt opgeslagen (het grondtal is altijd 2 aangezien we in het binaire talstelsel werken). IEEE voorziet in twee belangrijke standaarden: single precision en double precision. De single precision standaard gebruikt 32 bits, waarbij 1 tekenbit, 8 bits voor de exponent en 23 bits voor de mantisse. Een single precision getal ziet er dus zo uit:

TEEEEEEEEMMMMMMMMMMMMMMMMMMMMM

0 1 8 9

31

Een getal in double precision is opgebouwd uit 64 bits: 1 tekenbit, 11 exponentbits en 52 mantissebits. Twee belangrijke opmerkingen horen op zijn plaats bij deze twee standaarden:

1. De exponent wordt niet gewoon geschreven: de som van de exponent met 127 wordt genoteerd bij single precision, de som van de exponent met 1023 bij double precision.
2. Van de mantisse wordt het eerste cijfer (meest linkse) niet opgeslagen, dit is immers altijd 1

De tweede opmerking werd reeds hierboven uitgelegd (gelijkheid 1.4).

Voorbeeld:

In single precision:

0 0000000 00000000000000000000000000000000 = 0
1 0000000 00000000000000000000000000000000 = -0
0 1111111 00000000000000000000000000000000 = ∞
1 1111111 00000000000000000000000000000000 = $-\infty$
0 1000000 00000000000000000000000000000000 = $+1.0 \times 2^{128-127} = 2_{dec}$
0 1000001 10100000000000000000000000000000 = $+1.101 \times 2^{129-127} = 6.5_{dec}$
1 1000001 10100000000000000000000000000000 = $-1.101 \times 2^{129-127} = -6.5_{dec}$
0 0000001 00000000000000000000000000000000 = $+1.0 \times 2^{1-127} = 2^{-127}_{dec}$

Omrekening decimaal - floating point

Om een decimaal getal om te zetten naar de floating point notatie (single precision):

1. Getal omzetten naar binair
2. Getal normaliseren (komma verplaatsen en exponent aanpassen)
3. Getal na de komma = mantisse
4. Exponent = macht + 127
5. Tekenbit zetten

Voorbeeld:

+5.000

1. $5.000_{dec} = 101_{bin}$
2. $101_{bin} = 1.01_{bin} \times 2^2$ (let op: werken met machten van 2!)
3. Mantisse = 01
4. Exponent = $2 + 127 = 129 = 10000001_{bin}$
5. Tekenbit op 0 zetten

+5.000 wordt dus voorgesteld als:

0 100 0000 1 010 0000 0000 0000 0000

Voorbeeld:

-6.25

1. $6.25_{dec} = 110.01_{bin}$
2. $110.01_{bin} = 1.1001_{bin} \times 2^2$
3. Mantisse = 1001
4. Exponent = $2 + 127 = 129 = 10000001_{bin}$
5. Tekenbit op 1 zetten

-6.25 wordt dus voorgesteld als:

1	100	0000	1	100	1000	0000	0000	0000	0000
---	-----	------	---	-----	------	------	------	------	------

De omgekeerde omzetting (van floating point notatie naar decimaal) gaat als volgt:

1. Indien in hexadecimale weergave: zet om naar binair
2. Verdeel de bits in teken-, mantisse- en exponentbits
3. Zet 1. voor de mantisse
4. Zet de exponent om naar decimaal, trek er 127 vanaf
5. Bereken in het binair (gewoon komma verschuiven) het getal: $m \times 2^e$ waarbij e de exponent voorstelt (zie vorige stap)
6. Zet het getal om naar decimaal
7. Voeg het teken toe

Voorbeeld: $C0C80000_{hex}$

1. $C0C80000_{hex} = 11000000110010000000000000000000_{bin}$
2. Tekenbit: 1, Exponent: 100 0000 1, Mantisse: 1001000 0000 0000 0000
3. Mantisse: 1.1001_{bin}
4. Exponent: $10000001_{bin} = 129_{dec} \Rightarrow 129 - 127 = 2$
5. Getal: $1.1001 \times 2^2 = 110.01_{bin}$
6. Omzetten: $110.01_{bin} = 6.25_{dec}$
7. Tekenbit staat op 1 dus: -6.25

1.3 Bewerkingen op binaire getallen

Alle bewerkingen die we in het decimale talstelsel kunnen uitvoeren, zijn ook mogelijk in het binaire talstelsel. De berekeningen verlopen analoog. Aangezien we nu werken met grondtal 2, zal $1+1=0$ met rest 1. Dit is het analoge van $9+1$ in het decimale talstelsel.

1.3.1 Optellen

Voorbeeld:

$$\begin{array}{r} 10110 \\ + 101 \\ \hline 11011 \end{array}$$

1.3.2 Aftrekken

Wanneer je 0-1 moet berekenen, zitten we met het analoge geval van bijvoorbeeld 5-9 in het decimale talstelsel. In het binaire talstelsel gaan we niet 10 lenen, maar 1. Ga dus in de volgende 2-tallen een 1 lenen, dit wordt 10-1 wat ons 1 geeft.

Voorbeeld:

$$\begin{array}{r} 10110 \\ - 0101 \\ \hline 10001 \end{array}$$

Een andere manier om deze optelling te maken is 5 (0000 0101) als -5 (1111 1011) in 2's complement notatie te schrijven. In dat geval kan je beide getallen gewoon optellen.

1.3.3 Vermenigvuldigen

Bij de vermenigvuldiging werken we weer analoog als in het decimale talstelsel. Een voorbeeld wordt getoond in figuur 1.1. Bij het vermenigvuldigen met een macht van 2 hebben we een

$$\begin{array}{r} 10110 \\ 1101 \\ \times \quad \quad \quad \hline 10110 \\ 00000 \\ 10110 \\ 10110 \\ + \quad \quad \quad \hline 100011110 \end{array}$$

Figuur 1.1: Voorbeeld: 10110×1101

speciaal geval. Hier kunnen we de bits naar links verschuiven. Aan de rechterkant vullen we aan met nullen: $11011 \times 100 = 1101100$.

1.3.4 Delen

Ook het delen gebeurt weer op dezelfde manier als in het decimale talstelsel. Een voorbeeld wordt getoond in figuur 1.2. Bij delen door een macht van 2 verschuiven we de bits door naar rechts.

$$\begin{array}{r|l}
 11110011 & 1001 \\
 -1001 & \\
 \hline
 1100 & \\
 -1001 & \\
 \hline
 001101 & \\
 -1001 & \\
 \hline
 1001 & \\
 -1001 & \\
 \hline
 0000 &
 \end{array}$$

Figuur 1.2: Voorbeeld: 11110011 : 1001

1.4 BCD-getallen

Omdat het decimale stelsel zoveel wordt gebruikt, bestaat er een binaire representatie van het decimale stelsel: de BCD of Binary Coded Decimal. bij deze representatie schrijven we elk decimaal cijfer in zijn binaire code. Aangezien er 10 cijfers zijn in het decimale talstelsel, hebben we dus 4 bits nodig om alle mogelijkheden voor te stellen. We gebruiken de codering zoals weergegeven in tabel 1.4. Wanneer we getallen met 2 cijfers willen voorstellen, gebruiken we 8 bits. Bij getallen met 3 cijfers hebben we 12 bits nodig, enzovoort.

Voorbeeld:

1001	0010	1000
9	2	8

Zoals je zelf wel kan merken, hebben we hier veel meer bits nodig dan wanneer we de gewone representatie zouden gebruiken. Hiertegenover staat het voordeel dat er een absolute nauwkeurigheid wordt bekomen: de omzetting tussen decimaal en binair heeft geen verlies. Vele decimale getallen kunnen namelijk niet precies omgezet worden naar binair. In dat geval wordt het dichtste voorstellbare getal genomen, maar er blijft een kleine fout ten opzichte van de echte waarde... (probeer bijvoorbeeld het decimale getal 0.6 maar eens om te zetten naar zijn binaire voorstelling) Moderne processoren kunnen meestal rechtstreeks bewerkingen uitvoeren op BCD-getallen. Hiervoor worden speciale tellers (zie later) gebruikt, die per 4 bits kunnen optellen en eventuele overdracht kunnen waarmaken. Tenslotte kan een BCD-getal zo groot gemaakt worden als nodig: gewoon 4 bits toevoegen en je kan een extra cijfer voorstellen (het bereik is dus in theorie onbeperkt). Een programmeertaal zoals COBOL biedt rechtstreekse ondersteuning voor het werken met BCD-getallen.

BCD	Decimaal cijfer
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	-
1011	-
1100	-
1101	-
1111	-

Tabel 1.4: Overeenkomsten in de BCD-voorstelling

Let op bij bewerkingen op BCD-getallen! We moeten er namelijk voor zorgen dat geen enkel cijfer gelijk wordt aan 1010 tot en met 1111. Dit zou overeenkomen met de decimale **cijfers** 10,11,12,13,14,15 die helemaal niet bestaan!²

Voorbeeld:

$$\begin{aligned} & 0001 \ 0001 \ 1000 \ 0001 + 0001 \ 0100 \ 0010 \ 0000 \\ & = 0010 \ 0101 \ \mathbf{1010} \ 0001 \end{aligned}$$

We moeten alle cijfers groter dan 1001 wegwerken. We trekken van deze cijfers 1010 af en tellen 1 op bij het volgende cijfer (overdracht naar de volgende macht van 10):

$$\begin{aligned} & = 0010 \ (0101 + \mathbf{0001}) \ (1010 - \mathbf{1010}) \ 0001 \\ & = 0010 \ 0110 \ 0000 \ 0001 \end{aligned}$$

1.5 Symbolen

Tot nu toe hebben we het enkel gehad over het voorstellen van getallen. Maar ook letters, cijfers en tekens moeten kunnen worden voorgesteld. Denk maar aan een tekstbestand, waarin enkel letter voorkomen. Voor het opslaan van letters, cijfers en tekens maakt men gebruik van algemene afspraken. Deze worden vastgelegd in een standaard.

1.5.1 ASCII

De eerste standaard voor het voorstellen van tekens werd vastgelegd in ASCII (American Standard Code for Information Interchange). Hierbij worden 7 bits gebruikt voor het voorstellen van tekens, de ASCII-tabel wordt weergegeven in figuur 1.3. Merk op dat de codes voor

²Merk op dat de **getallen** 10,11,12,13,14,15 wel bestaan in het decimale talstelsel, maar niet de **cijfers** 10,11,12,13,14,15. Aangezien elk BCD-cijfer overeenkomt met een cijfer in het decimale talstelsel, moeten we dus zorgen dat we enkel volgende waarden krijgen: 0000 (voorstelling van cijfer 0) tot en met 1001 (voorstelling van cijfer 9).

de cijfers niet overeenkomen met de binaire voorstellingen van de cijfers. Zo is de ASCII-code van 5 gelijk aan 53(0110101), terwijl de binaire weergave van het getal vijf gelijk is aan 101. Met 7 bits kan men 128 tekens voorstellen, maar het werd al snel duidelijk dat er meer tekens nodig waren. In een latere versie werd 1 bit toegevoegd, waardoor er extra tekens konden worden voorgesteld (256). Dit noemt men de extended ASCII-tabel. Er zijn een aantal speciale karakters: linefeed (10), carriage return (13), backspace (8),... De linefeed en carriage return stammen uit de tijd van de oude typemachine.

000	<nul>	016	► (dle)	032	sp	048	Ø	064	¤	080	P	096	'	112	p		
001	@ (soh)	017	◄ (dc1)	033	!	049	1	065	A	081	Q	097	a	113	q		
002	§ (stx)	018	‡ (dc2)	034	"	050	2	066	B	082	R	098	b	114	r		
003	¥ (etx)	019	!! (dc3)	035	#	051	3	067	C	083	S	099	c	115	s		
004	♦ (eot)	020	¶ (dc4)	036	\$	052	4	068	D	084	T	100	d	116	t		
005	¤ (eng)	021	§ (nak)	037	¤	053	5	069	E	085	U	101	e	117	u		
006	♣ (ack)	022	- (syn)	038	&	054	6	070	F	086	V	102	f	118	v		
007	• (bel)	023	‡ (etb)	039	'	055	7	071	G	087	W	103	g	119	w		
008	□ (bs)	024	↑ (can)	040	<	056	8	072	H	088	X	104	h	120	x		
009	(tab)	025	↓ (em)	041	>	057	9	073	I	089	Y	105	i	121	y		
010	(lf)	026	<eof>	042	*	058	:	074	J	090	Z	106	j	122	z		
011	δ (vt)	027	← (esc)	043	+	059	:	075	K	091	[107	k	123	{		
012	♀ (np)	028	↳ (fs)	044	,	060	<	076	L	092	\`	108	l	124	:		
013	(cr)	029	↔ (gs)	045	-	061	=	077	M	093]	109	m	125	}		
014	▀ (so)	030	▲ (rs)	046	-	062	>	078	N	094	^	110	n	126	~		
015	* (si)	031	▼ (us)	047	/	063	?	079	O	095	_	111	o	127	△		
128	¢	143	฿	158	₭	172	₼	186	₪	200	₱	214	₩	228	£	242	₪
129	ū	144	€	159	ƒ	173	₭	187	₪	201	₱	215	₩	229	σ	243	₭
130	é	145	æ	160	á	174	«	188	₪	202	₱	216	₩	230	μ	244	₪
131	â	146	Œ	161	í	175	»	189	₪	203	₱	217	₩	231	χ	245	₪
132	ä	147	ô	162	ó	176	₪	190	₪	204	₱	218	₩	232	ø	246	₪
133	à	148	ö	163	ú	177	₪	191	₪	205	₪	219	₩	233	ø	247	₪
134	ã	149	ò	164	ñ	178	₪	192	₪	206	₪	220	₩	234	Ω	248	₪
135	ç	150	å	165	ñ	179	₪	193	₪	207	₪	221	₩	235	ø	249	₪
136	é	151	ù	166	¤	180	₪	194	₪	208	₪	222	₩	236	¤	250	₪
137	ë	152	ÿ	167	¤	181	₪	195	₪	209	₪	223	₩	237	¤	251	₪
138	ë	153	ø	168	ȝ	182	₪	196	₪	210	₪	224	¤	238	€	252	₪
139	í	154	ø	169	ȝ	183	₪	197	+	211	₪	225	¤	239	n	253	₪
140	í	155	¢	170	¤	184	₪	198	₪	212	¤	226	₪	240	≡	254	₪
141	í	156	£	171	¤	185	₪	199	₪	213	¤	227	₪	241	‡	255	₪
142	ã	157	¥														

Figuur 1.3: Extended ASCII tabel

1.5.2 EBCDIC

Naast ASCII werden ook andere standaarden ontwikkeld. Zo ook de EBCDIC, ontwikkeld door IBM en ontworpen voor de werking met prikkaarten bij mainframes. Ondertussen wordt EBCDIC niet meer gebruikt.

1.5.3 Unicode

ASCII is gebaseerd op de schrifttekens van ons alfabet. Naast ons alfabet zijn er nog een heleboel andere geschriften: Grieks, Hebreeuws, Arabisch, ... Twee belangrijke projecten werden opgezet om alle karakters van de verschillende schrifttekens te omvatten, ISO-10464 en Unicode. Unicode biedt een uniek getal voor elk teken, ongeacht het gebruikte platform, het gebruikte programma of de gebruikte taal. De Unicode-standaard is overgenomen door toonaangevende

bedrijven als Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, Unisys en vele anderen. Unicode wordt gebruikt in de besturingssystemen Windows NT, 2000, XP en Vista. Ook Java en het .Net platform, Mac OS X en KDE gebruiken Unicode voor de interne representatie van symbolen. Unicode wordt vereist door moderne standaards als XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML enzovoort, en vormt de officiële manier om ISO/IEC 10646 te implementeren. Het gebruik van Unicode in client-server of multi-tier toepassingen en in websites vormt een aanzienlijke kostenbesparing ten opzichte van het gebruik van de bestaande tekensets. Unicode zorgt ervoor dat één softwareproduct of één website zonder verdere aanpassingen kan worden gebruikt op meerdere platforms, in meerdere talen en landen. Gegevens kunnen worden uitgewisseld tussen verschillende systemen zonder dat hierbij gegevens verloren gaan of veranderen. Unicode maakt gebruik van 16 bits, wat toelaat 65536 karakters te definiëren. Omwille van achterwaartse compatibiliteit zijn de eerste 128 tekens dezelfde als bij ASCII.

Hoofdstuk 2

Logische poorten en schakelingen

2.1 Logische poorten

Logische poorten komen ontzettend veel voor in de informaticawereld. Niet alleen worden zeer veel poorten gebruikt bij het bouwen van een computer, ook bij het programmeren zijn de logische poorten belangrijk. Een logische poort is een eenheid met invoer en uitvoer. De poort legt vast welke uitvoer wordt geproduceerd bij een bepaalde invoer. We spreken hier steeds over digitale elektronica, aangezien de in- en uitgangen maar twee spanningsniveaus kunnen hebben(bv. 0V en 5V). We houden ons ook niet bezig met de elektronische opbouw van de poorten. In realiteit zijn de poorten opgebouwd uit enkele transistoren en eventueel andere elektronische componenten. We maken hier abstractie van en bekijken alles op het logische niveau. Ingangen en uitgangen kunnen bij logische poorten maar 2 waarden aannemen: waar(1) of onwaar(0). Deze kennen we ook als true, respectievelijk false.

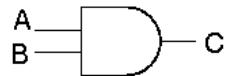
Wanneer we voor alle mogelijke invoerwaarden de uitvoerwaarden vastleggen, ligt het gedrag van de poort vast. Al deze gegevens weergegeven in een tabel noemt men een waarheidstabell. Om gemakkelijk aan te geven welke waarde in de tabel bij welke ingang of uitgang hoort, duiden we deze meestal aan met een letter. We kunnen deze letter echter ook verder gaan gebruiken als logische variabele en er mee rekenen(bijvoorbeeld vereenvoudigingen uitvoeren). We begeven ons hier in de wereld van de Booleaanse¹ algebra of schakelalgebra². Dit is een formalisme voor het weergeven van uitspraken in een logica, die twee toestanden kent: true en false. Bij de verschillende poorten zullen we steeds verwijzen naar de overeenkomende formules uit de schakelalgebra.

2.1.1 AND- of EN-poort

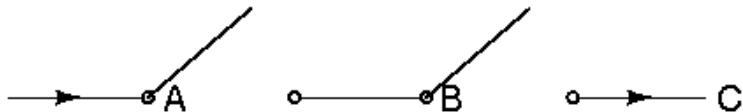
De eerste basispoort is de AND-poort. Deze wordt getoond in figuur 2.1. De AND-poort kan het beste vergeleken worden met twee in serie geplaatste schakelaars(figuur 2.2). Indien zowel A en B gesloten zijn (waar zijn of gelijk aan 1 zijn), zal de stroom door de schakelaars vloeien en zal C ook waar zijn. Dit is exact wat een AND-poort doet. De waarheidstabell van de AND-poort wordt weergegeven in tabel 2.1. De AND-poort wordt in de Booleaanse algebra

¹Deze naam komt van George Boole(1815-1864), de Engelse wiskundige die dit formalisme heeft ontwikkeld.

²De schakelalgebra werd in 1938 ontworpen door Claude Shannon. Deze algebra is een speciaal geval van de Boolese algebra.



Figuur 2.1: De AND-poort



Figuur 2.2: De AND-poort met schakelaars

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Tabel 2.1: Waarheidsstabel van de AND-poort

voorgesteld door middel van een punt. Dit punt is het symbool voor de vermenigvuldiging. Een AND-poort met ingangen A en B wordt dan voorgesteld als: $A \cdot B$. Direct kan men de relatie leggen met vermenigvuldigen van natuurlijke getallen:

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

De AND-poort kan het best als volgt worden geïnterpreteerd: “de AND poort geeft enkel 1 wanneer beide ingangen 1 zijn”.

2.1.2 OR- of OF-poort

De OF-poort wordt getoond in figuur 2.3. Deze poort kan het best vergeleken worden met twee

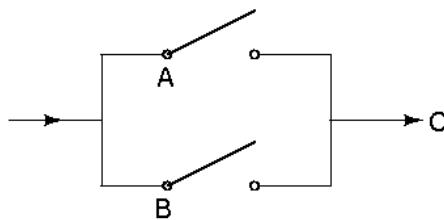


Figuur 2.3: De OR-poort

schakelaars die in parallel zijn aangesloten (figuur 2.4). Wanneer 1 van de twee schakelaars gesloten is, zal de stroom kunnen doorvloeien. De OF-poort wordt als volgt geïnterpreteerd: de OF-poort geeft 1 wanneer minstens 1 van de ingangen 1 is”. De waarheidstabellen voor de OF-poort is weergegeven in tabel 2.2. De OF-poort wordt in de Booleaanse algebra weergegeven door het optellingsteken: +. We kunnen dus weer de formules opstellen die overeenkomen met de OF-poort:

$$0 + 0 = 0$$

$$0 + 1 = 1$$



Figuur 2.4: De OR-poort met schakelaars

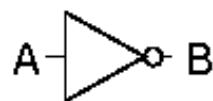
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Tabel 2.2: Waarheidstabel van de OR-poort

$$\begin{aligned} 1 + 0 &= 1 \\ 1 + 1 &= 1 \end{aligned}$$

2.1.3 NOT- of NIET-poort

De NOT-poort zet een 1 om in een 0 en omgekeerd. De NOT-poort wordt weergegeven in figuur 2.5. Ook hier kunnen we een vergelijking maken met schakelaars. Het betreft hier een



Figuur 2.5: De NOT-poort

speciale schakelaar, die stroom doorlaat wanneer hij open staat en stroom tegenhoudt wanneer hij gesloten is. De NOT-poort met behulp van schakelaars wordt weergegeven in figuur 2.6. Tenslotte geven we ook hier de waarheidstabel in tabel 2.3. Merk op dat de NOT-poort maar



Figuur 2.6: De NOT-poort met schakelaars

A	C
0	1
1	0

Tabel 2.3: Waarheidstabel van de NOT-poort

1 ingang heeft, in tegenstelling tot de OR- en AND-poorten. De uitgang wordt bij de NOT-poort ook wel de inverse genoemd, aangezien een 1 omslaat in 0, en omgekeerd. Aangezien de NOT-poort maar 1 ingang heeft, moeten we een speciaal symbool gebruiken bij de Booleaanse algebra. Het gebruikte symbool is een streep boven de betreffende variabele: \bar{A} . We kunnen dus stellen:

$$\bar{1} = 0$$

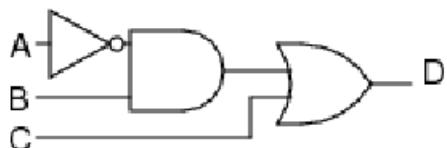
$$\bar{0} = 1$$

Een extra gegeven in verband met de NOT-poort is:

$$\overline{\bar{A}} = A$$

2.2 Logische schakelingen

Door verschillende van de eerder besproken basispoorten te combineren, kunnen logische schakelingen gevormd worden. Figuur 2.7 geeft een voorbeeld van zo'n gecombineerde schakeling. Hier wordt de uitgang van de NOT-poort als ingang van de AND-poort gebruikt. Daarna



Figuur 2.7: Een gecombineerde schakeling

wordt de uitgang van de AND-poort voor de ingang van de OR-poort gebruikt. De getoonde schakeling komt overeen met een formule uit de schakelalgebra: $D = \bar{A} \cdot B + C$. Ook van deze schakeling kan er een waarheidstabell worden opgesteld (zie figuur 2.4). De voorrangsregels voor

A	B	C	D
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Tabel 2.4: De waarheidstabell van de gecombineerde schakeling

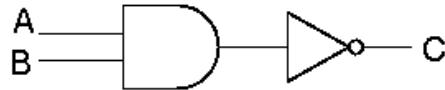
de Booleaanse algebra zijn als volgt (van hoge prioriteit naar lage prioriteit):

- De negatie (NOT)
- De vermenigvuldiging (AND)
- De optelling (OR)

Enkele speciale combinaties verdienen extra aandacht: de NAND-, NOR- en EXOR-poorten. De NAND- en NOR-poorten zijn goedkoper te fabriceren (door middel van goedkopere elektronische componenten), waardoor ze veel gebruikt worden.

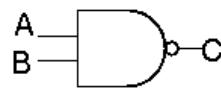
2.2.1 De NAND-poort

De NAND-poort is een aaneenschakeling van een AND- en een NOT-poort. Deze logische schakeling wordt getoond in figuur 2.8. Wanneer een NOT-poort aan de in- of uitgang van



Figuur 2.8: De NAND-poort

een andere poort wordt geplaatst, gebruikt men dikwijls een verkorte weergave door gewoon een bolletje te tekenen die de negatie voorstelt. Figuur 2.9 toont de verkorte weergave voor de NAND-poort. Dit is tevens het standaardsymbool voor deze poort. De waarheidstabell van



Figuur 2.9: Het symbool voor de NAND-poort

de NAND-poort staat in tabel 2.5. De NAND-poort heeft als Booleaanse formule: $C = \overline{A \cdot B}$.

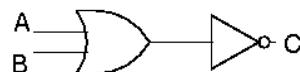
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

Tabel 2.5: Waarheidstabell van de NAND-poort

In deze formule is duidelijk dat eerst de AND-bewerking wordt uitgevoerd, en pas daarna de negatie (de NOT). De NAND-poort geeft enkel 0 indien beide ingangen 1 zijn.

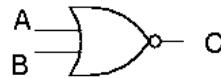
2.2.2 NOR-poort

De NOR-poort is een aaneenschakeling van een OR- en een NOT-poort. Deze schakeling wordt getoond in figuur 2.10. Ook voor deze poort bestaat er een standaardsymbool, dit wordt



Figuur 2.10: De NOR-poort

weergegeven in figuur 2.11. De formule van de NOR-poort is $C = \overline{A + B}$. De waarheidstabell staat in tabel 2.6. De NOR-poort geeft enkel 1 indien de beide ingangen 0 zijn.



Figuur 2.11: Het symbool voor de NOR-poort

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

Tabel 2.6: Waarheidstabel van de NOR-poort

2.2.3 XOR-poort

De XOR-poort of exclusieve-OF-poort (ook soms EXOR) zal enkel 1 geven indien de ingangen verschillend zijn van elkaar. De logische formule is dus $C = (\overline{A} \cdot B) + (A \cdot \overline{B})$. Het symbool van de XOR-poort wordt weergegeven in figuur 2.12 en de waarheidstabel in tabel 2.7. Probeer zelf de schakeling te tekenen in functie van de basisschakelingen.



Figuur 2.12: Het symbool voor de XOR-poort

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Tabel 2.7: Waarheidstabel van de XOR-poort

2.3 Rekenregels

Binnen de Booleaanse algebra zijn een aantal rekenregels ontwikkeld. Deze regels drukken logische waarheden uit. Ze kunnen allen bewezen worden.

2.3.1 Basisregels

Volgende basisrekenregels leggen de betekenis van de AND-, OR- en NOT-poorten uit:

$1 + 1 = 1$	$1 + 0 = 1$	$0 + 0 = 0$
$1 \cdot 1 = 1$	$1 \cdot 0 = 0$	$0 \cdot 0 = 0$
$\bar{0} = 1$	$\bar{1} = 0$	

We zien ook hier weer dat de + en . bewerkingen overeenkomen met die de bewerkingen

op natuurlijke getallen, uitgezonderd dat $1 + 1 = 1$ in de schakelalgebra.

Ook volgende rekenregels zijn “logisch” en dus eenvoudig na te gaan door het bekijken van de waarheidstabellen van de verschillende poorten:

$$\begin{array}{lll} A + 1 = 1 & A + 0 = A & A + A = A \\ A + \bar{A} = 1 & A \cdot 1 = A & A \cdot 0 = 0 \\ A \cdot A = A & A \cdot \bar{A} = 0 & \bar{\bar{A}} = A \end{array}$$

We bekijken in detail hoe een rekenregel het best wordt begrepen. Neem bijvoorbeeld $A \cdot A = A$. Indien een AND-operatie wordt uitgevoerd op eenzelfde variabele is dit gelijk aan de waarde van de variabele zelf. Dit is eenvoudig af te leiden uit de waarheidstabel van de AND-poort. Daarin staat immers dat $1 \cdot 1 = 1$ en $0 \cdot 0 = 0$. We hebben dus alle mogelijke waarden voor A bekijken en hiervoor geldt de gelijkheid.

2.3.2 Commutativiteit

De commutativiteit drukt het volgende uit:

$$\begin{array}{l} A + B = B + A \\ A \cdot B = B \cdot A \end{array}$$

Zowel de AND- als de OR-poort zijn dus commutatief: het maakt niet uit in welke volgorde de variabelen staan. Ook dit kan eenvoudig bewezen worden door het geven van de waarheidstabellen van de AND- en de OR-poort.

2.3.3 Associativiteit

De regels van de associativiteit:

$$\begin{array}{l} A + (B + C) = (A + B) + C = A + B + C \\ A \cdot (B \cdot C) = (A \cdot B) \cdot C = A \cdot B \cdot C \end{array}$$

2.3.4 Distributiviteit

$$\begin{array}{l} A + B \cdot C = (A + B) \cdot (A + C) \\ A \cdot (B + C) = A \cdot B + A \cdot C \end{array}$$

2.3.5 Absorptie

$$\begin{array}{l} A + A \cdot B = A \\ A \cdot (A + B) = A \end{array}$$

Door de voorrangsregels weten we dat de \cdot voorrang heeft op $+$. In feite staat er dus $A + (A \cdot B) = A$. We bewijzen deze gelijkheid ter illustratie:

$$\begin{aligned} A + A \cdot B &= A \cdot 1 + A \cdot B (\text{basisregels}) \\ &= A \cdot (1 + B) (\text{distributiviteit}) \\ &= A (\text{basisregels}) \end{aligned}$$

2.3.6 Regels van De Morgan

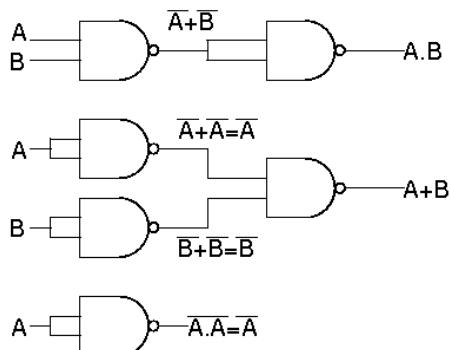
De regels van de Morgan³ worden als volgt geformuleerd:

$$\boxed{\begin{aligned} A + \bar{B} &= \bar{A} \cdot \bar{B} \\ \bar{A} \cdot \bar{B} &= \bar{A} + \bar{B} \end{aligned}}$$

De wetten van de Morgan en alle hierboven vermelde rekenregels worden vaak gebruikt om logische formules te vereenvoudigen. Vereenvoudiging is geldwinst in de industrie, aangezien men met minder poorten dezelfde schakeling bekomt. Dikwijls worden logische formules zo omgevormd dat ze quasi-enkel NAND- en NOR-poorten bevatten, omdat deze goedkoper te produceren zijn.

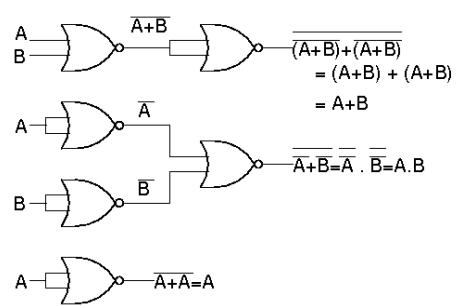
2.3.7 Computationeel compleetheid van NAND en NOR

We bekijken tenslotte een speciale eigenschap van NAND- en NOR-poorten. We kunnen immers zeggen dat de NAND- en NOR-poorten computationeel compleet zijn. Dit wil zeggen dat elke combinatorische functie kan gerealiseerd worden met enkel NAND's of enkel NOR's. We kunnen dus eender welke schakeling opbouwen door enkel gebruik te maken van NAND-poorten of NOR-poorten. Dit kan het best geïllustreerd worden door de AND-, OR- en NOT-poorten te construeren uit enkel NAND-poorten (of enkel NOR-poorten). Op deze manier bewijzen we dat elke mogelijke schakeling omzetbaar is naar een schakeling met als bouwstenen enkel NAND- of NOR-poorten. Figuur 2.13 en figuur 2.14 tonen deze schema's voor de NAND-poort, respectievelijk de NOR-poort.



Figuur 2.13: De NAND-poort is computationeel compleet

³Augustus De Morgan (1806-1871) was een belangrijke trendsetter op het gebied van logica. Op school was de Morgan zeker niet een van de betere leerlingen. Kort achter zijn geboorte verloor hij het zicht van zijn rechteroog en nam daardoor niet deel aan sportactiviteiten. De Morgan was dan ook vaak het slachtoffer van practical jokes. Later verwierf hij grote faam in de wereld van de wiskunde en logica. In 1828 werd hij professor wiskunde aan de University College in London.



Figuur 2.14: De NOR-poort is computationeel compleet

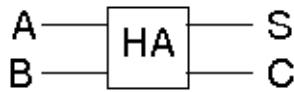
Hoofdstuk 3

Rekenschakelingen

Rekenschakelingen worden gebruikt om wiskundige operaties uit te voeren op binaire getallen. Ze worden in een computer gebruikt om basisinstructies te voorzien zoals: optellen, aftrekken, ...

3.1 Halve opteller (half adder)

De eerste belangrijke optelschakeling is de halve opteller. Deze schakeling kan twee bits optellen en heeft een twee-bits resultaat: een som en carry-bit. De carry-bit is de overdrachtsbit: ze geeft weer of er na de optelling overdracht is (bijvoorbeeld: $1+1=0$ met overdrachtsbit 1). De halve opteller wordt getoond in figuur 3.1. A en B zijn de twee bits die moeten worden



Figuur 3.1: De halve opteller (half adder)

opgeteld, S is de som (sum) en C is de overdrachtsbit (carry). Met de halve opteller willen we de waarheidstabell bekomen die wordt getoond in tabel 3.1. Immers, de optelling van 2 bits is als volgt gedefinieerd:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

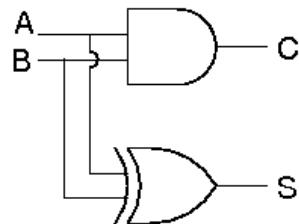
$$1 + 1 = 10$$

De waarheidstabell toont ons dat de som 1 is wanneer A en B verschillend zijn van elkaar. Daarnaast is de carry 1 wanneer zowel A als B 1 zijn. Deze schakelingen komen ons bekend

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

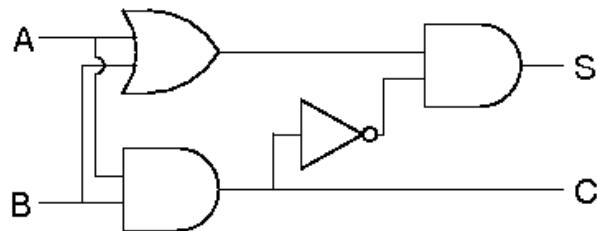
Tabel 3.1: De waarheidstabell voor de halve opteller

voor: een XOR- en een AND-poort. De halve opteller, gerealiseerd met logische poorten, wordt getoond in figuur 3.2. De halve opteller kan ook op andere manieren gerealiseerd worden. Een



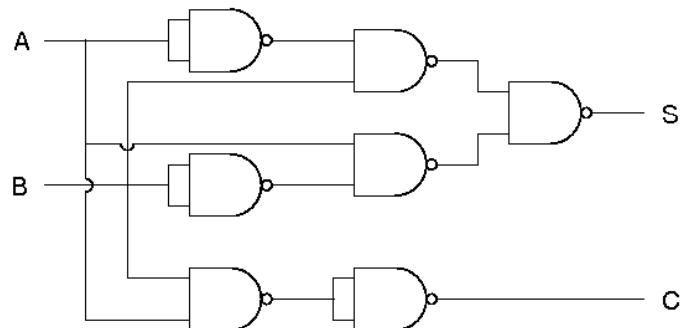
Figuur 3.2: De halve opteller (half adder), opgebouwd uit logische poorten

van de alternatieve denkpatronen is de volgende: “De overdracht is $A \text{ AND } B$. De sombit $S=A+B$ ($A \text{ OR } B$), behalve wanneer de overdracht = 1, dan is $S=0$ ”. Op basis van deze alternatieve formulering kunnen we een andere schakeling construeren die toch hetzelfde doet. Deze wordt weergegeven in figuur 3.3. De “behalve” in deze formulering kunnen we logisch voorstellen met een AND-poort. Deze AND-poort zal nu dienen als schakelaar: wanneer de schakelaar op 1 staat, laat de AND-poort (rechtsboven) stroom door. Indien de schakelaar op 0 staat laat de schakelaar de stroom niet door. Wanneer de carry dus 1 is, zal door de NOT-poort de schakelaar gesloten worden waardoor de som op 0 komt te staan. De halve opteller



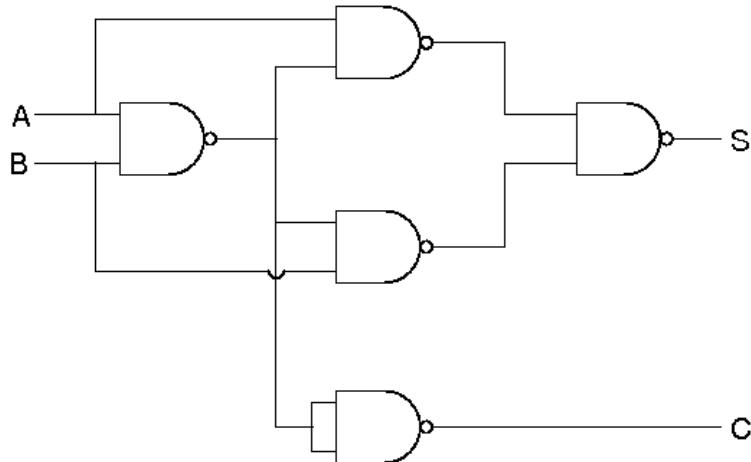
Figuur 3.3: Een alternatieve versie van de halve opteller, opgebouwd uit logische poorten

kan ook geconstrueerd worden op basis van enkel NAND-poorten. Dit wordt geïllustreerd in figuur 3.4. Deze schakeling kan zelfs met nog minder NAND-poorten gerealiseerd worden, dit



Figuur 3.4: De halve opteller, opgebouwd uit NAND-poorten

wordt getoond in figuur 3.5.



Figuur 3.5: De halve opteller, opgebouwd uit NAND-poorten (tweede versie)

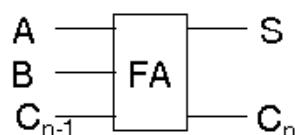
3.2 Volledige opteller (full adder)

Bij het optellen van twee bits moeten we dikwijs de carry van de vorige optelling mee in rekening brengen. De halve opteller houdt hier geen rekening mee. De hele opteller is een optelschakeling die naast twee invoerbits, ook de carry-bit van de vorige optelling als invoer heeft. Een voorbeeldoptelling wordt getoond in figuur 3.6. Hier wordt duidelijk dat we bij een optelling de mogelijkheid willen hebben om een carry in rekening te brengen.

$$\begin{array}{r}
 1011 \\
 + 1110 \\
 \hline
 0101 \text{ sombits} \\
 1 \ 1 \text{ carry} \\
 \hline
 11001 \text{ complete som}
 \end{array}$$

Figuur 3.6: Binaire optelling: carry of overdracht

De volledige opteller laat ons toe deze carry in rekening te brengen, deze wordt getoond in figuur 3.7. De waarheidstabellen voor de volledige opteller wordt getoond in tabel 3.2. Wanneer



Figuur 3.7: De volledige opteller

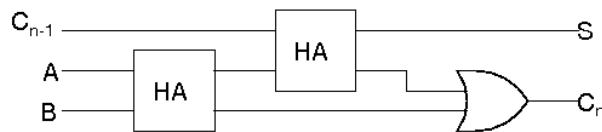
wij de tabel nader bekijken, kunnen we de bewerking opsplitsen in 2 eenvoudigere optellingen:

1. A+B
2. Het resultaat van 1 + de carry aan de invoer

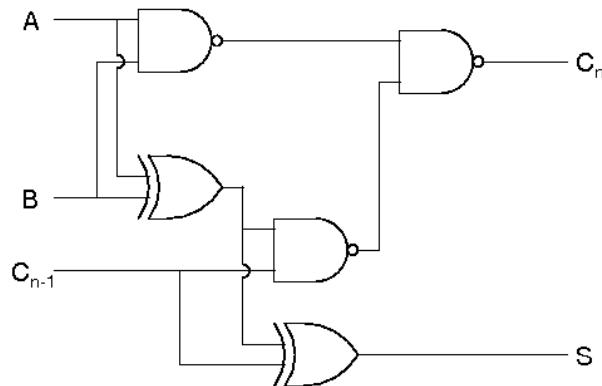
A	B	C_{n-1}	S	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabel 3.2: Waarheidstabell van de volledige opteller

De uiteindelijke sombit is die uit 2, dit is de sombit van de drie ingangen. De uiteindelijke carry komt OF van 1 OF van 2. Deze kan niet van beide tegelijk komen, ga dit zelf na. Een volledige opteller kan dus geconstrueerd worden op basis van halve optellers, dit wordt getoond in figuur 3.8. In figuur 3.9 wordt de volledige opteller getoond, opgebouwd uit enkel XOR- en NAND-poorten.



Figuur 3.8: De volledige opteller, opgebouwd halve optellers



Figuur 3.9: De volledige opteller, opgebouwd halve optellers

3.3 Intermezzo: maken van logische schakelingen

Hoe komen we nu tot het tekenen van een schakeling, vertrekkende vanuit een waarheidstable? We illustreren dit proces aan de hand van de volledige optelschakeling. Tabel 3.2 is het vertrekpunt.

3.3.1 Omzetten naar formulevorm

De eerste stap die meestal wordt ondernomen is het omzetten van de waarheidstabel naar een logische formule in de schakelalgebra. We moeten hierbij voor elke uitvoervariabele (in dit geval S en C_n) een formule opstellen in functie van de invoervariabelen: A , B en C_{n-1} . De formule wordt als volgt opgesteld:

- Zoek de uitvoervariabele waarvoor een formule gevraagd wordt op in de tabel.
- Voor elke 1 die in deze kolom staat, zullen we een term toevoegen aan de formule. De verschillende termen worden onderling gesommeerd (sommatie = OR-poort)
- De individuele termen worden als volgt opgesteld: alle invoervariabelen worden met elkaar vermenigvuldigd (vermenigvuldiging = AND-poort), waarbij een NOT wordt toegepast op een invoervariabele, indien er een 0 staat in de kolom van die invoervariabele

Een voorbeeld maakt dit duidelijker.

Voorbeeld:

We stellen een logische formule op voor S uit tabel 3.2. We kijken naar alle rijen waar een 1 staat voor S . Dit zijn rijen 1, 2, 4 en 7 (we beginnen te tellen vanaf 0, de eerste rij is rij 0 en de laatste rij is rij 7). Voor elk van deze rijen zullen we een term moeten toevoegen, dus we weten dat de formule van de volgende vorm zal zijn:

$$S = \dots + \dots + \dots + \dots$$

We zoeken nu de individuele termen. Eerst bekijken we rij 1. Hier staan voor de invoervariabelen A , B en C_{n-1} de respectieve waarden 0, 0 en 1 vermeld. Deze term wordt dus $\overline{A} \cdot \overline{B} \cdot C_{n-1}$. Onze formule ziet er dus momenteel als volgt uit:

$$S = \overline{A} \cdot \overline{B} \cdot C_{n-1} + \dots + \dots + \dots$$

We bekijken de tweede term. Hiervoor beschouwen we rij 2. Aangezien de waarden voor A , B en C hier 0, 1 en 0 zijn zal deze term gelijk aan $\overline{A} \cdot B \cdot \overline{C_{n-1}}$ zijn. Onze formule heeft nu volgende vorm:

$$S = \overline{A} \cdot \overline{B} \cdot C_{n-1} + \overline{A} \cdot B \cdot \overline{C_{n-1}} + \dots + \dots$$

Wanneer we deze redenering doorvoeren krijgen we de formule:

$$S = \overline{A} \cdot \overline{B} \cdot C_{n-1} + \overline{A} \cdot B \cdot \overline{C_{n-1}} + A \cdot \overline{B} \cdot \overline{C_{n-1}} + A \cdot B \cdot C_{n-1}$$

Afspraak: Vanaf nu zullen we $A \cdot B$ ook soms schrijven als AB .

De formule die we bekomen voor S en C_n zijn dan:

$$S = \overline{A} \cdot B \cdot C_{n-1} + \overline{A} \cdot B \cdot \overline{C_{n-1}} + A \cdot \overline{B} \cdot \overline{C_{n-1}} + A \cdot B \cdot C_{n-1}$$

$$C_n = \overline{A} \cdot B \cdot C_{n-1} + A \cdot \overline{B} \cdot C_{n-1} + A \cdot B \cdot \overline{C_{n-1}} + A \cdot B \cdot C_{n-1}$$

3.3.2 Vereenvoudigen van de formule

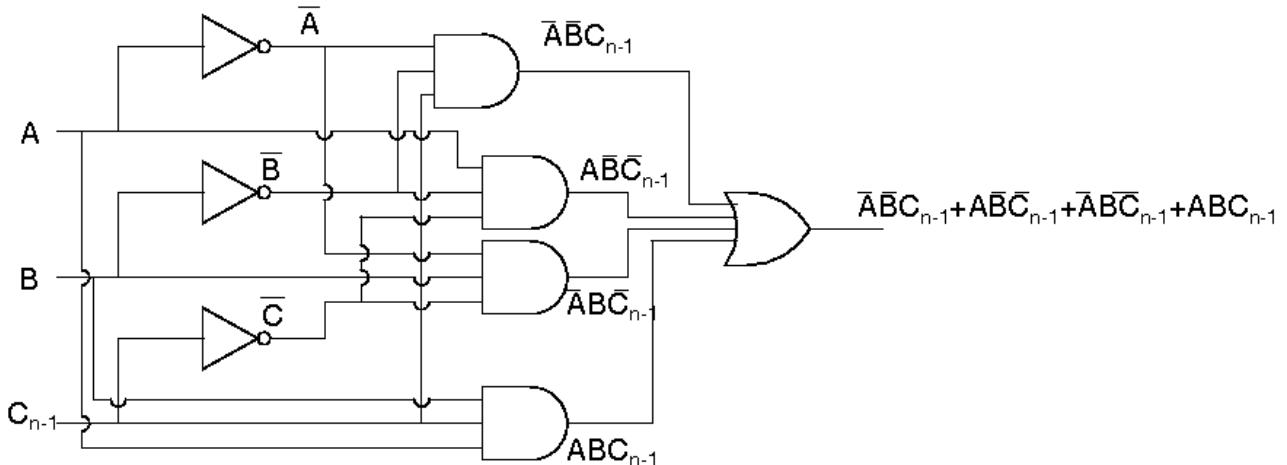
Om minder logische poorten te moeten gebruiken, kunnen we de formules proberen te vereenvoudigen. Dit kan gebeuren door toepassing van de verschillende rekenregels. Zo kunnen we de formule voor C_n als volgt vereenvoudigen:

$$\begin{aligned} C_n &= \overline{A} \cdot B \cdot C_{n-1} + A \cdot \overline{B} \cdot C_{n-1} + A \cdot B \cdot \overline{C_{n-1}} + A \cdot B \cdot C_{n-1} \\ &= \overline{A} \cdot B \cdot C_{n-1} + A \cdot \overline{B} \cdot C_{n-1} + A \cdot B (\overline{C_{n-1}} + C_{n-1}) \quad (\text{distributiviteit}) \\ &= \overline{A} \cdot B \cdot C_{n-1} + A \cdot \overline{B} \cdot C_{n-1} + A \cdot B \quad (\text{basisregels}) \end{aligned}$$

3.3.3 Omzetten van de formule naar een logische schakeling

De laatste stap die moet gebeuren is het omzetten van de vereenvoudigde formule naar een effectieve logische schakeling. Hiervoor zullen we in de schakeling voor elke formule een uitvoer krijgen, en voor elke vermelde variabele aan de rechterzijde van de formule een invoer. De logische poorten kunnen afgeleid worden uit de logische bewerkingen +, . en - (not). Uiteraard kunnen we ook de andere richting uit. Formules kunnen ook afgeleid worden uit een schema van een logische schakeling. Hiervoor volgt men de verschillende poorten en schrijft men de tussenbewerkingen erbij als hulpmiddel (zoals in figuur 2.13 op pagina 24).

We zetten nu de formules om naar een schakeling. Eerst werken we het gedeelte van S uit, daarna breiden we dit schema uit voor C_n . Figuur 3.10 toont de schakeling voor het implementeren van de logische formule voor S . Op deze figuur zien we dat het al gauw ingewikkeld kan worden bij het tekenen van de schakeling. Daarom is het verstandig om telkens de tussenresultaten te noteren bij de uitvoer van elk van de poorten. Merk op dat er poorten worden gebruikt met meer dan 2 ingangen. Dit is enkel om het schema te vereenvoudigen, een OR-bewerking op 4 invoersignalen kan verwezenlijkt worden door de eerste twee signalen naar een eerste OR-poort te sturen, de volgende 2 signalen naar een tweede OR-poort te sturen. Tenslotte worden beide uitgangen gecombineerd in een derde OR-poort. Hetzelfde geldt voor de AND-poorten. Nu breiden we deze tekening uit door de schakelingen voor C_n toe te voegen.

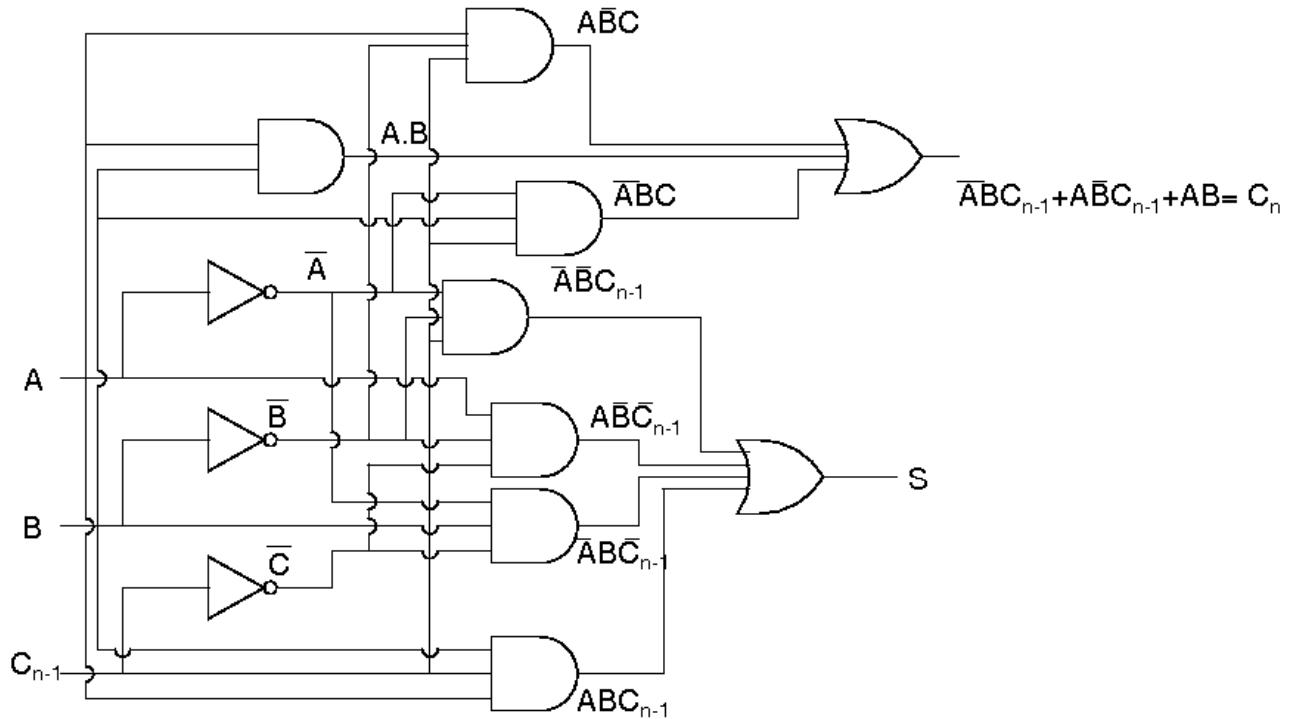


Figuur 3.10: De logische schakeling voor S

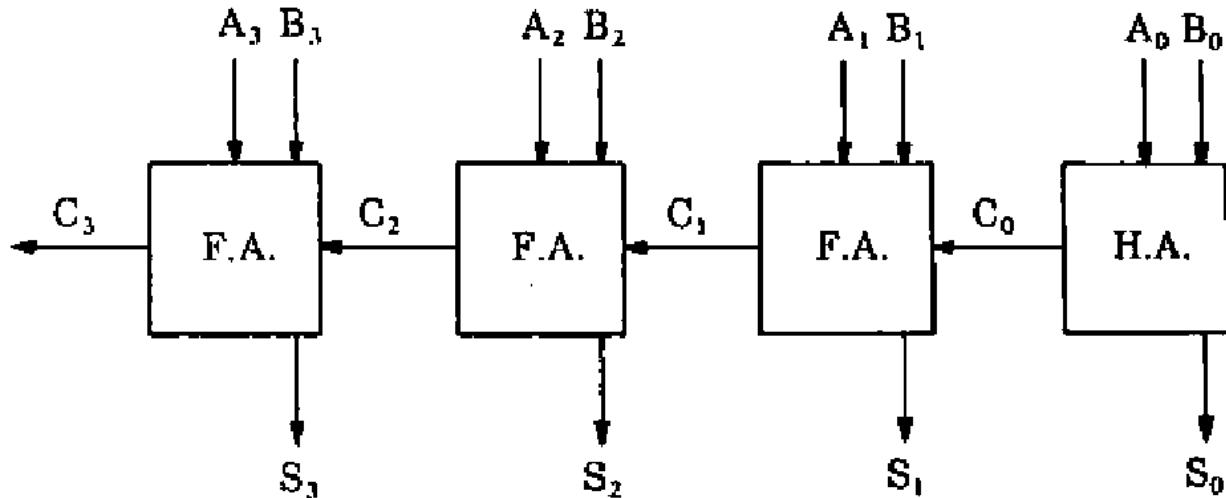
De gehele schakeling wordt getoond in figuur 3.11.

3.4 Parallel-opteller

Tot nu toe kunnen we enkel 2 bits optellen, eventueel met een carry-bit erbij. In realiteit willen we uiteraard binaire getallen bij elkaar optellen. We hebben dus nood aan een optelschakeling die twee volledige getallen kan optellen. De eerste variant van zo'n optelschakeling noemt de parallel-opteller. Figuur 3.12 toont deze schakeling. Deze schakeling telt twee binaire getallen $A = \dots A_3 A_2 A_1 A_0$ en $B = \dots B_3 B_2 B_1 B_0$ op en zet het resultaat in $S = \dots S_3 S_2 S_1 S_0$. De parallel-opteller bestaat uit één halve opteller, gevolgd door een aantal volledige optellers. De eerste schakeling is een halve opteller, omdat hier nog geen overdracht in rekening hoeft gebracht te



Figuur 3.11: De logische schakeling van de volledige opteller



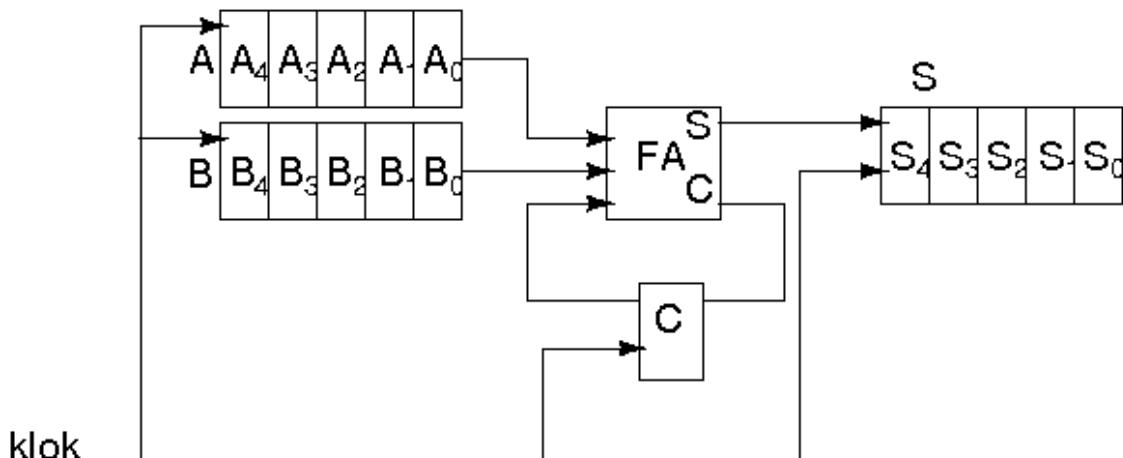
Figuur 3.12: De parallelopteller

worden. Het probleem bij deze schakeling is dat de tweede opteller moet wachten op de carry van de eerste opteller, de derde opteller moet wachten op de carry van de tweede opteller, enz. Door de propagation delay¹ duurt het even voor het signaal van de carry effectief aan de volgende opteller wordt doorgegeven. Elke opteller brengt een propagation delay met zich mee, wat deze schakeling niet echt optimaal maakt.

¹Propagation delay is de tijd tussen het versturen van een signaal en het effectief ontvangen aan het andere eind van het communicatiemedium

3.5 Serie-opteller

Bij een serie-opteller worden alle bits 1 per 1 in een optelschakeling gebracht. Op deze manier is er maar 1 volledige opteller nodig, wat enorm kostenbesparend werkt. De serie-opteller wordt weergegeven in figuur 3.13. Vroeger was de kostprijs per opteller zeer hoog, waardoor

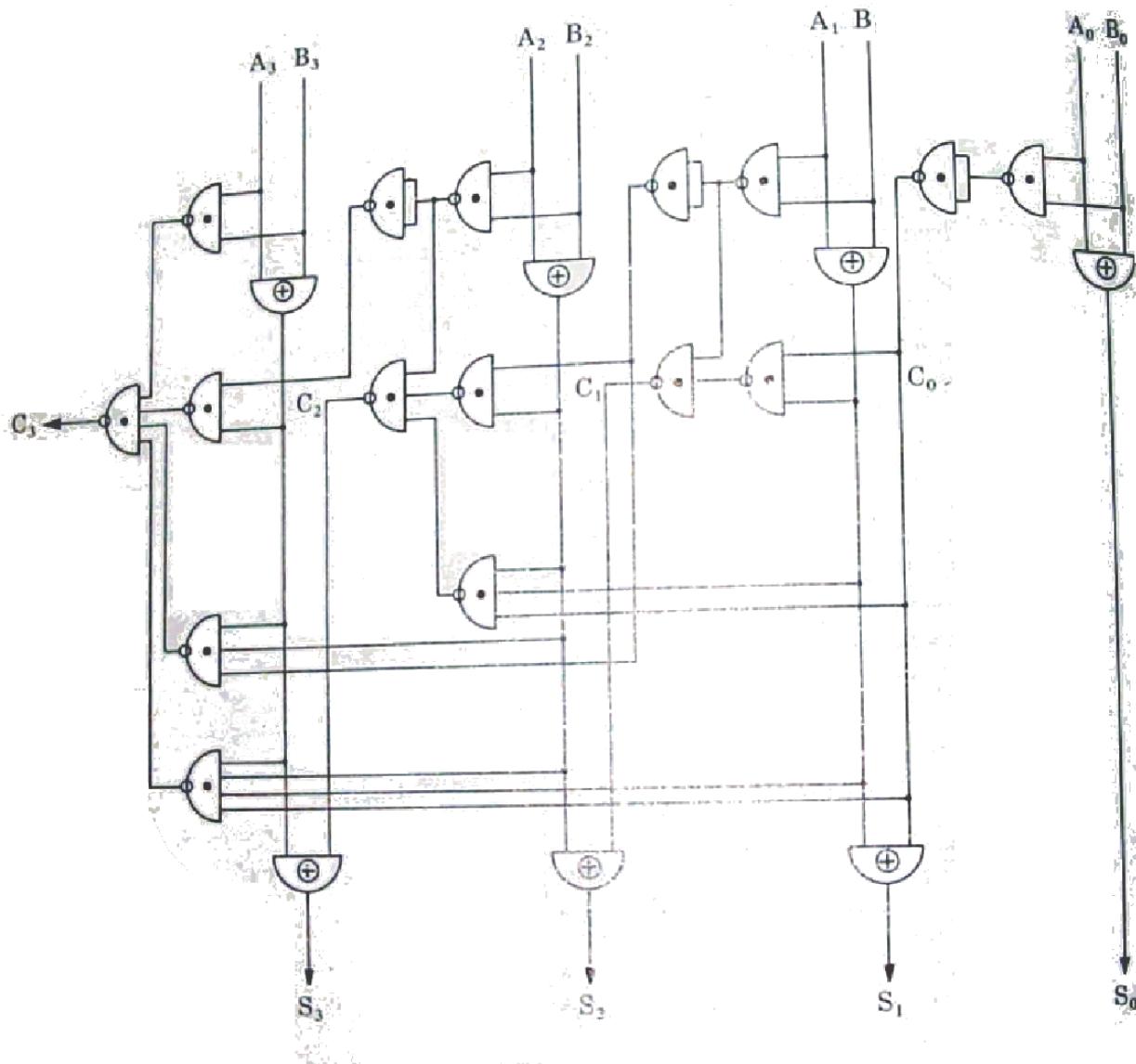


Figuur 3.13: De serie-opteller

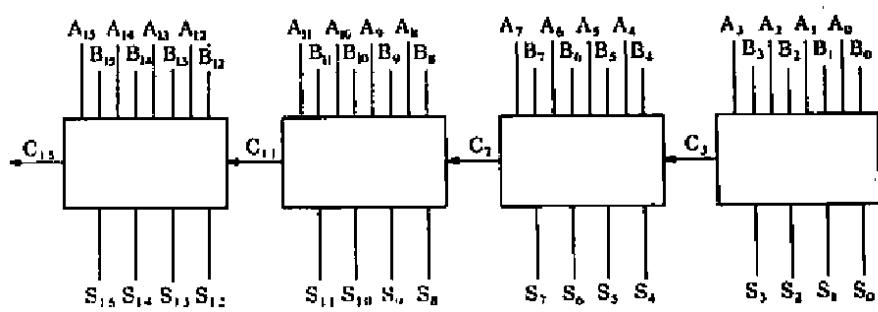
het uitsparen van optellers noodzakelijk was. Tegenwoordig is deze kostprijs niet zo belangrijk meer. Het nadeel van deze serie-opteller is dat de bewerkingstijd evenredig toeneemt met het aantal bits.

3.6 Simultaan-opteller

Om de nadelen van de parallel-opteller weg te werken, werd de simultaan-opteller ontworpen. Hierbij wordt de overdrachtbit via een schakeling berekend uit alle ingangen. Alle overdrachtbits worden op die manier elk afzonderlijk berekend en zijn dus allemaal op hetzelfde moment beschikbaar. Hierdoor zijn er geen wachttijden meer; het nadeel van de simultaan-opteller is dat hij ingewikkeld en dus duur is (veel schakelingen nodig). De simultaan-opteller wordt getoond in figuur 3.14. Vanwege de snelheid van de simultaan-opteller en de lagere kost van de parallel-opteller wordt vaak een combinatie van deze twee genomen: de simultaan-parallel-opteller. Figuur 3.15 toont een voorbeeld van zo'n opteller, waarbij overdrachten simultaan worden gegenereerd binnen groepjes van vier bits.



Figuur 3.14: De simultaan-opteller



Figuur 3.15: Een simultaan-parallel-opteller

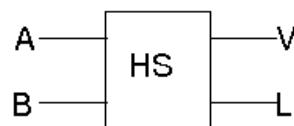
3.7 Halve aftrekker (half subtractor)

Analoog als bij de optellers, bestaat er een halve aftrekker. Deze heeft twee invoervariabelen: de twee bits die van elkaar moeten worden afgetrokken. Aan de uitgang is er het resultaat (V van Verschil) en de leenbit (L): deze geeft weer of er moet geleend worden voor de aftrekking. Tabel 3.3 geeft de waarheidstabellen voor de halve aftrekker. Het symbool voor de halve

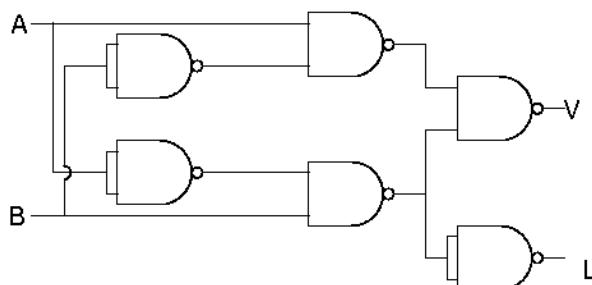
A	B	V	L
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Tabel 3.3: De waarheidstabellen voor de halve aftrekker

aftrekker wordt getoond in figuur 3.16. Vanuit de tabel kunnen we weer een logische schakeling opbouwen. Deze logische schakeling wordt getoond in figuur 3.17. Er worden enkel NAND-poorten gebruikt.



Figuur 3.16: De halve aftrekker



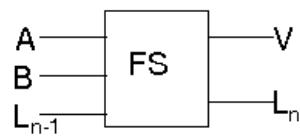
Figuur 3.17: De halve aftrekker op basis van NAND-poorten

3.8 Volledige aftrekker (full subtractor)

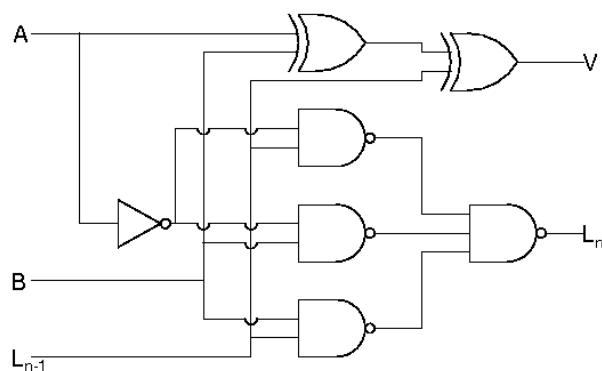
De laatste schakeling die we bekijken is de volledige aftrekker of full subtractor. Deze houdt rekening met de vorige leenbit. Dit wil zeggen dat indien de vorige keer geleend werd, er nu een extra 1-bit moet worden afgetrokken. De waarheidstabellen voor de volledige aftrekker wordt weergegeven in tabel 3.4, het symbool en het logische schema in figuur 3.18, resp. 3.19.

A	B	L_{n-1}	V	L_n
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Tabel 3.4: De waarheidstabel voor de volledige aftrekker



Figuur 3.18: De volledige aftrekker



Figuur 3.19: De volledige aftrekker op basis van logische poorten

Deel II

Computerhardware

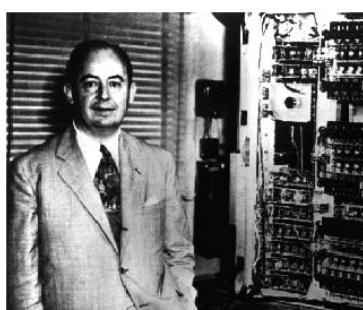
Hoofdstuk 4

Computerarchitectuur

Inhoudsopgave

4.1 von Neumann architectuur	38
4.2 Processor	40
4.3 Werkgeheugen	40
4.4 Permanent geheugen	40
4.5 In- en uitvoerapparaten (randapparatuur)	40
4.6 Chipsets en bussen	40

4.1 von Neumann architectuur



Figuur 4.1: John von Neumann

John von Neumann is met de naar hem vernoemde “von-Neumann-architectuur”, die de logische opbouw van alle moderne computers beschrijft, de (computer)geschiedenis ingegaan.

De in 1903 als zoon van een Joodse bankier in Boedapest geboren John von Neumann viel reeds vroeg in zijn leven op door zijn buitengewoon wiskundig talent. Tijdens de jaren '20 studeerde hij chemie in Berlijn en Zürich. Daarenboven doctoreerde hij nog eens wiskunde in Boedapest. Als privé-docent gaf hij zowel in Göttingen, Berlijn als Hamburg les. Toen hij van 1930 tot 1933 als gastprofessor mechanische fysica aan de universiteit van Princeton doceerde, was hij de jongste collega van Albert Einstein. Na het behalen van een professoraat aan het Institute for Advanced Study(IAS) in Princeton, besloot hij voorgoed in de VS te blijven. Von Neumann wordt beschouwd als één van de grootste wiskundigen van de 20e eeuw. Hij werkte o.a. rond de wiskundige grondbeginselen van de kwantummechanica, logica, economie en ballistiek.

Vanaf 1943 werkte von Neumann aan het Amerikaanse atoombomprogramma onder leiding van Robert Oppenheimer mee. Hij hielp o.a. bij de constructie van het ontstekingsmechanisme en de berekening van de optimale ontploffingshoogte van de bom die uiteindelijk op Nagasaki zou vallen. Hij maakte kennis met het ENIAC-team toen hij in 1944 op zoek was naar een hulpmiddel voor de verwerking van al deze gegevens. In zijn verslag over de EDVAC, de opvolger van de ENIAC, beschreef von Neumann de naar hem vernoemde computerarchitectuur. Deze zegt dat elke computer uit een stuurmechanisme, een rekenorgaan, een geheugen, een input en een output opgebouwd moet zijn.

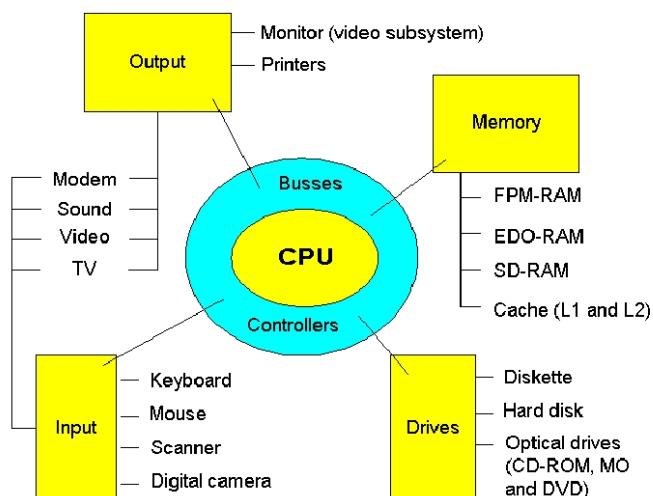
Na WO II keerde von Neumann naar het IAS terug om zich met de ontwikkeling van een elektronische computer voor wetenschappelijke doeleinden bezig te houden. Deze op zijn computerarchitectuur gebaseerd IAS-vacuümbuizencomputer was in 1952 bedrijfsklaar.

Naast zijn computerwerkzaamheden was hij een belangrijk militair adviseur, wiens invloed en erkenning tot in de hoogste kringen van de Amerikaanse regering en het leger reikte. John von Neumann overleed in februari 1957 aan kanker, veroorzaakt door de radioactieve straling waaraan hij tijdens de kernproeven blootgesteld was.

John von Neumann ontwikkelde in 1944 een computer architectuur model dat ook vandaag nog kan gebruikt worden. von Neumann deelde de computer op in 5 grote delen:

- CVE: De CVE of centrale verwerkseenheid. Dit is de processor, ook wel CPU(central processing unit) genoemd.
- Invoer: De invoerapparaten: muis, keyboard, scanner, ...
- Uitvoer: De uitvoerapparaten: scherm, printers, ...
- Werkgeheugen: Het werkgeheugen: FPM-RAM, DDR RAM, ...
- Permanent geheugen: Geheugen om informatie voor langere tijd op te slaan: diskette, harde schijf, CD-ROM, DVD-ROM, ...

Een aantal apparaten kunnen zowel voor invoer als uitvoer worden gebruikt. Voorbeelden hiervan zijn modems, geluid en video. Figuur 4.2 toont een schematische weergave van de von Neumann architectuur.



© Michael B. Karbo 1997

Figuur 4.2: von Neumann architectuur

4.2 Processor

In deze computerarchitectuur van von Neumann is de processor duidelijk het centrale gedeelte van ons computersysteem. Wanneer we programma's uitvoeren, zal de processor namelijk het meeste werk verrichten. De processor verwerkt opeenvolgende instructies en daardoor is het mogelijk om uiteindelijk een volledig programma uit te voeren. Hoofdstuk 6 gaat dieper in op de werking van de processor.

4.3 Werkgeheugen

Voor moderne software is het “geheugen” (ook wel registers genoemd) van de processor zelf echter veel te klein. Daarom wordt in een computerarchitectuur steeds een werkgeheugen voorzien. Dit geheugen is uiterst snel zodat de processor niet te lang hoeft te wachten op de nodige instructies die in dit geheugen bewaard worden. Deze geheugens moeten zo snel zijn, dat het niet mogelijk is fysieke geheugens te gebruiken die ook zonder stroom hun gegevens bewaren. We noemen deze geheugens dan ook vluchtlage geheugens. Meer informatie over geheugens wordt gegeven in hoofdstuk 5.

4.4 Permanent geheugen

Wat nu indien we de stroom van een computer zouden uitzetten? Hiervoor worden de bestendige (of permanente) geheugens voorzien, die ook bij stroomuitval hun gegevens kunnen bewaren. De permanente geheugens worden besproken in hoofdstuk 8 en hoofdstuk 9.

4.5 In- en uitvoerapparaten (randapparatuur)

Wanneer we tenslotte willen interageren met de computer, moeten we op een of andere manier signalen kunnen uitwisselen met het systeem. Dit gebeurt aan de hand van in- en uitvoerapparaten. De in- en uitvoerapparaten staan uitvoerig besproken in hoofdstuk 11. De poorten die gebruikt worden om deze apparaten aan te sluiten worden besproken in hoofdstuk 10.

4.6 Chipsets en bussen

De “lijm” die deze verschillende onderdelen van een computersysteem tesamen brengt, noemen we bussen. Ook extra chips zullen nodig zijn om de onderlinge werking van deze onderdelen in goede banen te leiden. De chipsets en bussen worden besproken in hoofdstuk 7.

Hoofdstuk 5

Het werkgeheugen

Inhoudsopgave

5.1 Geheugenkarakteristieken	42
5.1.1 Geheugencapaciteit	42
5.1.2 Informatiedichtheid	42
5.1.3 Toegankelijkheid	42
5.1.4 Adresseerbaarheid	43
5.1.5 Bestendige en vluchtbare geheugens	43
5.2 Cache-geheugens	43
5.2.1 Wat is cache-geheugens?	43
5.2.2 Locality of reference	44
5.2.3 Level 1 cache geheugens	44
5.2.4 Level 2 cache geheugens	45
5.3 Werkgeheugen en soorten geheugens	45
5.3.1 Wat is werkgeheugen?	45
5.3.2 Statisch RAM	45
5.3.3 Dynamisch RAM	47
5.4 Soorten DRAM geheugens	47
5.4.1 Vergelijking SRAM-DRAM	51
5.4.2 De geheugenpiramide	51
5.4.3 Geheugenmodules	52
5.5 Nieuwigheden en speciale RAMs	53
5.5.1 Nieuwigheden	54
5.5.2 Speciale RAMs	54
5.6 Detectie en foutencontrole	55
5.6.1 Detectie	55
5.6.2 Foutencontrole	56

5.1 Geheugenkarakteristieken

In dit deel bekijken we een aantal geheugenkarakteristieken. Deze zijn belangrijk wanneer we de verschillende soorten geheugens zullen bespreken.

5.1.1 Geheugencapaciteit

Ruwweg gezegd geeft de geheugencapaciteit de hoeveelheid informatie weer die op een geheugen kan worden opgeslagen. Deze hoeveelheid kan worden uitgedrukt in aantal bits, bytes, woorden van 32 of 64 bits, ... Merk op dat een geheugen met 10000 woorden van 40 bits evenveel informatie kan opslaan als een geheugen met 12500 woorden van 32 bits. Er is echter een duidelijk verschil tussen beide. Tabel 5.1 geeft een aantal veel gebruikte eenheden weer die worden gebruikt in verband met geheugencapaciteit.

Afkorting	Hoeveelheid	Aantal bits/bytes
1 Kb	1 kilobit	1024 bits
1 Mb	1 megabit	1024^2 bits = 1024 Kb
1 Gb	1 gigabit	1024^3 bits = 1024 Mb
1 KB	1 kilobyte	1024 bytes
1 MB	1 megabyte	1024^2 bytes = 1024 KB
1 GB	1 gigabyte	1024^3 bytes = 1024 MB

Tabel 5.1: Vaak gebruikte eenheden bij geheugencapaciteit

5.1.2 Informatiedichtheid

De informatiedichtheid van een bepaald geheugen is het aantal bits dat dit geheugen kan opslaan op een bepaalde oppervlakte-eenheid. Een CD-ROM schijfje van 700 MB, met een totale oppervlakte van ongeveer 90 cm^2 , geeft een informatiedichtheid van bijna 8 MB/cm^2 .

5.1.3 Toegankelijkheid

De toegankelijkheid van een geheugen bepaalt op welke manier de bits uit het geheugen kunnen opgevraagd worden. We onderscheiden hier drie belangrijke vormen:

- **Onmiddellijke toegang:** de tijd die nodig is om een woord uit het geheugen op te roepen is praktisch onafhankelijk van de plaats waar het woord zich bevindt. Geheugens met onmiddellijke toegang zijn statisch van aard, in de zin dat ze geen bewegende onderdelen hebben (op macroscopische schaal). Een belangrijk voorbeeld voor geheugens met onmiddellijke toegang is RAM(Random Access Memory), dat voor cache geheugens en werkgeheugens wordt gebruikt.
- **Sequentiële toegang:** hier bevindt de informatie zich in een welbepaalde volgorde in het geheugen. Het prototype van de geheugens met sequentiële toegang is de magneetband. De band beweegt met een bepaalde snelheid, en de informatie wordt weggeschreven in een welbepaalde volgorde. De toegangstijd is dus in hoge mate afhankelijk van waar de gegevens op de magneetband staan.

- **Cyclische toegang:** geheugens met cyclische toegang bevatten meestal bewegende onderdelen(bv. harde schijven), maar zijn soms ook statisch(de oudere magneetbellengeheugens,...). Deze geheugens zijn quasi-onmiddellijk toegankelijk, maar er is soms toch nog een korte wachttijd(bv. bij een harde schijf wachten tot de schijf is rondgedraaid en de leeskop zich boven de juiste blok informatie bevindt).

5.1.4 Adresseerbaarheid

Ook op het gebied van adresseerbaarheid kunnen we geheugens in verschillende groepen verdelen. In de meeste moderne computers zijn de geheugencellen 1 byte groot. Men noemt deze computers dan ook byte-georiënteerd. Een geheugenregister van het werkgeheugen bestaat meestal uit meerdere geheugencellen. Het adres van een geheugenregister is dan meestal het adres van de eerste geheugencel waaruit het register bestaat. We kunnen dus zeggen dat de geheugencellen en geheugenregisters van het werkgeheugen **individueel adresseerbaar** zijn.

De geheugencellen van permanente geheugens maakt men meestal niet individueel adresseerbaar. De informatie wordt vanaf het werkgeheugen in grote blokken naar het permanente geheugen verstuurd, en omgekeerd, in grote blokken teruggehaald. Op deze manier probeert men de nadelen van het cyclische of sequentiële karakter van bepaalde geheugens te elimineren. De blokken informatie die worden uitgewisseld tussen deze geheugens en het werkgeheugen zijn meestal van voorafbepaalde lengte. Zulk een blok heeft bijvoorbeeld een capaciteit van 64 of 512 bytes. In dat geval zal elk blok over een adres beschikken, we spreken dan van **blokadresseerbare** geheugens.

Sequentiële geheugens tenslotte zijn meestal **niet-adresseerbaar**. De informatie wordt in grote blokken van willekeurige lengte op bv. een magneetband geschreven, met spaties tussen blokken. Door het aantal spaties te tellen kan een programma uitmaken met welk blok van gegevens het te maken heeft.

5.1.5 Bestendige en vluchtbare geheugens

Bestendige geheugens zijn geheugens waarvan de inhoud behouden blijft indien de computer uitgeschakeld wordt, of wanneer de stroom uitvalt. In het Engels spreekt men van non-volatile memories. Niet-bestendige of **vluchtbare** geheugens verliezen hun inhoud bij het uitvalLEN van de stroom. Het voordeel van deze geheugens is dat ze meestal zeer snel informatie kunnen opnemen en afgeven.

5.2 Cache-geheugens

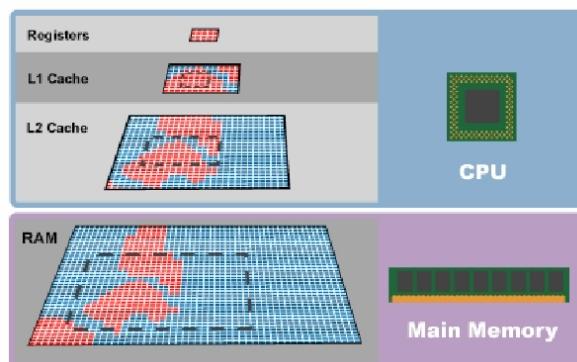
5.2.1 Wat is cache-geheugen?

De snelheid van de huidige processoren blijft verbazingwekkend snel stijgen. Ongeveer om de 2 jaar is er een verdubbeling van de snelheid. Om ervoor te zorgen dat de snellere processor genoeg informatie heeft om te verwerken, moet de snelheid van het geheugen mee stijgen. Het probleem is dat deze stijging veel rustiger verloopt. We zitten dus met een processor die gedurende een hele tijd werkloos is, doordat het werkgeheugen met lagere snelheid niet op tijd de nodige instructies of gegevens kan afleveren.

Om tegemoet te komen aan dit probleem, werden cache geheugens ingevoerd. Een cachegeheugen is een klein ultra-snel geheugen dat zal proberen tegemoet te komen aan de toekomstige aanvragen van de processor. Op deze manier hoeft de processor niet altijd het tragere werkgeheugen aan te spreken, zodat we een verbetering qua snelheid bekomen. We kunnen het cache geheugen bekijken als een soort buffer die de instructies/gegevens klaarthoudt voor uitvoering.

5.2.2 Locality of reference

Een belangrijk principe, waardoor cache-geheugens echt bijdragen tot een sneller systeem, is *locality of reference*. Dit principe zegt ons dat indien de processor recent een bepaalde locatie in het geheugen heeft gebruikt, de kans groot is dat in de nabije toekomst de processor een nabije locatie zal nodig hebben. Indien we uitgaan van dit principe kunnen we inzoomen op een bepaald gedeelte van het werkgeheugen, namelijk het gedeelte wat momenteel in uitvoering is. Wanneer we dit deel naar het cache-geheugen brengen, is de kans groot dat de processor de nodige informatie direct in het cache-geheugen zal vinden. Schematisch wordt dit weergegeven in figuur 5.1. We zien duidelijk dat het cache geheugen een kleiner gedeelte van het werkge-



Figuur 5.1: Principe van caching

heugen bevat. Op de figuur zijn twee caches te zien: level 1 en level 2. Zoals we in de volgende paragrafen zullen zien, worden tegenwoordig twee niveaus van caching gebruikt, om nog meer processortijd te kunnen benutten.

5.2.3 Level 1 cache geheugen

Het level 1 cache geheugen zit op de processor. Het zorgt voor een tijdelijke opslag van veel gebruikte gegevens (of waarvan verwacht wordt dat ze in de nabije toekomst zullen gebruikt worden) en instructies. Deze informatie wordt opgeslagen in blokken van 32 bytes. Level 1 cache geheugen wordt ook wel “primary cache” genoemd. Cache geheugens worden meestal gebouwd door middel van statisch RAM geheugen(SRAM, zie 5.3). Gedurende lange tijd was het level 1 cache 16 KB groot, maar op de nieuwste processoren is dit opgelopen tot 128 KB. Het level 1 cache geheugen heeft een “zero wait state”, wat wil zeggen dat de wachttijd verwaarloosbaar klein wordt. Dit heeft als gevolg dat het geheugen niet heel groot kan zijn; dan zou immers meer tijd moeten gestoken worden in het decoderen van het adres.

Voor het wegschrijven van gegevens en instructies werd lang een “write through” strategie gevuld. Hierbij werd alles wat in het cache geheugen zat ook volledig in het werkgeheugen bewaard. Tegenwoordig gebruikt men eerder “write-back”, hetgeen wil zeggen dat gegevens enkel in het cache geheugen zitten. Ze worden enkel naar het werkgeheugen verplaatst indien ze in een stuk van het cache zitten dat moet vervangen worden door andere gegevens of instructies. Door deze nieuwe strategie te volgen kan tot 10% aan performantie gewonnen worden, maar door de complexere boekhouding wordt het geheel ook duurder.

5.2.4 Level 2 cache geheugen

Level 2 cache geheugens, ook wel “secondary cache” genoemd, zijn een soort brug tussen het werkgeheugen en het level 1 cache. Dit level 2 cache zit ook op de processor. Het level 2 cache is meestal groter dan level 1 cache, maar is ook iets trager. Er wordt gebruik gemaakt van een speciale bus(de communicatielijn tussen processor en cache) om hoge snelheden te halen. Op de moderne processoren zit ook level 2 cache geheugen ingebouwd, dat meestal 512 KB groot is. Voor de ontwikkeling van level 2 cache geheugens wordt gebruik gemaakt van SRAM-chips.

Bij de eerste computers werd enkel een beperkt L1 cache geheugen gebruikt. De evolutie tot 2 cache levels wordt weergegeven in tabel [5.2](#).

5.3 Werkgeheugen en soorten geheugens

5.3.1 Wat is werkgeheugen?

Het werkgeheugen is als het ware de brug tussen de processor(eigenlijk het level 2 cache) en de harde schijf. Het verschil in snelheid tussen het werkgeheugen en de harde schijf is enorm, en daardoor zorgt meer werkgeheugen voor een groot verschil in de performantie van een systeem. De communicatie tussen het werkgeheugen en de processor gebeurt door middel van adres- en databussen. De adresbus wordt gebruikt om informatie door te geven in verband met adresering, de databus is de vervoerband voor de gegevens zelf.

De breedte van de adresbus bepaalt hoeveel geheugen er adreseerbaar is: hoe meer bits, hoe groter de adressen en hoe meer geheugen we kunnen aanspreken. In moderne systemen is de adresbus 36 bits breed, wat toelaat om 64 G geheugenplaatsen aan te spreken(indien elke adreseerbare geheugenplaats 1 byte groot is, verkrijgen we dus maximaal 64 GB werkgeheugen). De breedte van de databus bepaalt hoeveel gegevens in 1 bus cyclus kunnen worden verplaatst. Deze bus is tegenwoordig 64 bits breed, wat toelaat 8 bytes in 1 bus cyclus over te brengen van of naar het werkgeheugen. Een bus cyclus is een transactie tussen de processor en het werkgeheugen.

Voor werkgeheugen wordt vooral gebruik gemaakt van DRAM geheugenchips.

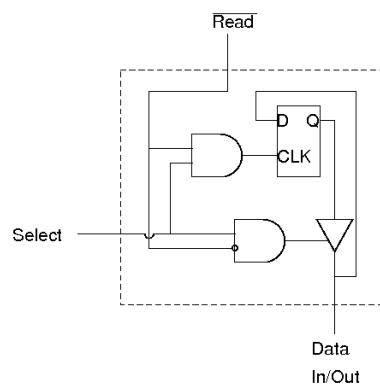
5.3.2 Statisch RAM

Statische RAM geheugens zijn opgebouwd uit bipolaire transistoren(4 tot 6 transistoren per opgeslagen bit). Ze behouden hun informatie zolang ze stroom krijgen. Een voorbeeld van een statische geheugencel opgebouwd uit een D-flipflop wordt getoond in figuur [5.2](#). SRAM

CVE	Caches
80486DX and DX2	8 KB L1
80486DX4	16 KB L1
Pentium	16 KB L1
Pentium Pro	16 KB L1 + 256 KB L2(sommige 512 KB L2)
Pentium MMX	32 KB L1
AMD K6 and K6-2	64 KB L1
Pentium II and III	32 KB L1
Celeron	32 KB L1 + 128 KB L2
Pentium III Cumine	32 KB L1 + 256 KB L2
AMD K6-3	64 KB L1 + 256 KB L2
AMD K7 Athlon	128 KB L1
AMD Duron	128 KB L1 + 64 KB L2
AMD Athlon Thunderbird	128 KB L1 + 256 KB L2
AMD Athlon X2 Dual-Core	128 KB L1 + 1 MB L2
AMD Phenom X4 Quad-Core (model 9850)	128 KB L1 + 512 KB L2 (x4) + 2 MB L3
AMD Phenom II X6 (model 1090T)	128 KB L1 + 512 KB L2 (x4) + 6 MB L3
Intel Core2 Duo E8600	6 MB L2
Intel Core2 Extreme QX9775	12 MB L2
Intel Core2 Quad Q9650	12 MB L2
Intel Pentium Extreme Edition 965	2x2 MB L2
Intel Pentium Dual-Core E5200	2 MB L2
Intel Core i3-550	4 MB L2
Intel Core i5-750	8 MB L2
Intel Core i7-970	12 MB L2

Tabel 5.2: Evolutie van cache geheugens

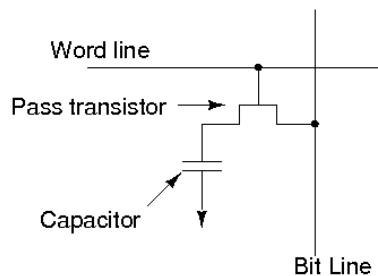
geheugen is ontzettend snel, maar ook heel duur. Door de snelheid en de beperkte grootte(door kost en om snelheid hoog te houden) wordt SRAM vooral gebruikt in cache-geheugens.



Figuur 5.2: SRAM geheugencel opgebouwd uit D-flipflop

5.3.3 Dynamisch RAM

In tegenstelling tot statisch RAM, bestaat dynamisch RAM geheugen uit condensators. Om de gegevens te blijven bewaren, moet deze condensator vele keren per seconde worden gerefreshed. Een condensator kan worden vergeleken met een soort batterij die voor een korte tijd stroom kan opslaan en weer afgeeft. Omdat de condensator zo klein is, verliest hij snel zijn lading; hij moet heropgeladen worden. Een geladen condensator zal een 1-bit voorstellen, een ontladen condensator stelt een 0-bit voor. DRAM gebruikt per opgeslagen bit 1 transistor en 1 condensator, wat de prijs enorm laag houdt tegenover SRAM. Het nadeel van DRAM is dat het trager is dan SRAM, doordat het refreshen de werking van de chips vertraagt. DRAM is ongeveer 6x trager dan SRAM, en wordt vanwege zijn lagere prijs vooral gebruikt voor het werkgeheugen. Figuur 5.3 toont hoe een DRAM-chip eruit ziet. De woord- en bitlijn worden gebruikt om een bepaalde bit te selecteren, de condensator(Engels: capacitor) zorgt voor de opslag van de bit.



Figuur 5.3: DRAM geheugencel opgebouwd uit een condensator

5.4 Soorten DRAM geheugens

We geven nu een overzicht van de verschillende soorten DRAM-geheugens die in de loop der jaren werden ontwikkeld. Deze geheugens worden vooral gebruikt voor werkgeheugen, aangezien SRAM-geheugens hiervoor meestal te duur is.

Eerst kort iets over snelheden die we bij de DRAM-types zullen tegenkomen. Wanneer een bepaald DRAM-geheugen kan werken aan 100 MHz, bedoelen we dat er 100 miljoen clock cycles per seconde kunnen plaatsvinden(1Hz is dus 1 cycle per seconde). Dit zal mee bepalen hoeveel gegevens het geheugen in 1 seconde kan verplaatsen naar de microprocessor.

Fast Page Mode DRAM(FPM DRAM)

FPM RAM is iets sneller dan het “gewone” DRAM. Het is het eerste type DRAM dat op commerciële computersystemen terug te vinden was. Het werd algemeen ondersteund door de verschillende moederbordvarianten, en had daardoor een groot(maar kort...) succes. Het duurde niet lang vooraleer er een verbeterde versie van FPM DRAM verscheen: EDO DRAM.

Extended Data Out DRAM(EDO DRAM)

Eind jaren '90 werd FPM DRAM geheugen als snel vervangend or EDO RAM. Bij EDO DRAM-geheugens is de data reeds toegankelijk terwijl de vorige toegang nog bezig is. Vooral wanneer het rijnummer hetzelfde is van de gevraagde data, kan EDO RAM deze gegevens veel sneller

ophalen. Hierdoor kan men een verbetering van ongeveer 30% op FPM DRAM halen. Het geheugen werkt op een maximale bussnelheid van 66 MHz. Door deze beperking werd ook EDO DRAM al gauw vervangen door nieuwkomers die wel hogere bussnelheden aankonden. EDO DRAM werd ook minder goed ondersteund dan FPM.

Burst Extended Data Out DRAM(BEDO DRAM)

Burst EDO DRAM is een verbetering(een “hack”) tegenover EDO DRAM, doordat men gecombineerde pipeline-technologiën heeft toegevoegd. Dit wil zeggen dat bij een data-transfer naar het geheugen, na de eerste adressering, de volgende adressen intern werden gegenereerd zodat er geen tijd meer verspild werd aan het ingeven van deze adressen. Deze technologie zorgde voor een serieuze verbetering in de snelheid, maar nog steeds kon nog niet worden gewerkt met bussnelheden boven de 66 MHz.

Eigenlijk was BEDO DRAM een poging om EDO DRAM te laten concurreren met SDRAM, maar BEDO DRAM verloor al snel de strijd. Dit vooral door politieke en economische redenen: veel producenten hadden al veel tijd en geld in SDRAM geïnvesteerd, en natuurlijk wilden ze er ook iets uithalen... Intel besliste(omdat ook zij mee in het onderzoek van SDRAM zaten) dat ze het BEDO DRAM geheugen niet meer gingen ondersteunen op hun hardware. Alhoewel BEDO DRAM bij snelheden onder de 100 MHz stabiever en sneller was dan SDRAM, is deze technologie dus een stille dood gestorven.

Synchronous DRAM(SDRAM)

Met synchroon DRAM bedoelt men dat dit soort RAM-geheugen aan de snelheid van de bus werkt(dus synchroon met de bus). Indien de bus aan 100 MHz werkt, zal ook het geheugen mee aan 100 Mhz draaien. Indien men een tragere bus van 66 MHz gebruikt, worden de mogelijkheden van SDRAM dus niet ten volle benut. Al gauw werden drie verschillende types SDRAM op de markt gebracht: PC66, PC100 en PC133. PC66 haalde het dubbele van de snelheid van EDO RAM. SDRAM buit het feit uit dat de meeste geheugentoegangen sequentieel zijn. Het zorgt ervoor dat geheugentoegangen met opeenvolgende adressen zeer snel kunnen uitgevoerd worden.

Enhanced Synchronous DRAM(ESDRAM)

ESDRAM is weer een soort “hack” op SDRAM. Bij ESDRAM wordt een kleine SRAM chip ingeplant, die als cache-geheugen voor de ESDRAM-chip wordt gebruikt. Meestal zal de nodige informatie zich in de SRAM-chip bevinden(zie gedeelte over caching), indien niet wordt verder in de chip gekeken. Door ESDRAM was het mogelijk om bursts tot 200 MHz te verkrijgen, een serieuze verbetering tegenover de vorige technologieën. ESDRAM wordt vooral gebruikt om L2-cache geheugens te implementeren. Voor het werkgeheugen heeft ESDRAM het moeten opgeven tegen zijn grote concurrent: DDR DRAM.

Double Data Rate RAM

DDR DRAM is een ander voorbeeld van synchroon DRAM, en werkt dus ook aan de snelheid van de bus. Traditioneel wordt op een bus enkel een geheugenoperatie uitgevoerd wanneer de klok omhoog gaat(dus van 0 naar 1). Bij DDR DRAM zijn operaties mogelijk zowel bij het

verspringen van de klok van 0 naar 1 als van 1 naar 0. Hierdoor krijgt men een verdubbeling van de kloksnelheid, zonder de kloksnelheid zelf aan te passen. Aangezien er telkens 64 bits tegelijk worden overgedragen, kan de snelheid van DDR geheugen berekend worden aan de hand van volgende formule:

$$\text{Snelheid} = \text{kloksnelheid} \times 2(\text{DDR}) \times 64(\text{aantal overgebrachte bits}) / 8(8 \text{ bits per byte})$$

Zo krijgen we bijvoorbeeld voor 1600 Mhz DDR RAM geheugen:

$$\text{Snelheid} = 100\text{Mhz} \times 2 \times 64/8 = 1600\text{MBps}$$

DDR DRAM werd in 1999 geïntroduceerd door NVIDIA in de geforce grafische kaarten. Intel bleef zich vastpinnen op SDRAM, omdat ze hierop meer verdienden. Producent AMD schakelde echter direct over door zijn Socket A-moederborden te laten werken met DDR DRAM. Het jaar 2000 werd voor AMD zeer winstgevend omwille van de introductie van DDR DRAM op zijn moederborden. Intel kon niet anders dan ook overschakelen naar DDR DRAM, alhoewel dit pas begin 2002 gebeurd is(en met tegenzin).

DDR DRAM geheugenchips worden ook vandaag(anno 2012) nog het meeste gebruikt, in de vorm van DDR3 (zie verder) en meestal in een dual of triple channel configuratie. De verschillende soorten DDR RAM worden benoemd door de snelheid die ze behalen. DDR266 werkt op een 133 MHz bus, maar door zowel bij 0 → 1 als bij 1 → 0 een operatie uit te voeren werkt dit toch aan 266 MHz. Dit wil zeggen dat er 266 MT/s(Miljoen Transfers per Seconde) plaatsvinden. Analoog werkt DDR333 op een 166 MHz bus, aan een snelheid van 333 MHz. Het snelste DDR dat ooit op de markt werd gebracht is DDR625.

Naast deze naamgeving wordt ook dikwijls het aantal MB dat het geheugen per seconde kan wegschrijven of uitlezen vermeld. Zo is PC1600 de DDR versie van PC100 SDRAM. Het maakt gebruik van DDR200-chips en kan 1600 MBps aan. Volgende berekening maakt een en ander duidelijker:

$$8 \text{ bytes} \times 200\text{Mhz} = 1.6 \text{ GBps}$$

We werken met 8 bytes aangezien de databus 8 bytes(64 bits) breed is. Dit wil zeggen dat er 8 bytes tegelijkertijd over de bus kunnen verstuurd worden. Analoog voor PC2700 vinden we:

$$8 \text{ bytes} \times 333\text{Mhz} = 2.7 \text{ GBps}$$

Dus PC2700 werkt op een 333 MHz bus. PC8000 werkt op een 1000 MHz bus.

Double Data Rate RAM 2 (DDR2) en 3 (DDR3)

Op moderne moederborden is het enkel mogelijk DDR2 of DDR3 geheugen te plaatsen. Zoals elke SDRAM implementatie zal DDR2 data opslaan in geheugencellen die op basis van een externe bus worden geklokt. DDR2 zal zoals DDR zowel bij het op- als het neergaan van de klok data doorsturen. Het belangrijke verschil met DDR is dat bij DDR2 de kloksnelheid van de bus dubbel zo groot is als de kloksnelheid van het geheugen zelf. Zo kunnen 4 data-eenheden doorgestuurd worden per cycle. De busfrequentie van DDR2 wordt verhoogd door elekrotechnische verbeteringen, on-die termination, prefetch-buffers en off-chip drivers. Ondanks deze verbeteringen is de latency groter bij DDR2. DDR heeft typisch read latencies van 2 tot 3 bus

cycles. DDR2 heeft latencies tussen 3 en 9 cycles. Daarom zal DDR beter zijn dan DDR2 op dezelfde bussnelheid. DDR2 kan echter op hogere bussnelheden draaien, waardoor het wel degelijk sneller is dan DDR.

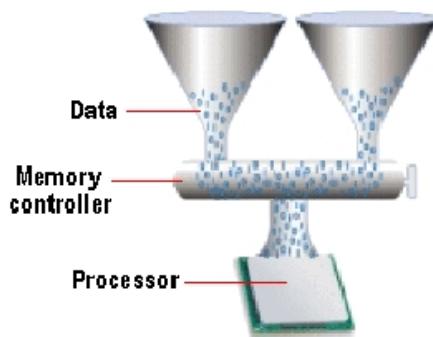
Bij DDR3 wordt nogmaals hetzelfde principe toegepast: tegenover DDR2 werkt DDR3 dubbel zo snel. Eigenlijk verbetert DDR3 enkel de interface naar het geheugen, de fysieke opbouw en eigenschappen van het geheugen zelf zijn dezelfde als die van DDR2. Ondertussen is DDR3 de standaard op het gebied van RAM-geheugens: alle moderne moederborden ondersteunen DDR3. Op grafische kaarten van NVidia en ATI wordt al jarenlang GDDR3 geheugen gebruikt. Dit is een geheugen voor grafische kaarten dat -in tegenstelling tot GDDR2- weinig of geen overeenkomsten heeft met DDR3 geheugen. Vandaag de dag komt GDDR3-geheugen vooral voor op de high-end grafische kaarten, de goedkopere grafische kaarten gebruiken de grafische variant van het minder performante DDR2.

Dual en triple channel

Dual Channel is een techniek die sinds 2003 op bijna alle moederborden wordt aangeboden ter verbetering van de snelheid van DDR geheugen. Dual Channel wordt toegepast op DDR, DDR2 en DDR3 geheugenmodules. Het is geen verbetering in het DRAM geheugen zelf, maar een toevoeging op het moederbord zodat twee individuele DRAM geheugenlatjes tegelijkertijd kunnen aangesproken worden. Dit geeft een verbetering van ongeveer 10%. Er moet wel bij vermeld worden dat Dual Channel in sommige gevallen enkel werkt indien:

- De DIMMs(zie [5.4.3](#)) in paren geïnstalleerd worden
- De DIMMs dezelfde geheugencapaciteit hebben
- De DIMMs dezelfde busbreedte hebben(8 bytes)
- De DIMMs beide single of beide double sided zijn(zie verder)

Schematisch wordt het dual channel principe getoond in figuur [5.4](#). Volgens Tom's Hardware,



Figuur 5.4: De Dual Channel techniek

een gerespecteerde website met tests en benchmarks, is er bijna tot geen verschil tussen single en dual channel. Dual channel haalt in zeer specifieke benchmarks tot 5% snellere resultaten, wat bijna verwarring veroorzaakt voor de gewone gebruiker. Een oudere studie van laptoplogic.com gaf een snelheidsverbetering tot 15% aan. Feit is dat de dual channel technologie zwaar

overschat wordt en ontrecht als marketing-middel wordt gebruikt voor het aanprijsen van computers of moederborden. De nieuwste technologie is triple channel, waarbij 3 geheugenlatjes tegelijk aangesproken kunnen worden. Triple channel wordt onder andere ondersteund op de Intel Core i7-9xx Bloomfield en Gulftown platformen.

5.4.1 Vergelijking SRAM-DRAM

We kunnen nu een vergelijking tussen SRAM en DRAM geheugens uiteenzetten. Tabel 5.3 geeft een overzicht van een aantal karakteristieken. Wat we uit deze tabel kunnen besluiten is

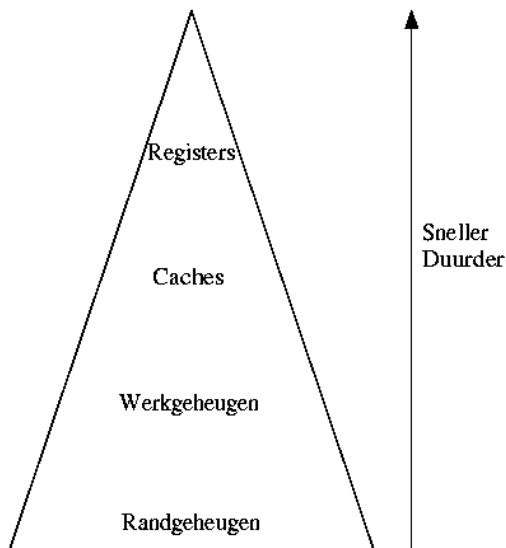
SRAM	DRAM
Snel	Traag
Groot	Klein
Duur	Goedkoop
Meer energieverbruik	Minder energieverbruik

Tabel 5.3: Een vergelijking tussen DRAM en SRAM

dat SRAM vooral voor cache geheugens en ultrasnelle registers gebruikt wordt (registers worden ook vaak door nog snellere geheugens geïmplementeerd). DRAM geheugens worden vooral voor het werkgeheugen van een computer gebruikt.

5.4.2 De geheugenpiramide

Aan de hand van voorgaande karakteristieken van geheugens wordt dikwijls een geheugenpiramide opgesteld. Een geheugenpiramide geeft weer hoe verschillende soorten geheugens zich tegenover elkaar afspiegelen qua grootte, prijs en snelheid. Figuur 5.5 geeft zo'n geheugenpiramide weer. In de geheugenpiramide zien we helemaal onderaan de goedkoopste maar tevens



Figuur 5.5: Geheugenpiramide

traagste geheugens. Daarbij horen onder andere de magneetschijfgeheugens(harde schijf), magneetbanden,... Een heel stuk sneller zijn de werkgeheugens, bijvoorbeeld DRAM geheugen. Dan komen we aan ultrasnelle cache geheugens, die meestal uit SRAM-cellen worden opgebouwd. Tenslotte zullen voor de registers in de microprocessor de snelste geheugensoorten gebruikt worden. Meer informatie over deze verschillende geheugens wordt getoond in tabel 5.4.

Level	Toegangstijd	Typische grootte	Technologie
Registers	1 - 3 ns	1 KB	CMOS
Level 1 cache	2 - 8 ns	8 KB - 128 KB	SRAM
Level 2 cache	5 - 12 ns	512 KB - 8 MB	SRAM
Werkgeheugens	5 - 70 ns	1 GB - 4 GB	DRAM
Harde schijf	3 000 000 - 30 000 000 ns	1 - 3 TB	Magnetisch

Tabel 5.4: Verschillende geheugentypes en hun kenmerken

5.4.3 Geheugenmodules

De geheugenchips die we tot nu toe gezien hebben zijn nog niet geschikt om op het moederbord van een computer te plaatsen. In realiteit wordt gebruik gemaakt van PCB(printed carton boards) om individuele geheugenchips op te plaatsen. Deze PCB's of geheugenmodules worden daarna op het moederbord geplaatst.

SIMM

De SIMMs(Single In-line Memory Module) zijn de eerste modules die voor de computer verschenen. Ze waren een basis verpakking voor de DRAM chips, en werden vooral gebruikt voor oudere systemen. Hun geheugencapaciteit was vrij beperkt: maximaal 128 MB op 1 module. Deze SIMM-modules konden single-sided of double-sided geproduceerd worden(double-sided wil zeggen dat aan beide kanten van de module chips voorkomen).

Deze SIMMs vormden het eerste aanpasbare systeem(modules konden worden toegevoegd en verwijderd) voor de 386, 486 en apple computers. De modules hadden 30 pins, maar later is men overgestapt naar 72 pins SIMMs. De 72 pins modules werden voor 486, 586 en Pentium varianten gebruikt, en konden maximaal 256 MB aan geheugen bevatten.

Vanaf de Pentium systemen werd de busbreedte tussen de processor en het geheugen verdubbeld van 32 bits naar 64 bits. De SIMMs die slechts een breedte hadden van 32 bits moesten dus aangepast worden... Hierop vond men echter het volgende: men plaatste de SIMMs steeds in paren, en liet zo'n paar van SIMM modules samenwerken om toch de 64 bits te verkrijgen. Deze SIMMs moesten dus altijd in paren geplaatst worden om te kunnen gebruikt worden door de Pentium systemen. Figuur 5.6 toont hoe een 72-pins SIMM eruit ziet.

DIMM

De opvolger van SIMM is DIMM(Dual In-line Memory Module). De 64 bits brede bus was een standaard geworden en daarom wordt bij DIMM ook 64 bits breed gewerkt. SIMM verloor meer en meer aan belangstelling door zijn beperking omtrent de busbreedte. Het voordeel bij DIMMs



Figuur 5.6: Een 72-pins SIMM

is dat deze niet meer in paren geïnstalleerd hoeven geïnstalleerd te worden. Ze bevatten meer pinnen: een DIMM heeft meestal 168 of meer pinnen. Bij SIMM modules zijn de elektrische contacten aan beide zijden van de module doorverbonden, bij DIMM modules worden deze gescheiden gebruikt. Hieronder worden de belangrijkste DIMM-varianten opgesomd:

- 72-pins SO-DIMM (niet hetzelfde als de 72-pin SIMM), gebruikt voor FPM DRAM en EDO DRAM
- 100-pins DIMM, gebruikt voor printer SDRAM
- 144-pins SO-DIMM, gebruikt voor SDR SDRAM
- 168-pins DIMM, gebruikt voor SDR SDRAM (en uitzonderlijk ook voor FPM/EDO DRAM in workstations/servers)
- 172-pins MicroDIMM, gebruikt voor DDR SDRAM
- 184-pins DIMM, gebruikt voor DDR SDRAM
- 200-pins SO-DIMM, gebruikt voor DDR SDRAM en DDR2 SDRAM
- 204-pins SO-DIMM, gebruikt voor DDR3 SDRAM
- 214-pins MicroDIMM, gebruikt voor DDR2 SDRAM
- 240-pins DIMM, gebruikt voor DDR2 SDRAM, DDR3 SDRAM en FB-DIMM DRAM

RIMM

De RIMM geheugenmodules zijn een speciale soort geheugenmodules ontworpen voor het DRDRAM geheugentype. Zij hebben 184 pins.

5.5 Nieuwigheden en speciale RAMs

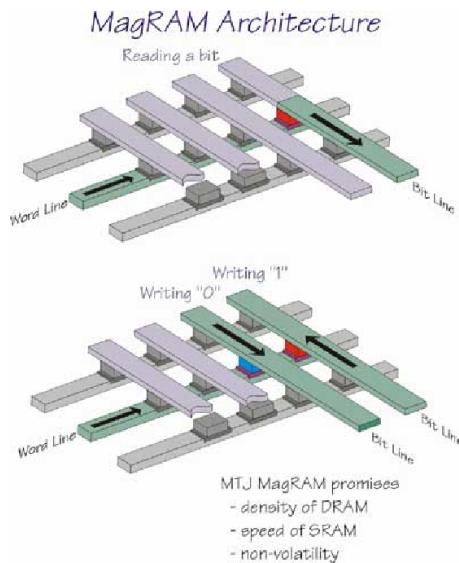
In dit gedeelte zullen we enkele nieuwigheden op het gebied van werkgeheugens bespreken, en ook een kort overzicht geven van speciale RAM-geheugens.

5.5.1 Nieuwigheden

Magnetoresistive RAM(MRAM)

Iets wat mensen altijd blijft storen is dat het soms lang kan duren voor een computer is opgestart. Magnetoresistive RAM wil aan dit euvel voorkomen. Magnetoresistive RAM behoudt zijn gegevens, ook als de stroom wordt uitgezet(en is dus een niet-vluchttig geheugen). Wanneer men zulke soort RAM-geheugens zou gebruiken voor het werkgeheugen, is een boot-procedure nagenoeg niet meer nodig. Ook voor gsm's en videospelletjes zou het een hele stap vooruit zijn.

Magnetoresistive RAM gebruikt -zoals de naam al verklapt- magnetisme in plaats van elektrische stroom om zijn gegevens op te slaan. Zo zal een 1 bit worden opgeslagen als een bepaalde magnetische lading tussen twee metalen lagen. Figuur 5.7 toont hoe MRAM qua architectuur in elkaar zit. MRAM is dus een kandidaat om de huidige DDR DRAM geheugenchips in de



Figuur 5.7: Magnetoresistive RAM geheugen

toekomst te gaan vervangen. In oktober 2008 werd een Japanse satelliet gelanceerd die als eerste MRAM gebruikte. Het voordeel in deze situatie is dat MRAM minder onderhevig is aan hoge temperaturen en stralingen in de atmosfeer. Momenteel werken heel wat firma's verder aan de ontwikkeling van MRAM. De kans is dus groot dat het in de toekomst effectief gebruikt zal worden in dagdagelijkse toestellen zoals GSM's, digitale camera's, laptops en ook gewone thuiscomputers. Ondanks de grote interesse in de ontwikkeling (en onderzoek) van MRAM is er nog steeds geen bedrijf dat uitsluitend MRAM chips produceert in hun fabrieken. De reden hiervoor is dat geheugen-producerende bedrijven op veilig willen spelen en dus de normale DRAM-markt voorzien van geheugens. Het is dus nog even wachten op een vendor die de sprong waagt en een fabriek opstart voor MRAM.

5.5.2 Speciale RAMs

Alhoewel de volgende RAM-geheugens bijna niet meer gebruikt worden, is het om historische redenen toch interessant om ze even te vermelden.

Video RAM(VRAM)

VRAM of Video RAM werd speciaal voor videokaarten ontwikkeld. Het voordeel aan dit soort RAM-geheugen is dat twee apparaten tegelijkertijd het RAM-geheugen kunnen gebruiken. Dit laat toe om zowel het beeldscherm de huidige gegevens te tonen, terwijl er nieuwe in worden geplaatst.

Windows RAM(RAM)

WRAM werd ontworpen door Samsung Electronics, en is een soort verbeterde versie van VRAM. Het geheugen wordt nu in grotere blokken geadresseerd(typisch een gebruik voor grafische toepassingen), windows genoemd. Dit soort RAM geheugen is ongeveer 25% sneller dan VRAM. Het werd gebruikt op de legendarische Matrox Millenium en ATI 3D Rage Pro grafische kaarten.

Synchronous Graphics RAM(SGRAM)

SGRAM is de laatste RAM-variant die speciaal voor grafische kaarten werd ontwikkeld. Zoals SDRAM werkt SGRAM synchroon, dus aan de snelheid van de bus. Door gebruik te maken van speciale masked writes, kon geselecteerde data in één keer aangepast worden. Ook block writes, waarbij bijvoorbeeld een heel gedeelte van het geheugen werd gevuld met een bepaalde opvulkleur voor het scherm, zonder dat naar de individuele delen moest gescheven worden, maken SGRAM speciaal. SGRAM maakt gebruik van single-ported geheugen: hierbij is er slechts 1 toegangspoort tot het geheugen. Dual-ported wordt gebruikt door VRAM en WRAM, waarbij er twee apparaten tegelijkertijd het geheugen kunnen gebruiken. Doordat geheugens sneller en sneller zijn geworden, is dit echter niet meer nodig, en zit de data op tijd klaar om uitgelezen te worden door het scherm. SGRAM is tevens sneller dan VRAM en WRAM. Tegenwoordig wordt in de grafische kaarten het DDR DRAM geheugen gebruikt.

5.6 Detectie en foutencontrole

5.6.1 Detectie

Tijdens het opstarten van een computer moet het aanwezige geheugen herkend worden, om het te kunnen gebruiken. Hiervoor werd lange tijd PPD of het Parallel Presence Detect systeem gebruikt. Daarbij werden enkele bits aan een geheugen toegevoegd waaruit men kon aflezen welk soort geheugen het was, welke snelheden het kon halen, enz. Vooral voor de eerste SIMMs en DIMMs werd PPD veel gebruikt. Later bleek echter dat PPD niet flexibel genoeg was om ook nieuwere geheugentypes te ondersteunen.

De opvolger van PPD is SPD of Serial Presence Detect, wat sinds de introductie van DDR DRAM tot op vandaag gebruikt wordt. Het maakt gebruik van een 8-bits EEPROM(zie [9.2.4](#)) chip op de geheugenmodule, waarin informatie over het type, snelheid, grootte en aan rijen/-kolommen van de geheugenmodule wordt opgeslagen.

5.6.2 Foutencontrole

Dikwijls voegt men aan geheugenchips extra bits toe om een zekere vorm van controle in te kunnen voeren op de juistheid van de opgeslagen informatie. Zo kan bijvoorbeeld een 9de bit worden opgeslagen voor elke 8 bits. Deze bit is dan bijvoorbeeld de even of oneven pariteitsbit voor die 8 bits. Indien er een fout wordt opgemerkt tijdens de pariteitscontrole, is er maar 1 oplossing: een shutdown van de machine...

Het spreekt voor zich dat bij betrouwbare systemen (zoals computersystemen voor o.a. kernreactoren, of ook gewone servers in bedrijven die (bijna) niet mogen heropgestart worden) zulk een systeem niet gewenst is.

Error Correcting Code (ECC)

Op betrouwbare systemen wordt dikwijls gebruik gemaakt van speciale algoritmes die fouten in de opgeslagen informatie kunnen ontdekken en verbeteren. Deze algoritmes noemt men ECCs. Er zal ook hier weer extra informatie worden opgeslagen:

- 5 bits voor een byte
- 6 bits voor 2 bytes
- 7 bits voor 3 bytes
- ...

Door het opslaan van extra bits kunnen fouten van 1 bit gecorrigeerd worden(bits kunnen bijvoorbeeld omslaan naar de inverse door een elektrische interferentie). Fouten van meerdere bits kunnen niet of eventueel via ingewikkeldere algoritmes(die ook meerdere extra bits gebruiken) verbeterd worden. ECC is een dure oplossing, en wordt dan ook alleen maar gebruikt op uiterst betrouwbare servers en speciale systemen.

Hoofdstuk 6

Processoren

Inhoudsopgave

6.1 Geschiedenis	58
6.1.1 Eerste processoren	58
6.1.2 32 vs 64 bits?	62
6.1.3 Dual-core en quad-core	63
6.2 Bouw van een processor	63
6.2.1 Onderdelen	63
6.2.2 Transistor density	64
6.2.3 Koeling	64
6.2.4 Van 10 µm naar 11nm technologie	65
6.3 Werking	66
6.3.1 Processorcyclus	66
6.3.2 Kloksnelheden	66
6.3.3 Buffers en cachegeheugen	68
6.3.4 Toekomstvoorspelling	68
6.3.5 Instruction sets	68
6.4 Registers	69

Een processor is in feite het brein van een computer; hij verwerkt opeenvolgende bevelen. Al deze bevelen vormen een programma (zie hoofdstuk 12). Zo'n programma bestaat uit machinetaal (een taal die de processor verstaat) en bevindt zich tijdens de uitvoering in het centraal geheugen. Daarvoor werd het eerst “geladen” vanuit de harde schijf, CD, tape, diskette, ... Pas dan kan het uitgevoerd worden.

Hierbij moet opgemerkt worden dat elke processor zijn eigen machinetaal heeft, niet elke processor spreekt dezelfde taal en bezit dezelfde mogelijkheden. Een processor wordt ook wel CPU (central processing unit) of CVE (centrale verwerkingseenheid) genoemd. We gebruiken vanaf nu regelmatig deze afkortingen.

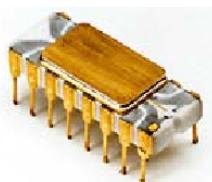
De eerste CPU's werden gebouwd met behulp van elektronenbuizen. Later gebruikte men

transistoren, vervolgens IC's (Integrated Circuits) en uiteindelijk verscheen de CPU op één enkele siliciumchip. De ontwikkeling van steeds nieuwe CPU's heeft geleid tot verschillende CPU-families.

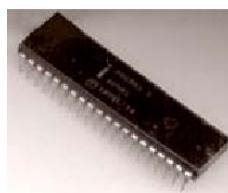
6.1 Geschiedenis

6.1.1 Eerste processoren

In 1971 kwam Intel met een CPU die gebouwd was op 1 chip, de INTEL 4004. Deze CPU had een kloksnelheid van 108 KHz en was hiermee de eerste commerciële microprocessor. Later volgde de INTEL 8080 en de zeer bekende en populaire INTEL 8086 processoren. Deze eerste processoren waren zeer beperkt qua mogelijkheden (de taal die ze spraken was eenvoudig). Figuur 6.1 en figuur 6.2 tonen deze INTEL 4004 en INTEL 8080 chips.



Figuur 6.1: De INTEL 4004 processor



Figuur 6.2: De INTEL 8080 processor

Ook andere fabrikanten kwamen met CPU's op de markt, zoals Motorola met de MOTOROLA 6800, die als microprocessor werd gebruikt voor de Macintosh computers van Apple.

Een overzicht van de belangrijkste processoren wordt getoond in tabel 6.1.

Tabel 6.1: Overzicht van belangrijkste processoren.

Jaar	Processor	Informatie
1971	Intel 4004	Deze 4-bit processor liet toe om met 4-bits binaire getallen te werken en ook 4-bits data te versturen naar andere componenten op het moederbord.
1974	Intel 8008	Deze 8-bit processor communiceerde met andere hardware over een 4-bits brede verbinding. Hij haalde een kloksnelheid van 0.74 MHz en kon slechts 1 KB RAM-geheugen adresseren (wat in die tijd enorm veel was).
Vervolg op volgende pagina		

Tabel 6.1 – vervolg van vorige pagina

Jaar	Processor	Informatie
1980	Intel 80186	De 80186 was een zeer populaire processor; veel verschillende versies werden hiervan ontwikkeld. De kopers konden kiezen tussen CHMOS en HMOS, 8-bit of 16-bit versies. De CHMOS chip kon op twee keer zo snelle kloksnelheid lopen van de HMOS chip.
1981	NEC V20 en V30	Deze processoren waren klonen van de 8088 en 8086. Ze werkten ongeveer 30% sneller dan de Intel chips.
1982	Intel 80286	Een 16-bits processor die tot 16 MB RAM-geheugen kon aanspreken. Deze processor introduceerde als eerste de “protected mode”.
1988	Intel 386	Deze 32-bits processor was een ware revelatie in de processorgeschiedenis. Hij kon 4 GB aan werkgeheugen aanspreken, wat een enorme verbetering was. Deze processor werd tevens vergezeld van een coprocessor: de 80387. De 80386 werkte op kloksnelheden van 12.5 MHz tot 33 MHz.
1991	Intel 486	Deze processor bevatte een interne coprocessor. Hij was ook veel sneller dan zijn voorganger: deze chip kon tot 120 MHz aan. Het werd uiteindelijk een van de meest verkochten processoren.
	AMD Am386 en Am486	AMD komt op de markt met eigen processoren, en wordt hierdoor de eerste echte concurrent voor Intel. Meer dan een miljoen Am386 processoren werden verkocht, onder andere in Compaq-computers.
1993	Intel Pentium 75	De Intel Pentium processoren lieten toe om gemakkelijker reële data te manipuleren zoals spraak, geluid, handschrift en foto's. Deze chips bevatten het equivalent van 3,1 miljoen transistoren.
1995	Intel Pentium Pro	In de herfst van 1995 werd de Intel Pentium Pro processor op de markt gebracht. Deze processor werd ontworpen om 32-bits server- en desktopapplicaties op te draaien. Hierdoor werden een heleboel perspectieven qua software geopend: computer-aided design, mechanical engineering en wetenschappelijke berekeningen. De Intel Pentium Pro processor werd vergezeld van een tweede snelheidsverhogende cache geheugenchip (zie 5.2). De processor bevatte 5,5 miljoen transistors.
1997	Intel Pentium MMX	Deze processor bracht een nieuwe techniek met zich mee: MMX. Dit is een verzameling speciale multimedia-instructies voor multimediale toepassingen.
	Intel Pentium II AMD-K6	Deze processor bevatte 7,5 miljoen transistoren. De AMD-K6 processor is een echte doorbraak op het vlak van concurrentie. De consument kan eindelijk kiezen tussen verschillende fabrikanten. De AMD-K6 processor blijft steeds een stuk goedkoper dan de alternatieven van Intel. Hierdoor boort AMD een nieuwe markt aan voor thuisgebruikers die aan lage prijs zeer degelijke processoren willen kopen voor desktop-applicaties. Intel blijft AMD meestal net een stapje voor qua snelheid en technologie, maar hierdoor kan AMD zijn prijzen nog meer drukken.

Vervolg op volgende pagina

Tabel 6.1 – vervolg van vorige pagina

Jaar	Processor	Informatie
1998	Intel Pentium II Xeon AMD-K6-2 Intel Celeron	Deze processor werd vooral gebruikt voor high-end servers en niet zozeer in desktop-computers. In 1998 bracht AMD de AMD-K6-2 processor uit, die 3DNow!-instructies bevatte. Dit was de directe tegenhanger van de MMX-instructieset en is dus ook op multimedia afgestemd. Intel brengt een lower-budget processor op de markt, om op die manier te differentiëren op de consumentenmarkt.
1999	Intel Pentium III en Pentium III Xeon AMD Athlon en Duron	De Pentium III bracht nieuwe instructies voor speciale toepassingen zoals advanced imaging, 3-D, streaming audio, video en speech recognition applicaties. 9,5 miljoen transistoren De AMD Athlon processoren betekenen het definitieve einde van de quasi-monopoliesituatie waarin Intel zich bevond. De Athlon is nu de directe tegenhanger van de Pentium processor. De race voor de snelste processor tussen Intel en AMD blijft duren. De Duron-processor de reactie van AMD op de Celeron processor. De Duron is dan ook goedkoper en afgestemd op de thuisgebruiker die standaard multimediale toepassingen wil gebruiken.
2000	Intel Pentium IV AMD AMD-K6-2	Gebruikers van de Intel Pentium 4 processor kunnen professionele kwaliteitsvideo maken, streaming video van hoge resolutie bekijken via het internet, 3D beelden genereren in real-time, muziek encoderen voor MP3-spelers. Tegelijkertijd kunnen gemakkelijke meerdere multimedia-applicaties draaien terwijl men ook nog verbonden is met het internet. Deze processor bracht 42 miljoen transistors met zich mee! Er werd ook voor het eerst gebruik gemaakt van HT (Hyper Threading) technologie. Hierbij wordt de processor logisch gezien in 2 verdeeld (voor het besturingssysteem komt het ook over alsof er 2 processoren zijn) en wordt de workload verdeeld over deze 2 processoren. Het besturingssysteem moet de Hyper-Threading technologie ondersteunen om er gebruik van te kunnen maken. Deze processor werd de mobiele versie van de reeds bestaande AMD-K6-2. Nog belangrijker was de introductie van de AMD-760-chipset. Deze liet als eerste toe om DDR-geheugen te gebruiken; Intel had deze technologie nog niet aangesproken wegens hun belangen in andere geheugentechnologien. Uiteindelijk is DDR-werkgeheugen de standaard geworden en was AMD hierbij een belangrijke pionier.
2001	Intel Itanium AMD Athlon XP	De Itanium processoren waren de eerste 64-bits processoren van Intel. Deze processor wordt gebruikt voor zeer-veeleisende en specifieke applicaties die gebruik kunnen maken van de 64-bits technologie. Deze processor, die speciaal werd ontworpen voor het Windows XP platform, was gebaseerd op AMD's QuantiSpeed-technologie.

Vervolg op volgende pagina

Tabel 6.1 – vervolg van vorige pagina

Jaar	Processor	Informatie
2003	Intel Pentium M	Met het oog op het opkomende succes van laptops werd de Intel Pentium M processor ontworpen. Deze energiesparende processor zorgt ervoor dat de gebruiker langer op de batterij van de draagbare computer kan werken. De Intel 855 chipset familie (zie 7.2) en de Intel PRO/Wireless netwerkverbinding zijn de belangrijkste componenten van de Intel Centrino mobiele technologie.
	AMD Opteron	De AMD Opteron bracht soelaas voor de AMD-fans die de 64-bits technologie wilden gebruiken. Als tegenhanger van de Intel Itanium werd het tijdperk van de 64-bits processoren helemaal op gang getrokken. De Opteron-reeks is duidelijk bestemd voor high-end systemen, waarbij performantie het kernwoord blijft. De Opteron-technologie is daarnaast uitermate geschikt om in multiprocessoromgevingen te gebruiken. Deze processor bevat een geïntegreerde memory controller, deze komt dus niet meer op de North Bridge voor!
	AMD Athlon 64	De Athlon 64-bitsprocessor biedt ondersteuning voor nieuwe toe-passingen die op de 64-bitstechnologie geschreven zijn. Verschillende varianten van deze processor zijn ondertussen in de omloop: de Athlon 64 X2 met dubbele processorcore, de Athlon 64 FX met HyperTransport technologie, de Athlon 64 for Desktops en de Mobile Athlon 64. Deze processor bevat een geïntegreerde memory controller, deze komt dus niet meer op de North Bridge voor!
2004	AMD Sempron	Omdat de AMD Duron processor uit de productielijn verdween, was er nood aan een nieuwe “value system” processor. De nieuwe AMD Sempron is dan ook een waardige opvolger voor de Duron en is een waardige tegenstander voor de Intel Celeron budgetprocessor. Ook hiervan werd een draagbare versie uitgewerkt, de Mobile AMD Sempron.
	AMD Celeron D en M	Intel differentieert in zijn Celeron-lijn rond een desktop- en notebookvariant, in de Celeron D resp. M processoren.
2005	AMD Turion	Ook AMD speelt in op de uitgebreide laptop-markt met deze Turion processor die weinig verbruikt en reeds op de 64-bits trein zit. Deze opvolger van de Itanium wordt gebruikt voor high-end serverplatformen.
	Intel Itanium 2	
	Intel Xeon Dual Core	Intel voorziet deze high-end server processor van 2 cores. Het begin van een nieuwe tendens in de processorindustrie.
	AMD Opteron Dual Core	Ook AMD stapt mee in de dual core wereld.
2006	Intel Core Duo	Intel lanceert de dual core technologie ook in de consumer-markt met deze processor. In juli 2006 komt ook nog de Intel Core 2 Duo uit.
Vervolg op volgende pagina		

Tabel 6.1 – vervolg van vorige pagina

Jaar	Processor	Informatie
	AMD Athlon 64 Dual-Core	Intel en AMD blijven elkaar op de hielen zitten. AMD brengt met deze Dual-Core processor een zeer krachtige gaming-processor op de markt.
	AMD Opteron Quad Core	De high-end server processor met vier kernen van AMD komt op de markt.
2007	Intel Core 2 Quad	Na de dubbele core volgt ook een quad-core (vier rekenkernen) processor van Intel.
2008	Intel Atom	De Intel Atom wordt voorgesteld als processor voor netbooks met een zeer laag energieverbruik. De processor levert voldoende kracht voor dagdagelijkse toepassingen zoals surfen en mailen.
	AMD Phenom	Nieuwste-generatie van snelle processoren met triple- of quad-core technologie.
	Intel i7	Deze high-end desktop processor levert 4 cores waarbij op elke core 2 threads tegelijk kunnen uitgevoerd worden. Dit betekent dat er 8 threads tegelijk worden uitgevoerd. De snelheid van de processor wordt gescaleerd naarmate er meer rekenkracht gevraagd wordt. Dit om stroom te besparen wanneer er geen of weinig berekeningen zijn. Een heel belangrijke wijziging is het toevoegen van een geïntegreerde memory controller op de processor zelf. AMD doet dit al sinds 2003 met de introductie van de AMD Opteron en Athlon 64.

Intel en AMD zijn ook vandaag de twee belangrijkste spelers op de processorenmarkt. De competitie voor de snelste processor zal waarschijnlijk nog een tijd blijven duren.

6.1.2 32 vs 64 bits?

Sinds enkele jaren worden quasi enkel nog 64 bits processoren op de markt verkocht. Het aantal bits van een processorarchitectuur slaat op de grootte van de interne registers die een processor gebruikt om gegevens te verwerken en berekeningen uit te voeren (zie verder). Het verdubbelen van dit aantal van 32 naar 64-bits betekent dus theoretisch dat de processor twee keer zoveel gegevens in dezelfde hoeveelheid tijd kan verwerken. In de praktijk blijken andere zaken belangrijker te zijn voor de verwerkingsnelheid, zoals de hoeveelheid cache en de mate waarin de processor meerdere instructies gelijktijdig kan verwerken.

Het voordeel van 64-bits processoren is dat applicaties die op zo'n processor draaien in staat zijn meer geheugen te adresseren. Met 32-bits processoren zijn programma's in staat maximaal 4GB geheugen te adresseren. Van deze 4GB staat maar 2GB (of in sommige gevallen 3GB) ter beschikking van de applicatie zelf, de rest wordt gebruikt door het besturingssysteem.

Het doorbreken van de 4GB grens is dus de belangrijkste reden voor softwarefabrikanten om over te stappen op 64-bits technologie. Hierdoor worden applicaties beter schaalbaar, vooral als het gaat om server applicaties die door meerdere gebruikers of programma's tegelijk benaderd kunnen worden. De volgende vraag is dan op welke 64-bits technologie over te stappen: er zijn

namelijk niet een maar twee 64-bits architecturen. Deze processoren zijn niet compatibel met elkaar: code die is gecompileerd voor de één kan niet worden uitgevoerd op de ander.

6.1.3 Dual-core en quad-core

Een multi-core CPU combineert 2 of meerdere onafhankelijke cores in 1 CPU. Een dual-core bevat 2 individuele microprocessors en een quad-core bevat er maar liefst 4. De verschillende cores delen meestal het cache-geheugen. Een systeem met n processoren kan in principe n aparte threads (deel van een proces dat op zichzelf kan uitgevoerd worden) tegelijk draaien. Denk bijvoorbeeld aan games: een thread voor het updaten van het scherm, een thread voor de afhandeling van de besturing en een thread voor het bijhouden van de score.

Stilaan worden meer en meer software-pakketten zo opgebouwd dat ze interessante mogelijkheden bieden voor het gebruik van meerdere cores.

6.2 Bouw van een processor

Figuur 6.3 toont een Pentium IV processor. In feite kan je niet veel zien aan deze afbeelding, maar achterliggend zitten er miljarden transistoren in dit kleine onderdeel geïntegreerd. Een microprocessor is in feite een geïntegreerde schakeling. De primaire grondstof voor zo'n schakelingen is silicium. Dit materiaal komt op aarde enorm veel voor in de vorm van siliciumoxyde, het hoofdbestanddeel van zand. Het siliciumoxyde wordt van zijn zuurstofatomen ontdaan waarbij een onzuivere vorm van silicium ontstaat. Na het verwijderen van de onzuiverheden wordt het silicium gesmolten bij een temperatuur van 1420 graden Celsius. Bij het afkoelen ontstaan dan zeer zuivere en regelmatige siliciumkristallen. Men vormt zo cylinders van ongeveer 10 tot 20 cm diameter en een halve meter lengte. Deze cylinders worden in schijfjes gezaagd van ongeveer 0,4 tot 0,5 mm dikte en dan fijn gepolijst tot een dikte van 0,2 mm. Zo ontstaat de zogenaamde "wafel" of het substraat waarop de geïntegreerde schakeling zal ontstaan.



Figuur 6.3: Een Pentium IV processor

6.2.1 Onderdelen

Volgende belangrijke onderdelen zijn te vinden op een processor:

- Arithmetic/Logic Unit (ALU): zorgt voor optellen, aftrekken, vermenigvuldigen en delen van gehele getallen.
- Floating Point Unit (FPU): zorgt voor alle bewerkingen op niet-gehele getallen. Vroeger zat deze functionaliteit in de coprocessor.

- Registers: tijdelijke opslagplaatsen voor de processor om gegevens en resultaten in op te slaan. Het is in feite een kladblok voor de processor en deze registers vormen het snelste geheugen van de processor. Wanneer we bijvoorbeeld over een 32-bits processor praten, bedoelen we dat het aantal bits per register 32 is.
- Bus Unit: regelt het dataverkeer naar en van de processor.
- Decision-making en instruction prediction: dit onderdeel probeert te voorspellen welke tak van een programma zal gekozen worden, om op die manier bepaalde gedeelten van een programma in te laden.
- Control Unit: de primaire component, eigenlijk “the brain within the brain”. Het controleert en coördineert de werking van de andere componenten.

6.2.2 Transistor density

Processoren worden steeds geavanceerder en krijgen meer functies. Hiervoor zijn meer transistoren nodig om deze bewerkingen uit te voeren. Als het aantal transistoren vergroot, moet hun grootte dalen. Hiervoor zijn verschillende redenen: de processor zou te groot en te zwaar worden, er zou te veel warmteontwikkeling zijn, de processor moet nog van stroom kunnen worden voorzien, ...

Transistoren zijn ondertussen enorm klein geworden: bij oudere chips was er ongeveer 1 micron¹ tussen de transistors, vandaag zijn de afstanden 0,18 tot 0,13 micron.

6.2.3 Koeling

Toen de 80386 werd ontwikkeld, die nog aan relatief lage snelheden werkte, was koeling nog niet zo belangrijk. Deze processor bevatte niet zo veel transistoren, waardoor hij niet heel warm werd. De koeling van de voeding was voldoende om de lucht te laten circuleren in de behuizing en op die manier alle componenten te koelen.

Rond de jaren 90 begon koeling van processoren een zeer belangrijk aspect te worden. Met de introductie van de 80486 en vooral de 80486DX2/66 met klokverdubbeling werd koeling onvermijdelijk, omdat de processor ontzettend warm werd -zelfs onder normale werkcondities.

Koelingproblemen

Processoren hebben, net als vele andere elektronische onderdelen, een temperatuurrange waarin ze normaal kunnen werken. Wanneer een processor toch overhit geraakt, kan dit leiden tot zeer uiteenlopende problemen. Het is dan ook vrij moeilijk een diagnose te stellen, omdat andere onderdelen van het computersysteem ook te leiden hebben van oververhitting van de processor. Meestal gaat een oververhitte processor wel gepaard met systeemcrashes, lockups en random reboots. Het moeilijke is dat natuurlijk ook andere problemen (software bijvoorbeeld) kunnen zorgen voor deze verschijnselen.

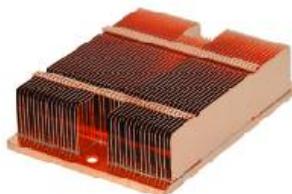
¹een micron is een kortere naam voor een micrometer, of één miljoenste van een meter.

AT en ATX behuizingen

De oorspronkelijke AT form factor behuizing was niet echt geschikt om de processor van een goede koeling te voorzien. De processor was te ver van de voeding geplaatst en de ventilators brachten de lucht naar buiten, waardoor de microprocessor niet rechtstreeks werd afgekoeld. Met de introductie van de ATX behuizingen werd vooral gekeken naar de verbetering van de koelingsmogelijkheden. Omdat de processoren en omliggende chips altijd maar meer en meer transistoren gingen bevatten, bleek een extra koeling echter onvermijdelijk.

Actieve en passieve koeling

De twee belangrijke typen van koeling zijn de passieve en actieve koelingen. De passieve koeling is een metalen raster dat de warmte kan afvoeren, meestal in koper door de uitstekende warmtegeleidende eigenschappen van dit materiaal. De naam passief komt van het feit dat er geen bewegende delen zijn in dit soort koeling. Het is dan ook een geruisloze oplossing, wat bij moderne (en dikwijls luide) computersystemen een groot voordeel kan bieden. Ook in bijvoorbeeld televisies en radios komen passieve koelingen voor. Figuur 6.4 toont een passieve koeling. De koperen vinnen die de warmte afvoeren zijn duidelijk te zien. Ook actieve koeling-



Figuur 6.4: Een passieve koeling

gen kunnen gebruikt worden. Hierbij wordt de processor gekoeld door een draaiende ventilator, meestal nadat eerst de warmte werd afgevoerd door een koperen plaat die op de processor wordt geplaatst. Om ervoor te zorgen dat de warmte goed kan ontsnappen, wordt een speciale warmtegeleidende gel aangebracht tussen de processor en het koelblok. Figuur 6.5 toont een actieve processorkoeling.



Figuur 6.5: Een actieve koeling

6.2.4 Van 10 µm naar 11nm technologie

In de jaren 70 werden de eerste processoren geproduceerd aan de hand van een 10 µm proces. Onder andere de Intel 4004 en 8008 processoren werden zo gemaakt. Ondertussen is er een

grote vooruitgang geboekt in de productie van processoren. Men kan de verschillende transistoren op een nog kleinere oppervlakte zetten wat veel voordelen met zich meebrengt: minder energieverbruik, meer transistoren (en dus snellere processoren) en minder warmteontwikkeling. Momenteel is de standaard een 22nm productie, deze wordt onder andere gebruikt door Intel bij de productie van de Ivy Bridge reeks. Tegen 2018 zou men al 16nm kunnen halen. De verste voorspelling in de roadmap is de 11nm productie, die rond het jaar 2022 zou gehaald worden.

6.3 Werking

6.3.1 Processorcyclus

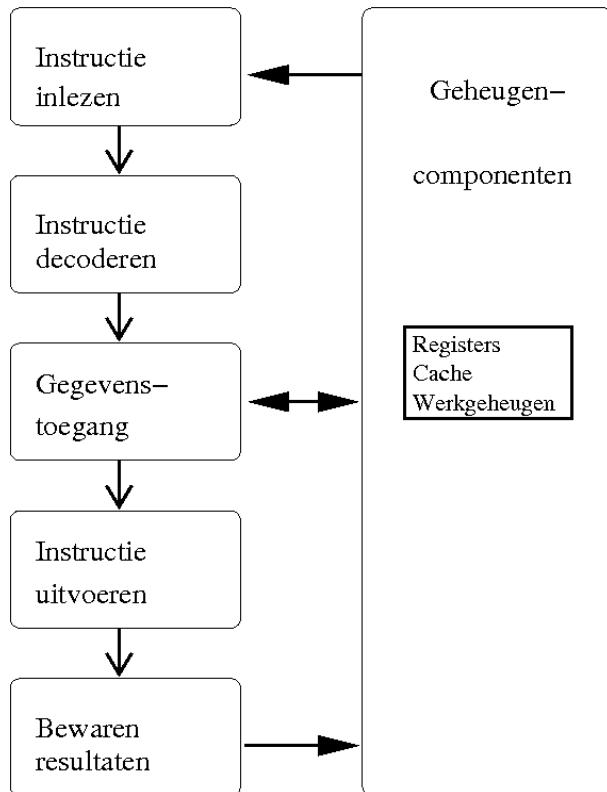
In principe zal een processor verschillende stappen ondernemen om uiteindelijk zijn werk te verrichten. Hij leest een instructie in, decodeert deze (interpreteren van wat er moet gedaan worden) en voert deze ten slotte uit. Bij dit uitvoeren kan het zijn dat er bepaalde gegevens uit het geheugen ingelezen of weggeschreven moeten worden. Elke processorarchitectuur volgt deze manier van werken op een of andere manier. Dikwijls worden sommige stappen gecombineerd, zodat er performantiewinst is. In figuur 6.6 wordt getoond hoe de processor zijn taak uitvoert. De werking is een opeenvolging van volgende cyclus, die men ook wel fetch-decode-execute cyclus noemt (zoals men kan zien komt er wel wat meer bij kijken):

- Inlezen instructie: de uit te voeren instructie wordt ingelezen vanuit het werkgeheugen, of beter nog de cache of een prefetch queue. Wanneer een instructie elke keer vanuit het werkgeheugen zelf zou moeten worden ingelezen, zou veel tijd worden verloren. Daarom worden de cache geheugens en prefetch queues hier in het bijzonder gebruikt.
- Instructie decoderen: er wordt gekeken naar wat de instructie doet. Dit kan een optelling zijn, een verplaatsing van een registerinhoud naar het geheugen of vice versa, ...
- Gegevenstoegang: indien er gegevens nodig zijn uit het geheugen, worden deze opgehaald.
- Instructie uitvoeren: de gedecodeerde instructie is nu klaar voor effectieve uitvoering.
- Bewaren resultaten: indien nodig, worden gegevens weggeschreven naar het werkgeheugen (of in de meeste gevallen de cache).

Meer informatie over mogelijke instructies wordt gegeven in hoofdstuk 12

6.3.2 Kloksnelheden

De kloksnelheid van een processor bepaalt hoeveel berekeningen per seconde kunnen worden uitgevoerd. Dit wordt weergegeven in MHz of hoeveel miljoen cycles per seconde de processor kan uitvoeren. Tegenwoordig zijn de snelheden opgedreven tot 4000 MHz, of 4 GHz (4 miljard cycles per seconde!). In feite zegt de kloksnelheid niet zo veel over de effectieve snelheid van de processor, het aantal instructies per seconde is hiervoor een betere maat (een instructie bestaat uit meerdere cycles). Maar zelfs het aantal instructies per seconde (vaak uitgedrukt in MIPS: Millions of Instructions Per Second) is geen absolute maat meer: bepaalde instructies zijn veel krachtiger dan andere en pipelining kan instructies laten overlappen. In 1987 noemde Bob



Figuur 6.6: Werking van een processor

Estall MIPS zelfs “Meaningless Indication of Processor Speed”. De kloksnelheid wordt bepaald aan de hand van 2 waarden, de multiplier en de front side bus (FSB) snelheid van de processor:

$$\text{kloksnelheid} = \text{FSB} \times \text{multiplier}$$

De FSB snelheid van de processor is in een ideale situatie hetzelfde als de FSB snelheid van het moederbord. Zo worden de snelheid van moederbord en processor optimaal benut.

Een voorbeeld dat goed aantoon hoe weinig de werkelijke kloksnelheid zegt over de uiteindelijke prestatie is de Intel Pentium rating die AMD gebruikt. AMD maakt processoren met een lagere kloksnelheid dan die van Intel. De AMD processoren kunnen qua prestatie echter zeer goed concurreren met Intel processoren met een hogere kloksnelheid. AMD geeft daarom een “Pentium Rating” aan zijn processoren. Een Athlon XP 1800+ is ongeveer vergelijkbaar met een Pentium 1,8 GHz, maar de AMD Athlon XP 1800+ draait maar op 1530 MHz (133 fsb x 11,5 multiplier).

In verband met de snelheid van processoren is het steeds interessant de wet van Moore² te vermelden. Als een van de oprichters van Intel stelde hij reeds in 1965 dat het aantal transistors per vierkante centimeter jaarlijks zou verdubbelen. Dit is niet helemaal uitgekomen, maar we kunnen toch een verdubbeling om het jaar en half à twee jaar vaststellen. De verdubbeling van het aantal transistors brengt meestal ook een ingrijpende snelheidsverhoging met zich mee.

²Gordon Moore paste zijn wet in 1975 aan naar een verdubbeling om de 2 jaar. Moore haalde een doctoraat in fysica en chemie aan California Institute of Technology. Moore ontving in 1990 de technologiemedaille van president George Bush.

6.3.3 Buffers en cachegeheugen

Het werkgeheugen is veel te traag in vergelijking met de verwerkingssnelheid van de processor. Daarom kunnen de opeenvolgende instructies beter dichter bij de processor worden bijgehouden. De processor zelf is voorzien van intern cachegeheugen. De instructies worden opgeslagen in het cache-geheugen, alsook gegevens die gebruikt worden. Een moderne processor heeft twee levels van cache geheugen: 1 uiterst snel en klein cachegeheugen en een tweede dat iets minder snel is en ook groter (zie [5.2](#)).

De buffers zijn een tijdelijke opslagplaats voor instructies die nog niet zijn uitgevoerd of gegevens die nog niet naar het werkgeheugen zijn weggeschreven. Op deze manier kan de processor verder werken en wordt de processorcapaciteit beter benut. Sommige van deze buffers worden prefetch buffers genoemd. Hierin worden de opeenvolgende instructies opgeslagen om ze daarna een voor een uit te voeren.

6.3.4 Toekomstvoorspelling

Een van de factoren die de snelheid van de processor bepaalt, is het kunnen voorspellen van de volgende uit te voeren instructies. Wanneer de processor weet wat de volgende instructies zijn, kunnen deze in de prefetch queue worden geladen en kan de processor maximaal benut worden. Een voorbeeld van zo'n voorspelling is:

```

1 | if (a==b){  

2 |     c=d;  

3 | }  

4 | else{  

5 |     c=e;  

6 | }
```

Van het moment dat de processor de waarde van a en b kent, en deze waarden niet meer worden aangepast in het programma tot aan de if, kan de processor al de juiste code inladen in zijn prefetch queue en het systeem om instructies op een rij klaar te zetten voor uitvoering wordt ook wel een pipeline genoemd. Het voorspellen van de code (in dit geval een if-structuur) die moet worden uitgevoerd noemt men ook wel branch prediction.

6.3.5 Instruction sets

Een instruction set is de verzameling van bevelen die een processor verstaat en kan uitvoeren. Alle processoren zijn voorzien van een basisinstructieset, instructies voor “gewoon” gebruik. Daarnaast bezitten processoren ook een aantal instructies voor optimalisatie op bepaalde gebieden:

- De MMX MultiMedia eXtension Technology is een groep van extra instructies voor de ondersteuning van multimedia-toepassingen.
- MMX maakt gebruik van de techniek *single instruction, multiple data* (SIMD), die toelaat om de processor tegelijkertijd een berekening uit te voeren op twee, vier of zelfs acht data-elementen -zonder verlies van snelheid. MMX bekomt dit door meerdere operanden (8-, 16- of 32-bits waarden) samen in 1 64-bits register te plaatsen en de berekening op alle waarden tegelijkertijd uit te voeren. Bijvoorbeeld kan een grafisch programma 8 1-byte

kleurenwaardes in een register zetten en de processor de taak geven om 10 op te tellen bij elk van de waarden. Op die manier kan de kleur van meerdere pixels tegelijkertijd aangepast worden.

- De MMX instructieset werd niet specifiek voor Multimedia ontwikkelt, maar vindt wel zijn voornaamste toepassingen in dit gebied.
- Vanaf de Pentium III werd een uitgebreidere versie van MMX geïntegreerd: SSE (Streaming SIMD Extensions)
- De tegenhanger van AMD voor MMX is AMD 3DNow!

6.4 Registers

Registers zijn in feite de geheugenplaatsen in de processor zelf, de registers vormen samen een kladblok waarop de processor zijn berekeningen uitvoert. Registers zitten ingebakken in de processor, zodat alles veel sneller kan verlopen. Het aantal registers en de grootte ervan (16-, 32- of tegenwoordig 64-bits) is afhankelijk van het type processor. Welke registers zoal voorkomen op een moderne processor, wordt verder bekijken in hoofdstuk 12.

Hoofdstuk 7

Chipsets en bussen

Inhoudsopgave

7.1 Bussen	70
7.1.1 De architectuur	70
7.1.2 Een geïntegreerde memory controller	71
7.1.3 De PCI-bus	72
7.1.4 USB (Universal Serial Bus)	74
7.1.5 AGP (Accelerated Graphics Port)	74
7.1.6 ATA	76
7.2 Chipsets	76
7.2.1 Processor-families	77
7.2.2 DMA (Direct Memory Access)	77

7.1 Bussen

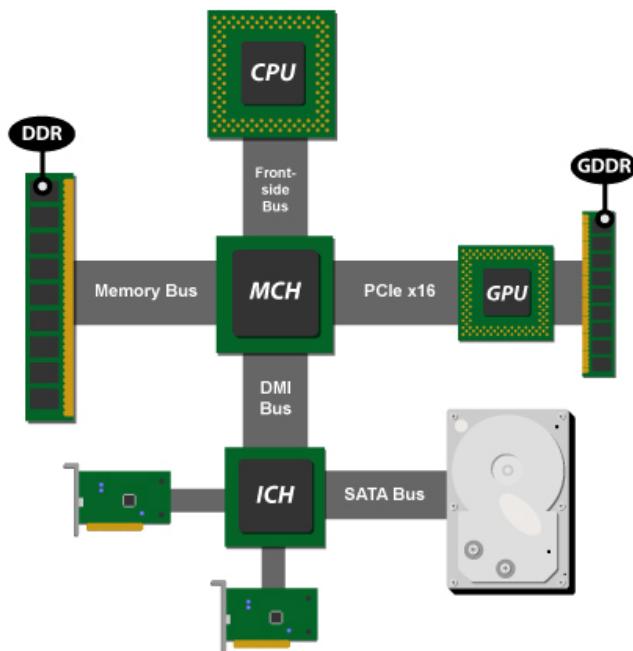
Overdracht van gegevens tussen de verschillende onderdelen van een computer gebeurt door middel van bussen. Gegevenspakketten (van 8, 16, 32, 64 of meer bits) worden voortdurend uitgewisseld tussen de CPU en andere onderdelen. Al deze bussen bevinden zich op het moederbord, de centrale printplaat waar alle onderdelen van een computer op gemonteerd kunnen worden.

Op een moederbord is er niet slechts 1 bus, er zijn er meerdere die afhankelijk van de aangesloten onderdelen op een bepaalde snelheid kunnen werken. Sommige onderdelen hebben namelijk zeer snel ontzettend veel gegevens nodig (bv. de processor), tegenover andere onderdelen die veel minder nodig hebben (bv. een aangesloten keyboard genereert niet zo veel data).

7.1.1 De architectuur

In figuur 7.1 wordt de architectuur van de bussen weergegeven bij een Intel chipset. De architectuur van een AMD systeem is zeer gelijkaardig. Hierbij zien we dat er een frontside bus aanwezig is, die ook wel de systeembus wordt genoemd. Deze systeembus heeft een zeer hoge

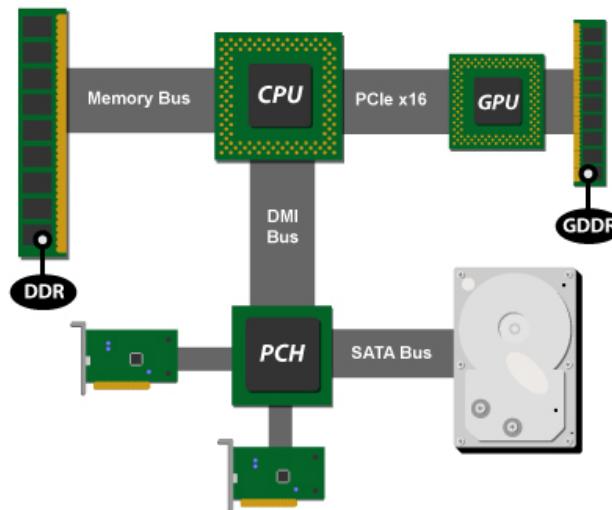
sneldheid en grote bandbreedte (veel bits kunnen tegelijk verstuurd worden) om te voorzien in de behoefte van de processor. De FSB (Front Side Bus) verbindt hierbij de processor met de North Bridge (MCH op de tekening). Doorgaans bestaat een computer-architectuur uit twee belangrijke chips: de north en de south bridge (ICH op de tekening). Deze onderverdeling vinden we bijvoorbeeld terug bij Intel en VIA chipsets. De North bridge is de verbinding tussen de CPU, het werkgeheugen en de AGP-poort. Daarnaast zal de North Bridge verbonden zijn met de South Bridge. Aan deze South Bridge hangen de -in vergelijking met de CPU-trage apparaten aan de verschillende bussystemen. Hierbij zijn PCI, USB, EIDE de meest gebruikte. Ook andere functies worden aan deze South Bridge gekoppeld, bijvoorbeeld geluid, netwerkadapters, ...



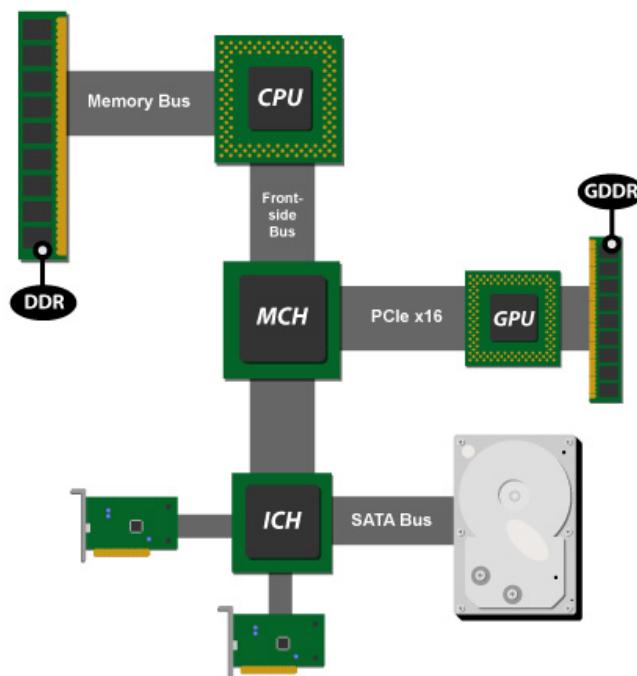
Figuur 7.1: Architectuur van een Intel chipset

7.1.2 Een geïntegreerde memory controller

Zoals beschreven in de vorige paragraaf zit de memory controller momenteel nog op de north bridge. Intel rust echter zijn nieuwe Intel i3, i5 en i7 processoren uit met een geïntegreerde memory controller, AMD doet dit zelfs al sinds 2003. Wanneer de memory controller op de processor zelf zit, worden de afstanden uiteraard verkleind en leidt dit tot snellere datatransfers tussen processor en werkgeheugen. Deze architectuur wordt getoond in figuur 7.2. De architectuur die AMD sinds 2003 gebruikt, is lichtjes verschillend. Deze wordt getoond in figuur 7.3. Het grote verschil is dat AMD enkel de memory controller naar de CPU heeft gehaald. Intel daarentegen haalt ook de PCI Express controller naar de CPU (al zijn hier ook uitzonderingen op bij Intel). De nieuwe architectuur bij Intel wordt ook de P55 architectuur genoemd. De south bridge noemt vanaf nu platform controller hub (PCH) en dit is de enige chip die nog echt onder de "chipset" valt.



Figuur 7.2: Intel computerarchitectuur met geïntegreerde memory controller



Figuur 7.3: AMD computerarchitectuur met geïntegreerde memory controller

7.1.3 De PCI-bus

In dit deel bespreken we de verschillende bussen die zijn ontwikkeld voor het aansluiten van uitbreidingskaarten, bijvoorbeeld netwerkkaarten of geluidskaarten. Deze bussen werken typisch aan een veel lagere snelheid dan bijvoorbeeld de front side bus.

PC bus, ISA bus en VESA bus

De eerste belangrijke bus die werd gebruikt in de computerwereld was de PC bus. Deze Personal Computer bus werkte aan 4.77 MHz en was 8 bits breed. Later werd deze PC bus verbreed tot 16 bits en versneld tot 8 MHz. Deze busarchitectuur noemde men de ISA-bus. ISA stond voor Industry Standard Architecture en werd op zijn beurt opgevolgd door de EISA (Extended ISA) bus, die 32 bits breed was en aan 8 MHz werkte. Daarnaast ontwikkelde men de VESA bus, die nog tot ongeveer 1994 werd gebruikt.

De PCI (Peripheral Component Interconnect) werd rond 1990 ontworpen door Intel. Rond 1992 kwam PCI 1.0 uit, een jaar later werd de nieuwere versie PCI 2.0 uitgebracht. VESA bleef lang populair, maar werd met de introductie van de Pentium toch vervangen door PCI. Vooral de introductie van Windows 95, met plug&play capaciteiten die door PCI werden ondersteund, maakte van PCI de marktstandaard. Ook tot op vandaag (anno 2012) is de PCI-bus (of althans zijn opvolger PCI Express) zeer populair, ze wordt gebruikt voor het aansluiten van modems, netwerkkaarten, geluidskaarten, ...

PCI bus

Het is belangrijk te weten dat de PCI bus eigenlijk losgekoppeld wordt van de systeembus door een South Bridge. Op deze manier kunnen beide bussen op verschillende snelheden werken, maar toch met elkaar communiceren. De PCI bus werkt aan 33 MHz, met een breedte van 32 bits. Met deze instellingen kunnen we een bitrate van 132 MB per seconde verkrijgen. Sommige varianten laten echter toe te werken aan 66 MHz tot 64 bits. Het laat hiernaast toe om burst-transfers uit te voeren. Deze technologie zorgt ervoor dat opeenvolgende geheugentoegangen sneller gemaakt worden. Hierbij wordt het eerste adres opgegeven van de door te geven data, en vervolgens worden de achterliggende datablokken gewoon doorgegeven, zonder dat eerst telkens het adres moet doorgegeven worden.

PCI maakt gebruik van een gemultiplexte adres- en databus. Dit wil zeggen dat over dezelfde lijn zowel de data als de adressen worden verstuurd. Hierbij wordt eerst het adres doorgegeven en nadien de effectieve data doorgestuurd.

PCI-X

De nieuwste variant van PCI is de PCI-X (PCI-extended) bus, waarbij de nieuwste technologieën zoals fibre channel, Ultra3 SCSI en gigabitnetwerken in het achterhoofd worden gehouden. Hier hebben we immers te maken met een hoge datarate waarbij we dus ook op onze bussen voldoende bandbreedte moeten voorzien. PCI-X voorziet in 1GB per seconde, dit op een 64 bits brede bus die aan 133 MHz draait. De PCI-X standaard is achterwaarts compatibel met PCI, waardoor oudere PCI kaarten toch nog kunnen gebruikt worden. De laatste versie van PCI-X, de PCI-X 2.0, laat toe om aan 266 of 533 MHz te werken wat bursts kan geven tot 2,1 GB/s resp. 4,3 GB/s!

PCI Express

PCI Express is een volledig nieuwe busarchitectuur, ook bekend onder de naam 3GIO. De aansluitingen werden veranderd, waardoor er geen compatibiliteit is met PCI. PCI Express heeft volgende eigenschappen:

- Seriële communicatie, waardoor later nog sneller kan gewerkt worden zonder problemen (zie ook [10.1](#)).
- Hoge bandbreedte: van 5 tot 80 Gbps (Gigabit per seconde)
- Kleine connectoren
- QoS: Quality of Service wat belangrijk is bij streaming toepassingen.

PCI Express wordt vandaag de dag vooral gebruikt om grafische kaarten aan te sluiten.

7.1.4 USB (Universal Serial Bus)

De Universal Serial Bus is een bus waarop gegevens in seriële vorm worden overgedragen en die zo universeel is opgezet dat er allerlei verschillende soorten apparaten kunnen worden op aangesloten. De USB bus werd ontwikkeld door een consortium van fabrikanten, waaronder Microsoft, Intel, Apple Computer, NEC, HP, IBM en Compaq. De bedoeling was een oplossing te bieden voor de problemen waarmee PC gebruikers dikwijls te kampen hebben: de wirwar van verschillende soorten kabels met verschillende soorten stekkers. Daarnaast is het plug&play concept zeer belangrijk voor gebruikers, hierbij wordt een apparaat automatisch gedetecteerd en geconfigureerd.

In de specificatie van USB wordt voorzien dat er (theoretisch) 127 apparaten tegelijkertijd kunnen worden aangesloten. Dat is heel wat meer dan er op een PC zonder USB kunnen aangesloten worden. Een ander voordeel is dat een apparaat dat via USB wordt aangesloten nooit een insteekkaart nodig heeft. De USB aansluitingen zitten aan de buitenkant van de computerkast waardoor gebruikers hun systeemkast niet meer hoeven open te vijzen. USB is ook hotpluggable, wat betekent dat men apparaten zonder probleem mag aansluiten of uittrekken wanneer de computer aanstaat.

USB wordt verder uiteengezet in hoofdstuk [11](#).

7.1.5 AGP (Accelerated Graphics Port)

De AGP poort is een point-to-point kanaal dat aan hoge snelheid werkt en dat het mogelijk maakt 1 apparaat met het moederbord te verbinden. Meestal is dit apparaat een grafische kaart. In feite hoort AGP niet thuis onder de noemer bus, aangezien er maar 1 apparaat kan worden op aangesloten. De term bus wordt gebruikt wanneer er meerdere apparaten kunnen worden aangesloten. We bespreken AGP echter in dit hoofdstuk, aangezien het een belangrijk onderdeel is van het moederbord.

Geschiedenis

AGP werd het eerst geïntroduceerd door Intel in 1997 in hun 440LX chipset, die bedoeld was voor de Pentium II. AGP werd al snel de standaard voor grafische kaarten, alhoewel ook nog grafische PCI kaarten werden gemaakt voor moederborden zonder AGP slot. De eerste AGP versie, AGP 1.0 (of AGP 1x), heeft een 32 bits breed kanaal dat aan 66 MHz werkt. Dit kan zorgen voor datatransfers van 266 MB/s. Een groot voordeel van AGP (zie ook figuur [7.1](#)) is dat AGP rechtstreeks het geheugen kan aanspreken, wat bij PCI niet helemaal mogelijk was. Figuur [7.4](#) toont een typische AGP grafische kaart.



Figuur 7.4: AGP grafische kaart

Nieuwe AGP versies

Al snel volgden nieuwere versies van AGP, toen de vraag naar zeer performante grafische kaarten (vooral door gamers) groter werd. De nieuwere versies zijn:

- AGP 2x: gebruik makend van een 32 bits kanaal, met een verdubbeling van de hoeveelheid data die verstuurd wordt tijdens 1 clockcycle. Dit zorgt voor een effectieve verdubbeling van de datarate tot 533 MB/s
- AGP 4x: nog steeds een 32 bits kanaal, nu met een viervoudige hoeveelheid data per clock cycle, met als gevolg een AGP poort die 4x zo snel werkt. (1066 MB/s) Vanaf deze AGP 4x standaard spreekt men van AGP 2.0. AGP 2.0 is weer achterwaarts compatibel met 1x of 2x AGP kaarten.
- AGP 8x: de nieuwste versie van AGP, namelijk AGP 3.0, laat toe om tot 2133 MB/s te versturen. Hierbij wordt 8 keer zoveel data per clock cycle verstuurd. Ook AGP 8x werkt dus nog steeds op een 66 MHz bus. AGP 8x is ook weer compatibel met de vorige versies van AGP.

Het enige belangrijke bij de compatibiliteit is het voltageniveau, maar de aansluitingen laten niet toe dat een kaart op een verkeerd voltageniveau wordt gebruikt.

AGP Pro

AGP Pro is een andere AGP variant waarbij een langer slot (niet AGP compatibel) wordt gebruikt, met een sterkere voeding. Dit laat toe om zeer krachtige (en dure) grafische kaarten te gebruiken, die bijvoorbeeld in de animatiewereld veel gevraagd zijn. Ook andere zware grafische toepassingen kunnen gebruik maken van deze standaard, zoals CAD (Computer Aided Design) toepassingen, programma's voor ingenieurs of architecten, ...

Toekomst

De AGP poort is stilaan aan het uitsterven en zal op termijn volledig vervangen worden door de nieuwe PCI-Express standaard. (zie [7.1.3](#)) Intel kondigde in 2004 aan dat in hun nieuwe chipsets AGP ondersteuning zal plaatsmaken voor PCI-Express. Ook Nvidia en ATI -twee belangrijke producenten van grafische chips- volgen mee, de NV40 resp. R420 zullen de laatste grafische chips zijn die AGP ondersteunen.

7.1.6 ATA

De implementatiedetails van ATA reiken buiten de scope van deze cursus. ATA wordt gebruikt om CD-ROM lezers, CD en DVD schrijvers, maar vooral harde schijven aan te sluiten om een computersysteem. Tegenwoordig wordt reeds veel gebruik gemaakt van de nieuwe standaard, SATA (Serial ATA) die hoogstwaarschijnlijk ATA op termijn zal vervangen.

7.2 Chipsets

Naast de microprocessor en de systeembussen zijn er een aantal ondersteunende chips nodig voor de werking van een moderne computer. Een chipset is een verzameling van chips (Engelse “set” = verzameling) die op een moederbord voorkomen en die bepaalde taken op zich nemen. Zoals eerder in dit hoofdstuk vermeld worden de functies van de north bridge op recente processoren verplaatst naar de CPU zelf. We kunnen een chipset van een moederbord opsplitsen in 3 belangrijke functies:

- System controller (North Bridge of op CPU op nieuwere systemen): De system controller regelt een aantal zaken in verband met timing. De functies die in een moderne system controller aanwezig zijn, zaten reeds in de eerste chipsets op oudere PC's. Een van de belangrijke functionaliteiten die pas later is toegevoegd, is power management. Men had bij de fabricatie van de eerste PC's ook nooit kunnen voorspellen dat onze computers ooit draagbaar zouden worden. De standaardonderdelen van de system controllers zijn:
 - Timers en oscillators: deze zorgen voor de timing van de microprocessor (frequentie), het geheugen en de rest van de computer.
 - De interrupt controller: deze zorgt voor onderbrekingen in programma-uitvoeringen wanneer iets belangrijks gebeurd is. (zie ook hoofdstuk 12)
 - De DMA controller: deze regelt het gegevenstransport van en naar het interne geheugen, zonder tussenkomst van de processor. Hierdoor is de processor in staat meer tijd te besteden aan andere instructies. DMA wordt uitvoerig besproken in sectie 7.2.2.
 - Energiebeheer: met de huidige trend van draagbare computers, is het belangrijk dat we zo lang mogelijk kunnen blijven werken zonder externe stroomvoorziening. Dit kan enkel wanneer we rekening houden met energiebesparende maatregelen. Hiervoor zal de Power Unit in de system controller zorgen.
- Memory controller (North Bridge): de geheugencontroller heeft een belangrijke taak. Hij spreekt het geheugen aan, bepaalt de snelheid en de manier waarop het geheugen wordt gebruikt en voegt eventueel een ECC (zie 5.6.2) toe. Tegenwoordig wordt de memory controller zowel bij Intel als AMD processoren niet meer op de North Bridge geplaatst maar rechtstreeks op de processor zelf. Voor meer info, zie 7.1.2.
- Peripheral controller (South Bridge): de peripheral controller regelt de verbindingen naar interfaces waar randapparaten zijn op aangesloten. Het gaat hier om:
 - De businterface: de South bridge zal bijvoorbeeld een PCI bus verbinden met de systeembus.

- De floppy-drive interface: deze regelt de toegang tot 1 of meerdere floppy-drives.
- De harddiskinterface: deze regelt de toegang tot 1 of meerdere harde schijven.
- De toetsenbordcontroller: deze vertaalt de scancodes die door het toetsenbord worden gegenereerd naar code die het besturingssysteem begrijpt.
- De in/uitvoerpoortcontroller: deze verwerkt de in- en uitvoer van onder andere seriële en parallelle poort.

De South Bridge is verantwoordelijk voor communicatie met seriële en parallelle port, muis, toetsenbord, modem en geluidskaart, harde schijven, USB interfaces, ...

7.2.1 Processor-families

Een bepaalde processor wordt meestal samen ontwikkeld met een bijhorende chipset. Zo is er voor elke processor een chipset die de zonet vermelde functionaliteiten op zich neemt. Figuur 7.5 toont een aantal chipsets die werden ontwikkeld voor specifieke processoren.

7.2.2 DMA (Direct Memory Access)

DMA is een techniek die toelaat dat bepaalde apparaten rechtstreeks werkgeheugen kunnen aanspreken om te lezen en/of schrijven, zonder dat daarvoor de processor moet gestoord worden. Zeer veel hardware maakt gebruik van DMA, zoals harde schijf controllers, grafische kaarten, netwerkkaarten en geluidskaarten.

DMA is uiterst belangrijk in een moderne computerarchitectuur, aangezien apparaten met verschillende snelheden kunnen communiceren zonder hiervoor de processor te beladen met een grote hoeveelheid interrupts (zie ook hoofdstuk 12). Als DMA niet zou bestaan, zou de processor steeds data moeten verwerken van zo'n apparaat, om het dan bijvoorbeeld weg te schrijven naar het geheugen. De processor zou hierdoor belast worden, waardoor hij geen andere taken op zich kan nemen op zo'n moment. Daar moderne computergebruikers willen surfen terwijl ze naar hun favoriete muziek luisteren en meestal nog veel meer, is DMA echt een must geworden. Een ander typisch voorbeeld van DMA is bij het inlezen van data vanaf een diskette. Dit inlezen gaat ontzettend traag, en het is dus interessant om de DMA controller het inlezen verder te laten afhandelen, in plaats van de processor enorm te gaan beladen met deze tijdsrovende taak.

Werking van DMA

Om DMA te kunnen laten werken, maakt men gebruik van een DMA controller. De processor zal meestal een transfer van data initialiseren (bijvoorbeeld van de netwerkkaart naar het werkgeheugen bij het downloaden van een webpagina), maar het vervolg van de transfer zal worden afgehandeld door de DMA controller.

In principe bestaan er twee belangrijke vormen van DMA: Third-Party en First-Party DMA. Bij Third-Party DMA wordt de ingebouwde ISA DMA controller (die relatief traag geworden is) gebruikt. Deze is nog wel voldoende snel om bijvoorbeeld te gebruiken voor geluidskaarten of het diskettestation. Voor snelle apparaten zoals harde schijven is de Third-Party DMA controller te traag geworden. In dit geval wordt in het toestel zelf een DMA controller ingebouwd, wat men First-Party DMA noemt. Deze zal de controle overnemen van de bus om zijn

Feature	VIA K8T890	VIA K8T800 Pro	VIA K8T800	VIA K8M800
North Bridge				
Processor Support	AMD Athlon™64, Athlon™64FX & Sempron™ processors	AMD Athlon™64, Athlon™64FX & Opteron™ processors	AMD Athlon™64, Athlon™64FX & Opteron™ processors	AMD Athlon™64, Athlon™64FX & Opteron™ processors
Front Side Bus	1GHz/16-bit HyperTransport Bus Link	1GHz/16-bit HyperTransport Bus Link	800MHz/16-bit HyperTransport Bus Link	800MHz/16-bit HyperTransport Bus Link
Bus Architecture	Asynchronous	Asynchronous	Synchronous	Synchronous
Memory Support	DDR memory controller integrated directly into processor*			
External Graphics	PCI Express x16	AGP8X/4X	AGP8X/4X	AGP8X/4X
Integrated Graphics	N/A	N/A	N/A	S3 Graphics UniChrome™ Pro IGP
PCI Express Peripheral Support	4 PCI Express x1	N/A	N/A	N/A
Video/Display	N/A	N/A	N/A	MPEG2 decoder HDTV output
South Bridge	VIA VT8237	VIA VT8237	VIA VT8237	VIA VT8237
North/South Bridge Link	Ultra V-Link (1066MB/s)	Ultra V-Link (1066MB/s)	8X V-Link (533MB/s)	8X V-Link (533MB/s)
Audio	VIA Vinyl™ 6-channel Audio (AC97 integrated) VIA Vinyl™ Gold 8-channel Audio (PCI companion controller)	VIA Vinyl™ 6-channel Audio (AC97 integrated) VIA Vinyl™ Gold 8-channel Audio (PCI companion controller)	VIA Vinyl™ 6-channel Audio (AC97 integrated) VIA Vinyl™ Gold 8-channel Audio (PCI companion controller)	VIA Vinyl™ 6-channel Audio (AC97 integrated) VIA Vinyl™ Gold 8-channel Audio (PCI companion controller)
Networking	VIA Velocity™ Gigabit Ethernet (PCI companion controller) VIA Integrated 10/100 Fast Ethernet	VIA Velocity™ Gigabit Ethernet (PCI companion controller) VIA integrated 10/100 Fast Ethernet	VIA Velocity™ Gigabit Ethernet (PCI companion controller) VIA integrated 10/100 Fast Ethernet	VIA Velocity™ Gigabit Ethernet (PCI companion controller) VIA integrated 10/100 Fast Ethernet
Serial ATA	2 x SATA 150 devices SATALite™ interface for 2 additional SATA devices (4 total)	2 x SATA 150 devices SATALite™ interface for 2 additional SATA devices (4 total)	2 x SATA 150 devices SATALite™ interface for 2 additional SATA devices (4 total)	2 x SATA 150 devices SATALite™ interface for 2 additional SATA devices (4 total)
V-RAID	RAID 0, RAID 1, and RAID 0+1* & JBOD (SATA)	RAID 0, RAID 1, and RAID 0+1* & JBOD (SATA)	RAID 0, RAID 1, and RAID 0+1* & JBOD (SATA)	RAID 0, RAID 1, and RAID 0+1* & JBOD (SATA)
IDE	ATA133 (up to 4 devices)			
USB 2.0	8 ports	8 ports	8 ports	8 ports
PCI Devices/Slots	6 slots	6 slots	6 slots	6 slots
Modem	MC97	MC97	MC97	MC97
I/O Protocols	I/O APIC / LPC Super I/O			
Power Management	ACPI/APM/PCI/PM/HTSTOP	ACPI/APM/PCI/PM	ACPI/APM/PCI/PM	ACPI/APM/PCI/PM

Figuur 7.5: Chipsets

datatransfer uit te voeren. Dit fenomeen wordt ook wel bus-mastering genoemd, aangezien het apparaat even de baas wordt van het kanaal. Bus mastering gaat veel sneller dan Third-Party DMA, aangezien de ingebouwde DMA controllers van de apparaten veel nieuwer en sneller zijn.

DMA kanalen

Bij het opstarten van een PC worden dikwijls de DMA kanalen getoond. Deze kanalen worden gebruikt om de data tussen het werkgeheugen en het DMA apparaat uit te wisselen. Er zijn in totaal 7 bruikbare DMA kanalen. Merk op dat deze enkel op de ISA bus worden gebruikt. Elk DMA kanaal heeft een 16 bits adresregister en 16 bit telregister. Om een transfer te laten verlopen stelt de DMA controller het juiste adresregister en telregister in, samen met de richting van de transfer (van/naar het werkgeheugen). Daarna wordt het desbetreffende apparaat gesignaliseerd dat de transfer kan beginnen.

Sommige apparaten hebben een vast DMA kanaal, zoals de floppy die op DMA kanaal 2 hangt.

Hoofdstuk 8

Permanente magnetische gegevensopslag

Inhoudsopgave

8.1	Algemene principes van magnetische geheugens	81
8.1.1	Fysische eigenschappen	81
8.1.2	Dichtheid van (magnetische) materialen	83
8.1.3	Lees- en schrijfkop bij magnetische geheugens	84
8.1.4	Indeling van magnetische schijven	85
8.2	Magnetische tapes (magneetbanden)	87
8.2.1	Geschiedenis	87
8.2.2	Organisatie van gegevens op magneetband	88
8.2.3	Tape drives	89
8.2.4	Tapes als backup	89
8.3	Floppy disks (diskettes)	90
8.3.1	Geschiedenis	90
8.3.2	De $5\frac{1}{4}$ inch floppy disk	90
8.3.3	De $3\frac{1}{2}$ inch floppy disk	90
8.3.4	Floppy disk drive	92
8.3.5	Lees- en schrijfkop	93
8.3.6	Toekomst	93
8.4	Harde schijven	93
8.4.1	Geschiedenis	93
8.4.2	Opbouw van een moderne harde schijf	94
8.4.3	Opbouw van de magnetische schijven	95
8.4.4	Luchtcirculatie en koeling	98
8.4.5	Snelheid van harde schijven	100
8.4.6	Evolutie	101

8.5 Geperfectioneerde gegevensopslag: RAID	101
8.5.1 Inleiding	101
8.5.2 RAID concepten	103
8.5.3 RAID niveau's	105
8.5.4 RAID controllers	109
8.5.5 Praktijk	110

In dit hoofdstuk bekijken we permanente magnetische geheugens. Dit zijn geheugens die ook zonder stroom hun informatie kunnen behouden.

8.1 Algemene principes van magnetische geheugens

Wanneer we informatie willen opslaan op een medium en -nog belangrijker- deze informatie willen behouden bij stroomuitval, moeten we gebruik maken van eigenschappen van materialen. Zo is magnetisme een fysische eigenschap van een materiaal die zeer interessant is bij gegevensopslag. Er bestaan namelijk materialen die hun magnetisme kunnen *bewaren*, wat kan leiden tot het bewaren van informatie.

8.1.1 Fysische eigenschappen

Soorten magnetismen

In het algemeen kunnen we materialen onderbrengen in 3 categorieën:

- **Diamagneten:** dit zijn materialen die niet of zeer weinig beïnvloed worden door magnetische velden. Typische voorbeelden zijn hout en glas.
- **Paramagneten:** dit zijn materialen die gemagnetiseerd worden door een magnetisch veld, maw. de elektronen nemen de richting van magnetisatie over. Bij het wegvalLEN van het magnetisch veld valt echter ook hun magnetisatie weg. Voorbeelden van paramagnetische materialen zijn aluminium en platium.
- **Ferromagneten:** deze materialen worden ook gemagnetiseerd door een magnetisch veld en bewaren deze magnetisatie wanneer het magnetische veld wordt opgeheven. Voorbeelden zijn ijzer en nikkel.

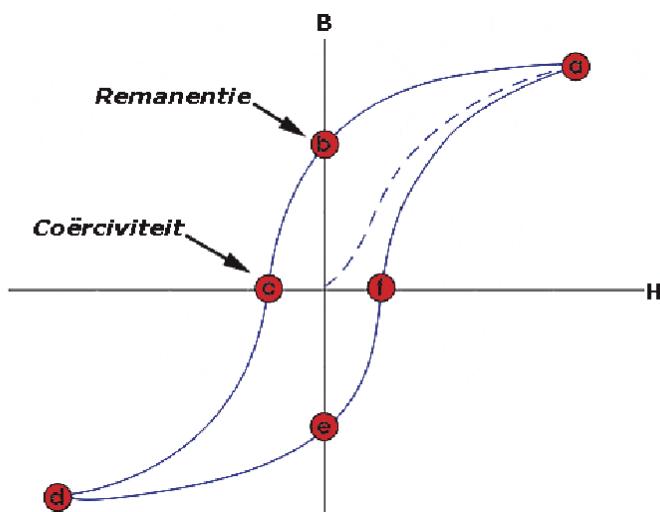
Het is vrij duidelijk dat ferromagnetische materialen interessant zullen zijn om gegevens te bewaren. Zij hebben sterke magnetische eigenschappen en verliezen hun magnetisatie niet wanneer het aanwezige magnetische veld wordt verwijderd. Elektronen in een materiaal creëren een klein magnetisch veldje wanneer ze rond de nucleus (kern) van een atoom draaien. Meestal heffen deze kleine magnetisaties elkaar op, maar bij ferromagneten wordt een ordening van de elektronen op atomisch niveau verkregen. Verschillende elektronen staan nu in dezelfde richting en vormen samen een zogenaamd magnetisch domein. Binnen zo'n domein is er magnetisatie in een bepaalde richting, maar tussen verschillende domeinen kan de richting van magnetisatie verschillen.

Hysteresis

Het “onthouden” (vasthouden) van magnetisatie kan sterk of zwak zijn, deze eigenschap noemt men hysteresis. Van een bepaald materiaal kunnen we een hysteresis-lus opstellen, waarin twee belangrijke punten kunnen worden aangeduid:

- **Coerciviteit:** dit is het punt waarop er geen enkele interne magnetisatie is.
- **Remanentie:** dit is de interne magnetisatie die blijft behouden na het wegvalLEN van het externe magnetische veld.

Figuur 8.1 toont de hysteresis-lus van een ferromagnetisch materiaal. Op deze grafiek zijn er



Figuur 8.1: Hysteresis-lus voor een ferromagnetisch materiaal

twee assen: de X-as stelt het externe magnetisch veld voor (H), de Y-as het interne magnetische veld (B). We vertrekken in volgende bespreking op het nulpunt, d.w.z. het kruispunt van de twee assen.

- Op het nulpunt is er geen interne of externe magnetisatie. Wanneer het externe magnetische veld H wordt verhoogd, komen we terecht in punt a.
- Punt a: het interne magnetische veld B kan niet meer vergroot worden, er treedt een punt van verzadiging op. Ook al vergroten we het externe magnetische veld H , toch zal de interne magnetisatie dezelfde blijven.
- Punt b: wanneer we het externe magnetische veld H wegnemen (stilaan laten afnemen), komen we terecht in punt b. Dit punt noemt men ook wel *remanentie*, aangezien het externe magnetische veld H volledig weg is, maar er toch een interne magnetisatie overblijft. Dit is dan ook het zogenaamde geheugen-effect dat we willen bereiken om uiteindelijk informatie te bewaren. Wanneer we nu een omgekeerd magnetisch veld (in de andere richting) aanbrengen, zal de interne magnetisatie B terug afnemen, we komen nu in punt c.

- Punt **c**: op dit punt is de interne magnetisatie volledig weg, dit noemt men *coerciviteit*. Wanneer het externe magnetische veld wordt opgedreven (nog steeds in de tegengestelde richting in vergelijking met het initieel aangebrachte magnetische veld) neemt de interne magnetisatie in deze richting weer toe. Op de grafiek is dit te zien wanneer we vanaf punt c naar punt d gaan. De interne magnetisatie B wordt negatief, wat duidt op een magnetisatie in de omgekeerde richting.
- Punt **d**: ook wanneer we een omgekeerd magnetisch veld aanbrengen, en we blijven dit externe magnetische veld versterken, zal er ooit een punt komen waarop de interne magnetisatie niet meer sterker wordt. Dit is het geval in punt d, waar dus opnieuw verzadiging optreedt.
- Punt **e**: wanneer het externe magnetische veld weer wordt weggenomen, komen we op het punt e, waar we opnieuw *remanentie* bekomen. Het verschil met de remanentie uit punt b is dat deze remanentie in de omgekeerde richting wordt behouden (lees: het magnetische veld is omgedraaid tov. punt b).
- Punt **f**: bij het opnieuw aanbrengen van een magnetisch veld H, zoals we dat ook initieel hebben gedaan, komen we terecht op punt f waar er geen enkele interne magnetisatie is. Dit noemen we *coerciviteit*.
- Punt **a**: wanneer het externe magnetische veld H verder wordt opgedreven komen we opnieuw uit op het punt van verzadiging. De lus is nu compleet.

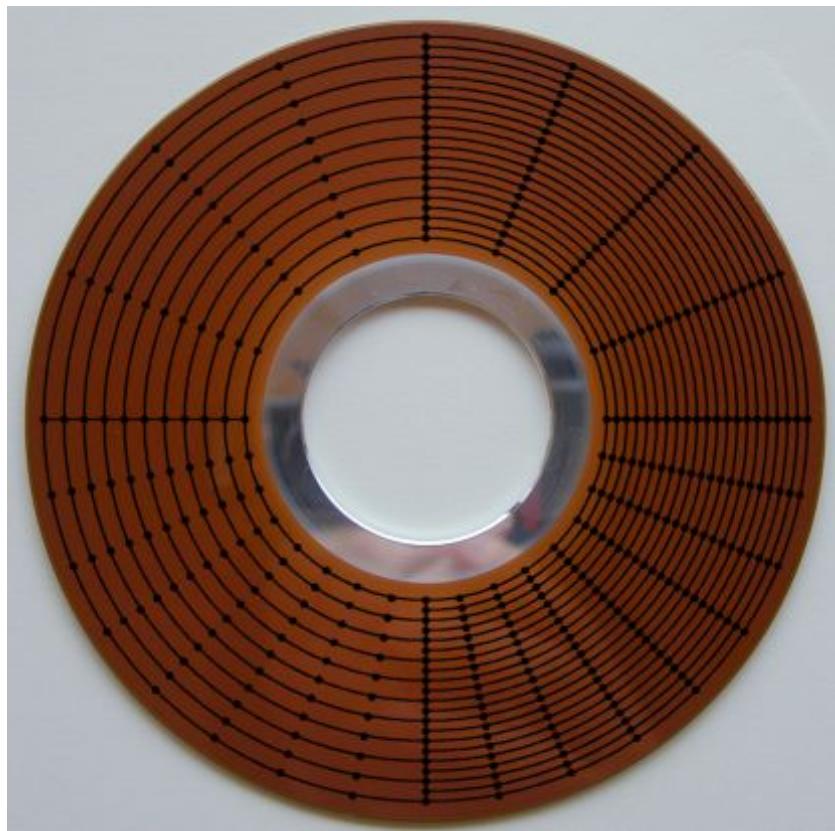
De sterkte van een magnetisch materiaal wordt meestal berekend aan de hand van het product van de coerciviteit en remanentie. Hoe groter de remanentie, hoe meer de twee mogelijke toestanden in b en e van elkaar verschillen. Hoe meer ze van elkaar verschillen, hoe beter ze te onderscheiden zijn. Dit is van belang om later de informatie terug uit te kunnen lezen. Voldoende coerciviteit zorgt voor de stabiliteit van de twee toestanden, zodanig dat kleine magnetische stoornissen (magnetisch ruis) geen invloed hebben.

8.1.2 Dichtheid van (magnetische) materialen

De oppervlakte-dichtheid van een materiaal geeft weer hoeveel informatie er op een bepaalde hoeveelheid plaats past. Dit noemt men ook wel informatie-dichtheid. Meestal wordt deze informatie-dichtheid uitgedrukt in bits/inch^2 (bits per square inch, bits per vierkante inch waarbij 1 inch = 2,54 cm). De oppervlakte-dichtheid van een materiaal wordt meestal als volgt berekend:

$$\text{Oppervlaktedichtheid} = \text{TPI} \times \text{BPI}$$

Hierbij is TPI het aantal tracks per inch (aantal sporen per inch) of ook wel de track density genoemd. BPI is het aantal bits per inch per track, of ook wel de lineaire densiteit (recording density). Figuur 8.2 maakt een en ander duidelijk. Op deze figuur is duidelijk te zien dat het rechtergedeelte een hogere track density (track = spoor) heeft dan het linkergedeelte. Er zijn duidelijk meer sporen (concentrische cirkels) te zien. Daarnaast kan ook vastgesteld worden dat het bovenste gedeelte van deze schijf een lagere lineaire densiteit heeft dan het onderste gedeelte. Inderdaad, op het bovenste gedeelte komen meer bolletjes voor op een bepaalde lengte, dan in het onderste gedeelte. Belangrijk om te weten is dat op een fysisch medium (harde schijf, floppy disk, ...) de lineaire densiteit varieert van spoor tot spoor.

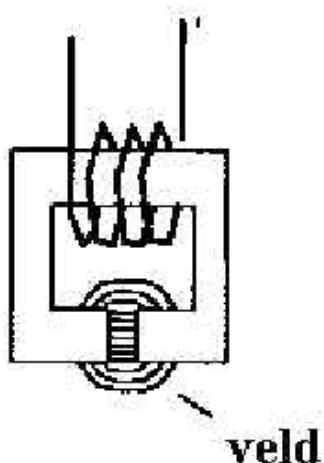


Figuur 8.2: Informatie-dichtheid

8.1.3 Lees- en schrijfkop bij magnetische geheugens

De lees- en schrijfkoppen bij magnetische geheugens zijn gebaseerd op een zeer eenvoudig concept: een stukje ijzer in U-vorm wordt omwonden met een elektrische spoel om op die manier een klassieke elektromagneet te maken. Deze kop baseert zich op een bepaald principe uit de fysica: magnetische flux. Een stroom in een spoel brengt immers een magnetische flux te weeg in de kern. Dit geldt in het algemeen voor geleiders: een geleider waar stroom door vloeit genereert een magnetisch veld, waarvan de richting volgens de regel van de kurkentrekker kan bepaald worden. Figuur 8.3 maakt duidelijk hoe deze kop (ruwe schets) gefabriceerd werd. Het magnetische veld kan in 2 richtingen worden opgewekt, naargelang de richting waarin de stroom vloeit. We zien duidelijk dat we hier een extern magnetische veld aanmaken, wat een interne magnetisatie zal te weeg brengen in het magnetiseerbare materiaal (zie 8.1.1). Deze magnetische polarisatie van de cel onder de lees- en schrijfkop zorgt voor de opslag van een magnetisatierichting. Bij het uitlezen gebeurt net het omgekeerde: het bewegend magnetisch materiaal wekt een stroomje op in de spoel. Het aldus verkregen signaal is een zeer zwakke stroom in 1 van de 2 richtingen, dat onmiddellijk versterkt moet worden om het te kunnen aflezen. Er zijn een heleboel factoren die een rol spelen bij de goede werking van het lezen en schrijven:

- Hoe sterker de stroom is die men door de spoel stuurt, hoe sterker het opgewekte magnetische veldje, maar hoe meer energie er zal gebruikt worden...
- Het aantal windingen van de spoel speelt ook een rol, alsook het materiaal waaruit de



Figuur 8.3: Een eenvoudige voorstelling van een elektromagnetische kop

kern van de spel gemaakt is.

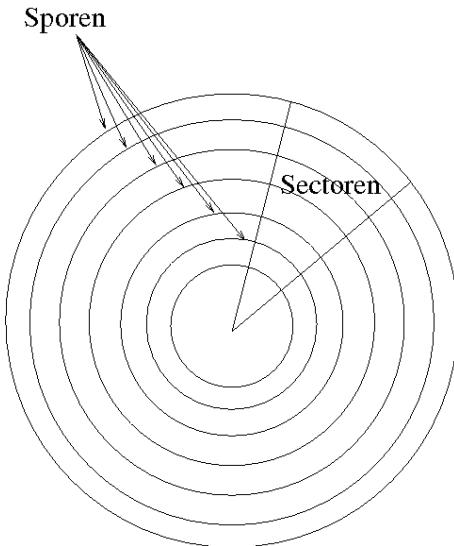
- Hoe dunner het magnetiseerbare materiaal is, hoe beter.
- Hoe gevoeliger de laag, hoe beter.
- De afstand van de kop tot de schijf speelt een dubbele rol:
 - Hoe dichter de kop bij de schijf komt, hoe sterker het magnetische veld op de schijf.
 - Hoe dichter de kop bij de schijf, hoe kleiner de oppervlakte van de schijf die door het schrijven beïnvloed wordt, hoe kleiner dus de geheugenelementen, hoe dichter men ze bij elkaar kan plaatsen. Dit leidt tot een grotere informatiedichtheid.

Belangrijk is te vermelden dat vele moderne lees- en schrijfkoppen (anno 2012) niet meer volledig gebaseerd zijn op het bovenstaande concept. Met name de lees- en schrijfkoppen van harde schijven zijn enorm geëvolueerd. Zo worden tegenwoordig magneto-resistieve koppen gebruikt, die niet meer met magnetische inductie werken maar die een totaal ander fysisch fenomeen hanteren: magneto-resistiviteit. Wanneer een magneto-resistieve kop over het schijfoppervlak passeert, zal deze boven bepaalde magnetisaties komen. De weerstand van de leeskop wordt hierdoor beïnvloed, dit wordt gemeten en zo kan data worden uitgelezen. Magneto-resistiviteit wil dus zeggen dat de weerstand (resistiviteit) verandert onder invloed van een magnetisch veld. De huidige harde schijven hebben dan ook 2 koppen die onzettend dicht bij elkaar staan: een leeskop die volgens het principe van magneto-resistiviteit werkt en een schrijfkop die volgens het dunne-film inductie principe werkt.

8.1.4 Indeling van magnetische schijven

Sporen, sectoren en cilinders

We bespreken hier algemeen de opdeling van magnetische schijven. Deze worden zowel gebruikt bij harde schijven als bij floppies. Een magnetische schijf is telkens opgedeeld in sporen, elk spoor is verder opgedeeld in een aantal sectoren. Dit wordt duidelijk in figuur 8.4. Wanneer



Figuur 8.4: Indeling van magnetische schijf in sporen en sectoren

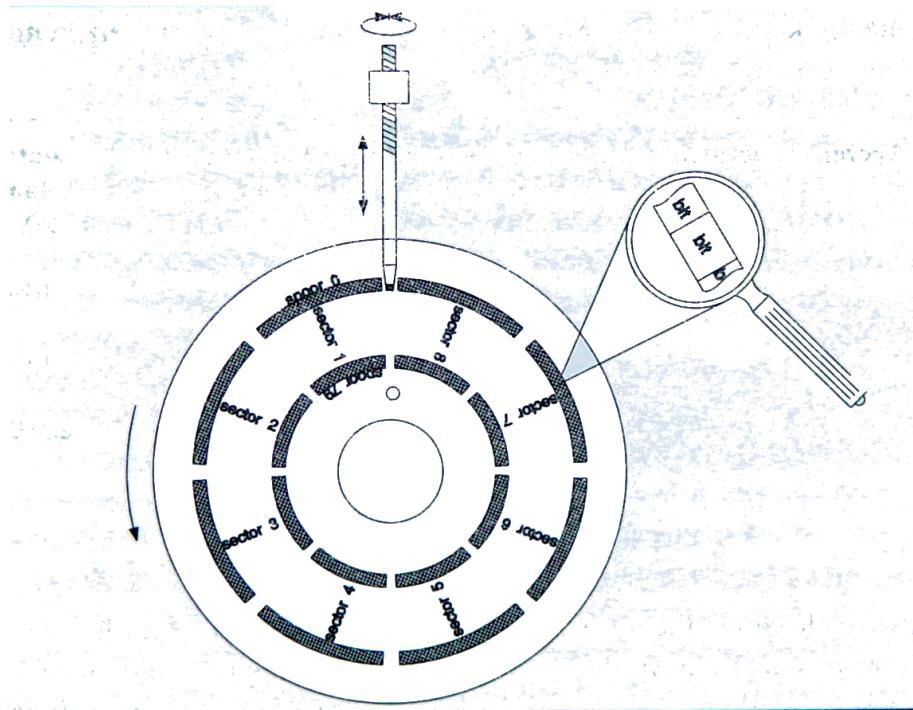
meerdere schijven boven elkaar worden geplaatst (bv. bij harde schijven) noemt men meerdere sporen boven elkaar een cilinder. De onderverdeling in sectoren kan op 2 manieren gebeuren:

- *Harde sectorindeling*: het begin van de sectoren wordt voor eens en voor altijd door constructie vastgelegd. Bij de floppies, die we bekijken in sectie 8.3 wordt harde sectorindeling gebruikt. Typische grootte-orden voor de capaciteiten per sector zijn 128, 256, 512 of 1024 informatiebytes per sector. De floppy gebruikt er bijvoorbeeld 512.
- *Zachte sectorindeling*: de onderverdeling van de sectoren wordt bepaald door informatie die op de schijf zelf is opgeslagen. Zulk een schijf bevat bij constructie nog geen informatie in de vorm van gaten of inkepingen, die informatie wordt er achteraf door de computer op aangebracht, meestal bij het installeren van het schijfgeheugen, in de vorm van bitpatronen. Uiteraard verliezen we een klein deel van de geheugencapaciteit door de opslag van deze formatteringsgegevens. Oude harde schijven moesten na aanschaf door de gebruiker geformatteerd worden (low-level formatting, zie verder). Moderne harde schijven worden vanuit het fabriek voorgeformatteerd, al gebeurt dit ook software-matig.

Interleaving

Bij de eerste magneetschijfgeheugens waren de controllers die de gegevens effectief inlazen en doorgaven aan de processor nog relatief traag. Dit zorgde ervoor dat wanneer een sector was ingelezen, het een tijdje duurde voor deze gegevens verwerkt waren en de volgende sector kon worden ingelezen. Door deze wachttijd, was het niet mogelijk om alle sectoren direct naast elkaar op de schijf te nummeren. Als we namelijk de naast elkaar liggende sectoren nummeren als volgt: 0, 1, 2, ... komen we in de problemen. Tegen de tijd dat de ingelezen sector 0 was behandeld, was door de rotatie van de schijf sector 1 al gepasseerd. Hierdoor moest een volledige rotatie gewacht worden alvorens sector 1 opnieuw onder de schijf passeeerde. De oplossing voor dit fenomeen was interleaving: er werden net zoveel sectoren tussengelaten als nodig voor de verwerking van een bepaalde sector. Stel dat het verwerken van een sector zoveel tijd inneemt dat ondertussen 4 andere sectoren onder de leeskop gepasseerd zijn. We

kunnen dan een nummering als volgt nemen: 1, 6, 3, 8, 5, 2, 7, 4. Dit noemen we een 5-to-1 interleaving. Dit wordt duidelijk in figuur 8.5. Interleaving wordt bij moderne harde schijven niet meer toegepast, aangezien de moderne controllers de hoeveelheid gegevens gemakkelijk kunnen verwerken zodat dadelijk de volgende sector kan worden ingelezen. Ze gebruiken dus een 1-to-1 interleaving (= geen interleaving). Bij floppies wordt interleaving nog wel gebruikt.



Figuur 8.5: Interleaving: de sectoren worden op een speciale manier genummerd

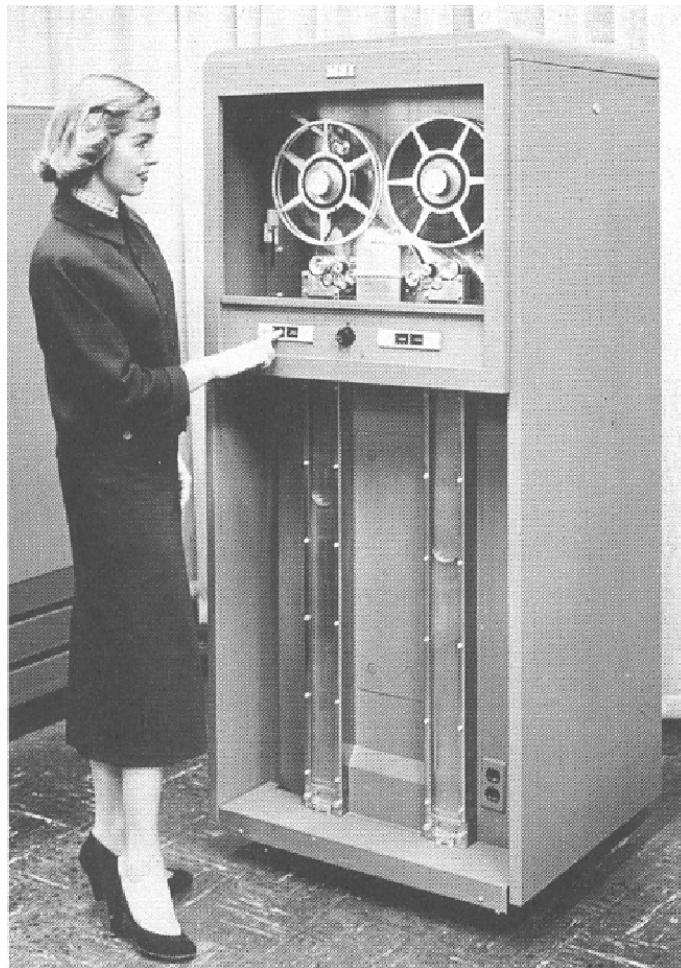
8.2 Magnetische tapes (magneetbanden)

8.2.1 Geschiedenis

Magnetische tapes maken gebruik van een cassettes met daarop een plastic lint, waarop een magnetiseerbare laag zit. We kennen de magnetische tapes van overal: de audio-cassettes, video cassettes (VHS), digital audio tape (DAT), digital linear tape (DLT), ... De audio-cassette werd zelfs lange tijd gebruikt bij PC's begin jaren '80.

De magneetband werd in 1889 uitgevonden door de Deen Valdemar Poulsen. Hij werd in het begin voornamelijk gebruikt voor het opnemen en vertraagd overseinen van telefonische boodschappen, later ook voor het opnemen van muziek en dergelijke. Sinds 1940 worden magneetbanden gebruikt in de informatieverwerking. In het begin gebeurde dit in analoge vorm, zoals voor het registreren van muziek en gesproken taal. Tegenwoordig werken magneetbanden digitaal, de bits worden voorgesteld door magnetische vlekjes.

Magneetbanden zijn de oudste computergeheugenmedia. Sommige computers uit de vijftiger jaren hadden een magneetband als centraal geheugen (werkgeheugen!), eigenlijk zelfs als enige geheugen. Figuur 8.6 toont de eerste commerciële IBM tape drive (let ook op de uitstekende marketing van IBM...), de IBM 701.



Figuur 8.6: De IBM 701 tape drive

8.2.2 Organisatie van gegevens op magneetband

Vanwege het kleine aantal sporen en het feit dat er maar 1 leeskop aanwezig is, verschillen banden grondig van magneetschijven (harde schijven, zie 8.4) en optische schijven. We hebben hier te maken met een volledig sequentieel geheugen. De informatie op een magneetband is niet adreseerbaar. Wanneer we bepaalde gegevens willen bereiken, moeten we de tape laten afspelen tot we aan de gevraagde gegevens komen. De informatie wordt op de band geplaatst in blokken. Deze blokken een vaste lengte geven zou een belemmering betekenen in het gebruik en zou als gevolg hebben dat heel wat blokken slechts gedeeltelijk zouden gevuld zijn. Men koos daarom voor blokken van variabele lengte. Op het einde van elke blok worden ook een aantal testbits voorzien. Men spreekt dikwijls van horizontale foutencontrole. Tussen twee blokken wordt een blokhiaat (ruimte) gelaten die 1 tot 2 cm lang is. Een blokhiaat is een onbeschreven gedeelte van de band met in het begin en op het einde een speciaal teken dat door de leeskop herkend wordt. Het hiaat heeft tot doel een zekere tijd te voorzien voor het afremmen en het terug op snelheid brengen van de band. Ondertussen kan de foutencontrole uitgevoerd worden. Een bijkomende reden is dat er door de hoge snelheid altijd een zekere rek van de band optreedt waardoor het onmogelijk zou zijn de band plots op een bepaalde positie te laten stoppen. Vanwege de wisselvallige lengte van de blokken zal men een magneetband zelfs niet

blokadresseerbaar kunnen maken, zoals een harde schijf. Een magneetband is helemaal niet adresseerbaar. Tenslotte kan nog gezegd worden dat vanwege de openingen er een belangrijk rendementsverlies van de band optreedt.

8.2.3 Tape drives

De toestellen die worden gebruikt om tapes te beschrijven en te lezen noemen we tape drives. Dit zijn machines die de tape van 1 wiel naar een ander wiel kunnen brengen, met een zeer fijne en gesofisticeerde motor. Op de moderne tape drives zijn de twee wielen niet heel duidelijk meer te zien, aangezien ze geïntegreerd zitten in de cassette. De eerste tape drives konden een densiteit aan van 556 karakters per inch. Er is een zeer ingewikkelde afweging nodig tussen de gebruikte blokgroote, de grootte van de gebruikte databuffer in de tape drive, het percentage van de tape dat verloren wordt door inter-blok hiaten om tot een uiteindelijke schrijf/lees performantie te komen.

8.2.4 Tapes als backup

Alhoewel tapes een oude en omslachtige manier van data-opslag lijken, worden ze nog steeds zeer veel gebruikt. Het voordeel aan tapes is dat ze relatief goedkoop zijn, een hoge densiteit hebben, eenvoudig off-site te halen zijn (naar een andere fysieke locatie) en ook eenvoudig kunnen gewisseld worden in de zogenaamde automatische tapewisselaars. Het is om deze redenen dat tapes het meest populaire backup-medium vormen. Hierbij wordt meestal gebruik gemaakt van een compressiemethode om de data kleiner te maken alvorens ze weg te schrijven. Het marktsegment van de tapes is echter stilaan aan het stagneren, in tegenstelling tot innovaties op het gebied van harde schijven. Het is dan ook erg waarschijnlijk dat het belang van tapes voor backups op termijn zal wegebben. Figuur 8.7 toont een IBM 3592 tape drive, die tot 40 MB per seconde kan opslaan. Het is duidelijk dat alleen al binnen IBM de evolutie van de tape drives enorm is! Tegenwoordig worden ook optische media (DVD, Blu-Ray) en vooral harde schijven gebruikt als backup-medium. Harde schijven hebben als groot voordeel dat ze veel sneller werken dan optische media of tapes. Het nadeel van harde schijven is dan weer dat ze relatief gemakkelijk beschadigd worden bij transport (off-site halen van backup) en dat hun betrouwbaarheid op lange termijn nog niet zo goed gekend is.



Figuur 8.7: De IBM 3592 tape drive

8.3 Floppy disks (diskettes)

Floppy disks, ook bekend als floppies of diskettes, werden zeer veel gebruikt in de jaren '80 en '90. Ze waren het middel bij uitstek voor het uitwisselen van bestanden en software en voor het maken van backups. Gedurende lange tijd -voor de introductie van harde schijven- werden floppies zelfs gebruikt om het besturingssysteem op te bewaren, samen met software en andere belangrijke data. Een typisch voorbeeld hiervan was het populaire besturingssysteem DOS.

8.3.1 Geschiedenis

Rond 1967 kreeg de opslagafdeling van IBM de opdracht een systeem te ontwikkelen om de microcode van de System/370 mainframes van te kunnen laden. Deze 370 machines waren de eerste IBM machines met halfgeleider geheugens, die hun informatie verloren wanneer de stroom werd uitgeschakeld. De tape drives bestonden al, maar IBM wou een sneller systeem dat ook eenvoudig kon gebruikt worden op updates van het besturingssysteem naar hun klanten te sturen.

David Noble, die onder het management van Alan Shugart werkte, probeerde een aantal zaken uit. Initieel herwerkten ze de bestaande tape drives, maar dit idee werd al gauw opgegeven. Het resultaat van hun onderzoek was een 20 cm (8 inch) grote floppy die ze "memory disk" noopten, en een opslagcapaciteit van 80 KB (kilobytes!) had. Het eerste ontwerp bevatte geen bescherming, maar al snel werd een plastieken enveloppe gebruikt om de schijf te vrijwaren van stof en vuil.

De Japanse uitvinder Yoshiro Nakamatsu claimde dat hij dit principe reeds eerder had uitgevonden, in 1950, en hierdoor moest IBM een verkoopslicentie aankopen bij het produceren van de eerste floppy disk systemen. Later volgden al gauw nieuwe varianten van deze eerste floppy, in verschillende maten. In de volgende secties bespreken we de meest gebruikte.

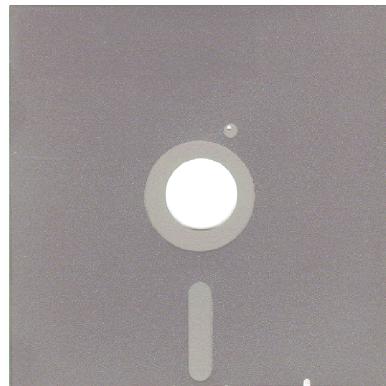
De naam floppy komt van de buigzame zachte schijf die gebruikt werd. Deze schijf werd gemagnetiseerd, zoals reeds eerder aangegeven.

8.3.2 De $5\frac{1}{4}$ inch floppy disk

Deze 5,25 inch floppy disk werd in de jaren '80 het meest populair opslagmedium. Het werd dan ook verkocht bij het gros van de personal computers. Eens het besturingssysteem was ingeladen van een floppy, werden andere floppies gebruikt om applicatiesoftware in te laden. Deze 5,25 inch floppy disks hadden een geheugencapaciteit van 360 KB. Aan het einde van de jaren '80 geraakte de 5,25 inch floppy van de markt, aangezien zijn opvolger -de 3,5 inch- een grotere opslagcapaciteit had en een grote doorbraak kende. Figuur 8.8 toont een 5,25 inch floppy disk. Een groot nadeel van de 5,25 inch floppy is dat hij buigzaam was, waardoor data kon verloren gaan wanneer een floppy per ongeluk ergens tussen verzeild geraakte.

8.3.3 De $3\frac{1}{2}$ inch floppy disk

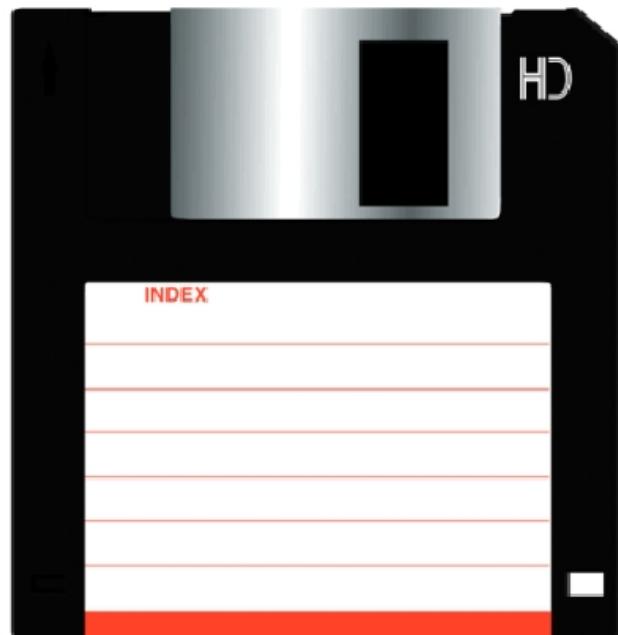
Doorheen de jaren '80 werden de beperkingen van de 5,25 inch floppies duidelijk, toen machines sneller en sneller werden. Een aantal nieuwe floppy-varianten werden ontwikkeld, waaronder de 2 inch, de 2,5 inch, de 3 inch en 3,5 inch. Ze hadden allen een groot voordeel tegenover de 5,25



Figuur 8.8: De 5,25 inch floppy

inch floppy: de stevigheid was enorm verbeterd. Daarnaast werd ook een write-protect slider toegevoegd, waarmee men een diskette kon beveiligen tegen overschrijven.

Toen Apple computer in 1984 het Sony formaat (90x94mm) koos voor zijn Macintosh computers, werd deze grootte de facto standaard in de Verenigde Staten. Deze nieuwe 3,5 inch diskettes hadden een metalen beschermingskap, die ervoor zorgt dat de interne buigzame disk niet kan beschadigd worden. De 3,5 inch diskette kende een ware revolutie: initieel konden deze floppies slechts 360 KB (single-sided) aan data opslaan, later 720 KB (double-sided double-density) en de meest geavanceerde versie 1440 KB (high-density). Figuur 8.9 toont een 3,5 inch floppy disk.



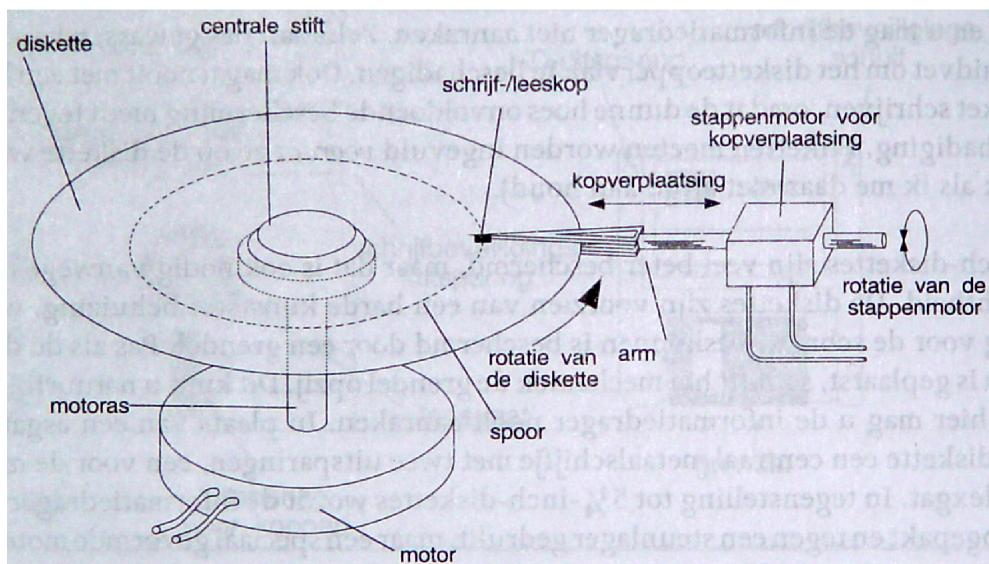
Figuur 8.9: De 3,5 inch floppy

8.3.4 Floppy disk drive

De floppy disk drive is het elektromechanische apparaat waarmee floppies kunnen beschreven en uitgelezen worden. De floppy disk drive bevat een klikmechanisme dat de metalen beveiliging wegschuift van zodra een floppy in de drive wordt gestoken. Daarna kan de floppy gebruikt worden voor lezen en schrijven. Belangrijke onderdelen in de floppy drive zijn:

- Lees- en schrijfkoppen: aan beide zijden van de diskette bevinden zich de koppen, die in hetzelfde frame zijn vastgemaakt. Ze bewegen dus steeds simultaan. De koppen zitten niet direct op beide zijden op dezelfde plaats, om te vermijden dat er interactie is tussen twee schrijf operaties aan beide kanten. Dezelfde kop wordt gebruikt voor lezen en schrijven en een tweede, bredere kop wordt gebruikt om de diskette te wissen. Dit kan gebeuren juist voordat de schrijfkop het materiaal opnieuw beschrijft. Dit laat toe om de data te schrijven op een “schone lei”.
- Drive motor: een zeer kleine roterende motor die centraal op de diskette zit, en die ervoor zorgt dat de floppy draait aan 300 of 360 toeren per minuut.
- Stepper motor: deze motor zorgt ervoor dat de lees- en schrijfkop heen en weer kan bewegen naar het juiste spoor.
- Mechanische frame: het frame dat de beschermende metalen slider opzij schuift.
- Circuit board: de printplaat waarop alle nodige elektronica vastzit.

De interne lees- en schrijfkop van een floppy disk drive is opgebouwd zoals aangegeven in figuur 8.10.



Figuur 8.10: Interne werking van floppy disk drive

8.3.5 Lees- en schrijfkop

De interne lees- en schrijfkop van een floppy disk drive raakt het oppervlak van de floppy disk. Dit is dan ook de reden dat een floppy disk zo traag werkt. Omdat de lees- en schrijfkop de floppy raakt, kan de floppy niet zeer snel draaien aangezien dit te veel warmte-ontwikkeling met zich mee zou brengen. Hierbij zouden de fysische eigenschappen van het plastiek wijzigen waardoor gegevens verloren kunnen gaan of zelfs het plastieken schijfje zou smelten. Het is om deze reden dat de floppy maar aan snelheid van 300 of 360 toeren per minuut draait, dit alleen wanneer er geschreven of gelezen wordt van de floppy (wanneer er niet gelezen of geschreven wordt staat de floppy stil).

8.3.6 Toekomst

Floppies worden vandaag de dag minder en minder gebruikt. Ze brengen soms nog soelaas bij een gecrashte computer, waarbij de diskette als laatste redmiddel kan gebruikt worden. Spijtig genoeg blijft de betrouwbaarheid van floppies ondermaats, waardoor ze in de toekomst ongetwijfeld zullen verdwijnen in de computerwereld.

8.4 Harde schijven

8.4.1 Geschiedenis

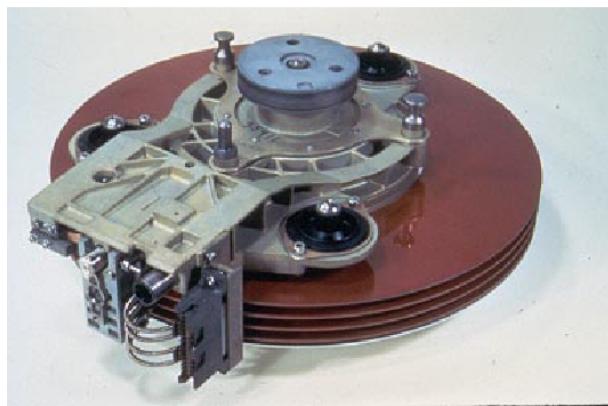
De eerste computer die standaard een harde schijf had, was de IBM 305 RAMAC, die in 1955 op de markt werd gebracht. Figuur 8.11 toont deze grote en “krachtige” (in die tijd althans) machine. RAMAC stond voor “Random Access Method of Accounting and Control”. Deze



Figuur 8.11: De eerste harde schijf in een IBM 305 RAMAC

IBM 305 had 50 schijven die elk 24 inch (60 cm) groot waren. De harde schijf zelf was bekend onder de naam IBM 350. In totaal konden hiermee 5 miljoen karakters opgeslagen worden. Bij deze RAMAC raakten de lees- en schrijfkoppen de schijven, waardoor de schijven op lage snelheid moesten draaien. De schijven draaiden aan een snelheid van 1200 toeren per minuut. In 1962 werd de IBM 1301 schijf uitgebracht, de eerste harde schijf met vliegende koppen. Doordat de koppen boven het schijfoppervlak vlogen, konden hogere snelheden gehaald worden. De IBM 1301 draaide immers aan 1800 toeren per minuut.

Een volgende grote stap kwam er in 1973, toen IBM de Winchester op de markt bracht, met als officiële naam IBM 3340. Figuur 8.12 toont een IBM 3340 Winchester schijf. Er zijn veel



Figuur 8.12: De IBM 3340

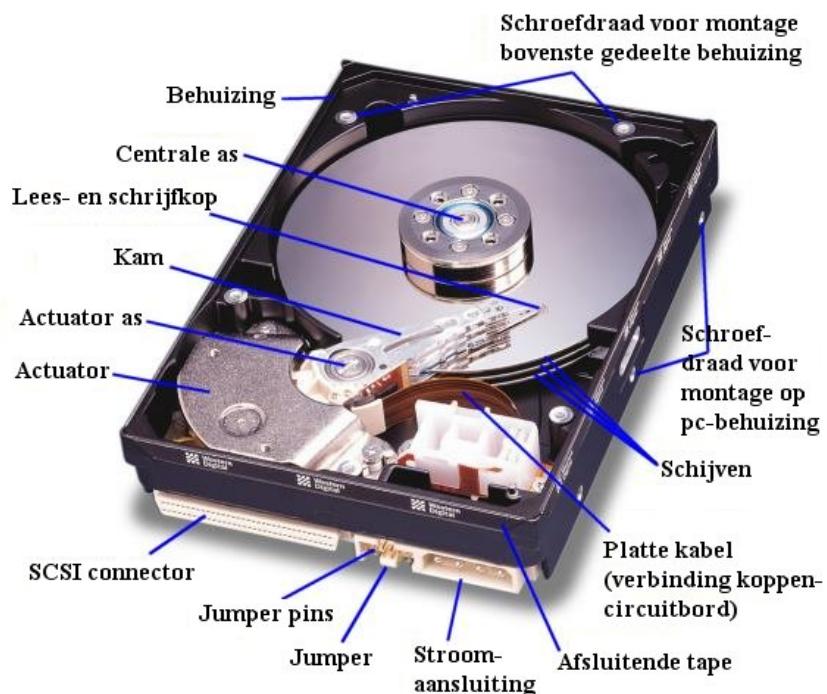
verklaringen waarom deze schijf de naam Winchester meekreeg. Een verklaring is dat IBM in deze naam verwijst naar het bekende Winchester 30-30 geweer, waarbij de eerste 30 verwijst naar het kaliber en de tweede 30 naar het gewicht van de lading. De IBM 3340 had twee 30 MB volumes, wat de gelijkenis volledig maakte. Een andere verklaring is dat IBM de schijf Winchester noemde naar het laboratorium waar deze schijf werd ontwikkeld. De IBM 3340 was de eerste harde schijf dat een volledige afgesloten kop/schijf geheel bevatte (Head/Disk Assembly of HDA). Harde schijven werden nog lange tijd "Winchesters" genoemd, naar dit populaire product van IBM.

Gedurende lange tijd waren harde schijven grote, onhandige apparaten, die enkel geschikt waren voor datacenters of grote kantoren. In industriële omgevingen kon de harde schijf nog niet gebruikt worden, aangezien ze nog te broos en kwetsbaar was. Ook in thuisomgevingen was de harde schijf nog niet in gebruik, wegens hun grootte en hoge energieverbruik.

Pas rond de jaren '80, toen Seagate Technology de ST-506 schijf introduceerde, werd de harde schijf pas echt populair. De ST-506 was de eerste 5,25 inch form factor harde schijf. Dit zeggen dat de grootte van de harde schijf nu een industriële standaard werd. Met zijn 5 MB opslagcapaciteit, werd de ST-506 een van de belangrijke voorlopers van de huidige harde schijf. IBM maakte in zijn eerste personal computers (IBM PC/XT) gebruik van een ST-412 harde schijf, die 10 MB kon opslaan en dezelfde 5,25 inch standaard volgde.

8.4.2 Opbouw van een moderne harde schijf

Harde schijven bestaan uit een pakket magnetische schijven en lees- en schrijfkoppen. Zoals de naam het zelf zegt, worden in dit geval harde, onbuigzame schijven gebruikt om de gegevens

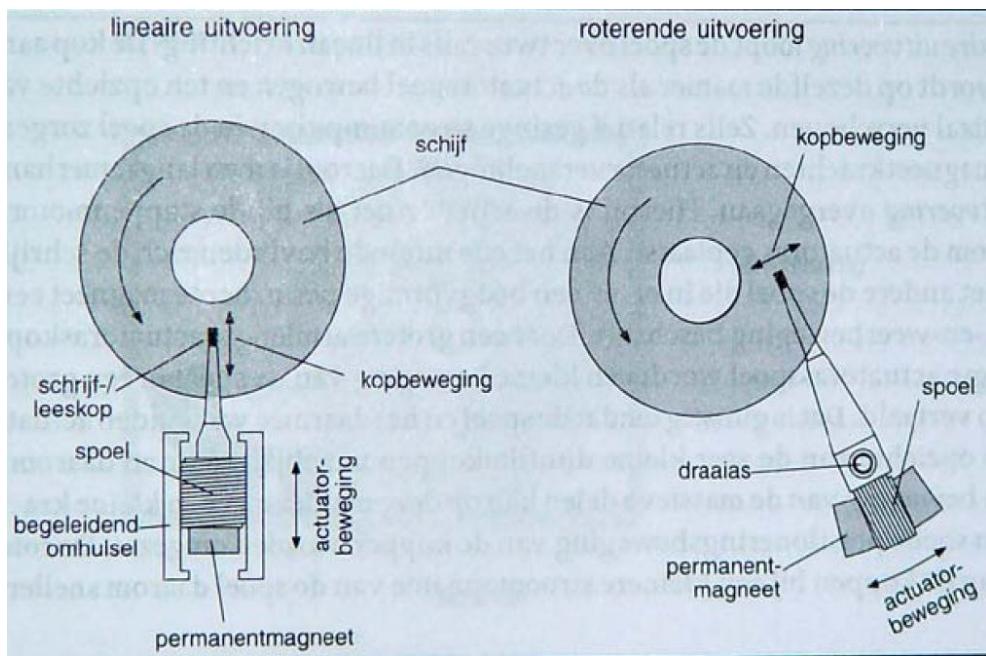


Figuur 8.13: Opbouw van een moderne harde schijf

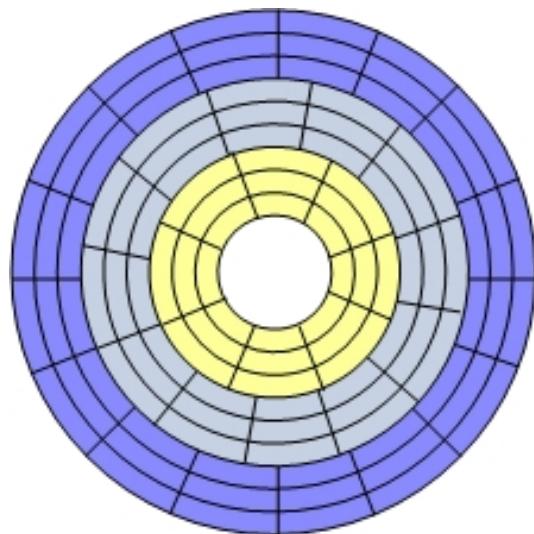
te bewaren. Figuur 8.13 toont de bouw van een moderne harde schijf. Op deze afbeelding kan men zien dat er niet 1, maar meerdere harde “platters” (ronde schijven) boven elkaar worden gemonteerd, om zo nog meer opslagcapaciteit te bieden. Elk van de schijven is aan beide zijden beschrijfbaar en heeft een aparte lees- en schrijfkop. De lees- en schrijfkoppen van de verschillende platters bewegen samen, aangezien ze gezamenlijk op dezelfde kam zijn gemonteerd. Deze kam kan bewogen worden door de actuator. Deze motor werkt duidelijk op een andere manier dan de lineaire motor van een floppy disk drive. De actuator is een roterende motor die slechts over een beperkte afstand kan bewegen (de koppen moeten immers enkel van het midden tot de buitenzijde van de schijven kunnen schrijven). Deze manier van werken zorgt ervoor dat de koppen op een uiterst precieze manier kunnen bewegen. Uiteindelijk worden de lees- en schrijfkoppen verbonden met het circuitbord van de harde schijf, waarop de nodige electronica aanwezig is om het geheel consistent te laten werken. Figuur 8.14 toont nog eens duidelijk het verschil tussen de motorprincipes van floppy disk drives (lineair) en harde schijven (actuator).

8.4.3 Opbouw van de magnetische schijven

Wel vermelden we hier extra dat het aantal sectoren per spoor varieert op moderne harde schijven. Dit is ook logisch, als men figuur 8.4 bekijkt. Op deze figuur kan men duidelijk zien dat een sector op een van de buitenste sporen veel meer informatie zal bevatten dan een spoor op een van de binnenste ringen. In realiteit neemt men geen variabele sector-grootte, maar een variabel aantal sectoren per spoor. Het feit dat het aantal sectoren per spoor kan variëren, noemt men zone bit recording. Hierbij wordt de harde schijf in een aantal zones onderverdeeld. Binnen deze zone is het aantal sectoren per spoor constant. In afbeelding 8.15 wordt het principe van zone bit recording geïllustreerd. Wanneer een harde schijf moet gepartitioneerd



Figuur 8.14: Motor van floppy disk drives (links) en harde schijven (rechts)



Figuur 8.15: Zone-bit recording

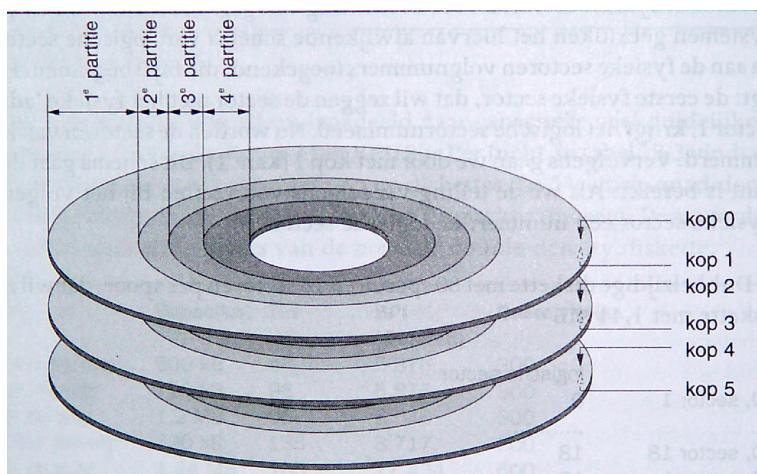
worden, gebeurt dit in 3 stappen:

- **Low-level formatting:** de “echte” formatteringsstap. Deze stap maakt de juiste fysieke structuren aan (sporen, sectoren, controle-informatie). Wanneer deze stap voor de eerste keer (en in de meeste gevallen ook ineens de laatste) wordt uitgevoerd, begint men met een volledige schone lei. Moderne schijven worden 1x low-level geformateerd bij fabricage, en worden nooit meer opnieuw low-level geformateerd. Op een moderne ATA of SCSI harde schijf is het zelfs niet mogelijk om na fabricage nog lower-level te formatteren.
- **Partitionering:** deze stap verdeelt de harde schijf in verschillende logische delen die verschillende harde schijf volumes (partities) zullen vormen op het systeem. Deze stap

wordt meestal door besturingssystemen uitgevoerd.

- **High-level formatting:** deze stap wordt ook in de meeste gevallen uitgevoerd door het besturingssysteem. Hierbij wordt gekozen voor een bepaald formaat (FAT, NTFS, EXT2, EXT3, HPFS, ReiserFS) dat meestal ook samenhangt met het gebruikte besturingssysteem. Hierbij worden logische structuren van de harde schijf opgebouwd en meestal ook opgeslagen op de schijf zelf. Zo wordt bij FAT bijvoorbeeld een allocatietafel van de bestanden opgeslagen op de schijf.

Figuur 8.16 geeft aan hoe de partities over een harde schijf (één harde schijf bevat meerdere magnetiseerbare schijven) verdeeld worden. Wanneer een schijf geformatteerd wordt, en er



Figuur 8.16: Partities op een harde schijf

dus controle-informatie op de schijf wordt bijgehouden, zal sowieso een gedeelte van de opslagcapaciteit verloren gaan. Dit verschil kan tot 20% bedragen! Vele fabrikanten vermelden de ongeformatteerde capaciteit, de werkelijke capaciteit is dus kleiner. Daarnaast geven de fabrikanten de decimale capaciteit op, niet de binaire. Dit heeft te maken met het feit dat 1 kilo (en analoog voor mega en giga) in het decimale talstelsel voor 1000 staat, en in het binaire talstelsel voor 1024. Als de basiseenheid kleiner wordt genomen (decimaal dus), zal een fabrikant kunnen zeggen dat hij meer van die eenheden kan opslagen, in tegenstelling tot grotere eenheden. Zo zal een harde schijf van 72 GB (decimaal uitgedrukt) uiteindelijk maar effectief 67 GB (binair uitgedrukt) kunnen opslagen. Probeer deze berekening zelf ook eens te maken! Dit is uiteraard pure marketing, wat soms tot frustraties van de eindgebruikers kan leiden.

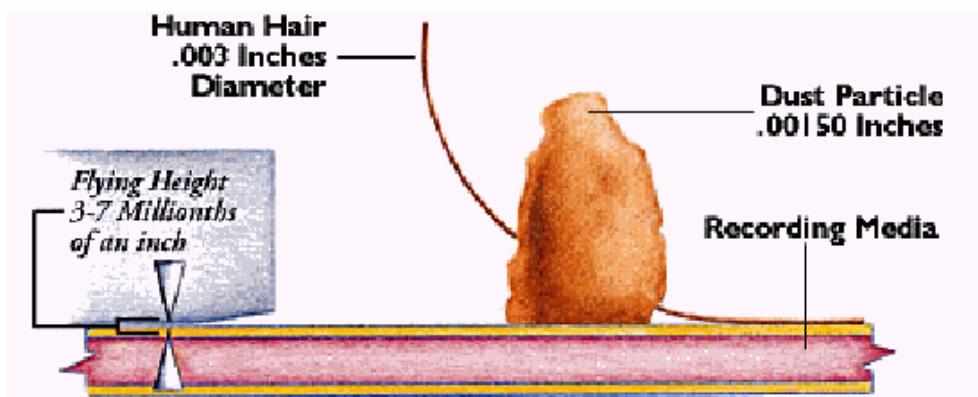
Bij harde schijven wordt dikwijls gebruikt gemaakt van een ECC algoritme om fouten op te vangen. Voor meer informatie over error correcting codes, zie 5.6.2. Een interessante evolutie is de volgende: stel dat we onze bits *nog* dichter bij elkaar zetten op een harde schijf. Hierdoor gaan er waarschijnlijk meer fouten optreden, maar dit kunnen we dan weer opvangen door ECC-algoritmen. Zo zal men dus meer informatie opslagen wanneer men de bits zeer dicht bij elkaar legt en veel ECC-bits gebruikt, ten opzichte van harde schijven waarbij de bits ver uit elkaar liggen en weliswaar minder ECC bits worden toegevoegd. De toekomst brengt ons schrijven, waarbij de bits zeker en vast nog dichter bij elkaar kunnen staan. Degelijke ECC's zullen dus enorm belangrijk worden! De ECC's worden berekend en gecontroleerd in de harde schijf controller, zodat een besturingssysteem hier meestal niets van merkt.

Lees- en schrijfkop

De lees- en schrijfkoppen van harde schijven zijn over de jaren sterk geëvolueerd. Vooral de vlieghoogte is altijd al een belangrijk aspect geweest. Zoals eerder besproken, zal een kop die het schijfoppervlak raakt zeer veel warmte ontwikkelen, waardoor de draaisnelheid eerder beperkt moet blijven. Daarom werd al snel gekozen voor een vliegende lees- en schrijfkop. Hierbij zijn er verschillende mogelijkheden. Men zou de kop kunnen vastmonteren op kam, waarbij een zeer kleine afstand wordt bewaard tussen het oppervlak en de schijf. Dit is uiteraard een fragiel ontwerp, en wordt dan ook nooit gebruikt.

Een moderne lees- en schrijfkop vliegt over het schijfoppervlak zoals een vliegtuigje. De kop bevat ook effectief twee vleugels, die gedragen worden door de luchtverplaatsing die veroorzaakt wordt door de draaiing van de schijf. Zo blijft de lees- en schrijfkop steeds op een constante afstand verwijderd van de schijf. Wanneer de schijf niet meer draait, zal de kop ook niet meer vliegen. In dit geval zal de kop toch op het schijfoppervlak rusten, dit gebeurt op een speciaal daarvoor voorzien spoor (dat overigens geen data bevat). Sommige moderne harde schijven bevatten zelfs een sensor die meet hoe hoog de kop boven het oppervlakte vliegt en kunnen dit bijsturen.

Het vliegen boven de schijf gebeurt op onzettend kleine afstand. Dit om het signaal nog zo duidelijk mogelijk te maken, en het te genereren magnetische veld zo klein mogelijk te houden. Een moderne kop zweeft zo op ongeveer 0,0003 mm (!) van het schijfoppervlak. Dit is vooral opmerkelijk als we de diameter van een menselijk haar bekijken (0.01 mm) of nog straffer, de diameter van een stofpartikel (0,0038 mm). Figuur 8.17 geeft een idee van de uiterst kleine afstanden die gebruikt worden.



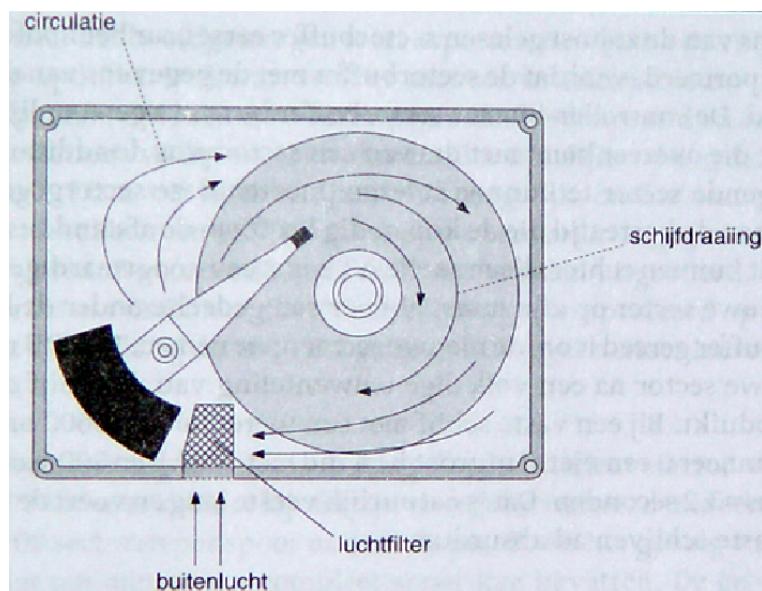
Figuur 8.17: Afstand van de lees- en schrijfkop tot de schijf

8.4.4 Luchtcirculatie en koeling

Aangezien een stofpartikel veel groter is dan de afstand tussen kop en materiaal, kan zo'n partikel enorme schade veroorzaken. Het is dan ook enorm belangrijk dat er geen enkel stofdeeltje in de behuizing van de harde schijf kan geraken. Gebeurt dit toch, en komt zo'n stofdeeltje op de verkeerde plaats terecht (lees: onder de lees- en schrijfkop) hebben we een zogenaamde head crash. Zo'n klein stofdeeltje zal toch zorgen voor een enorme kras (of zelfs erger) op de harde schijf. In de meeste gevallen wil dit dan ook zeggen dat de schijf onbruikbaar wordt. We zouden de schijf hermetisch (luchtdicht) kunnen verpakken, maar het probleem is dat de

koppen enkel vliegen wanneer er toch lucht aanwezig is. Een harde schijf wordt dan ook nooit vacuüm gemaakt.

Om de schijf te beschermen tegen stofdeeltjes, is ze afgesloten van de buitenwereld met behulp van luchtfilters. Het is immers belangrijk dat er nog wel lucht naar binnen kan, om de ontwikkelde warmte te kunnen afvoeren. Harde schijven hebben immers een minimale en maximale temperatuur waaronder ze normaal werken. Figuur 8.18 toont de opbouw van de luchtfilters en geeft ook een idee van de luchtcirculatie die plaatsvindt. Vooral de spin-motor van de harde



Figuur 8.18: Een harde schijf bevat luchtfilters

schijf brengt veel warmte met zich mee. Soms zorgt de interne luchtcirculatie voor te weinig koeling, zodat men nood heeft aan extra koeling. Dit kan -net zoals bij processoren- op actieve of passieve manier verwezenlijkt worden. Figuur 8.19 toont een voorbeeld van een actieve koeling op een harde schijf.



Figuur 8.19: Een actieve koeling op een harde schijf

8.4.5 Snelheid van harde schijven

Draaisnelheid

De harde schijf draait aan een constante snelheid van 3600 tpm (toeren per minuut) tot 15000 tpm. Voor gewone PC's is een rotationele snelheid van 7200 tpm veel gebruikt. Een laptop streeft naar minder energieverbruik en minder warmte-ontwikkeling, waardoor bij laptops snelheden van 4200 en 5400 tpm normaal zijn. SCSI systemen, ontwikkeld voor zeer snel datatransport draaien meestal aan 15000 tpm.

Toegangstijd (access time)

De hoeveelheid gegevens die geschreven kunnen worden door een harde schijf, binnen een bepaald tijdsbestek, hangt af van verschillende factoren. De toegangstijd tot een harde schijf wordt berekend aan de hand van volgende formule:

$$\text{Toegangstijd} = \text{Command Overhead} + \text{Seek Time} + \text{Settle Time} + \text{Latency}$$

Hierbij is:

- Command overhead: De tijd die verloopt bij het doorgeven van het commando naar de harde schijf. Ongeveer 0,5 ms.
- Seek time: Het verplaatsen van de koppen naar het juiste spoor. Bij harde schijf specificaties geeft men dikwijls een van volgende tijden op:
 - Average: de gemiddelde tijd om van een spoor naar een ander spoor te gaan. Meestal ligt dit rond de 8 tot 10 ms.
 - Track-to-track: de tijd om van een spoor naar een aanliggend spoor te gaan. Meestal rond de 1 ms.
 - Full stroke: de tijd om de kop van het binnenste naar het buitenste spoor te brengen. Dit ligt meestal rond de 15 tot 20 ms.
- Settle time: dit is de tijd dat nodig is om de verplaatste lees- en schrijfkoppen te laten stabiliseren. De actuator zal nog heel even navibreren nadat de koppen verplaatst werden, waardoor het nog niet direct mogelijk is om data te lezen of te schrijven.
- Latency: de tijd om de schijf te laten draaien tot de gewenste sector. Men kan geluk hebben en de data onmiddellijk onder lees- of schrijfkop krijgen, maar in het slechtste geval moet nog een volledig disk spin gewacht worden alvorens het lees- of schrijfproces van start kan gaan. Bij een 7200 tpm harde schijf neemt een halve omwenteling (gemiddeld dus) 4,2 ms in, een volledige omwenteling zo'n 8,3 ms. Bij een ultrasnelle SCSI schijf aan 15000 tpm kan dit verlaagd worden tot 4 ms voor een volledige omwenteling!

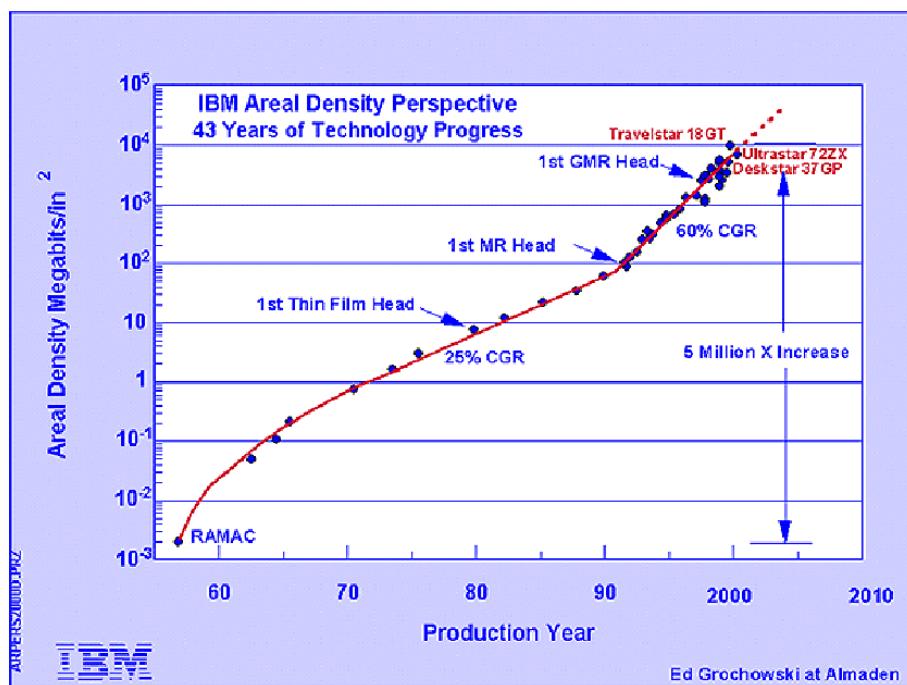
Caching

Er is een soort hype ontstaan rond caching bij harde schijven, ondanks hun geringe verbetering van de performantie. Zo komen op huidige harde schijven cache-geheugens voor van 32 of 64 MB. Deze cache-geheugens werken ook volgens het principe van "locality of reference" (zie 5.2.2), zoals normale caches op processoren. Dit principe baseert er zich echter op dat opeenvolgende data-toegangen in de buurt van elkaar zijn. Daarom zal de cache niet zo'n groot

verschil uitmaken wanneer opeenvolgende leesoperaties ver uit elkaar liggen. Het blijft moeilijk te berekenen of in te schatten hoeveel de interne cache de performantie zal verbeteren. Ook wanneer er zeer grote files worden weggeschreven (denk aan filmpjes of andere bestanden van ongeveer 700 MB) zal de cache niet veel soelaas brengen, aangezien deze relatief klein is.

8.4.6 Evolutie

De evolutie die de harde schijf doorheen de jaren gekend heeft is uiterst opmerkelijk. Voor 1990 verdubbelde de capaciteit elke 36 maanden. Na 1990 verdubbelde (en verdubbelt) de capaciteit nagenoeg om de 18 maanden. Aangezien de rotatiesnelheid dezelfde is gebleven, betekent dit tevens dat om de 18 maanden de datadoorvoer verdubbelt. De snelheid van positionering verdubbelt maar om de 10 jaar, wat dus niet zo'n grote invloed heeft op de totale snelheid van een harde schijf. Vooral de oppervlakte-densiteit blijft enorm stijgen, wat ook duidelijk wordt in figuur 8.20.



Figuur 8.20: De evolutie van de oppervlakte-densiteit

8.5 Geperfectioneerde gegevensopslag: RAID

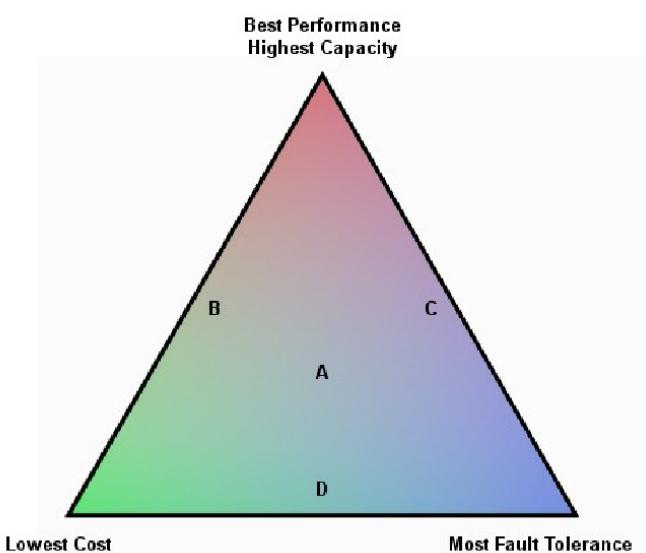
8.5.1 Inleiding

De traditionele harde schijven kunnen geen gelijke tred houden met de steeds sneller wordende processoren en centrale geheugens. We kunnen dan ook echt spreken van een soort *gap* tussen de snelheid van een computersysteem en het permanente geheugen. Men tracht de snelheid van magnetische schijven op te drijven door hogere informatiedichtheid in de sporen, maar deze snelheidswinst volstaat niet en heeft het nadeel dat de kans op fouten verhoogd.

Tegelijk met de vraag naar snellere, is er ook de vraag naar steeds grotere permanente geheugens die altijd beschikbaar (high availability) en betrouwbaar zijn (reliable). Hiervoor werden verschillende systemen ontwikkeld, die samen onder de noemer RAID vallen. RAID stond oorspronkelijk voor Redundant Array of Inexpensive Disks. Het werd ontwikkeld door de Universiteit van Californië op Berkeley. Het idee was om goedkopere (en kleinere) harde schijven te combineren tot een groot en betrouwbaar geheel. Tegenwoordig kosten grotere harde schijven relatief niet veel meer dan kleinere schijven, waardoor de afkorting tegenwoordig ook wel Redundant Array of Independant Disks wordt genoemd. De benaming maakt duidelijk dat we een aantal harde schijven gaan combineren tot een systeem dat redundantie bevat, met andere woorden betrouwbaarheid. We kunnen dus stellen dat RAID zich op 3 terreinen zal situeren:

- Veiligheid: gegevens zijn meer dan één keer aanwezig (redundantie). Zelfs bij het uitvallen van een schijf gaan er geen gegevens verloren. Alle RAID-levels (zie verder), behalve RAID 0, voorzien in redundantie.
- Capaciteit: soms hebben we zeer grote harde schijven nodig, waardoor het niet mogelijk is dit met 1 harde schijf te bekomen. We kunnen dan verschillende harde schijven combineren tot 1 groter geheel. Bij het groeperen van deze schijven gaat er uiteraard plaats verloren voor instellingen om de hercombinatie mogelijk te maken.
- Performantie: door tegelijk te lezen van verschillende schijven, kan de snelheid enorm opgedreven worden.

Wanneer we het over RAID hebben, maar in het algemeen over betrouwbare of performante systemen, kunnen we ons dikwijls baseren op de volgende regel: *Fast, cheap, good: pick two*. Inderdaad, we kunnen nooit deze drie factoren (snelheid, prijs en performantie) verkrijgen zonder in 1 van de factoren in te leveren. Dit wordt ook duidelijk in figuur 8.21. In deze figuur



Figuur 8.21: Fast, cheap, good: pick two!

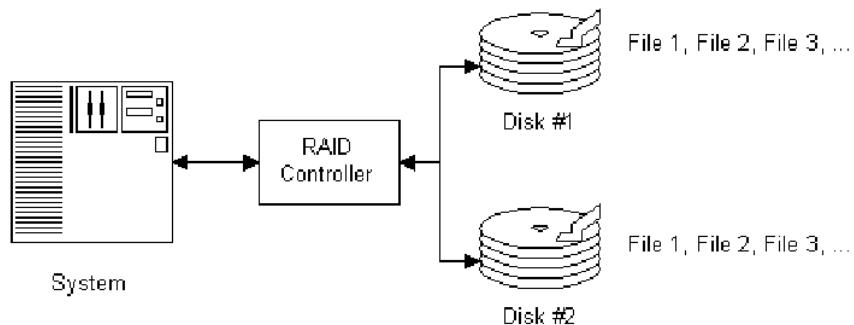
stelt punt A de ideale afweging voor tussen prijs, snelheid en betrouwbaarheid. In de meeste gevallen wordt 1 van de drie factoren echter opgeofferd, wat ons in punt B, C of D brengt.

8.5.2 RAID concepten

Schijfspiegeling (disk mirroring)

Schijfspiegeling bestaat ervan dat alle informatie dubbel voorzien wordt. Er zijn dus minimaal 2 harde schijven die identieke informatie bevatten. Alles wat naar het permanente geheugen wordt geschreven, wordt dan ook dubbel weggeschreven. Indien 1 van de 2 harde schijven het begeeft (bv. door head crash) kan de andere het gewoon overnemen. Op deze manier kunnen we een quasi-perfecte bedrijfszekerheid garanderen. Naast het veiligheidsvoordeel is er ook het snelheidsvoordeel. Schijfspiegeling maakt de toegangstijd van het lezen korter, aangezien het besturingssysteem zal uitrekenen welke van de twee sectoren het vlugst onder de leeskop kan gebracht worden.

Het nadeel van schijfspiegeling is uiteraard de kost. Als men een systeem heeft dat 600 GB moet kunnen opslaan, wil dit bij schijfspiegeling zeggen dat men 1200 GB zal moeten aankopen! Een ander nadeel is dat het schrijven iets langer duurt: de twee schrijfoperaties gebeuren normaal na elkaar, elk met hun eigen access time. In een duplex-systeem kan dit wel opgevangen worden (zie hieronder). Figuur 8.22 geeft weer hoe spiegeling wordt geïmplementeerd.



Figuur 8.22: Schijfspiegeling

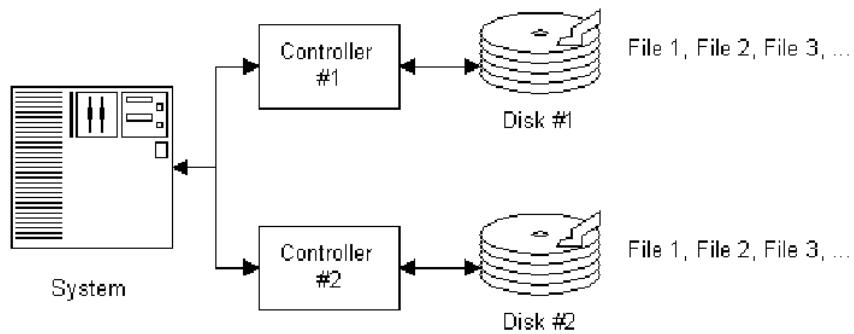
Duplexing

Bij schijfspiegeling wordt ervan uitgegaan dat er 1 schijfcontroller is voor beide harde schijven. Door aparte controllers te gebruiken per schijf, wat men duplexing noemt, kan de snelheid nog opgedreven worden. In dit geval kan namelijk in parallel gelezen worden van de harde schijven. Ook het schrijven zal sneller kunnen gebeuren aangezien de twee harde schijven tegelijk de informatie kunnen wegschrijven. Figuur 8.23 geeft aan hoe duplexing werkt.

Schijvenspreiding (disk striping)

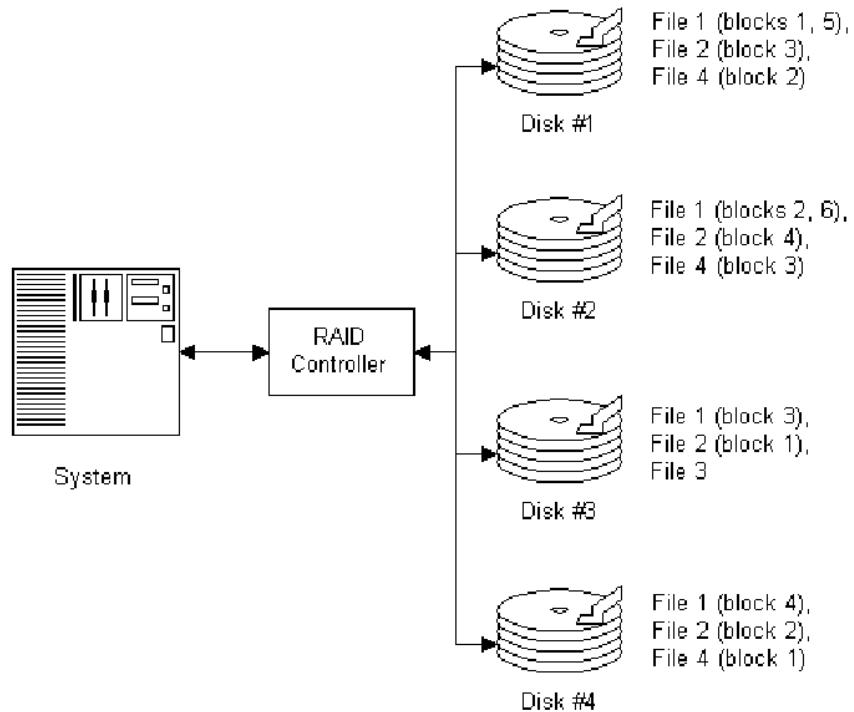
Het zijn vooral de wachttijden die de oorzaak zijn van de steeds groeiende snelheidskloof tussen transistor- en magneetschijvengeheugens.

Schijvenspreiding zal de gegevens die moeten worden weggeschreven, verspreiden over meerdere harde schijven. De schijvenscontroller ziet alle harde schijven als 1 geheel, maar in feite worden alle gegevens verspreid om zo de last te verdelen. Dit "uitsmeren" van gegevens over verschillende schijven brengt een grote mate van parallelisme met zich mee. Stel dat we bijvoorbeeld



Figuur 8.23: Duplexing

spreiding over 5 schijven gebruiken. Indien we 100 MB moeten wegschrijven, wordt eigenlijk op de 5 schijven parallel 20 MB weggeschreven. Het is al snel duidelijk dat dit een enorme performantiewinst met zich zal meebrengen. De spreiding kan gebeuren op basis van individuele bytes (byte-level striping), of op basis van volledige blokken (block-level striping) van gegevens. In principe kan zelfs een bit-level striping worden toegepast, waarbij elke opeenvolgende bit op een volgende schijf wordt weggeschreven. Figuur 8.24 toont de werking van een systeem met schijvenspreiding.



Figuur 8.24: Schijvenspreiding

(N+1)-concept (parity)

Het (N+1)-concept is gebaseerd op een extra controle op N dataschijven, die in dit geval op de (N+1)-de schijf wordt opgeslagen. Met behulp van het (N+1)-concept kunnen kleine occasionele fouten opgespoord en/of verbeterd worden. Voor de N aanwezig schijven wordt meestal gebruik gemaakt van schijvenspreiding, van deze N schijven wordt de controle dus op schijf (N+1) bewaard. Voor het (N+1)-concept wordt dikwijls gebruik gemaakt van de XOR-poort voor de berekening van de extra pariteitsbit. Dit komt door volgende interessante eigenschap van de XOR-poort:

$$A \text{ XOR } B = C \Rightarrow B \text{ XOR } C = A \text{ en } A \text{ XOR } C = B$$

Deze formule kan ook uitgebreid worden naar meerdere symbolen. Zo zal bijvoorbeeld als $A \text{ XOR } B \text{ XOR } C \text{ XOR } D \text{ XOR } E = F$, ook $A \text{ XOR } B \text{ XOR } C \text{ XOR } D \text{ XOR } F = E$. Dit is een interessant aspect voor foutentolerantie: als we op de (N+1)-de schijf de XOR opslaan van alle andere schijven, kunnen we eenvoudig een weggevallen schijf herstellen. Stel inderdaad dat we werken met 2 schijven met data, en 1 schijf met pariteitscontrole door de XOR-poort. We slaan dus op onze derde schijf telkens $A \text{ XOR } B$ op. Wanneer nu schijf A of B uitvalt, kunnen we ze eenvoudig herberekenen aangezien $A = B \text{ XOR } C$ (indien schijf A uitviel) en $B = A \text{ XOR } C$ (indien schijf B uitviel). De pariteit wordt echter niet alleen gebruikt bij het wegvalen van een volledige harde schijf. Elke keer er data wordt ingelezen van de harde schijven wordt gekeken of de bijhorende pariteitsbit juist staat. Indien dit zo is, weten we dat de gegevens niet veranderd zijn. Wanneer onze pariteitscontrole echter niet correct is, weten we dat er ergens een (of meerdere...) bits zijn omgeslagen van 1 naar 0 of omgekeerd.

Schijfspiegeling vs. het (N+1)-concept

Het grote voordeel van schijfspiegeling is de quasi perfecte bedrijfszekerheid, meer bepaald de beveiliging tegen het uitvallen van een volledige harde schijf. Schijfspiegeling biedt echter geen foutentolerantie in de zin dat kleine occasionele fouten zouden opgespoord en/of verbeterd worden. Daarvoor is het (N+1)-concept veel beter geschikt.

Een bijkomend voordeel van schijfspiegeling is een zekere snelheidswinst bij het lezen. Het (N+1)-concept zal echter vanwege het parallelisme dat door de schijvenspreiding teweeg gebracht wordt, nog vlugger werken, vooral als duplexing wordt toegepast.

Doordat alle schijven dubbel aanwezig zijn, is schijfspiegeling op het eerste gezicht tamelijk duur. Voor grote systemen is het (N+1)-concept inderdaad meestal goedkoper dan schijfspiegeling, maar voor kleine systemen wegen de kosten van de complexere schijvenbestuurder relatief zwaar, zodat het (N+1)-concept dan duurder wordt dan schijfspiegeling.

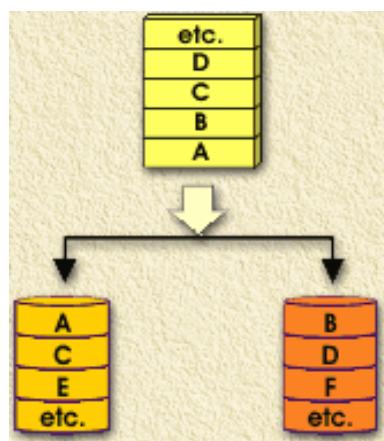
8.5.3 RAID niveau's

JBOD

Bij Just a Bunch Of Disks worden verschillende harde schijven achter elkaar “geplakt”, om zo meer schijfruimte te verkrijgen. Wanneer schijf 1 volg zit, wordt verder gegaan op schijf 2, daarna schijf 3, ... Deze techniek noemt men ook wel spanned volumes.

RAID 0: data striping without parity

RAID 0 zal data striping uitvoeren over ten minste 2 schijven. Daarnaast is er geen enkele vorm van redundantie bij RAID 0. Deze versie profiteert wel maximaal van de snelheidswinst van schijvenspreiding omdat ze niet afgeremd wordt door de foutenbeveiligingstesten. RAID 0 is wegens deze eigenschappen eigenlijk alleen maar geschikt voor situaties waar een snelle opvraging van gegevens het belangrijkste criterium is. RAID 0 wordt tegenwoordig standaard ondersteund op vele moederborden. Het is dan ook een techniek die voor de thuisgebruiker interessant kan zijn: geen extra gegevensbeveiliging, maar wel een performantiewinst. Er zijn minstens 2 schijven nodig om RAID 0 te kunnen toepassen: de striping vereist inderdaad minstens 2 schijven. Figuur 8.25 illustreert RAID 0. Een test op Tom's hardware (www.tomshardware.com) toont aan dat de aanschaf van een RAID0 systeem met 2 schijven wel degelijk een serieuze snelheidswinst kan opleveren. Het RAID0 systeem uitbreiden naar 4 schijven levert dan weer weinig tot geen verschil op tegenover de versie met 2 schijven.



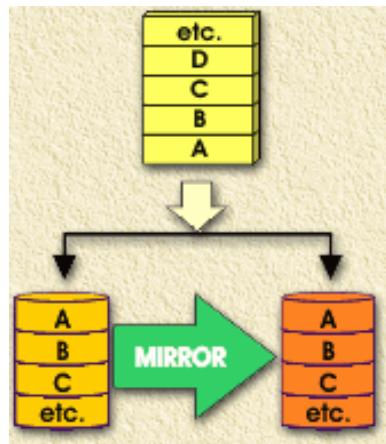
Figuur 8.25: RAID 0

RAID 1: mirroring or duplexing

Deze RAID versie werkt niet met een schijvenspreiding, maar met schijfspiegeling. Afhankelijk van het aantal controllers spreekt men ook wel van duplexing. Doordat alles gespiegeld moet worden, zal het schrijven een stuk trager verlopen. Het lezen kan in de meeste gevallen wel versneld worden, RAID 1 is dus geschik voor lees-intensieve toepassingen. Daarnaast geeft RAID 1 uiteraard ook een bescherming tegen het wegvalLEN van een volledige schijf; buiten de beveiliging tegen deze panne's geeft RAID 1 geen foutenbeveiliging. Zoals RAID 0 heeft men voor RAID 1 minstens 2 schijven nodig. Figuur 8.26 illustreert RAID 1. Ook RAID 1 wordt tegenwoordig ondersteund door een aantal moederborden.

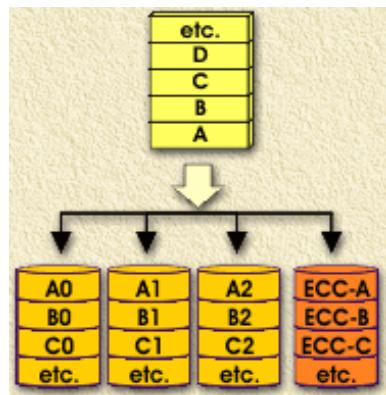
RAID 2: Bit-level striping with Hamming code ECC

Hammingcodes laten toe om fouten van 1 bit te verbeteren en fouten van 2 bits te detecteren. Bij RAID 2 worden deze Hammingcodes gebruikt voor foutentolerantie. Daarnaast wordt ook elke bit op een andere schijf geschreven. Level 2 wordt bijna niet gebruikt, en dit voor een aantal redenen. Het is eerder duur en vraagt meestal een groot aantal schijven, er is een complexe



Figuur 8.26: RAID 1

controller nodig en de performantie is zeker niet fantastisch. Maar de belangrijkste reden is dat het oorspronkelijk idee achter level 2 was de foutencontrole (ECC) ook in de harde schijf zelf te laten gebeuren, en dit is tegenwoordig zo bij elke harde schijf. Figuur 8.27 toont de werking van RAID 2.



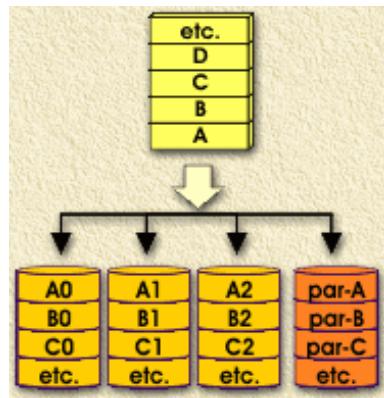
Figuur 8.27: RAID 2

RAID 3: Byte-level striping with dedicated parity

Bij RAID 3 wordt de data op byte-niveau verspreid over de verschillende schijven. Daarnaast wordt een extra schijf toegevoegd om de pariteitscontrole op te slaan. Het verschil met RAID 4 is dat bij RAID 3 de stripe size kleiner is dan 1024 bytes, waar dit bij RAID 4 juist groter is. RAID 3 heeft twee bedoelingen:

- De bestaande schijven inschakelen in een systeem van schijvenspreiding, om de snelheid op te drijven.
- Een extra beveiliging toevoegen zodanig dat wanneer één van de schijven defect is in die mate dat hij zijn foutenverbeterende en -detecterende taak niet meer kan uitvoeren, die taak door een extra harde schijf kan overgenomen worden.

Bij RAID 3 hebben we minstens 3 schijven nodig: 2 voor de striping en 1 voor de extra pariteitsbits. Figuur 8.28 toont de werking van een RAID 3 systeem.



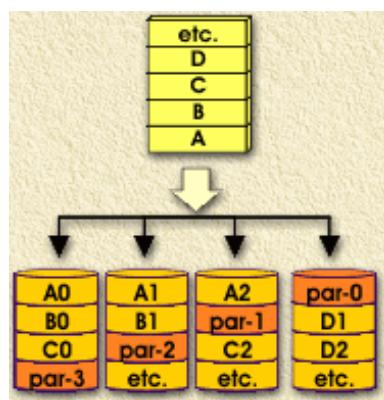
Figuur 8.28: RAID 3 en RAID 4

RAID 4: Block-level striping with dedicated parity

Analoog als bij RAID 3, alleen wordt er nu gestriped met een stripe-grootte groter dan 1024 bytes. Ook hier hebben we minstens 3 schijven nodig. De illustratie van RAID 4 is analoog aan die van RAID 3, en wordt getoond in figuur 8.28.

RAID 5: Block-level striping with distributed parity

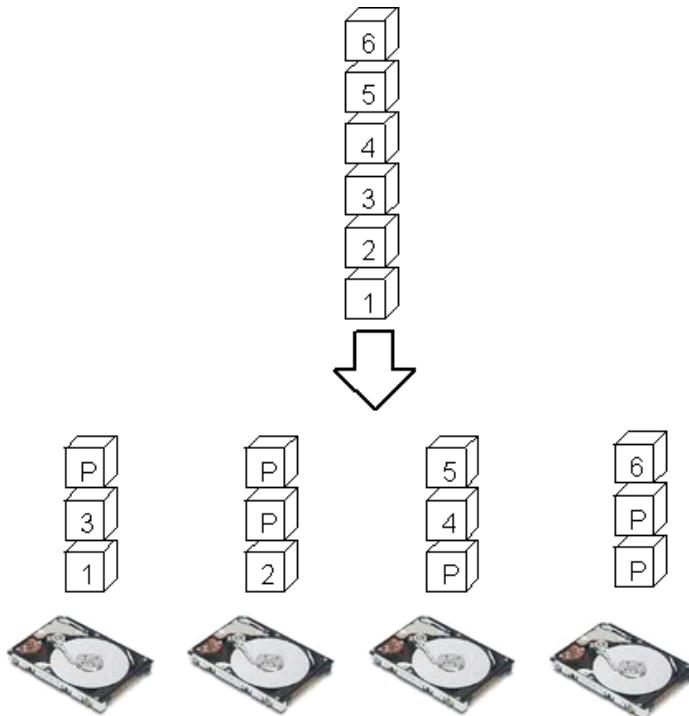
RAID 5 is vergelijkbaar met RAID 4, alleen worden de pariteitsbits nu verspreid over de verschillende schijven. Dit heeft als voordeel dat alle schijven nu gebruikt kunnen worden om data te lezen, waar we bij RAID 4 altijd 1 schijf kwijt zijn (de dedicated schijf met pariteit). Voor een implementatie van RAID 5 hebben we minstens 3 schijven nodig. Figuur 8.29 toont de werking van een RAID 5 systeem.



Figuur 8.29: RAID 5

RAID 6: Block-level striping with dual distributed parity

RAID 6 is weer analoog aan RAID 5, met het verschil dat hier 2 verschillende soorten pariteitscontroles worden opgeslagen. Dit heeft het voordeel dat er zelfs 2 (!) schijven mogen uitvallen, dan nog kunnen de gegevens hersteld worden. Uiteraard is RAID 6 vrij duur, waardoor het minder gebruikt wordt. Voor RAID 6 heeft men minstens 4 schijven nodig. Figuur 8.30 toont tenslotte de werking van RAID 6.



Figuur 8.30: RAID 6

8.5.4 RAID controllers

RAID is op twee manieren implementeerbaar:

- Software: bij software RAID zal het besturingssysteem de taak van de RAID-controller op zich nemen. Er is dus geen speciale hardware nodig, enkel voldoende schijven om een bepaald RAID-niveau toe te passen. Het voordeel van software-controllers is dat ze goedkoop zijn, er is geen speciale hardware voor nodig. Meestal is software RAID echter beperkt tot levels 0, 1 en 5. Als voorbeeld kunnen we het besturingssysteem Windows 2003 Server nemen, waarbij er ondersteuning is voor mirroring, spanning, striping en RAID 5.
- Hardware: bij hardware RAID zal er een speciale RAID controller zijn die de taak van het distribueren van gegevens, berekenen van pariteitbits, ... op zich zal nemen. Dit is uiteraard een duurdere oplossing, alhoewel voor sommige RAID-niveaus tegenwoordig een ingebakken (eenvoudige) RAID-controllers aanwezig is op het moederbord zelf. Dit is zo voor bijvoorbeeld RAID 0 en RAID 1. De RAID controller heeft meestal zijn eigen

processor, om de juiste berekeningen en operaties uit te voeren nodig voor een RAID-systeem.

8.5.5 Praktijk

Op de huidige RAID-markt is er een grote verscheidenheid aan systemen. Zo zijn er veel fabrikanten die zich niet houden aan de oorspronkelijke Berkeley-indeling, maar met eigen varianten op de markt komen zoals een RAID 01 of RAID 53. Een belangrijk verkoopsargument bij betrouwbare RAID-systemen is het zogenaamde MTBF, of Mean Time Between Failure. Dit is de gemiddelde tijd tussen 2 fouten die op het systeem voorkomen. Zo kunnen RAID-systemen een MTBF van 1 miljoen uren hebben, dit wil zeggen meer dan 114 jaren onafgebroken gebruik!

Hoofdstuk 9

Optische geheugens en “leesgeheugens”

Inhoudsopgave

9.1 Optische geheugens	111
9.1.1 CD-ROM	111
9.1.2 CD-R en CD-RW	113
9.1.3 De DVD-RAM	115
9.1.4 DVD-ROM	115
9.1.5 DVD±R en DVD±RW	116
9.1.6 Blu-ray	117
9.2 “Leesgeheugens”	118
9.2.1 Read-only Memory(ROM)	118
9.2.2 Programmable Read-only Memory(PROM)	118
9.2.3 Erasable Programmable Read-only Memory(EPROM)	118
9.2.4 Electronically Erasable Programmable ROM(EEPROM)	118
9.2.5 Flashgeheugens	119
9.2.6 Solid state drives (SSD)	121

In dit hoofdstuk bekijken we de optische permanente geheugens, waarvan CD en DVD de meest bekende vormen zijn. Daarnaast kijken we naar de huidige markt van de leesgeheugens met zijn nieuwste varianten, waarvan de USB-memorystick ongetwijfeld een van de meest gebruikte voorbeelden is.

9.1 Optische geheugens

9.1.1 CD-ROM

Geschiedenis

In een paar jaar tijd is de Compact Disk - Read-Only Memory (CD-ROM) van dure luxe tot een goedkoop een onmisbaar medium geëvolueerd. De CD-ROM heeft -dankzij zijn grote opslagcapaciteit en toepasbaarheid- nieuwe wegen geopend voor de personal computer. De

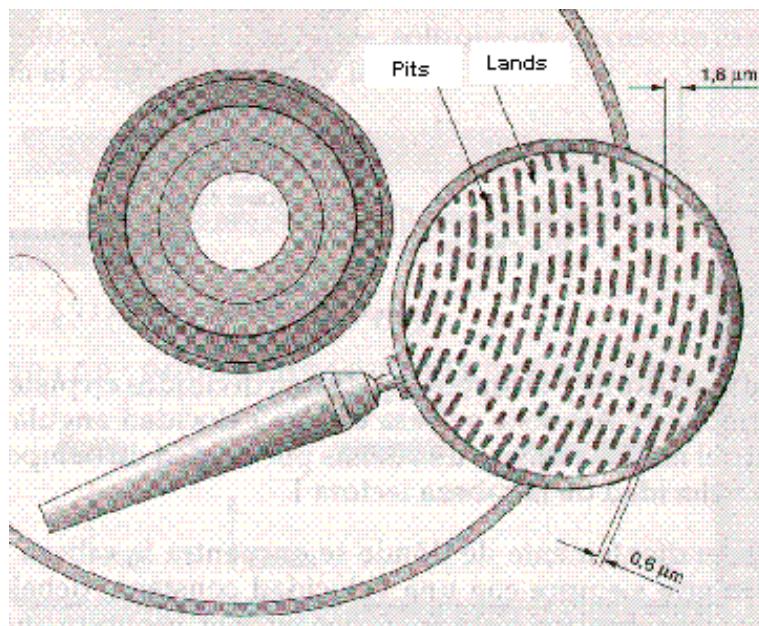
CD-ROM drive heeft ondertussen de floppy drive vervangen, maar laat ons vooral nieuwe toepassingen ontdekken die tevoren nooit denkbaar waren. De grote “multimedia revolutie” is mede een gevolg van de opkomst van de goedkope CD-ROM drives.

De CD-ROM “Yellow Book” standaard werd ontwikkeld in 1985 door Sony en Philips. Microsoft en Apple Computer sprongen al snel mee op de trein en maakten van de CD-ROM een onmisbaar product. John Sculley, CEO van Apple in dit tijd, voorspelde reeds in 1987 dat CD-ROMs de computermarkt volledig zouden veranderen...en niets is minder waar!

Zoals de naam het al zegt, gebruiken CD-ROM drives compact discs, die van hetzelfde fysieke formaat zijn als de schijfjes die we gebruiken voor muziek (de Audio CD). Dankzij een speciale formattering is het namelijk mogelijk om hier informatie in op te slaan. Een CD-ROM drive is onontbeerlijk omwille van de volgende reden: alle huidige software wordt meestal verspreid via CD-ROMs. In de beginjaren was software (en ook besturingssystemen) beschikbaar op floppy en soms CD-ROM, tegenwoordig is CD-ROM de nieuwe standaard (en er is zelfs een evolutie merkbaar richting DVD-ROM).

Opbouw van een CD-ROM

Een CD-ROM is een schijfje met een diameter van ongeveer 12 cm, gemaakt uit plastiek. De gegevens worden hier niet opgeslagen met behulp van magnetisatie (zoals bij harde schijven), maar met behulp van ingebrande putjes in de oppervlakte van de CD-ROM schijf. Men spreekt van zogenaamde **pits** en **lands** voor putjes resp. bergjes in het materiaal. Zo worden er miljoenen kleine putjes gebrand op het schijfje. CD-ROMs worden gedrukt in een fabriek, meestal in massale aantallen voor bijvoorbeeld software, muziek, ... Figuur 9.1 toont een CD-ROM schijfje. Op de tekening is ook te zien dat de breedte van de gaatjes zo'n $0,6 \mu\text{m}$ breed zijn, en dat de afstand tussen twee gaatjes (gezien langs de diameter) $1,6 \mu\text{m}$ is. Interessant om



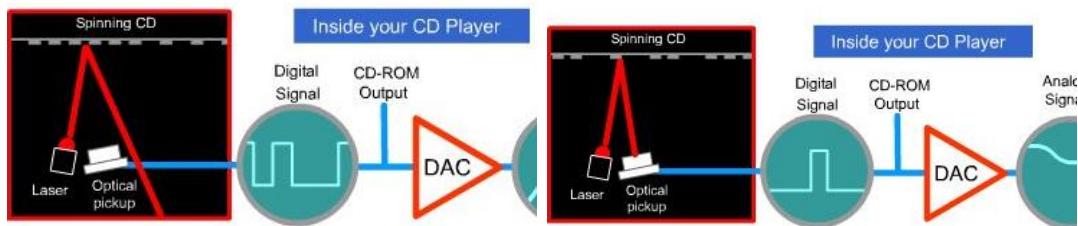
Figuur 9.1: CD-ROM

weten is dat bij de productie van een CD-ROM onmiddelijk vrij veel fouten kunnen voorkomen op het schijfje. Deze fouten worden verbeterd door de foutenverbeterende code, die zo'n 10%

van de informatie op een CD-ROM inhoudt! Voor elke 2048 bytes gegevens worden 280 bytes aan error correcting codes bijgevoegd. De foutencontrole situeert zich op 3 niveaus: per symbool (1-bit fout), per frame (aantal opeenvolgende fouten) en eventueel per sector (cross-interleaved Reed-Solomon code).

CD-ROM drive

De CD-ROM drive is een toestel dat het mogelijk maakt om CD-ROM's te lezen. De belangrijkste component is een beweegbare laser, met een golflente van 780 nm. Deze laser wordt al dan niet gereflecteerd in een bepaalde richting wanneer de straal op een pit resp. land terecht komt. Zo kan men het onderscheid maken tussen wel en niet reflecteren, wat ons dan weer een manier geeft om 1-en en 0-en op te slaan (in realiteit wordt gebruik gemaakt van een “al dan niet overgang” van reflectie naar niet-reflectie). De laser wordt geïllustreerd in figuur 9.2 waar duidelijk wordt dat de laser op een andere manier reageert op pits dan op lands. Op deze



Figuur 9.2: Lezen van een CD-ROM door een laser: links een pit, rechts een land

manier is het dus mogelijk een digitaal signaal op te slaan op een CD-ROM. Een belangrijk verschil met harde schijven is dat het spoor op een CD-ROM een spiraalvorm is (vergelijkbaar met een grammofoonplaat), tegenover de concentrische cirkels bij een harde schijf. Uitgerekend is dat spoor van een CD-ROM zo'n 5 km lang! Figuur 9.3 toont hoe een volledige CD-ROM drive eruit ziet. De CD-ROM drive laat het schijfje niet aan constante snelheid draaien. Om ervoor te zorgen dat er een constante bitstroom is, draait het schijfje sneller aan de binnenkant dan aan de buitenkant. Er moet wel gezegd worden dat deze techniek niet meer in alle moderne CD-ROM spelers wordt toegepast.

Beschermingslaag

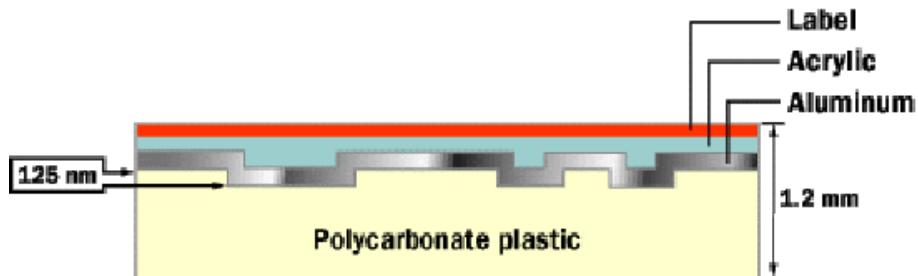
Een CD-ROM bevat verder een beschermingslaag, en aan de andere kant meestal een label om aan te geven wat er op het schijfje staat. Deze opbouw wordt getoond in figuur 9.4. Op deze tekening is ook duidelijk te zien dat de label-kant van de CD-ROM veel minder goed beschermd is dan de gegevens-kant. Hieruit kunnen we afleiden dat het het beste is een CD op zijn gegevens-kant te leggen, in plaats van op zijn label-kant! De minste kras op de label-kant kan tot onherstelbare schade leiden.

9.1.2 CD-R en CD-RW

De nieuwste CD-drives zijn niet gewone CD-ROM lezers, ze kunnen ook zelf CD's schrijven (CD-R of CD-Recordable) of zelfs herschrijven (CD-RW of CD-ReWritable). CD-R's en CD-



Figuur 9.3: Een CD-ROM drive



Figuur 9.4: Doorsnede van een CD-ROM

RW's kunnen typisch tot 700 MB aan gegevens bevatten.

CD-R

Bij een CD-R bevat het schijfje geen aluminium, maar een materiaal dat dye (een soort goudlegering) genoemd wordt. Dit materiaal heeft bepaalde reflectieve eigenschappen, die veranderd kunnen worden door het verbranden ervan. De kleur van een CD-R komt overeen met het soort dye dat werd gebruikt voor de productie ervan. Wanneer een CD-R beschreven wordt, veranderen de eigenschappen van dit materiaal, waardoor het dikwijls ook met het blote oog kan gezien worden welk stuk van de CD-R beschreven is. Vanaf een bepaalde temperatuur wordt het dye absorberend, wat wil zeggen dat het geen licht meer zal weerkaatsen. Analoog als bij de pits en lands hebben we nu een systeem om al dan niet reflectie te voorzien op bepaalde plaatsen. Voor een CD-R gebruikt men de laser op hoog vermogen om te schrijven en laag vermogen om te lezen.

De standaard voor het schrijven van CD-R's werd vastgelegd in het “Orange Book”.

Het grote nadeel van de CD-R is dat wanneer er een fout wordt geschreven, deze niet ongedaan kan gemaakt worden. Het was dan ook niet verwonderlijk dat men een nieuwe technologie ontwierp: de CD-RW.

CD-RW

Om een CD-RW te kunnen lezen, beschrijven en wissen zijn er ook drie golflengtes nodig om deze CD-RW te gebruiken:

- Het hoogste laser vermogen: ook wel “Write Power” genoemd, die het materiaal in een lichtabsorberende staat brengt. Dit noemt men de amorfie staat van het materiaal. De amorfie staat ervoor dat het licht quasi niet wordt gereflecteerd.
- Het middelste laser vermogen: ook wel “Erase Power”, genoemd, die het materiaal opnieuw reflectief maakt voor bepaalde lichtgolven. Hierbij wordt het materiaal kristallijn gemaakt. Kristallijne gedeeltes reflecteren dus veel van het licht.
- Het laagste laser vermogen: ook wel “Read Power” genoemd, die de status van het materiaal niet beïnvloedt, zodat deze kan gebruikt worden om de CD-RW te lezen.

Bij een CD-RW wordt het verschil tussen al dan niet reflecteren minder groot, wat als gevolg heeft dat sommige geschreven CD-RW's niet door alle oudere CD-ROM spelers kunnen worden gelezen. Vooral CD-lezers in oudere radio's hebben vaak problemen met muziekcd's die op een CD-RW zijn geschreven.

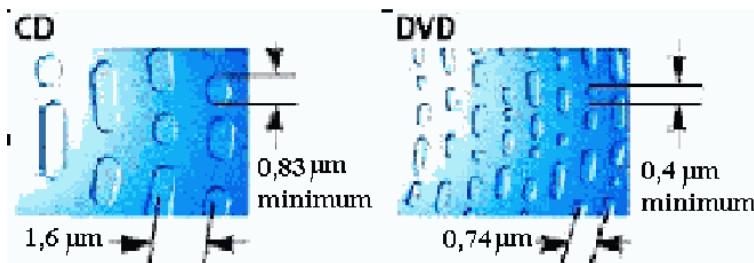
9.1.3 De DVD-RAM

De DVD-RAM is een optische schijf die is ingesloten in een harde behuizing, net zoals een floppy disk. Het formaat is echter niet compatibel met andere DVD-varianten (zie verder). De prijs van deze schijven alsook de dure branders hebben ervoor gezorgd dat de DVD-RAM nooit echt een commercieel succes geworden is.

9.1.4 DVD-ROM

Opbouw van een DVD-ROM

De DVD-ROM of Digital Versatile Disc heeft dezelfde afmetingen als een CD-ROM. De DVD heeft echter een grotere capaciteit en kan tot 4,7 GB aan gegevens bevatten. Dit wordt verwezenlijkt door de pits en lands dichter bij elkaar te zetten dan bij een CD-ROM. Dit wordt getoond in figuur 9.5. Uiteraard zal ook de laser dan fijner moeten zijn, deze heeft een golf-



Figuur 9.5: Vergelijking tussen CD-ROM en DVD-ROM

lengte van 650 nm. DVD's zijn enorm populair als medium voor films van hoge kwaliteit, maar kunnen dus ook gebruikt worden om computergegevens op te slaan. Naargelang het aantal beschreven kanten, spreekt men over:

- DVD-5: single sided, single layer: 4,7 GB
- DVD-9: single sided, double layer op 1 kant: 8,5 GB
- DVD-10: double sided, single layer op beide kanten: 9,4 GB
- DVD-14: double sided, double layer op 1 kant, single layer op andere kant: 13,3 GB
- DVD-18: double sided, double layer op beide kanten: 17,1 GB

Ook bij DVD's worden error correcting codes toegevoegd. Bij DVD's gebeurt dit aan de hand van de Reed-Solomon code.

DVD-ROM drive

De normale snelheid van een 1x DVD-drive is 1350 kB/s, bij een 2x, 4x, 8x, ... DVD drive verdubbelt iedere keer de datadoorvoer. Een DVD-ROM drive ziet er aan de buitenkant niet verschillend uit van een CD-ROM drive. Enkel de interne werking is dus beduidend anders door het gebruik van een andere laser.

Regiocode's

Elke DVD-video disc bevat één of meerdere regiocodes, die aangeven in welk gedeelte van de wereld de disc wordt gedistribueerd en waar dus de discs zouden moeten worden afgespeeld. De commerciële DVD-video speler specificatie stelt dat een speler enkel de discs mag afspelen die in de desbetreffende regio is gesitueerd. Dit om tegen te gaan dat bijvoorbeeld films die in Amerika net zijn uitgebracht al kunnen worden afgespeeld in Europa. Deze restrictie is uiteraard ontzettend interessant voor een regio-na-regio release (waarbij prijs en inhoud kan variëren naargelang het marktsegment) van films. In praktijk bestaan er echter veel DVD spelers die elke DVD-ROM afspelen, eventueel mits een kleine aanpassing aan het toestel. De regiocodes zijn volledig gebaseerd op encryptie.

9.1.5 DVD±R en DVD±RW

DVD±R's en DVD±RW's werken op een zeer gelijkaardige manier als de CD-R's en CD-RW's. Er zijn twee soorten:

- DVD-R(W): Deze formaten werden ontwikkeld door het DVD Forum.
- DVD+R(W): Deze formaten werden ontwikkeld door de DVD Alliance. Ze zijn vrij gelijkaardig aan de DVD-R(W) standaard in gebruik, standaardisatie en compatibiliteit.

De meeste DVD recordable drives ondersteunen ook beide standaarden. Ingaan op de implementatiedetails zou ons te ver leiden. Indien men verzekerd wil zijn van de beste compatibiliteit, kiest men het beste voor DVD-RW. Een en ander hangt wel af van de DVD recorder (speler) die men bezit.

9.1.6 Blu-ray

De Blu-ray Disc is een optisch mediaformaat dat informatie opslaat op een schijf met dezelfde afmeting als een CD of DVD. Dit gebeurt met behulp van een blauwe laser van 405 nm golflengte. Standaard DVD's en CD's gebruiken een rode laser met een golflengte van 650 nm en 780 nm respectievelijk.

De Sony/Philips-ingenieurs die de Blu-Ray Disc ontwierpen in 1995, hadden de taak om een optische vervanger te ontwerpen voor de DV-camcorder. Belangrijk was hierbij dat door de gebruiker zelf kan worden opgenomen. Massa-rePLICATIE voor distributie van videofilms en data was bij het ontwerp van ondergeschikt belang. Daar het inmiddels, na tien jaar, duidelijk geworden is dat de DV-markt waarschijnlijk zal worden vervangen door camcorders uitgerust met solid-state geheugen, hebben de verkopers van Blu-Ray Disc de koers moeten verleggen. De in Japan geïntroduceerde Blu-Ray Disc is een tafelmodel dat HD-video kan opnemen.

De tijd tussen ontwerp en introductie was bijzonder lang, bijna tien jaar. Dit is mede veroorzaakt door gebrek aan belangstelling in de industrie, maar ook door de hinderlijk hoge prijs van de blauwe laser (EUR 700). De fabricage van blauwe lasers en de basisoctrooien zijn in handen van slechts één (Japanse) fabrikant: Nichia.

De kortere golflengte van de gebruikte blauwe laser maakt het mogelijk een hogere informatiedichtheid te behalen zodat meer informatie op dezelfde schijf kan worden opgeslagen. Een schijf heeft de capaciteit van ongeveer 25 GB of twee uren van HDTV-audio + -video tussen 36 en 72 Mbps. Het is tevens mogelijk om meerdere lagen boven op elkaar te zetten welke elk 25 GB hebben, hierdoor is het mogelijk om meer dan 50 GB op één schijf op te slaan. De Blu-Ray Disc heeft een zeer dunne bescherm laag, 100 micrometer, en werd bij de eerste versies ter bescherming tegen krassen en vingerafdrukken verpakt in een cartridge. In een later stadium werden deze vervangen, zodat ze qua uiterlijk hetzelfde zijn als de huidige cd's en dvd's. De coating van de Blu-ray Disc bevat op dit moment ook een harde bescherm laag van 0,1 mm.

Door de hoge informatiedichtheid is het niet eenvoudig mogelijk om van aangepaste DVD-rePLICATIEapparatuur gebruik te maken om platen te fabriceren. De concurrent HD DVD heeft een dikkere bescherm laag en een lagere dichtheid, 17 GB per laag, en kan met minder problemen op bestaande DVD-rePLICATIEapparatuur gefabriceerd worden, omdat HD DVD voortborduurt op de bestaande DVD. HD DVD spelers kunnen eenvoudig compatibel gemaakt worden met CD en DVD. In maart 2008 werd het HD DVD consortium opgeheven, doordat Toshiba zijn medewerking aan HD DVD opzegde. Toshiba was een van de grote merken die achter de commercialisatie van HD DVD stond, maar moest toch de handdoek in de ring gooien na de stijgende populariteit van Blu-Ray. Nochtans had HD DVD veel potentieel: eenvoudige productie (gebaseerd op de DVD) en goedkope HD DVD schijfjes...

De eerste Blu-ray recorder werd onthuld door Sony op 3 maart 2003, en werd geïntroduceerd op de Japanse markt in april dat jaar. Op 1 september 2003 kondigden JVC en Samsung hun Blu-ray-producten aan op de Internationale Funkausstellung in Berlijn, Duitsland.

Ook de nieuwe PlayStation 3 is uitgerust met Blu-ray.

9.2 “Leesgeheugens”

Leesgeheugens worden vooral gebruikt om gegevens uit te lezen, en eventueel om af en toe gegevens naar te schrijven. Typisch kan men uit zulke geheugens snel lezen, maar het schrijven neemt meer tijd in beslag.

9.2.1 Read-only Memory(ROM)

Uit ROM-geheugens kan men enkel lezen, men kan ze dus niet beschrijven. Wanneer deze chips worden geproduceerd, wordt hun inhoud voor eens en voor altijd vastgelegd. Men noemt ze ook wel eens dode geheugens, omwille van deze kenmerken. ROM-geheugens zijn bestendige geheugens: de informatie is niet verloren wanneer de stroom wordt uitgeschakeld. ROM-geheugens worden gebruikt voor de BIOS van een moederbord, alhoewel daarvoor tegenwoordig meer en meer flashgeheugens (zie [9.2.5](#)) worden gebruikt.

9.2.2 Programmable Read-only Memory(PROM)

De Programmable ROM geheugens zijn interessanter dan ROM, in die zin dat de gebruiker van zo'n PROM 1x kan schrijven naar een PROM chip, en dat vanaf dan alle informatie bewaard blijft. Wanneer er een fout gebeurd tijdens het schrijven, is de chip waardeloos. Om gegevens naar deze PROM-geheugenchip te schrijven, wordt gebruik gemaakt van een zogenaamde “PROM programmeur”, een toestel dat bepaalde zekeringen in de chip kan doen doorbranden om zo het onderscheid te maken tussen een 0 en een 1. Dit schrijven neemt enorm veel tijd in beslag: schrijven gaat zo'n 1000x trager dan bij SRAM. PROM chips worden dikwijls gebruikt om besturingssystemen van eenvoudige handpalm-computers op te draaien.

9.2.3 Erasable Programmable Read-only Memory(EPROM)

EPROM is de uitbreiding van PROM, waarbij nu verschillende malen naar de geheugenchip kan geschreven worden. Eens de data op de chip staat, kan ze terug gewist worden door blootstelling aan UV-stralen gedurende 20 tot 40 minuten. Hiervoor wordt gebruik gemaakt van een speciaal toestel. Na dit uitwissen kan er opnieuw informatie naar geschreven worden, maar indien de chip vele malen is uitgewist, verslechteren de opslagcapaciteiten van de chip. Een EPROM chip zal zijn informatie ongeveer 10 jaar vasthouden.

9.2.4 Electronically Erasable Programmable ROM(EEPROM)

Het grote nadeel van de EPROM-chip is dat men een speciaal UV-toestel nodig heeft om de chip te wissen. EEPROM is een EPROM chip die nu elektronisch kan worden gewist, waardoor ze veel handiger te hanteren is dan EPROM. Omdat de EEPROM-chip elektronisch kan gewist worden, is er natuurlijk ook meer electronica aanwezig op deze chip. Dat maakt ze dan ook duurder tegenover de EPROM. Daartegenover staat het feit dat men geen dure UV-straler meer moet bezitten. De voordelen halen het dus van het prijsverschil, waardoor EEPROM-geheugens meer worden gebruikt dan hun UV-tegenhanger. Ook bij de EEPROM-geheugens treedt er na ongeveer 10 jaar geheugenverlies op. Er zijn twee belangrijke soorten EEPROMs:

- Flash EEPROM = flash-geheugen

- Full-featured EEPROM

Wanneer we beide varianten vergelijken, bekomen we tabel 9.1.

Flash EEPROM	Full-featured EEPROM
In 1 “flash” een blok wissen	Wissen op kleinere schaal
Goedkoop	Duur
Minder ruimte	Meer ruimte(2.5x meer)
1000x wissen	100 000x wissen

Tabel 9.1: Full-featured vs. flash EEPROM

9.2.5 Flashgeheugens

Flash EEPROM of kortweg flashgeheugens zijn populair geworden doordat een aantal verbeteringen aan deze geheugenchips werden aangebracht:

- Het aantal mogelijk herbeschrijvingen werd verhoogd
- De informatiedichtheid werd verhoogd(dus meer informatie op dezelfde oppervlakte)
- De fabricagekosten werden fel verlaagd(en dus ook de effectieve kost)
- De snelheid werd verhoogd
- De betrouwbaarheid werd beter en beter

Daarnaast kwam er een groeiende vraag voor flashgeheugens vanuit de wereld van zak- en draagbare computers, waardoor flash EEPROM tot een van de populairste geheugens is uitgegroeid.

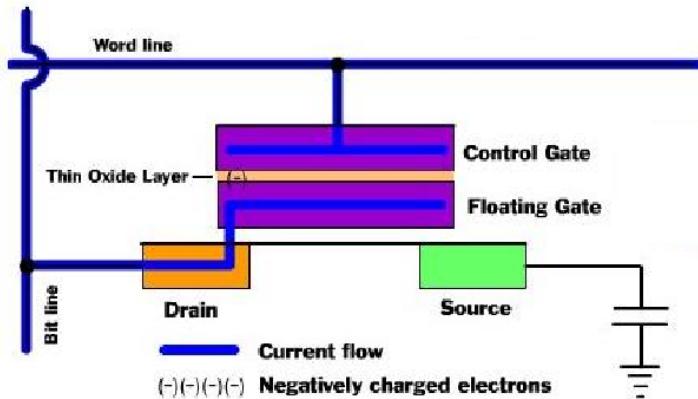
Geschiedenis

NOR flash werd het eerste ontwikkeld in 1988 door Intel. Er is relatief veel tijd nodig om op deze geheugens te schrijven of ze te wissen, maar ze hebben wel een volledig adresseerbaar geheugen waardoor random access mogelijk is tot elke locatie in het geheugen. Dit maakt het geheugen uitstekend geschikt voor het bewaren van de BIOS van een computer, die op die manier op verwisselbare (weliswaar traag) firmware werkt. Het idee achter dit geheugen bleef: weinig schrijven, veel lezen. Deze geheugens hadden de mogelijk om 10000 tot 1000000 keer gewist te worden. Het was ook de basis voor het later ontwikkelde Compactflash. De nieuwste Compactflash kaartjes zijn echter gebaseerd op de NAND technologie.

In 1989 brachten Samsung en Toshiba NAND flashgeheugens op de markt. Het had een betere wis- en schrijfperformantie, een hogere informatiedichtheid en een lagere kost per bit. Daarnaast was het 10x duurzamer dan de NOR-variant. Het nadeel is dat er enkel sequentieel kon naar geschreven worden, wat het geschikt maakte voor USB sticks en geheugenkaartjes zoals SmartMedia, MMC, Secure Digital (SD), Memory Stick en xD-picture kaarten.

Werking van flashgeheugens

Flashgeheugen slaagt zijn informatie op in een rij van “floating gate transistors”, die ook wel cells worden genoemd. Figuur 9.6 toont een floating gate transistor. Elk van deze cells zal



Figuur 9.6: Floating gate transistor, gebruikt in flashgeheugens

meestal 1 bit informatie opslaan, alhoewel nieuwere flash versies meerdere bits per cell kunnen opslaan, door het aantal elektronen dat op de cell zit te laten variëren.

In NOR flashgeheugen ziet elke cell er uit als een standaard MOSFET transistor, behalve dat er twee poorten worden gebruikt in plaats van één. Één van deze poorten is de control gate (CG), de tweede is de floating gate (FG). Tussen deze twee poorten bevindt zich een oxide laag. Omdat de FG geïsoleerd is door deze oxide laag, zullen alle elektronen die erop worden gezet vastzitten en op die manier informatie opslaan. Wanneer er elektronen zitten op de FG, zullen ze het elektrische veld van de CG gedeeltelijk aanpassen (opheffen), wat het voltage van die cell zal wijzigen. Wanneer de cell wordt gelezen, zal er al dan niet elektrische stroom vloeien, afhankelijk van dit voltage. Deze aan- of afwezigheid van stroom wordt gemeten en vertaald in een 1 of 0. In het geval van meerdere bits per cell wordt het voltageniveau gemeten om zo te weten te komen hoeveel elektronen er aanwezig waren.

Een NOR flash cell wordt geprogrammeerd door elektronen van de bron (source) naar een drain te laten vloeien, waarna een relatief groot voltage wordt gezet op de CG zodat de elektronen op de FG springen. Dit noemt men ook wel hot-electron injection. Om het geheugen terug te wissen van een NOR cell, wordt een hoog voltage gezet tussen de CG en de bron, waardoor de elektronen worden weggehaald. De meeste NOR flashgeheugens zijn verdeeld in zogenoamde erase-segmenten, die men blokken of sectoren noemt. Zo'n volledig blok wordt steeds in zijn geheel gewist. Het programmeren kan echter per byte of per woord gebeuren.

NAND flash gebruikt tunnel injection voor het schrijven en tunnel release voor het wissen.

Mogelijkheden

Alhoewel deze flashgeheugens random kunnen beschreven worden, kan er enkel gewist worden per volledig blok. Met andere woorden kunnen deze flashgeheugens random write aanbieden, maar geen random rewrite of erase. Wanneer we ze vergelijken met een harde schijf is een andere limiet het aantal erase-write cycli, zodat men moet oppassen om op dit soort geheugen

harde schijf-gebaseerde applicaties te zetten, zoals het besturingssysteem. Om te vermijden dat sommige delen in een flashgeheugen te veel beschreven worden in vergelijking met andere, wordt door de producent in het geheugen een voorziening gemaakt die de load verspreid over het volledige geheugen. Zo is er een gelijke en geleidelijke degradatie van het volledige geheugen.

De kost per byte van flashgeheugens blijven veel groter dan die van harde schijven. Samen met het feit dat het aantal erase-write cycli beperkt is, maakt dat flashgeheugens nog niet direct concurrentie voor de harde schijf zullen vormen.

Men gebruikt flash-geheugens in camera's, GSM's, printers, palmtops en ook als gegevensdrager zelf(denk aan de USB-sticks en flash geheugenkaarten voor draagbare MP3-spelers en fototoestellen).

9.2.6 Solid state drives (SSD)

Solid-state drives zijn de gedoodverfde opvolgers voor harde schijven. Zeker in moderne laptops komen solid-state geheugens dikwijls voor, omdat ze minder energie verbruiken en veel sneller zijn dan gewone harde schijven. In een solid-state drive zitten geen draaiende onderdelen, de gegevens worden opgeslagen op NAND-flash gebaseerde geheugencellen. Het NAND-flash geheugen is persistent, wat wil zeggen dat de data ook bij stroomuitval blijft bewaard. Er bestaan ook DRAM SSD schijven, gebaseerd op DRAM en dus wel vluchtig. Deze DRAM SSD schijven worden gebruikt in toepassingen die heel snel moeten zijn maar waarbij persistentie (bewaren bij stroomuitval) niet nodig is. De meeste SSD geheugens bevatten ook een cache geheugen, om het opvragen van data te versnellen. Dit cache is meestal opgebouwd uit DRAM-cellene. SSD drives kosten nog vrij veel ten opzicht van harde schijven: in september 2011 betaal je voor een SATA solid-state drive van 240 GB al snel 400 euro! De voordelen zijn echter legio:

- Snellere start-up: er zijn geen draaiende onderdelen.
- Zeer snelle toegangstijden, aangezien er geen lees- en schrijfkop moet bewegen. Op CeBIT 2009 demonstreerde OCZ een 1TB grote SSD schijf die een schrijfsnelheid haalde van 654 MB/s en een leessnelheid van 712 MB/s. Gewone harde schijven komen zelfs niet in de buurt van deze lees- en schrijfsnelheden.
- Geruisloos aangezien er geen draaiende onderdelen zijn.
- Zeer laag energieverbruik en warmte-productie (belangrijk bij bv. laptops).
- Zeer hoge betrouwbaarheid.
- Uiterst geschikt in extreme omstandigheden: kan tegen extreme warmte of koude, schokken, hoogte, ...

Omwille van de hoge snelheid van SSD-schijven worden ze meer en meer gebruikt als opstartschaif, m.a.w. de schijf waarop het besturingssysteem staat geïnstalleerd. Omdat ook de kwaliteit van de geheugencellen in SSD geheugens mettertijd achteruit gaat na veel wissen en herschrijven, probeert men slimme algoritmes in te bouwen die de data verspreiden over alle geheugencellen. Zo slijt het SSD geheugen geleidelijk aan en op alle plaatsen even veel. Dit noemt men ook wel *gracefull degradation*. Er is een zeer geleidelijke en aanvaardbare slijtage van gans het geheugen. Wanneer zo'n algoritme niet gebruikt zou worden, zou het geheugen

op bepaalde plaatsen enorme slijtage ondervinden. Daardoor zou de levensduur aanzienlijk verkort worden.

MLC vs. SLC

Solid-state drives zijn onder te verdelen in twee typen: SLC en MLC. Een SSD bestaat uit verschillende cellen. Iedere cel heeft een analoge waarde. Deze analoge waarde, doorgaans een lading, spanning of weerstand, wordt onderverdeeld om tot een digitale waarde te komen. Bij Single-Level-Cell SSD's (SLC) wordt de analoge waarde van een cel verdeeld in twee bereiken: een hoog bereik en een laag bereik. Hierdoor slaat iedere cel effectief dus n bit op. Bij Multi-Level-Cell SSD's (MLC) wordt de analoge waarde in meer bereiken verdeeld, meestal vier. Hierdoor worden effectief dus meer bits per cel opgeslagen. Dit verschil heeft tot gevolg dat SLC's betrouwbaarder, duurzamer en sneller zijn, terwijl MLC's juist als voordeel hebben dat ze data veel compacter kunnen opslaan. Hierdoor kunnen MLC's met dezelfde opslagcapaciteit goedkoper worden geproduceerd dan SLC's.

Hoofdstuk 10

Poorten

Inhoudsopgave

10.1 Parallelle vs. seriële communicatie	124
10.2 De parallelle poort	124
10.2.1 Aansluitingen en gebruik	124
10.2.2 Standaardisatie en modes	125
10.3 De seriële poort	125
10.3.1 Aansluitingen en gebruik	125
10.3.2 Asynchrone vs. synchrone communicatie	127
10.4 PS/2 poort	127
10.5 De USB poort	128
10.5.1 Ontwerp	128
10.5.2 Standaardisatie	129
10.5.3 Host controller	130
10.5.4 Apparaatklassen	131
10.5.5 Stroomvoorziening	131
10.5.6 USB 3.0	131
10.6 FireWire (IEEE 1394)	131
10.6.1 Ontwikkeling	132
10.6.2 Werking	132
10.6.3 Standaarden	132
10.6.4 Hot swap gevraagd...	133
10.7 eSATA	133
10.8 Bluetooth	134
10.9 VGA-poort	134

In dit hoofdstuk overlopen we de belangrijkste poorten, die gebruikt worden om uiteindelijk in- en uitvoerapparaten aan te sluiten aan een computersysteem.

10.1 Parallelle vs. seriële communicatie

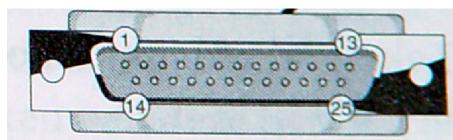
Bij alle poorten (en dus ook verbindingen) die we zullen bekijken kunnen we het onderscheid maken tussen parallelle en seriële poorten. Bij parallelle communicatie worden meerdere signalen tegelijk verstuurd (over meerdere draden), waar bij seriële communicatie alles over 1 draad na elkaar wordt verstuurd. Bij parallelle poorten wordt meestal gewerkt in veelvouden van 8 bits (8, 16, 32, 64 bits tegelijk). Daarnaast worden enkele draden gebruikt die dienen voor synchronisatie, aarding en kloksturing.

Bij seriële communicatie wordt gebruik gemaakt van een draad in de ene richting, een draad in de andere richting en tenslotte twee nuldraden (een signaal wordt telkens uitgemeten in vergelijк met een zogenaamde nuldraad). De bits worden 1 per 1 verstuurd. Het is dus vrij duidelijk dat seriële communicatie trager zal verlopen dan parallelle communicatie, aangezien we bij deze laatste de doorvoer kunnen vergroten door de parallelle draden te gebruiken. Deze theorie blijft echter enkel geldig op korte afstanden, van zodra de afstanden langer worden krijgt men last van zogenaamde signal skew. Dit is het verschijnsel dat gelijktijdig verstuurde bits (op zo'n parallelle verbinding) niet gelijktijdig aankomen, doordat er per draad bijvoorbeeld kleine verschillen kunnen zijn qua materiaalzuiverheid. Hierdoor kan men niet meer detecteren welke bits nu samen aangekomen zijn, maw. de synchronisatie wordt enorm moeilijk. Hoe hoger de snelheid waarop men zendt, hoe meer last men heeft van deze signal skew. De maximale snelheid voor een parallelle poort is door dit fenomeen reeds bereikt en er zit dan ook weinig toekomstmuziek in deze verbindingen. Kijken we bijvoorbeeld naar ATA, die met parallelle communicatie zijn snelheidslimiet had bereikt en nu stilaan wordt vervangen door een seriële variant, SATA. Uiteraard hebben seriële verbindingen geen last van signal skew, hun snelheid kan dan ook dikwijls nog serieus verhoogd worden.

10.2 De parallelle poort

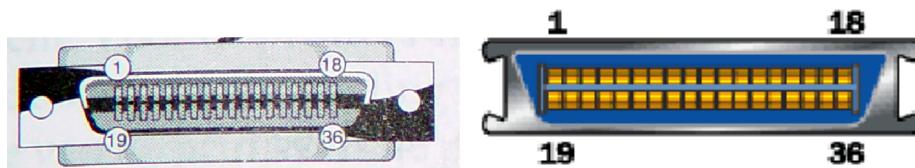
10.2.1 Aansluitingen en gebruik

De parallelle poort op een computer bestaat uit een 25-draads SUB D connector. Deze wordt getoond in figuur 10.1. Opmerkelijk is dat aan de andere kant van de parallelle kabel (bv.



Figuur 10.1: De parallelle poort

bij de printeraansluiting) meestal gebruik wordt gemaakt van een ander soort connector, de Centronics aansluiting, die wordt getoond in figuur 10.2. Wanneer we gebruik maken van de parallelle poort om te printen, verwijst men meestal naar deze poort door de benaming LPT. Op een parallelle poort worden 8 bits tegelijk verzonden. Op de parallelle poort wordt een parallelle kabel aangesloten, die tot zo'n 3 meter lang kan zijn. Langer is in sommige omgevingen wel mogelijk, maar meestal zal de signal skew voor moeilijkheden zorgen.



Figuur 10.2: De Centronics aansluiting

10.2.2 Standaardisatie en modes

De standaard van de parallelle poort wordt vastgelegd in IEEE 1284, en hierbij worden 5 verschillende werkmodi voorzien. Afhankelijk van de gebruikte standaard is communicatie in 1 richting (van apparaat naar pc of omgekeerd) of in 2 richtingen mogelijk.

- Compatibility mode: ook wel SPP mode (Standard Parallel Port) genoemd. In deze modus is er enkel verkeer mogelijk van de PC naar een randapparaat. Dit is de standaard die ooit door IBM werd vastgelegd en gebruikt wordt voor printers. In deze modus wordt gebruik gemaakt van 18 draden, waarvan de meeste data van de pc naar de printer sturen. De printer kan wel signaliseren naar de PC (bijvoorbeeld bij problemen), maar echt data kan er niet verzonden worden van de printer naar de PC (dit is ook niet nodig).
- Nibble Mode: hierbij worden 4 bits tegelijk verstuurd, waarbij er enkel verkeer kan gaan van een apparaat naar de PC. Gecombineerd met de compatibility mode, is dit wat door Hewlett Packard "Bi-tronics" wordt genoemd.
- Byte Mode: hierbij worden 8 bits tegelijk verstuurd. Ook hier kan er enkel verkeer van een apparaat naar de PC. Wanneer men dit combineert met de compatibility mode, spreekt men van een zogenaamde bidirectionele poort.
- EPP: Enhanced Parallel Port, een bidirectionele verbinding (wel maar half-duplex, wat wil zeggen dat beide kanten niet tegelijk kunnen verzenden). Deze wordt gebruikt voor het aansluiten van alle randapparatuur behalve printers, zoals CD-ROM spelers, tapes, harde schijven, netwerkadapters, ... In deze modus kan tot 2 MB per seconde worden verstuurd!
- ECP: Extended Capability Port, bidirectionele half-duplex verbinding. Deze mode wordt vooral gebruikt bij nieuwe printers en scanners en werd mede ontwikkeld door HP en Microsoft. Bij deze manier van werken wordt er aan datacompressie (aantal gegevens kleiner maken door speciale algoritmes, zonder informatieverlies) om op die manier efficiënter gebruik te maken van de parallelle poort.

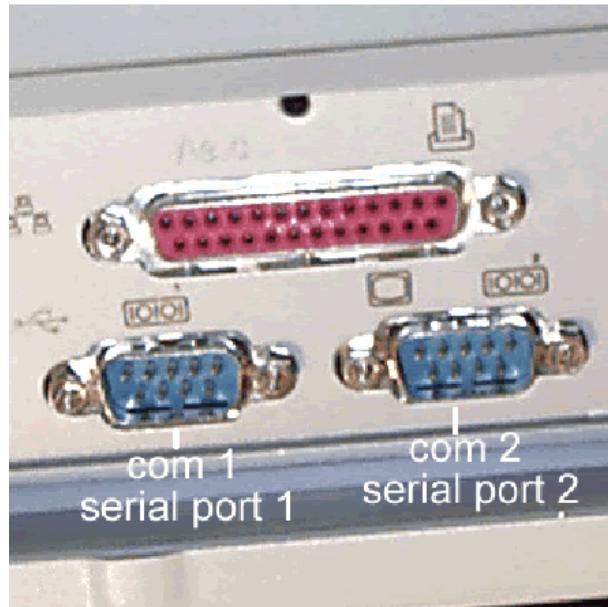
Op termijn zal de parallelle poort waarschijnlijk volledig vervangen worden door nieuwere technologieën zoals USB.

10.3 De seriële poort

10.3.1 Aansluitingen en gebruik

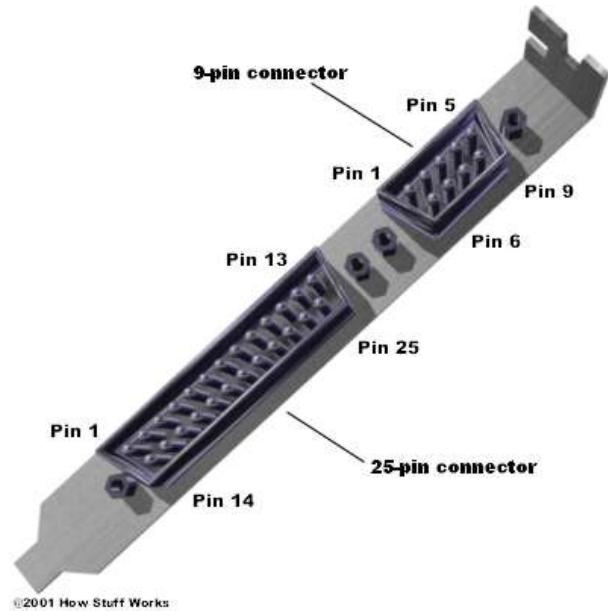
Seriële verbindingen zijn zeer geschikt om lange afstanden te overbruggen, zoals reeds werd aangegeven in 10.1. De seriële poort van een computer is in het bijzonder een seriële verbinding,

en wordt ook dikwijls aangegeven met de naam RS-232. Figuur 10.3 toont een achterkant van een computer, met onderaan 2 seriële en bovenaan 1 parallelle poort. Op deze figuur kan



Figuur 10.3: Een parallele poort en onderaan twee seriële poorten

men zien dat de seriële poort een mannelijke connector heeft op het moederbord, en dat de bijhorende stekker vrouwelijk. Bij de parallele poort is dit net het omgekeerde. Naast deze 9-polige aansluiting bestaat er nog een 25-polige connector voor seriële communicatie, die getoond wordt in figuur 10.4. Seriële poorten werden lange tijd gebruikt om bijvoorbeeld modems aan te



Figuur 10.4: Een 9- en 25-polige seriële poort

sluiten. Bij alle moderne computersystemen hebben een of meerdere seriële poorten, vooral om oudere randapparatuur te blijven ondersteunen. (bv. ook een muis kon worden aangesloten op

de seriële poort). Tegenwoordig zijn andere soorten seriële poorten meer populair, zoals USB en Firewire.

10.3.2 Asynchrone vs. synchrone communicatie

Seriële poorten, ook wel COM poorten genoemd, werken bi-directioneel. Dit wil zeggen dat communicatie in beide richtingen kan verlopen. Seriële poorten worden aangestuurd door een speciale controle-chip, de zogenaamde UART-controller. UART staat voor Universal Asynchronous Receiver/Transmitter. Deze UART-controller krijgt de parallelle uitvoer van de systeembus en zet deze om in seriële informatie om uiteindelijk door de seriële poort te sturen. De asynchronous staat voor de asynchrone manier van data verzenden. In het algemeen zijn 2 belangrijke vormen van communicatie bekend:

- Synchrone communicatie: verzender en ontvanger zijn perfect op elkaar afgestemd. Ze maken gebruik van een timing-systeem waardoor ze precies weten hoe de communicatie zal verlopen. Synchrone communicatie is niet eenvoudig: de hardware van beide partijen moet zeer goed op elkaar afgestemd zijn, de lijn van betrouwbaar en storingsvrij zijn.
- Asynchrone communicatie: hier is de omgeving waarin zender en ontvanger zich bevinden minder kritisch. De lijn mag onbetrouwbaar zijn, fouten kunnen gebeuren en worden gedetecteerd of zelfs opgelost. In dit systeem wordt een byte als volgt doorgestuurd: startbit, 8 databits, pariteitsbit, stopbit. Deze manier van verzenden wordt gebruikt bij de seriële poort.

10.4 PS/2 poort

Deze poort werd (en wordt nog steeds) lange tijd gebruikt om een toetsenbord en muis aan te sluiten op een computersysteem. Er wordt gebruik gemaakt van een kleine 6-polige DIN-poort (Deutsche Industrie Norm). Er zijn twee modi voor de PS/2 poort: de streaming mode waarin een apparaat continu data blijft doorgeven aan de computer, en de remote mode waarbij de computer de informatie opvraagt aan het aangesloten systeem. Figuur 10.5 toont een PS/2 poort.



Figuur 10.5: Een PS/2 poort

10.5 De USB poort

We bekijken USB meer in detail dan de andere poorten, omdat USB een zeer veel gebruikte poort is met goede toekomstperspectieven.

10.5.1 Ontwerp

USB (Universal Serial Bus) voorziet in een seriële manier van communiceren. USB wordt niet alleen gebruikt om apparaten aan te sluiten op computers, maar ook bij bijvoorbeeld de Sony PS2, Nintendo Revolution en verscheidene PDA's. Er zijn twee belangrijke connectoren voor USB apparaten, de USB A en B aansluitingen, deze worden getoond in figuur 10.6. Ook de USB poort op de computer wordt in deze figuur getoond. De USB aansluitingen zijn speci-



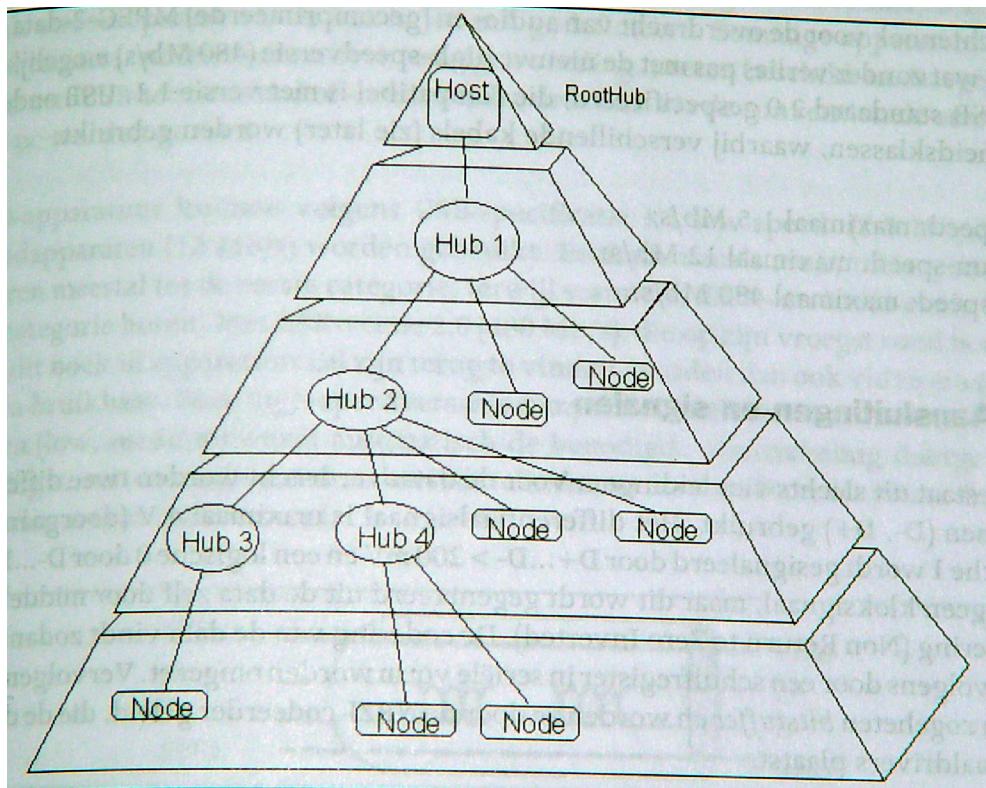
Figuur 10.6: Van link naar rechts: USB A, USB B, USB poort

aal ontworpen om aan een aantal belangrijke eisen te voldoen zoals: lage productiekost, geen schroeven nodig, onmogelijk verkeerd in te steken, connectoren dwingen juiste topologie af voor hubs, goed aarding, ... Het USB systeem heeft een ontwerp dat bestaat uit een zogenoamde host controller en meerdere apparaten die in een boomstructuur zijn aangesloten. Er is een limiet van 5 levels aan knopen (hubs die aangesloten worden op 1 USB-poort en zelf meerdere USB-aansluitingen aanbieden) in de boomstructuur; er kunnen maximaal 127 apparaten aangesloten worden op 1 host controller (hubs tellen ook mee als apparaten!). Figuur 10.7 toont een USB hub: 1 aansluiting wordt vertakt in 4 USB aansluitingen. Figuur 10.8 toont hoe



Figuur 10.7: USB hub

we USB hubs kunnen gebruiken om een aantal apparaten aan te sluiten op 1 USB connectie.



Figuur 10.8: Typische gelaagde structuur van USB met behulp van hubs

Aangezien een modern computersysteem meestal meerdere USB host controllers heeft, kunnen dus ontzettend veel (meer dan we waarschijnlijk nodig hebben) apparaten aangesloten worden via USB. Het idee achter USB is de ontwikkeling van één universeel aansluitingssysteem dat kan gebruikt worden om verschillende soorten randapparaten aan te sluiten. Daarnaast zijn plug&play (insteken en direct kunnen werken) en hot swapping (apparaat insteken of uittrekken terwijl de computer aanstaat) belangrijke kernbegrippen in de USB-technologie.

USB wordt gebruikt voor het aansluiten van muisen, toetsenborden, gamepads, joysticks, scanners, digitale cameras, printers, harde schijven en netwerkapparaten. Het is duidelijk dat de mogelijkheden van USB virtueel onbeperkt zijn en dat USB een enorm groot deel van de markt inneemt op het gebied van poorten. De enige apparaten die nog niet worden aangesloten zijn schermen, omdat ze een te hoge datarate nodig hebben die (nog) niet voorzien is bij USB.

10.5.2 Standaardisatie

USB wordt gestandaardiseerd door het USBIF, het USB Implementers Forum, waarin belangrijke bedrijven zitten zoals Apple Computer, Hewlett-Packard, NEC, Microsoft, Intel en Agere. Over de tijd zijn er vernieuwingen aangebracht op het USB-platform, waardoor er verschillende versies bestaan: 0.9, 1.0, 1.1 en 2.0. Alle versies zijn weer achterwaarts compatibel, wat wil zeggen dat een 2.0 apparaat kan aangesloten worden op een 1.1 systeem. Moderne systemen bieden minstens USB 2.0 aan:

- USB 1.1: maximale snelheid van 12 Mbps, of 1,5 MB per seconde. Maximale kabellengte

3 m.

- USB 2.0: maximale snelheid van 480 Mbps, of 60 MB per seconde. Maximale kabellengte 5 m.
- USB 3.0: maximale snelheid van 4,8 Gbps, of 600 MB per seconde. Maximale kabellengte 5 m.

10.5.3 Host controller

De hardware die de host controller bevat heeft een programmeerinterface. Deze interface wordt ook wel de Host Controller Device (HCD) genoemd. Bij de ontwikkeling van versie 1.0 en 1.1 waren er twee HCD implementaties:

- OHCI: Open Host Controller Interface, ontwikkeld door Compaq en als standaard aanvaard door USB-IF.
- UHCI: Universal Host Controller Interface, ontwikkeld door Intel waarbij licentiekosten betaald moesten worden indien fabrikanten hiervan gebruik wilden maken. (dit is natuurlijk niet de eerste keer dat Intel een marketing-stap zette die niet overal in goede aarde viel...)

De twee concurrerende systemen hebben ertoe geleid dat besturingssystemen en apparaten ondersteuning moeten bieden voor beide implementaties, wat de uiteindelijke kost zeker niet ten goede komt. Tijdens de ontwikkeling van USB 2.0 eiste USB-IF slechts 1 implementatie, wat nu de EHCI (Extended Host Controller Interface) wordt genoemd. Enkel EHCI ondersteunt high-speed transfers. Elke EHCI controller heeft ook nog 4 virtuele HCD implementaties voor ondersteuning van full speed en low speed apparaten. De virtuele HCD op Intel en Via chipsets maken gebruik van UHCI, alle andere fabrikanten gebruiken OHCI. De 4 soorten transfersnelheden zijn:

- Low Speed: 1,5 Mbps (192 KBps) trage standaard die vooral gebruikt wordt voor HID (Human Input Devices) apparaten gebruikt wordt zoals een muis, toetsenbord en joystick. Deze apparaten hebben toch geen hoge invoersnelheid waardoor er geen problemen zijn.
- Full Speed: 12 Mbps (1.5 MBps) lijn die vóór de 2.0 standaard uit was voor de hoogste datatransmissiesnelheid zorgde. Vele apparaten met de 1.1 specificatie, vallen nog terug op deze datarate, die relatief traag is, zeker voor bijvoorbeeld externe harde schijven. Full Speed apparaten verdelen de bandbreedte van de USB poort op een FCFS (First Come First Served, of diegene die het eerste aanvraagt, krijgt het eerste bandbreedte toegewezen) waarbij het regelmatig voorkomt dat op een bepaald moment geen bandbreedte meer vrij is voor juist aangesloten apparaten.
- Hi-Speed: 480 Mbps (60 MBps) dat pas sinds USB 2.0 mogelijk is. Niet alle USB 2.0 apparaten zijn sowieso Hi-Speed.
- Super-Speed: 4,8 Gbps (600 MBps) dat pas sinds USB 3.0 mogelijk is.

Alle USB apparaten zouden in principe moeten aangeven op welke snelheid ze werken. De USB-IF deelt de juiste logo's uit aan fabrikanten die aangeven wat de snelheid is van een apparaat. Wanneer Hi-Speed apparaten in een Full Speed hub wordt gestoken, moeten deze ook overschakelen naar de lagere datarate.

10.5.4 Apparaatklassen

De verschillende apparaten die op USB kunnen worden aangesloten, worden onderverdeeld in een aantal apparaatklassen. Deze klassen worden bepaald door de USB-IF, en bepalen een verwacht gedrag van een bepaald apparaat zodat wanneer het apparaat wordt aangesloten, drivers van die desbetreffende klasse kunnen gebruikt worden. Zo zijn er in de meeste besturingssystemen standaard drivers aanwezig voor een aantal van deze klassen. De meest voorkomende klassen zijn:

- USB human interface device (HID) apparaten: toetsenborden, muizen, ...
- USB mass storage apparaten: memorystick, harde schijven, multimediacartaalreaders, digitale cameras, MP3-spelers, ... Bij deze klasse worden apparaten voorgesteld als een bestandensysteem.
- USB communications device class (CDC): modems, netwerkkaarten, faxmachines, ...
- USB printer device apparaten: printers.
- USB audio device class: geluidskaarten.
- USB video device class: webcams, capturing apparaten.

10.5.5 Stroomvoorziening

USB voorziet stroom tot 5V met een stroomsterkte van 500mA. Dit wil zeggen dat apparaten die een grotere stroomsterkte nodig hebben of een grotere spanning, zelf voor externe stroomtoevoer moeten zorgen. Een typisch randgeval bestaat hier bij externe harde schijven: sommige (kleinere en minder performante) externe harde schijven kunnen volledig van de door de USB poort voorziene stroom werken, andere hebben externe stroom nodig via een adapter, nog andere werken door middel van bijvoorbeeld 2 USB aansluitingen. Deze beperkte stroomvoorziening is in feite een van de enige nadelen die momenteel aan USB verbonden zijn. Bijvoorbeeld FireWire zorgt voor beter stroomvoorziening dan USB.

10.5.6 USB 3.0

Door de komst van USB 3.0 is er nog een mooie toekomst weggelegd voor USB. Er zijn al enkele fabrikanten die (op duurdere moederborden) USB 3.0 aanbieden en ook enkele USB 3.0 devices zijn al te koop. Het zal waarschijnlijk tot het laatste kwartaal van 2011 duren vooraleer USB 3.0 mainstream wordt op de moederborden.

10.6 FireWire (IEEE 1394)

FireWire (ook bekend als i.Link of IEEE 1394) is een seriële busstandaard die toelaat om aan hoge snelheid te communiceren. FireWire is eigenlijk de opvolger van de SCSI parallelle interface. Bijna alle moderne videocameras hebben een firewire-aansluiting en ook vele PC's hebben deze aansluiting tegenwoordig standaard. Ook de alom bekende en populaire Apple iPod heeft een Firewire aansluiting.

10.6.1 Ontwikkeling

FireWire werd oorspronkelijk ontwikkeld door Apple Computer in de jaren '90, op basis van de tragere IEEE 1394 standaard. IEEE had deze standaard vooropgesteld als opvolger voor de SCSI bus. Apple's ontwikkeling werden afgerond in 1995 en het ontwikkelde product werd later beschreven in IEEE 1394-1995, IEEE 1394a en IEEE 1394b. Sony's implementatie van deze standaard is gekend als de i.Link, en gebruikt enkel de 4 signaalpinnen. De stroomvoorziening op de andere twee draden wordt genegeerd door deze apparaten, die extern van stroom worden voorzien.

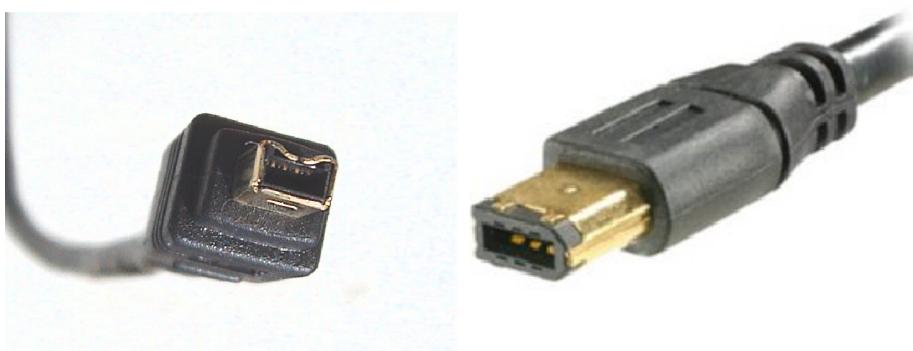
FireWire wordt vooral gebruikt voor het aansluiten van externe harde schijven en video- en audioapparaten. Het grote voordeel van FireWire tegenover USB is zijn hogere snelheid en betere stroomvoorziening. De initiële royalties die producerende bedrijven van FireWire-apparaten moesten betalen hebben echter de verspreiding van FireWire tegengehouden, in tegenstelling tot USB.

10.6.2 Werking

FireWire kan tot 63 randapparaten in een acyclische netwerkstructuur verbinden. FireWire voorziet daarnaast ook in peer-to-peer communicatie, zoals communicatie tussen een printer en scanner. Zoals USB laat FireWire meerdere hosts per bus toe, en met behulp van FireWire kan men zelfs een IP netwerk opzetten tussen 2 computers. Plug&play en hot swapping worden ondersteund, en de stroomvoorziening biedt tot 45 Watt per poort, waardoor er zwaardere randapparaten toch van stroom kunnen voorzien worden zonder externe bronnen.

10.6.3 Standaarden

FireWire 400, vastgelegd in de IEEE 1394a standaard, biedt data rates van 100, 200, of 400 Mbps (eigenlijk 98,304, 196,608 of 393,216 Mbps). De kabellengte is gelimiteerd tot maximaal 4,5 meter maar het is mogelijk om tot 16 keer door te koppelen aan andere apparaten wat het mogelijk maakt om tot 72m ver te gaan. Figuur 10.9 toont een 4-pins en 6-pins FireWire connector voor FireWire 400. FireWire 800 (dit is eigenlijk Apple's naam voor de 9-pins versie



Figuur 10.9: 4-pins(links) en 6-pins(rechts) FireWire connector

van de IEEE1394b standaard) werd in 2003 op de markt gebracht, en laat toe om tot 786,432

Mbps door te sturen. Deze nieuwste standaard heeft ervoor gezorgd dat FireWire nu toch een pak sneller is dan Hi-Speed USB. Figuur 10.10 toont een 9-pins connector voor FireWire 800.



Figuur 10.10: 9-pins FireWire connector

10.6.4 Hot swap gevaar...

Bij het uittrekken of insteken van een apparaat via FireWire, wanneer de computer aanstaat, bestaat het gevaar van een kortsluiting. Dit kan ertoe leiden dat het aangesloten apparaat onomkeerbaar beschadigd wordt, meestal aan de controller voor de FireWire interface. Het wordt dan ook aangeraden om FireWire apparaten aan te sluiten wanneer zowel het apparaat als de computer uitgeschakeld zijn.

10.7 eSATA

eSATA, gestandaardiseerd in 2004, is een variant op SATA die bedoeld is voor het aansluiten van externe apparaten zoals harde schijven. Ondanks het feit dat eSATA hogere datasnelheden kan halen dan USB en FireWire, wordt deze interface nog niet zo veel gebruikt voor het aansluiten van bv. externe harde schijven. Dat is wellicht te wijten aan het feit dat de meeste huidige computers zijn uitgerust met USB en/of Firewire, maar nog altijd met eSATA. Stilaan beginnen de meeste fabrikanten echter al 1 of meerdere eSATA aansluitingen te voorzien op de moederborden. Figuur 10.11 toont een eSATA kabel met de bijhorende eSATA interface.



Figuur 10.11: eSATA kabel met eSATA interface

10.8 Bluetooth

Bluetooth is een industriële specificatie voor draadloze communicatie op korte afstand (PAN of Personal Area Network). Bluetooth laat toe om bijvoorbeeld een PDA, GSM, laptop, PC, printer en digitale camera met elkaar te verbinden via de korte golf radiofrequentie, waarbij vooral wordt gekeken naar een zo laag mogelijk energieverbruik. Afhankelijk van de vermogensklasse van het product, moeten de twee communicerende apparaten dichter bij elkaar liggen, met een maximale afstand van 100m. Er zijn hierbij drie vermogensklassen:

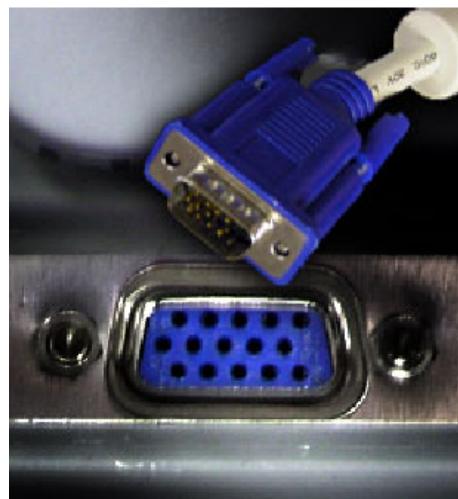
- Class 3 (1 mW) wordt het meest gebruikt en laat transmissies toe tot op 10m.
- Class 2 (2.5 mW) wordt het minste gebruikt en laat transmissies toe tot op 20m.
- Class 1 (100 mW) heeft de verste transmissiebereik: tot 100m.

Bluetooth werd oorspronkelijk ontwikkeld door Ericsson, en werd later verder gestandaardiseerd in de Bluetooth Special Interest Group (SIG). Typische voorbeelden van Bluetooth gebruik zijn de volgende:

- Wireless netwerken tussen desktop en laptop, of tussen desktops indien weinig bandbreedte gevraagd.
- Bluetooth randapparaten zoals printers, muizen en keyboards
- Bestandsverdracht (beelden, MP3, ...) tussen PDA's, GSM's onderling.
- Sommige MP3 spelers (traag...) en digitale camera's om alle bestanden naar de computer over te zetten.
- Car kits en headsets voor GSM's
- Medische toepassingen
- Communicatie tussen GPS receiver en PDA.

10.9 VGA-poort

De VGA-poort is een 15-polige SUB D connector, die wordt getoond in figuur 10.12. Tegenwoordig wordt meer en meer gebruik gemaakt van de DVI aansluiting. DVI staat voor Digital Video Interface en werd ontwikkeld om de beeldkwaliteit te maximaliseren op digitale schermen zoals flatscreens en projectoren. Figuur 10.13 toont een DVI-poort.



Figuur 10.12: VGA-poort met bijhorende stekker



Figuur 10.13: DVI-poort

Hoofdstuk 11

Randapparatuur

Inhoudsopgave

11.1 Toetsenbord	136
11.1.1 Geschiedenis	136
11.1.2 Opbouw	137
11.1.3 De microcontroller	138
11.1.4 Interrupts	138
11.2 Muis	140
11.2.1 Geschiedenis	140
11.2.2 Klassieke muis: opbouw en werking	140
11.2.3 Optische muis: opbouw en werking	141
11.2.4 RSI: Repetitive Strain Injury	142
11.3 Scherm	142
11.3.1 Eigenschappen van schermen	143
11.3.2 Geschiedenis	143
11.3.3 Werking van CRT schermen	144
11.3.4 Werking van TFT schermen	144
11.4 Geluid	145
11.5 Netwerken en modems	145
11.6 Scanner	146
11.7 Printer	146
11.8 Stroomvoorziening	147

11.1 Toetsenbord

11.1.1 Geschiedenis

Het toetsenbord is het belangrijkste invoerapparaat dat op een computer wordt aangesloten. Het toetsenbord bevat alfabetische en numerieke tekens en verder controlotoetsen en functie-

toetsen. Over de tijd heen werden toetsenborden uitgebreid met meer en meer toetsen, om in te gaan op de ook steeds bredere functionaliteit die computersystemen aanboden. Er zijn verschillende soorten layouts voor keyboards, waarbij azerty en qwerty de meest bekende zijn. Deze indelingen stammen eigenlijk nog terug van de oude typemachines, waarbij de afstand tussen twee opeenvolgende letters (lees: hamertjes die op het papier slaan) werd gemaximaliseerd om zo geen vastzittende hamertjes te hebben. Deze optimalisatie is gebaseerd op de Engelse taal, waarbij statistisch wordt bepaald hoe groot de kans is dat twee bepaalde letters na elkaar worden ingetyped.

Doordat de meeste mensen azerty of qwerty gewoon waren, werd deze indeling overgenomen bij de keyboards. Alhoewel keyboards geen last hebben van deze jamming, werken we dus nog wel met dat oude systeem. Daardoor typen we nog niet echt optimaal: het zou sneller kunnen en ook beter vanuit orthopedisch standpunt. Daarom werd door tegenstanders van de qwerty/azerty stijl het dvorak toetsenbord ontwikkeld, dat meer inspeelt op de nieuwe eisen, maar spijtig genoeg (in onze Europese landen) weinig gevolg kende.

11.1.2 Opbouw

Belangrijke aspecten in de bouw van een toetsenbord zijn:

- Plaats van de toetsen in de zogenaamde key matrix.
- Hoe ver een toets wordt ingedrukt, en hoe dit gefilterd kan worden.
- De snelheid waarmee een blijvend ingedrukte toets moet worden doorgegeven (zogenaamde typematics).

Ook een toetsenbord heeft een geïntegreerde kleine microprocessor om de toetsaanslagen te verwerken (de controller). Figuur 11.1 toont de key matrix, die ervoor zorgt dat elke toets een signaal kan sturen naar de controller: een wirwar van circuits. Wanneer een toets wordt inge-



Figuur 11.1: Key matrix

drukt, wordt een bepaald circuit gesloten waardoor er stroom kan door vloeien. De controller krijgt van deze circuits signalen doorgegeven. Figuur 11.2 toont een toetsenbord-controller. Wanneer zo'n circuit gesloten wordt, wordt dat circuit vergeleken met een karaktermapping in zijn ROM. Deze karaktermapping zet een bepaalde ingedrukte toets om in een zogenaamde



Figuur 11.2: Toetsenbordcontroller

scancode. Wanneer meer dan 1 toets tegelijk wordt ingedrukt, wordt gekeken of de combinatie een speciaal nieuw karakter produceert, zoals bijvoorbeeld een letter indrukken terwijl de shift-toets wordt ingedrukt. De karaktermapping die voorkomt in het keyboard kan overruled worden door een computer-karaktermapping. Bij typematics (wanneer de toets lang wordt ingedrukt) zal het keyboard deze toetsaanslag met een bepaalde snelheid doorgeven. Deze snelheid is softwarematig instelbaar, en kan variëren van 2 tot 30 toetsaanslagen per seconde.

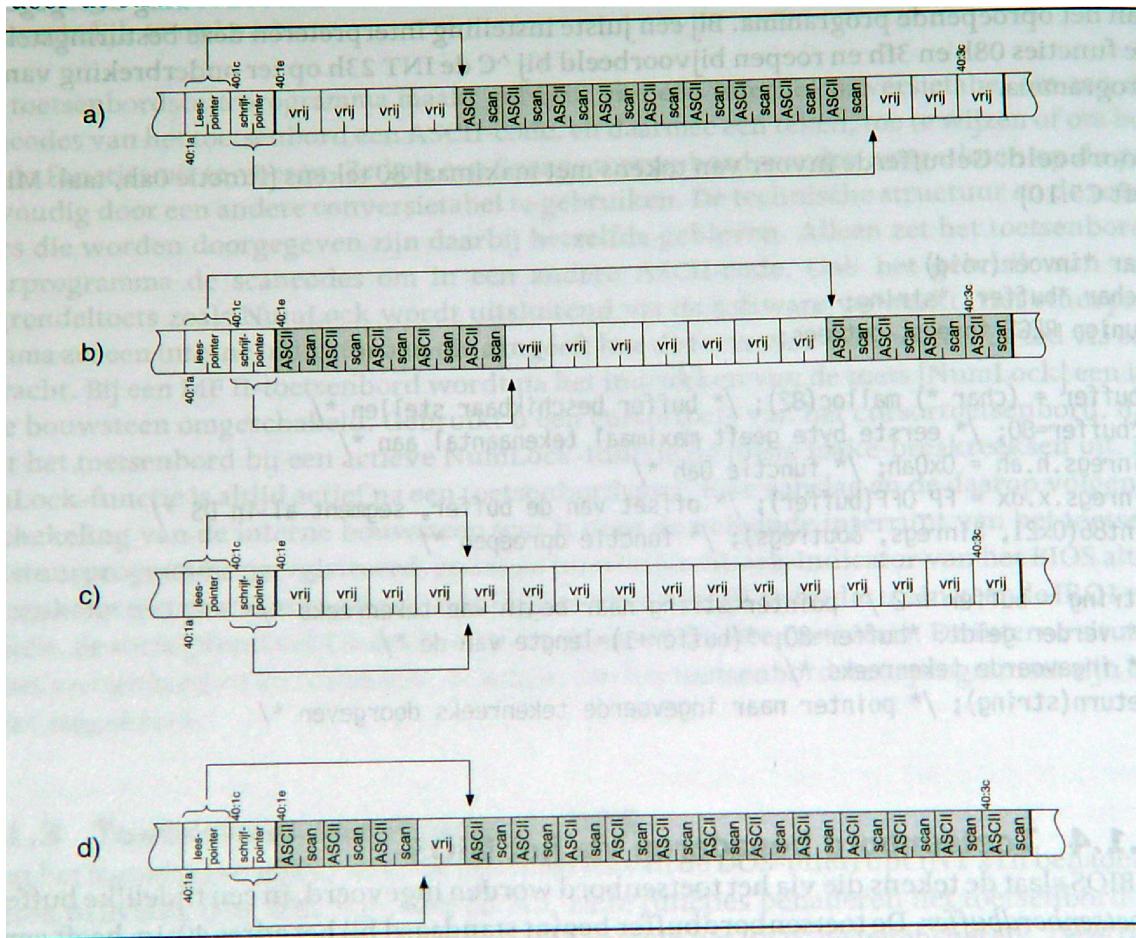
11.1.3 De microcontroller

Wanneer toetsen zeer snel achter elkaar worden ingedrukt, kan de microcontroller deze even bishouden in zijn buffer, dat meestal 16 karakters lang is. Daarna wordt deze stroom van toetsindrukken als scancodes doorgegeven naar de computer (meestal via USB of PS/2). De computer checkt dan of er eventueel een speciaal systeemcommando (zoals ctrl-alt-del) werd ingedrukt. Indien dit niet zo is, wordt de invoer gewoon doorgestuurd naar de lopende applicatie die de toetsaanslagen verder verwerkt.

11.1.4 Interrupts

Interrupts worden besproken in sectie 12.4, toch gaan we er voor de volledigheid al even op in. Wanneer de scancode aankomt op het moederbord, wordt er eigenlijk een interrupt gegenereerd. Dit is een soort signaal dat aan de processor laat weten dat er iets speciaals is gebeurd, bijvoorbeeld in- of uitvoer die klaar is. In dit geval wordt IRQ kanaal 1 gebruikt. De processor krijgt dus te horen dat het toetsenbord invoer heeft en zal deze invoer dan ook verwerken. Dit gebeurt aan de hand van en zogenaamde interrupt-handler, een stukje code dat deze invoer verder afhandelt. Ook in de computer zelf komt een toetsenbord-buffer voor waar achtereenvolgens de nog niet afgehandelde toetsaanslagen worden opgeslagen. Deze buffer is een zogenaamde FIFO (First In First Out) circulaire buffer met lees- en schrijfpointer. Figuur 11.3 maakt dit concept duidelijker. We bekijken deze buffer aan de hand van de tekeningen a, b, c en d:

- Afbeelding a: de leespointer wijst naar het volgende uit te lezen teken voor het besturingssysteem. Er wordt verwezen naar een bepaalde ASCII-scancode. Momenteel staan er in de buffer 9 nog niet behandelde scancodes.



Figuur 11.3: Circulaire buffer met lees- en schrijfpinger

- Afbeelding b: ondertussen staat de leespointer 8 scancode verder, wat wil zeggen dat het besturingssysteem 8 scancode verder heeft verwerkt. De schrijfpointer is echter ook verder komen te staan, zelfs zo ver dat er terug naar het begin van de buffer wordt gesprongen. Dit is echter geen probleem aangezien we werken met een circulaire buffer. We moeten enkel opletten dat de schrijfpointer nooit naar tekens wijst die de leespointer nog niet heeft behandeld.
- Afbeelding c: lees- en schrijfpinger wijzen hier naar hetzelfde elementje, er is momenteel geen scancode die nog niet behandeld is, de buffer is volledig leeg.
- Afbeelding d: er is nog maar 1 plaatsje vrij voor een scancode. Wanneer de gebruiker nu dus snel veel toetsen tegelijk indrukt, is de kans groot dat een aantal van deze toetsaanslagen genegeerd worden aangezien de buffer vol zit.

11.2 Muis

11.2.1 Geschiedenis

De muis werd in 1968 voorgesteld door Doug Engelbart, toen noemde het nog de x-y position indicator (eigenlijk een benaming die meer zin heeft). Doug Engelbart was naast de uitvinder van de muis ook grondlegger van andere ontzettend belangrijke computerconcepten zoals: windows op een scherm, online publishing, video-conferencing, e-mail, cooperating software, ... Figuur 11.4 toont deze eerste muis, samen met zijn uitvinder Doug Engelbart. Begin jaren '70



Figuur 11.4: Doug Engelbart met zijn uitvinding: de x-y position indicator

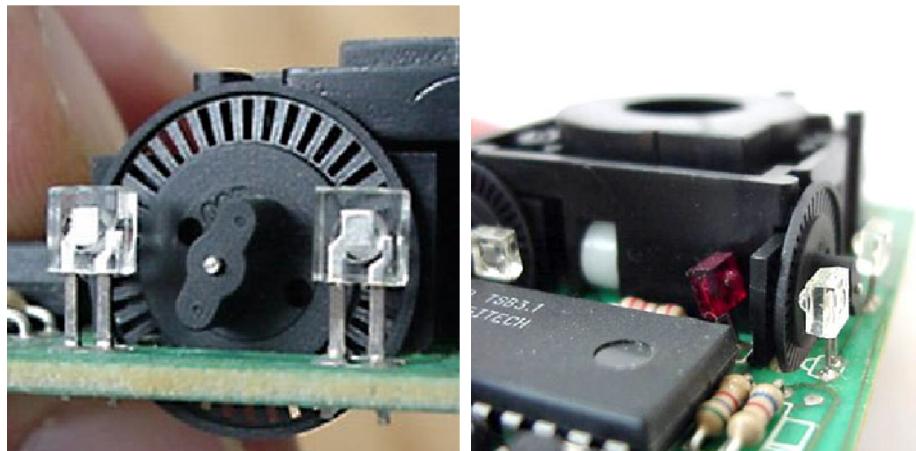
werden de wieltjes die rechtstreeks met de ondergrond contact maakten, vervangen door een bal die kon rollen in elke richting. De beweging van de bal werd gedetecteerd door 2 wieltjes die de X en Y bewegingen doorgaven. Tot op vandaag werken we nog steeds met deze variant van deze grote uitvinding.

11.2.2 Klassieke muis: opbouw en werking

Figuur 11.5 toont een doorsnede van een moderne muis met bal. Op deze figuur is duidelijk te zien dat er een bal in de muis zit. Deze bal rolt tegen twee staafjes, die op hun beurt een wieltje laten draaien. In dit wieltje zitten gaatjes, waar een lichtstraal wordt doorgezonden. Wanneer het wieltje draait, zal de lichtstraal onderbroken worden en kan gedetecteerd worden welke beweging er wordt gemaakt en hoe snel. Figuur 11.6 illustreert deze werking verder. De muiswieltjes worden in figuur 11.5 aangegeven door nummer 3, het lichtdetecterende systeem door nummer 5. Ook op muizen komt uiteraard een kleine controller voor, die de gedetecteerde beweging doorgeeft aan de computer, door het genereren van een interrupt. Ook een indrukken van bepaalde muisknoppen zal uiteraard leiden tot het doorgeven van een bepaald signaal naar het computersysteem.



Figuur 11.5: Doorsnede van een moderne muis met bal



Figuur 11.6: Muiswieltjes en lichtdetectoren

11.2.3 Optische muis: opbouw en werking

Tegenwoordig wordt meer en meer gebruik gemaakt van optische muizen. Bij optische muizen wordt niet gebruikt gemaakt van een balletje met wieltjes, maar van een LED dat licht naar het "rol"-oppervlakte uitstraalt. Het weerkaatste licht wordt gedetecteerd door een lichtgevoelige CMOS-sensor die het patroon (vergelijkbaar met een soort mini-fototoestel) doorstuurt naar de DSP (Digital Signal Processor) die zal bekijken hoe het patroon is verplaatst tegenover het vorige gecapteerde patroon! Als lichtstraler wordt dus meestal een LED (Light Emitting Diode)

gebruikt. De LED en het gedeelte dat het weerkaatste signaal weer oppikt wordt getoond in figuur 11.7. Het grote voordeel van optische muizen is dat ze beter bestand zijn tegen vuil



Figuur 11.7: LED en rechts het kanaaltje dat naar de CMOS chip leidt

(dit hoopt op op de wieltjes bij een klassieke muis) en dat de beweging precieser kan worden gemeten. Een nadeel is dat sommige oppervlakken niet geschikt zijn om met een optische muis op te werken.

11.2.4 RSI: Repetitive Strain Injury

Door langdurig gebruik van het keyboard en muis kunnen de spieren van onze handen overbelast worden. RSI uit zich in zeer vervelende klachten: chronische pijnen in hand, pols, schouder, nek die ontstaan uit de repetitieve bewegingen. Repetitive Strain Injury kan aan de andere kant eenvoudig vermeden worden door voldoende rustpauzes in te lassen, af en toe andere bewegingen te doen met de handen en een juiste zithouding. Sommige muismodellen werden dan weer ontwikkeld om RSI tegen te gaan, hierbij wordt de muis zelf niet bewogen, maar ligt er een bal bovenaan waar de mouse-pointer op het scherm kan verplaatst worden. Dit noemt men een trackball. Ook met zo'n speciale muizen is RSI nooit 100% te vermijden.

RSI kan ook u overkomen! Uit een recent onderzoek blijkt dat maar liefst de helft van de tweedejaarsstudenten en meer dan tweederde van de derdejaarsstudenten informatica aan een vorm van RSI lijden. Neem dus voorzorgen!

11.3 Scherm

Het scherm is het belangrijkste uitvoerapparaat op een computersysteem. Het laat ons toe te zien wat er gebeurt en geeft de veranderingen die we met invoerapparaten bekomen weer. Het scherm wordt ook wel monitor genoemd, naar de behuizing waarin het scherm wordt gezet. De meeste schermen gebruiken de cathode ray tube technology (CRT), maar de huidige markt wordt sterk beïnvloed door de LCD schermen (Liquid Crystal Display), de plasma en projectietechnologieën. Omdat ze minder energie verbruiken (en dus ook minder warmte ontwikkelen), platter zijn en dus een mooier design hebben, worden TFT LCD schermen meer en meer gebruikt. Het grote nadeel van de TFT technologie is de nog relatief hoge prijs. Een ander nadeel van TFT schermen is dat hun vernieuwingsfrequentie (refresh frequency) laag is tegenover de CRT. Voor de meeste kantoortoepassingen is echter geen snelle vernieuwingsfrequentie nodig.

11.3.1 Eigenschappen van schermen

Schermgrootte en aspect ratio

De schermgrootte geeft aan hoeveel inch een scherm aan zichtbare pixels bevat. Populaire schermgroottes zijn 15, 17, 19 en 21 inch. Uiteraard zal de schermgrootte ook de resolutie (zie verder) beïnvloeden: een bepaalde resolutie zal er fijn uitzien op een klein scherm en groffer op een groot scherm. De aspect ratio heeft te maken met de verhouding van de hoogte en de breedte. De twee meest bekende formaten zijn 4:3 en 16:9. Voor computerschermen wordt meestal nog de 4:3 standaard genomen, alhoewel 16:9 of het breedbeeldformaat een flinke opmars kent.

Resolutie en dot pitch

Resolutie geeft weer hoeveel individueel aanstuurbare kleurstippen er zijn, ook wel pixels genoemd. De resolutie wordt uitgedrukt in 2 termen: het aantal pixels in de breedte en in de hoogte. Typische resoluties zijn tegenwoordig 640x480, 800x600, 1024x768, 1280x1024 en 1600x1200. Dot pitch geeft weer hoeveel plaats er is tussen twee pixels op een scherm. Hoe dichter de pixels tegen elkaar worden gezet, hoe fijner en beter het scherm.

Vernieuwingsfrequentie (refresh rate)

De refresh rate is het aantal keer dat het scherm hertekend wordt per seconde. Refresh rates zijn vrij belangrijk, aangezien een te lage refresh rate flikkering tot gevolg zal hebben. Het is veel vermoeiender en slechter voor de ogen om op zo'n scherm te werken. De refresh rate en de resolutie zijn afhankelijk van elkaar, een hogere resolutie betekent een lagere mogelijke refresh rate. Dit wil zeggen dat deze twee instellingen tegen elkaar moeten afgewogen worden.

Kleurdiepte

De kleurdiepte bepaalt hoeveel verschillende kleuren er kunnen afgebeeld worden. Dit wordt berekend aan de hand van het aantal bits dat per kleur kan worden doorgegeven. Met een 24 bits kleur, 8 bits per rode, groene en blauwe component, kunnen we 10 miljoen kleuren vormen. Met 16 bits per kleur verkrijgen we 65536 verschillende kleuren.

11.3.2 Geschiedenis

Rond de jaren '70 maakte men gebruik van eenvoudige schermen waarop groene lettertjes werden afgebeeld. Over de jaren heen heeft zich een grote evolutie afgespeeld op het terrein van de schermen:

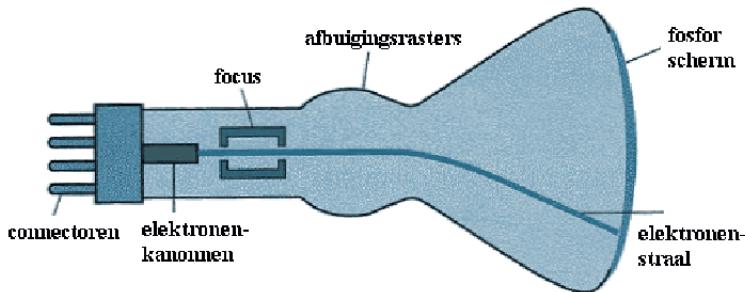
- 1981: IBM brengt de Color Graphics Adapter (CGA) op de markt, die 4 kleuren kon tonen en een maximale resolutie van 320 op 200 pixels had.
- 1984: IBM introduceert de Enhanced Graphics Adapter (EGA). EGA laat toe om tot 16 verschillende kleuren te tonen aan een resolutie van 640x350 pixels.
- 1987: IBM ontwikkelt de Video Graphics Array (VGA) display. De huidige computersystemen ondersteunen de VGA standaard en deze monitoren worden nog steeds gebruikt.

- 1990: IBM brengt de Extended Graphics Array (XGA) uit, die een resolutie van 800x600 pixels aanbiedt in true color (16,8 miljoen kleuren) en een 1024x768 resolutie met 65536 kleuren.

De dag van vandaag zijn de meeste schermen UXGA schermen. UXGA ondersteunt tot 16,8 miljoen kleuren aan een resolutie van 1600x1200. Een en ander is weer afhankelijk van de grafische kaart, die bepaalde resoluties zal ondersteunen met een daarbij horend aantal kleuren.

11.3.3 Werking van CRT schermen

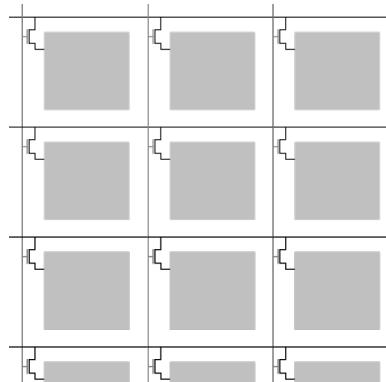
Een CRT scherm lijkt enorm hard op een gewone televisie. Zowel de televisie als een CRT scherm bevatten een beeldbuis. Men maakt hierbij gebruik van een elektronenkanon, dat elektronen afschiet door een rooster. Dit rooster zorgt dat de elektronen juist afbuigen en worden geschoten op een fluorescerende laag vooraan het beeldscherm. Figuur 11.8 illustreert de werking van een CRT monitor.



Figuur 11.8: Werking van een CRT scherm

11.3.4 Werking van TFT schermen

TFT is de afkorting van Thin Film Transistor, de belangrijkste componenten waaruit een TFT scherm is opgebouwd. Het TFT scherm is een variant van de LCD (Liquid Crystal Display) schermen. TFT LCD's worden gebruikt in televisies, computerschermen, GSM's, draagbare videogames, navigatiesystemen, MP3-spelers, ... Op hele kleine LCD-schermen, zoals die van rekenmachines, kan een voltage aangebracht worden op een individueel segment van het scherm. Dit is natuurlijk niet mogelijk voor grote schermen, aangezien er dan miljoenen verschillende verbindingen gemaakt moeten worden. Elke pixel zou 3 verbindingen nodig hebben, namelijk voor de R(ood), G(roen) en B(lauw) component. Om dit probleem te omzeilen, maakt men gebruik van pixeladressering per rij en kolom. Hierdoor verminderen het aantal verbindingen van miljoenen naar duizenden. In figuur 11.9 wordt dit geïllustreerd. Indien alle pixels in een bepaalde rij worden aangestuurd met een positief voltage en alle pixels in een bepaalde kolom met een negatief voltage, dan heeft de pixel in de kruising de hoogste voltage en wordt deze geactiveerd. Het probleem is dat de pixels op dezelfde rij of kolom ook een gedeelte van het voltage krijgen, waardoor deze ook gedeeltelijk aangestuurd worden. Om aan dit probleem te ontsnappen wordt elke pixel voorzien van een transistor switch die toelaat om elke pixel individueel te adresseren. De lage verliesstroom van de transistor zorgt ervoor dat er tussen 2 refreshes ook een lichte stroom aanwezig blijft op de pixel. Elke pixel is een kleine condensator



Figuur 11.9: Opbouw van matrix in een TFT scherm

met een laag isolerend vloeibaar kristal (liquid crystal), dit geheel wordt omgeven door 2 ITO lagen. In deze cursus gaan we niet dieper in op de constructie en opbouw van TFT schermen, dit is een vrij complexe materie.

11.4 Geluid

De eerste personal computers hadden enkel een kleine speaker, die beperkt was tot het voortbrengen van piepjess. Toch wisten bijvoorbeeld spelletjesmakers heel veel uit deze apparaatjes te halen. Al snel kwam de zogenaamde soundblaster op de markt, genoemd naar een bekend produkt van Creative. Zo'n geluidskaart bevat een DAC (Digital to Analog Convertor) die een digitaal geluid kan omzetten in een analoog geluid dat uiteindelijk door luidsprekers wordt gestuurd. De geluidskaarten worden ook voorzien van een microfooningang en eventueel een midi-interface voor communicatie met bijvoorbeeld midi-keyboards. De eerste multimedia-machines ontstonden met de introductie van deze geluidskaarten en de eerste CD-ROM's. De mogelijkheden waren dan ook zeer uitgebreid: spelletjes, muziek-CD's afspeLEN, ...

Tegenwoordig is de markt van de geluidskaarten (meestal geïntegreerd op het moederbord, of op een PCI-slot) geëvolueerd naar zogenaamde 5.1 systemen, waarbij een 3-dimensionaal effect wordt bekomen zoals dat vroeger enkel in cinema's beschikbaar was. De 5.1 kan verklaard worden aan de hand van het aantal benodigde luidsprekers: 2 achteraan, 2 vooraan, 1 center speaker ($2+2+1 = 5$) en de .1 die voor de extra bas luidspreker staat. Er zijn nu zelfs al 6.1 en 7.1 systemen op de markt.

11.5 Netwerken en modems

Met behulp van netwerkkaarten is het mogelijk om LAN-netwerken aan te leggen, waarbij een aantal computersystemen binnen een bepaald geografisch gebied (klein) worden verbonden tot 1 netwerk. Modems worden dan weer gebruikt voor WAN-verbindingen, waarbij een computer meestal wordt aangesloten op het bestaande telefoonnetwerk, nu echter om aan datacommunicatie te doen. In een modem vindt het proces van modulatie en demodulatie plaats, om informatie te sturen op de telefoondraad resp. te ontvangen en te digitaliseren.

Het vak “Datacommunicatie en netwerken” gaat veel dieper in op de mogelijkheden en theorie rond netwerken.

11.6 Scanner

Scanners laten ons toe om documenten in digitale vorm op de computer te zetten. Het document dat gescand moet worden, wordt beschenen met een zeer sterke lichtbron die door een spiegel slechts een deel van het origineel beschijnt. Door het draaien van deze spiegel wordt telkens een volgend deel ingescand. Per lijn wordt de reflectie opgenomen door een CCD (Charge Coupled Device), zoals die ook voorkomen in bijvoorbeeld videocamera's. De werking wordt geïllustreerd in figuur 11.10.



Figuur 11.10: Werking van een scanner

11.7 Printer

Printers vormen met het scherm een van de belangrijkste uitvoerapparaten van een moderne computer. Waar we volledig overgeschakeld zijn op digitaal aanmaken en bewerken van teksten, worden toch nog vele zaken op papier afgedrukt. Hiervoor maken we gebruik van een printer. Er zijn een aantal belangrijke soorten printers:

- De matrixprinter: de matrixprinter bevat een printkop met daarop een aantal kleine pennetjes. Deze pennetjes drukken inkt op papier, daarna schuift het papier op. Doordat er gewerkt wordt met de impact van deze pennetjes (een zogenaamde impactprinter), kan er eenvoudig een dubbel gemaakt worden. Daarom worden matrixprinters dikwijls nog gebruikt om bijvoorbeeld rekeningen af te drukken.
- Lijnprinters: een lijnprinter bevat ongeveer 130 wieltjes, waarbij op elk wieltje alle nodige tekens zijn aangebracht. Deze wieltjes draaien tot de juiste regel is gevormd, dan slaan hamertjes de letters tegen een lint dat uiteindelijk het papier zal bedrukken. Ook dit is een zogenaamde impactprinter.
- Thermische printers: deze printers werken met een speciaal soort papier, dat op bepaalde plaatsen wordt verhit. Het nadeel is uiteraard het speciale papier wat hiervoor moet voorhanden zijn.
- Inkjetprinters: deze printers werken met kleine spuitmondjes waardoor minuscule inktdruppeltjes op het papier worden gespoten.
- Laserprinters: bij laserprinters wordt gebruik gemaakt van het principe dat ook bij kopieermachines bekend is. Hierbij wordt een drum eerst volledig elektrisch neutraal gemaakt. Daarna wordt de rol volledig negatief geladen. Op plaatsen waar een laser heeft geschenen, komt een positieve lading te zitten. Daarna wordt het papier tussen een toner en

de drum gerold, waarbij de inkt op de plaatsen komt te zitten waar positieve ladingen zaten. Dit komt doordat het tonerpoeder negatief geladen is en aangetrokken wordt door de positieve lading. Tenslotte wordt het papier door een verwarmingseenheid gestuurd die de inkt fixeert op het papier.

11.8 Stroomvoorziening

Computers zijn zeer gevoelig aan stroomuitval. Ondanks het feit dat vele data opgeslagen wordt op de harde schijf, is de kans groot dat tijdens een stroomuitval gegevens verloren gaan. Het gaat dan vooral om gegevens die in het werkgeheugen worden bewerkt en die nog niet werden weggeschreven naar de harde schijf. Om problemen te vermijden bij stroomuitval, worden vele computers beschermd met een zogenaamd UPS-systeem (Uninterruptible Power Supply). De stroom wordt dan voorzien door een batterij-unit of een andere stroombron zoals een dieselgenerator. Zeker in het geval van grote serverparken met kritieke services (bv. banktransacties, websites, ...) is het belangrijk dat de servers constant online blijven. Wanneer de stroom uitvalt nemen de batterijen of generatoren het werk over en voorzien (een deel van) de computers van stroom. Meestal is dit slechts voor een korte periode, bv. 1 uur. Nadien moet de stroompanne verholpen zijn, anders zullen de servers zichzelf afsluiten. In het geval van dieselgeneratoren kan er uiteraard langer stroom voorzien worden, zolang de dieseltank niet leeg geraakt. Afbeelding 11.11 toont een eenvoudig UPS-systeem: een batterij met stroomvoorziening voor 1 computer.



Figuur 11.11: UPS systeem

Deel III

Software en inleiding tot besturingssystemen

Hoofdstuk 12

Software

Inhoudsopgave

12.1 Wat is software?	149
12.1.1 Programma's, instructiesets, machinecode	149
12.1.2 Assembler	150
12.1.3 Hogere programmeertalen	150
12.2 Registers	152
12.2.1 Breedte van registers	152
12.2.2 Algemene registers	153
12.2.3 Segmentregisters	154
12.2.4 Speciale registers	155
12.3 Geheugensegmentatie	156
12.3.1 Geheugenmodellen	156
12.3.2 Meer over de werking van enkele registers	156
12.4 In- en uitvoer	158
12.4.1 Interrupts	158
12.4.2 Exceptions (uitzonderingen)	160
12.5 RISC of CISC?	160
12.5.1 RISC-processoren	161
12.5.2 CISC-processoren	161
12.5.3 RISC vs. CISC	161
12.5.4 Het post-RISC tijdperk	162

12.1 Wat is software?

12.1.1 Programma's, instructiesets, machinecode

Een *programma* is een reeks bevelen of opdrachten die de processor moet uitvoeren of laten uitvoeren. Het is dus belangrijk dat de processor deze opdrachten begrijpt. Niet elke processor

begrijpt dezelfde bevelen, dit is eigen aan elk model van een processor. Voor elke processor wordt een zogenaamde *instructieset* bepaald. Dit zijn de opdrachten die de processor begrijpt en kan uitvoeren.

Deze instructieset bestaat uit opdrachten die eigenlijk gewoon een reeks van enen en nullen zijn. Als wij deze bits bekijken dan is dit meestal wel in hexadecimale vorm, dat is dan toch al iets makkelijker. Een instructieset van een processor wordt ook vaak de *machinecode* genoemd.

12.1.2 Assembler

Voor elke opdracht onderscheiden we de opdracht zelf (ook wel: opcode of operatie) en de parameters of operanden. Hoeveel parameters er zijn en hoe lang deze zijn (aantal bytes) hangt af van de opdracht. Sommige opdrachten hebben geen parameters, andere één, ... De opcode zelf kan ook uit een verschillend aantal bytes bestaan. Veel codes bestaan uit 1 byte, maar enkel daarmee zouden we slechts 256 verschillende opdrachten kunnen definiëren, daarom bestaan er ook opdrachten van 2 bytes. Al deze codes kunnen we vinden bij de specificaties van een processor. Maar met al deze -zeer elementaire- codes een programma opbouwen is zeer moeilijk, je moet alle instructies met hun machinecode kennen. Direct machinecode schrijven is moeilijk, maar mogelijk. Er is -indien geschreven in hexadecimale code- geen compiler nodig, deze code kan direct worden uitgevoerd. Het schrijven van hexadecimale code kan niet in een gewone editor, aangezien deze meestal enkel ascii ondersteunen.

Omdat ons brein beter is met woorden dan met codes, heeft men voor elke code een woord bedacht. Zo kunnen we de opdrachten beter onthouden. We spreken van een mnemomische code, of *assembler* code. Met assembler is het makkelijker programmeren terwijl je geen voordelen verliest tegenover de echte machinetaal. Er is een 1 op 1 mapping: 1 bevel in machinetaal = 1 bevel in assembler! Een compiler kan dan deze woorden lezen en omzetten naar machinecode die de computer begrijpt. Dit omzetten noemen we compileren.

12.1.3 Hogere programmeertalen

Hogere vs. lagere programmeertalen

Als er dan toch al een compiler is die onze geschreven woorden omzet in machinecode, kunnen we het ons makkelijker maken en een taal ontwikkelen om programma's te schrijven, een programmeertaal. Een compiler zal deze taal dan moeten begrijpen, en zal ook moeten weten welke machinecodes hij mag creëren. Een compiler is eigen aan een programmeertaal en aan een processortype. Een oude compiler zal weinig gebruik maken van nieuwe mogelijkheden van een nieuw type processor. Het schrijven van compilers is een van de meest ingewikkelde uitdagingen in de computerwereld.

In feite kunnen we stellen dat assembler ook een soort programmeertaal is, die heel dicht staat tegen de machinecode. Voor elke instructie is er een overeenkomende assembler-instructie. We kunnen dan ook meestal machinecode omzetten naar assembler, dit noemen we de-assembleren of de-compilieren.

In de loop der tijd werden verschillende hogere programmeertalen ontwikkeld. Bekende voorbeelden zijn C, Java, C++, C#, Eiffel, Visual Basic, ... Het grote voordeel van deze hogere programmeertalen is dat de code die we erin schrijven veel dichter staat bij de probleemwereld, dan bij lager-niveau talen. Als we bijvoorbeeld de object-georiënteerde programmeertalen

bekijken, zien we dat deze op een zeer natuurlijke manier de werkelijkheid voorstellen. Een ander groot voordeel aan een hogere programmeertaal is dat de code die geschreven wordt kan gecompileerd worden naar machinecode voor verschillende processoren. Zo kan een stukje C-code met behulp van de juiste compilers zowel gecompileerd worden naar machinecode voor een Pentium processor en naar machinecode voor een Sun Sparc processor. We hebben enkel een C-compiler nodig voor Pentium processoren en een C-compiler voor Sun Sparc processoren.

Bibliotheken

In hogere programmeertalen lossen programmeurs dikwijls problemen op die andere programmeurs ook al hebben tegengekomen. Voor veel gebruikte functionaliteiten (zoals het berekenen van random getallen, het afdrukken op het scherm, ...) worden zogenaamde libraries of bibliotheken geschreven, die vanuit de eigen code kan worden opgeroepen. Typische voorbeelden in C-code zijn de oproepen van de functies cout, cin die in de bibliotheek iostream zitten.

Ontwikkeling in teamverband

Naarmate de programmeertalen ontwikkelden en software ingewikkelder en groter werd, rees ook de vraag naar ontwikkeling van software in teamverband. Hierbij wordt een groot programma opgedeeld in zogenaamde modules, die individueel gecompileerd en getest kunnen worden. Nadien worden deze stukken bij elkaar gezet om zo de totale functionaliteit van het programma te bekomen. Dit bijeenzetten van stukken programma's noemen we linken. De linker zal ook een vrij ingewikkelde taak op zich nemen: het aanpassen van adressering (2 programmeurs uit aparte teams hebben misschien hetzelfde geheugen aangesproken...) met al zijn verwijzingen.

Semantiek van een programmeertaal

Wanneer we in een programmeertaal iets schrijven, heeft dit steeds een bepaalde betekenis. Bekijken we bijvoorbeeld het volgende C-fragment:

```

1 #include <iostream>
2 int main(){
3     int i, j;
4     i = 0;
5     j = i+1;
6     return 0;
7 }
```

Op lijn 3 zien we dat er twee variabelen worden gedeclareerd: i en j. Op lijn 4 wordt i op 0 gezet en tenslotte wordt j gezet op het getal dat 1 hoger is dan i op lijn 5. De betekenis die deze lijnen verklaart noemen we de semantiek van de taal. We kunnen in feite twee soorten fouten maken bij het programmeren:

- Syntaxfouten: dit zijn de taalregels van een bepaalde programmeertaal. Bijvoorbeeld het correct openen en sluiten van haakjes, gebruik van de juiste keywords, ... Wanneer we hierop fouten maken geeft de compiler ons een foutmelding en kunnen we achterhalen wat de fout is.
- Semantische fouten: dit zijn fouten in de achterliggende betekenis van een instructie. Ze worden niet opgemerkt door de compiler, en dat maakt ze net zo moeilijk opspoorbaar.

Een typisch voorbeeld van een syntaxfout is de volgende:

```

1 #include <iostream>
2 int main(){
3     int i;
4     i = 0
5     return 0;
6 }
```

Wanneer we dit laten compileren met de GNU C++ compiler, krijgen we volgende foutmelding:
Compiling source file(s)...

main.cpp
main.cpp: In function ‘int main()’:
main.cpp:5: error: syntax error before ‘return’
main.cpp:6:2: warning: no newline at end of file

De compiler geeft dus zelf al aan dat we een syntaxfout hebben gemaakt, wat ons dichtbij de juiste oplossing brengt: de puntkomma op lijn 4 was vergeten... Het is echter ook mogelijk om fouten te maken op de semantiek van het programma:

```

1 int main(){
2     int get1, get2, som;
3     cout << "geef_getal_1";
4     cin >> get1;
5     cout << "geef_getal_2";
6     cin >> get2;
7     som = get1 + get1;
8     cout << "De_som_is_" << som;
9 }
```

De compiler zal hier geen fout op geven (in het beste geval misschien een warning), waardoor het programma tot een uitvoerbaar programma wordt gecompileerd. Er staat wel degelijk een fout in de optelling, waar *get1* in plaats van *get2* bij zichzelf wordt opgeteld. De semantiek van het programma is nu verkeerd, aangezien we eigenlijk het eerste met het tweede getal wilden optellen.

12.2 Registers

Voor we verder gaan met effectieve Assembler bevelen, moeten we eerst weten welke registers zoal vorhanden zijn in een processor. De registers kunnen vergeleken worden met een soort kladblok voor de processor, waarop die zijn berekeningen kan uitvoeren. We bespreken in dit deel de registers van de intel x86-familie. Al deze registers zijn ook vandaag nog op moderne processoren terug te vinden, alhoewel er op zo'n nieuwe processor veel meer registers voorkomen voor bijvoorbeeld multimediale taken.

12.2.1 Breedte van registers

Een register wordt steeds gekenmerkt door een breedte: in de eerste architecturen waren de registers 8 bits breed, maar men breidde deze al snel uit tot 16-bits registers (vanaf de 8086).

Vanaf de 80386 werden de registers 32-bits breed, waarbij wel werd gedacht aan de achterwaartse compatibiliteit (backwards compatibility) met de vorige processoren. De nieuwste generatie processoren zijn 64-bits processoren, wat wil zeggen dat de registers hierbij 64 bits breed zijn. Merk op dat een processor pas echt nuttig kan gebruikt worden wanneer hij volledig wordt ondersteund door het besturingssysteem. Zo was er voor de 80386 niet direct een besturingssysteem vorhanden dat direct van de 32-bits registers gebruik maakte. Pas in 1991 kwam een versie van linux uit met ondersteuning voor 32-bits processoren.

12.2.2 Algemene registers

De algemene registers worden gebruikt voor de snelle opslag van gegevens, deze registers kunnen door de programmeur vrij gebruikt worden. In de 80386 werden 8 algemene registers geïntroduceerd: EAX, EBX, ECX, EDX, ESI, EDI, EBP en ESP. Ook vandaag nog zijn dit de enige algemene registers op een Intel 32-bit architectuur¹. We zullen deze verschillende registers in detail bekijken. In het algemeen kunnen we stellen dat de processor de inhoud van één of meerder registers leest, ze naar de ALU of FPU stuurt voor bewerking en nadien opnieuw opslaat in de registers. We moeten hierbij het verschil tussen gegevens en instructies nader bekijken: een instructie geeft weer welke bewerking moet worden uitgevoerd, de gegevens zijn datgene waarop deze bewerkingen worden uitgevoerd. Toch komen beide samen voor in het geheugen: dit gemengde geheugenmodel werd ontwikkeld door von Neumann. Het belangrijkste verschil tussen instructies en gegevens is:

- Gegevens worden door bepaalde instructies ingeladen in registers.
- Instructies zelf worden automatisch in de prefetch queue in de processor ingeladen vanuit het werkgeheugen.

De opdrachteenheid in de processor zal er dan voor zorgen dat de opeenvolgende instructies uit de prefetch queue worden gehaald, decodeert deze en stuurt vervolgens de opdracht naar de CVE. Deze opeenvolging wordt ook wel de *fetch-decode-execute*-cyclus genoemd. We geven nu wat meer uitleg over de algemene registers:

- **EAX:** accumulator, ter beschikking voor de meest geoptimaliseerde instructies. Voorbeeldcode in Assembler zou de volgende kunnen zijn: OUT 70h,AL. Hierbij wordt de waarde van AL uitgevoerd via de poort met hexadecimaal nummer 70. Merk op dat we -wegens achterwaartse compatibiliteit- de eerste 16 bits kunnen aanspreken via AX. Binnen de eerste 16 bits (AX) zijn de eerste 8 en de volgende 8 aanspreekbaar via AL (A Lower) en AH (A Higher). Het aanspreken van AX maakt het mogelijk om 16-bits programma's die geschreven werden voor de 8086 uit te voeren op de 80386. Bij de eerste microprocessors was AX het enige register waarmee gegevens konden worden opgeteld (optellen = accumuleren, vandaar de naam accumulator). Ook vandaag nog zijn veel opdrachten geoptimaliseerd voor dit register. Wanneer we bijvoorbeeld de vermenigvuldigingsinstructie bekijken, bevat EAX één van de factoren, en na de bewerking het resultaat. We kunnen dus stellen dat EAX een zeer belangrijk en veel gebruikt register is.

¹Meer informatie over de opbouw van moderne processoren kan je vinden in de uitgebreide Intel manuals. Zelfs wanneer je niet alles wil doornemen kan je hier en daar toch een aantal interessante zaken lezen. De documentatie is te vinden op <http://www.intel.com/products/processor/manuals/index.htm>

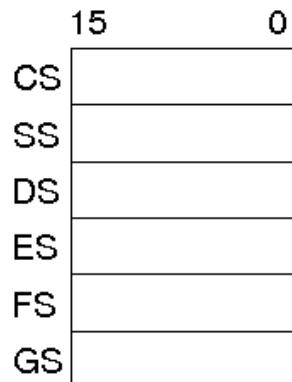
	31	16	15	0
EAX		AH	AL	
EBX		BH	BL	
ECX		CH	CL	
EDX		DH	DL	
ESI			SI	
EDI			DI	
EBP			BP	

Figuur 12.1: De algemene registers

- **EBX:** basisregister, dat wordt gebruikt voor tijdelijke opslag van gegevens. Daarnaast wordt EBX gebruikt als basis voor het verwijzen naar gegevensobjecten bij indirecte adressering.
- **ECX:** de counter (telregister), dat vooral wordt gebruikt bij het uitvoeren van lussen (bv. for-lus). Bij elke uitvoering van de lus wordt ECX met 1 verlaagd, tot ECX gelijk wordt aan 0. Het is voor een Assembler-programmeur geen verplichting om ECX te gebruiken voor lussen, maar het wordt wel dikwijls gedaan.
- **EDX:** dataregister, voor tijdelijke opslag van gegevens. Bij in- en uitvoerinstructies heeft DX een speciale functie: het bevat in dat geval het adres van de aan te spreken poort.
- **ESI:** source index (bronindex), voor tijdelijke gegevensopslag en als wijzer. Meestal wordt ESI gebruikt als index van een gegevensobject binnen een veld of soortgelijke structuur. Daarnaast wordt dit register gebruikt om vergelijkingen tussen twee registers te maken: ESI bevat dan het adres van het bronregister.
- **EDI:** destination index (doelindex), voor tijdelijke gegevensopslag en als wijzer. Als tegenhanger van ESI wordt in EDI het adres van het doelregister opgeslagen bij een vergelijking.
- **EBP:** base pointer (basispointer), die verwijst naar de basis van een stackframe. Deze basis van een stackframe wordt vaak gebruikt om de argumenten van een functie aan te spreken.
- **ESP:** de stack pointer, die aangeeft waar het huidige stackelement zich bevindt. SP bevat de eerste 16 bits van ESP.

12.2.3 Segmentregisters

In [12.3](#) bekijken we wat een segment juist is en waarom er geheugensegmentatie wordt toegepast. Voorlopig is het voldoende te weten dat het geheugen (logisch gezien) opgedeeld wordt in



Figuur 12.2: De segmentregisters

kleinere stukken, segmenten genaamd. Volgende segmentregisters zijn aanwezig in de 80386 (en vele opvolgers hiervan):

- **CS:** codesegment, bevat het adres van het segment dat de opdrachten omvat die onmiddellijk worden geadresseerd. De opdrachten worden binnen dit segment verder geadresseerd met behulp van de instruction pointer, EIP (zie [12.2.4](#)).
- **DS:** databasegment, segment met de gegevens van het programma dat momenteel draait.
- **SS:** stacksegment, het segment dat wordt gebruikt als stack. De stack wordt gebruikt om data eventjes opzij te zetten en later weer terug te halen.
- **ES, FS, GS:** extra databasegmenten voor bijvoorbeeld opdrachten met strings of met gegevens die buiten het databasegment liggen.

12.2.4 Speciale registers

Er worden ook enkele speciale registers gebruikt in de processor, waar de programmeur meestal zelf geen controle over heeft:

- **EIP:** de instruction pointer, die aangeeft welke de volgende uit te voeren instructie is. Elke keer er een instructie wordt uitgevoerd, wordt deze verhoogd met de grootte van de uitgevoerde instructie. IP kan ook aangesproken worden, dit zijn de eerste 16 bits van het register EIP.
- **EFLAG:** het vlagregister, waarvan de eerste 16 bits met FLAG opgevraagd kunnen worden. Volgende vlaggen worden opgeslagen in het vlagregister (in totaal dus 32 vlaggen): carry (wordt op 1 gezet indien er overdracht is voor de doeloperand), parity (1 indien een resultaat een even aantal 1-bits bevat), zero (1 indien het resultaat van een bewerking 0 is), sign (tekenbit), trap (wordt voor debugging gebruikt, indien op 1 wordt na elke instructie een interrupt gegenereerd), overflow (1 indien resultaat te groot of te klein), interrupt enable (instellen of de processor mag onderbroken worden voor het afhandelen van hardware-interrupt requests).

12.3 Geheugensegmentatie

12.3.1 Geheugenmodellen

In moderne processoren zijn er 3 geheugenmodellen beschikbaar:

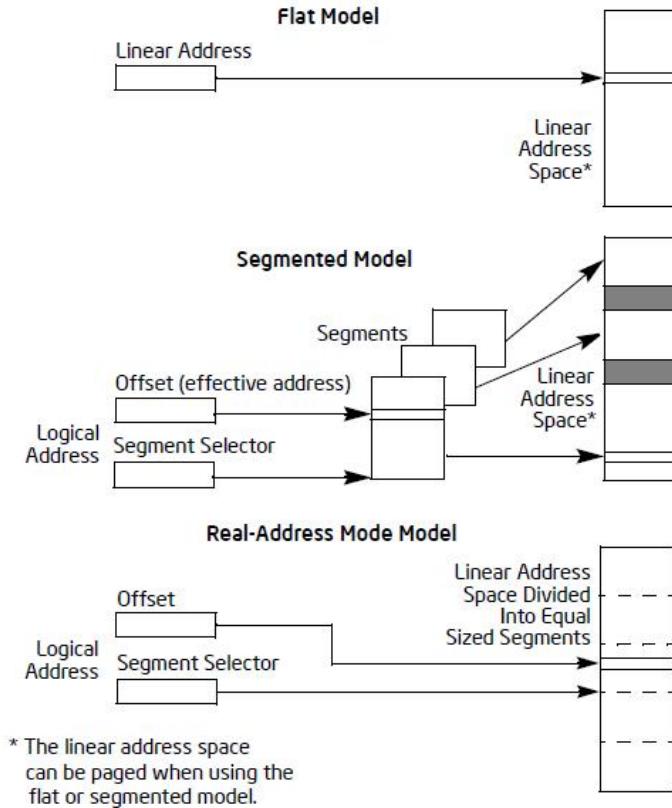
- **Flat memory model:** Voor een programma lijkt het geheugen als 1 groot aansluitend geheel. Men noemt dit ook wel een lineaire adresruimte, aangezien alle geheugenlocaties mooi op elkaar volgen als 1 grote lijst. Code, data en stapels zitten allemaal in deze adresruimte. Men kan bij dit model elke byte individueel adresseren.
- **Segmented memory model:** Voor een programma lijkt het geheugen opgesplitst in een aantal onafhankelijke adresruimtes, die men ook wel segmenten noemt. Code, data en stapels bevinden zich typisch in aparte segmenten. Om een byte te adresseren gebruikt het programma een logisch adres. Dit adres bestaat uit een segmentnummer en een offset (een verplaatsing binnen het segment). Men kan segmentatie gemakkelijk vergelijken met straten en huisnummers. Een straat geeft een segment aan, het huisnummer geeft elk huis in de straat een uniek adres. Het huisnummer is dus de offset. In de processor moeten de segmenten gemapt worden op het echte geheugen, dat gewoon lineair is en dus niet opgedeeld. De vertaling van deze adressen gebeurt volledig transparant voor een programma, het merkt er niets van. Het grote voordeel aan dit geheugenmodel is dat de betrouwbaarheid van programma's vergroot. Aangezien de stapel bijvoorbeeld in een apart segment voorkomt, kan deze niet per ongeluk stukken code of data overschrijven. Het feit dat een stack buiten zijn eigen voorziene geheugenbereik gaat noemt men ook wel een stack overflow.
- **Real-address mode memory model:** Het laatste model is het geheugenmodel dat afstamt van de 8086 processoren. Het wordt voorzien om compatibiliteit te voorzien met programma's die nog voor de 8086 processor werden geschreven. Bij het real-address geheugenmodel wordt het geheugen ook gesegmenteerd in vaste segmenten van 64 KB. De maximale grootte van de lineaire adresruimte is hier 2^{20} bytes. Dit komt van de 20-bits brede adresbus die voorkwam op de oude 8086 processor.

De verschillende geheugenmodellen worden geïllustreerd in figuur 12.3.

12.3.2 Meer over de werking van enkele registers

Codesegment en opdrachtteller CS:IP

Nu we weten dat een geheugenadres meestal is opgebouwd uit 2 delen (segmentnummer en offset) kunnen we meer in detail bekijken wat het doel is van enkele belangrijke registers. De instructionpointer is de offset van de volgende te lezen opdracht binnen een bepaald segment dat wordt gebruikt om code op te slaan. Het paar codesegment:opdrachtteller (CS:IP) bepaalt het volledige adres van de volgende instructie in het geheugen. Aan de hand van de opdrachtcodes weet de processor hoeveel bytes hij moet lezen. Na het lezen wordt de instructionpointer verhoogd met dit aantal bytes. We komen nu met CS:IP te wijzen naar de volgende uit te voeren instructie. Dit inlezen en vermeerderen van CS:IP voert de processor zelfstandig uit, de programmeur moet zich hier niets van aantrekken. In feite zitten we hier in het hart van de computer, aangezien het bijhouden van welke de volgende uit te voeren instructie enorm



Figuur 12.3: De 3 geheugenmodellen

belangrijk is. We kunnen dus gerust stellen dat het verhogen van de Instruction Pointer met het aantal bytes dat de instructie groot is de hartslag van de computer is! Door dit verhogen weet de processor wat het volgende is dat hij moet gaan doen.

Bij dit inlezen spelen prefetch-wachtrij en buseenheid natuurlijk een belangrijke rol. De code wordt eerst in de prefetch-wachtrij gelezen, en dit per aantal bits dat over de databus kan. Een databus van 32 bit breed kan pas “fetchen” als er 4 bytes vrij zijn in de wachtrij. Op deze manier wordt de bandbreedte van de databus optimaal gebruikt. Wanneer de processor bezig is (bv. 2 registers optellen) kan de buseenheid de wachtrij al opvullen zonder te storen. Hierdoor staat de volgende opdracht al klaar. Wel blijft een rechtstreekse geheugentoegang van de processor zelf natuurlijk voorrang hebben. (bv. MOV register, geheugen) Dit is echter ook voorspelbaar en kan ook al op voorhand gebeuren. Het wordt al gauw duidelijk dat een processor een zeer ingewikkelde taak op zich neemt, en dat we hier zelfs nog maar een deel van het verhaal vertellen.

Datasegment en adressering

Bij het lezen of schrijven van data in het geheugen wordt de offset in een register klaar gehouden, het paar DS:offset verwijst dan naar de effectieve data. Om tegelijkertijd te lezen of te schrijven in een ander segment kan men de overige segmentregisters ES, FS en GS gebruiken. Het overschrijven van een datasegmentregisters heeft desastreuze gevolgen: er zal in het verkeerde segment geschreven of gelezen worden!

Stacksegment en stackpointer

Een laatste segment heeft te maken met het oproepen van functies. Wanneer een functie wordt opgeroepen, moet CS:IP aangepast worden, omdat we nu in een andere functie willen verder gaan. Als we achteraf terug willen springen van waar we komen, moeten we het huidige CS:IP paar bewaren, dit gebeurt op de zogenaamde stack. Daarnaast wordt de stack gebruikt om bepaalde variabelen in op te slaan en om parameters mee te geven aan de desbetreffende functie.

12.4 In- en uitvoer

Een processor werkt in principe onophoudelijk voort, de EIP (instruction pointer) schuift bij elke uitgevoerde instructie verder op. Ook wanneer het lijkt dat een computer niets doet, is deze eigenlijk bezig met "wachten". Om een processor iets anders te laten doen moet hij onderbroken worden (interrupt). Er bestaan eigenlijk drie belangrijke soorten interrupts: software-interrupts, hardware-interrupts en exceptions (uitzonderingen)

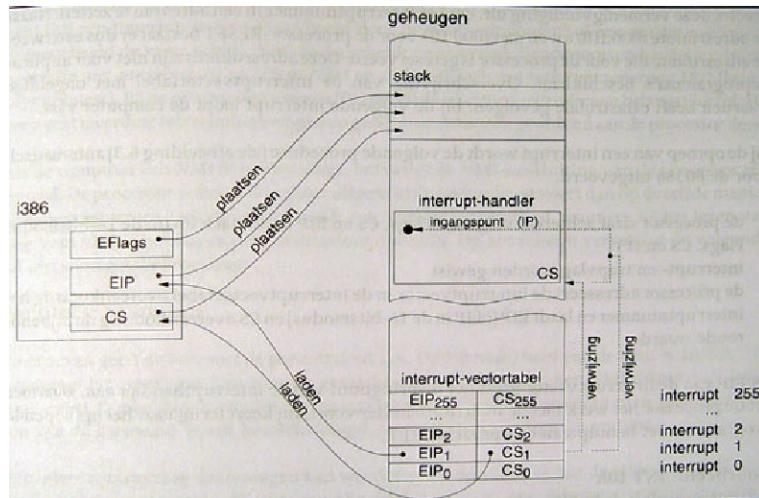
12.4.1 Interrupts

Software-interrupts

Een software-interrupt wordt verwezenlijkt via een zogenaamde INT-opdracht in de code. Een voorbeeld is INT 10h. Dit zal een bepaalde interrupt-routine gaan uitvoeren, namelijk die van interrupt 16 (hexadecimaal).

In de x86 architectuur worden de eerste 1024 bytes (1 KB) van het RAM-geheugen gereserveerd voor de interruptvectortabel. In de interruptvectortabel zelf zitten adressen van interrupt-handlers die een bepaalde interrupt zullen afhandelen. Aangezien deze adressen 4 bytes lang zijn, zullen er in de interruptvectortabel in totaal 256 mogelijke interrupts kunnen gezet worden. Elke vector (=adres) van een interrupt in de interruptvectortabel bestaat uit 2 bytes segmentnummer en 2 bytes offset (de verplaatsing binnen dit segment). Bij het oproepen van INT 10h wordt de interrupt-routine (we verklaren dadelijk wat dit is) opgeroepen waarvan het adres te vinden is op 0000:0064 (dus segment 0 offset 64). Dit is te verklaren door het feit dat we interrupt 16 willen oproepen (= 10 hexadecimaal), en dan elke vector 4 bytes lang is. Om het adres van de interruptvector te vinden moeten we dan het interruptnummer vermenigvuldigen met 4, wat ons $4 \times 16 = 64$ oplevert. Dan wordt verder gewerkt met het segmentnummer en de offset dat op deze plaats in de interruptvectortabel wordt gevonden. Inderdaad, de interruptroutine (interrupt-handler) kan nu worden opgeroepen. Figuur 12.4 maakt een en ander duidelijk. Het toont de eerste 1024 bytes van het geheugen, waarbij onderaan er dus 256 keer een CS:EIP paar voorkomt dat het adres bevat van de juiste interrupt-handler voor die interrupt. De interrupt-handler zelf is een stuk code ergens in het geheugen dat die interrupt op een correcte manier zal kunnen afhandelen. Om dit alles te verduidelijken, bekijken we een voorbeeld gebaseerd op figuur 12.4.

Stel dat interrupt 1 (we tellen vanaf 0) moet uitgevoerd worden. Dit komt overeen met de instructie INT 1h. In dat geval wordt in de interrupt-vectortabel gekeken naar adres $1 \times 4\text{bytes} = 4$, en daar vinden we EIP_1 en CS_1 . Deze bevatten dan weer het adres van de uit te voeren interrupt-handler. Inderdaad, als we nu deze CS en EIP in onze processor laden (wat ook op de figuur duidelijk wordt), zullen we als volgende instructie de eerste instructie van



Figuur 12.4: De interruptvectortabel en interrupt-handlers

de interrupt-handler uitvoeren. Dit wordt aangegeven in de figuur als het ingangspunt van de interrupt-handler. In tegenstelling tot het oproepen van een zelfgeschreven functie is het oproepen van interrupts veel eenvoudiger. Door het uitvoeren van de instructie INT 1h zullen automatisch de huidige EIP en CS en de vlagregisters op de stapel gezet worden. Dit is belangrijk aangezien we na de uitvoering van de interrupt-handler moeten kunnen terug springen naar de functie van waaruit de interrupt werd opgeroepen. Er zijn dus geen speciale regels (bv. caller's-rules) waar we rekening mee moeten houden.

Hardware-interrupts

Interrupts worden, in tegenstelling tot software-interrupts, niet in de assembler-code opgeroepen. Ze komen vanuit een hardware-apparaat, bv. het aflopen van een timer, de harde schijf die klaar is met inlezen en dit wil signaliseren aan de processor, een seriële poort die gegevens heeft ontvangen, ... Binnen de hardware-interrupts bestaan er twee belangrijke soorten:

- De niet-maskeerbare interrupt (NMI): deze komt sowieso de processor storen en de interrupt-handler wordt altijd uitgevoerd.
- De maskeerbare interruptaanvraag IRQ (interrupt request): deze kan tijdelijk worden uitgesteld, door het aan- of afzetten van de zogenaamde interrupt-vlag IE (Interrupt Enable). Met behulp van het commando CLI (Clear Interrupt) kunnen we het binnenkomen van interrupts maskeren (dus negeren), bij het uitvoeren van STI worden interrupts weer toegelaten.

Wanneer een interrupt wordt opgeroepen met behulp van een INT-instructie, wordt automatisch voor het uitvoeren van de interrupt-handler CLI uitgevoerd. Dit om te vermijden dat er andere interrupts komen terwijl een reeds openstaande interrupt wordt afgehandeld. Hardware interrupts zijn vastgelegd aan de hand van de IRQ-kanalen. Deze liggen op de meeste systemen als volgt vast:

- IRQ 0: timer (hoogste prioriteit)
- IRQ 1: toetsenbord

- IRQ 2: gereserveerd
- IRQ 3: COM 2
- IRQ 4: COM 1
- IRQ 5: LPT2
- IRQ 7: LPT1
- IRQ 8: real-time klok
- IRQ 14: schijfcontroller

Deze IRQ-nummers kunnen op een computer eenvoudig bekijken worden bij het opstarten ervan.

12.4.2 Exceptions (uitzonderingen)

De oorzaak van een exception is de processor zelf. Meestal komt dit door een bepaalde fout die gebeurt is in de processor. Het is dan ook vergelijkbaar met een software-interrupt, maar dan niet-maskeerbaar. Ook hier zijn er weer verschillende soorten:

- Fault: resulteert in exception, vóór dat de huidige opdracht afgehandeld is. De EIP wijst dus naar opdracht die verantwoordelijk is voor de exception. De processor kan dan bijvoorbeeld opnieuw proberen de opdracht uit te voeren. Voorbeeld: "segment niet aanwezig" Oplossing: ander segment inladen?
- Trap: resulteert in een exception, nadat de huidige opdracht is uitgevoerd, EIP wijst dus naar de volgende uit te voeren instructie. Dit heeft men nodig wanneer het programma correct wordt uitgevoerd, maar het programma onderbroken moet worden. Voorbeeld: breakpoints in een debugger (hierbij stopt men de uitvoering van het programma bij een bepaalde instructie om de registers te bekijken (of variabelen) en op die manier na te gaan of alles correct verloopt).
- Abort: levert niet het adres van de fout (in tegenstelling tot fault en trap). Het programma kan in dit geval dus niet altijd worden hervat. Een typisch voorbeeld van een abort is een hardware-fout, die dan resulteert in het heropstarten van het systeem.

We geven ter verduidelijking nog een aantal voorbeelden van exceptions in figuur 12.5. Exception 0 is een deling door 0, exception 1 is de zogenaamde trap, die een exception zal genereren na elke uitgevoerde instructie. Dit is handig voor het debuggen van programma's. Exception 3 wordt gebruikt voor breakpoints, exception 5 wanneer er buiten de grenzen van een array wordt gegaan, ...

12.5 RISC of CISC?

Men onderscheidt twee categorieën processoren: de CISC (Complex Instruction Set Computer) en RISC (Reduced Instruction Set Computer) architecturen. De processoren die in de pc's zitten (zoals die van Intel of AMD) zijn van het type CISC, terwijl de computers van bijvoorbeeld Sun Microsystems (de Sparc processoren) van het RISC type zijn.

	Seg high	Seg low	Offset high	Offset low
Byte 3				
Byte 2				
Byte 1				
Byte 0				
000H				
Divide error				
004H				
Single-step				
008H				
NMI pin				
00CH				
1-byte breakpoint				
00DH				
Overflow (INTO)				
010H				
Bound				
014H				
Undefined Opcode				
018H				
Coprocessor not avail				
01CH				
Double fault				
020H				
Coproc seg overrun				
024H				
Segment not present				
028H				
Invalid task state seg				
02AH				
Stack seg overrun				
030H				
General protection				
034H				
Page fault				
038H				
Unassigned				
03CH				
Coprocessor error				
040H				
32-255 User defined				
14-31 Reserved				
080H				

The interrupt vector table is located in the first 1024 bytes of memory at addresses 000000H through 0003FFH.

There are 256 4-byte entries (segment and offset in real mode).

Figuur 12.5: Mogelijke exceptions

12.5.1 RISC-processoren

Het concept van RISC-processoren werd rond 1974 ontwikkeld door John Cocke van IBM. Zijn argument was dat van de computers van die tijd slechts 20 % van de mogelijke instructies gebruikten. Een processor van het RISC-principe zou dan weinig instructies toelaten, waardoor er minder transistoren zouden nodig zijn en de prijs dus ook zou dalen. Door het aantal transistoren en instructies te verminderen tot wat het meest gebruikt wordt, krijgt de computer meer gedaan in een kortere periode. De meest bekende RISC-processor is zeker en vast de Sun Microsystems SPARC-processor. Opvolgers hiervan waren de zogenaamde MIPS RISC microprocessoren (Microprocessor without Interlocked Pipe Stages). Vanuit dit principe werd dan weer de Motorola 68000 processor ontwikkeld, die lang een populaire processor bij Apple macintosh systemen was. De enige huidige RISC-processor op de consumentenmarkt is de PowerPC processor van Apple. Ook de bekende ARM-processoren zijn van het type RISC. Bij RISC-processoren ligt de nadruk vooral op software.

12.5.2 CISC-processoren

CISC (Complex Instruction Set Computer) is een retroactieve definitie die puur werd uitgevonden om ze te kunnen onderscheiden van de RISC processoren. In tegenstelling tot RISC hebben CISC processoren zeer veel verschillende en complexe instructies. Het argument hiervoor is dat chips zo ontwikkeld kunnen worden dat deze complexe instructies op hardware niveau kunnen uitgevoerd worden. Door de hoge kost van geheugen (een tijdje geleden...) en permanente opslag werden CISC processoren als beter beschouwd, doordat ze kleine, snelle code bevatten. De evolutie heeft ons echter geleerd dat software-grootte niet echt een probleem meer is. Bij CISC-processoren ligt de nadruk vooral op hardware.

12.5.3 RISC vs. CISC

RISC, voor- en nadelen:

- RISC-processoren kunnen veel goedkoper en veel sneller gemaakt worden, een Apple Mac G3 is qua performantie een stuk beter dan zijn Intel-equivalent.
- Instructies worden 4x zo snel uitgevoerd.
- Meer lijnen code om dezelfde resultaten te behalen. Dit verhoogt de grootte en de complexiteit van software.

CISC, voor- en nadelen:

- CISC processoren zijn duurder in constructie...ware het niet dat de markt (lees: de thuisgebruiker) zich toegespitst heeft op de CISC processoren. Dit komt omdat verschillende producenten zich op deze markt stortten: Intel, AMD, Cyrix,... Door deze concurrentiestrijd vielen de CISC processoren toch goedkoper uit.
- Men lijkt geleidelijk aan af te stappen van CISC-processoren. Het is veel makkelijker voor de hardwaremakers om de uitvoering van RISC instructies te parallelizeren dan de uitvoering van CISC-instructies.

12.5.4 Het post-RISC tijdperk

Het probleem van vandaag is dat een processor moeilijk volledig RISC of CISC kan genoemd worden. Zo werden de RISC processoren ingewikkelder qua complexiteit (vergelijk de G4 processor maar met de PPC 601) en CISC chipsets werden meer efficiënt. Het resultaat van al deze ontwikkelingen is dat een processor RISC of CISC wordt genoemd op basis van zijn voorgangers...een PowerPC 601 processor ondersteund bijvoorbeeld meer instructies dan een Pentium.

Hoofdstuk 13

Besturingssystemen

Inhoudsopgave

13.1 Doelen en functies van besturingssystemen	164
13.1.1 Geschiedenis	164
13.1.2 Functies	165
13.2 Process management	166
13.2.1 Processen	166
13.2.2 Procestabel	167
13.2.3 Multiprogramming	168
13.2.4 Interproces communicatie	168
13.2.5 Sheduling	169
13.2.6 Enkele scheduling algoritmen	170
13.2.7 Deadlocks	171
13.3 Memory management	173
13.3.1 Swapping	173
13.3.2 Paging	174
13.3.3 Virtueel geheugen	175
13.4 File management	175
13.4.1 Blokgrootte	175
13.4.2 Allocatie	175
13.4.3 Beheer van vrije ruimte	176
13.4.4 Veiligheid	176

Zoals we in de vorige hoofdstukken hebben gezien, is een modern computersysteem een complex geheel van interne hardware en randapparaten. Het is een onmogelijke taak om als programmeur vertrouwd te zijn met alle specifieke eigenschappen van deze componenten. Van daar dat deze complexiteit door middel van een besturingssysteem wordt teruggedrongen tot een bepaald niveau. Zoals het woord zelf zegt, is een besturingssysteem de bestuurder van alle hardware die is aangesloten op een computer. Het zal een abstractie maken van alle gebruikte

hardware, het besturingssysteem biedt als het ware een eenvoudige en uniforme interface naar de hardware aan.

13.1 Doelen en functies van besturingssystemen

13.1.1 Geschiedenis

Om beter te begrijpen hoe een modern besturingssysteem in elkaar zit, overlopen we kort de geschiedenis ervan.

Batchsystemen

De eerste computersystemen die werden ontwikkeld werkten volgens het principe van batch. Bij batch worden verschillende applicaties 1 na 1 gedraaid op een bepaald systeem. Door de hoge kost van de eerste computersystemen was het belangrijk om de processor bijna constant bezig te houden. Om hieraan te voldoen werden de programma's -toen nog ingegeven op ponskaarten- aan een batch-operator gegeven die de ponskaarten inlas en opsloeg op tape. Daarna werd de tape overgebracht naar het eigenlijke computersysteem, waar de echte verwerking plaats vond.

Batchsystemen met multiprogramming

Onder multiprogramming verstaan we het feit dat er zich 2 of meer programma's tegelijk in het geheugen bevinden. De introductie van multiprogramming volgde al gauw uit het feit dat de processor soms lange tijd werkloos was doordat hij moest wachten op in- of uitvoer. Men kwam op het idee om tijdens die periodes van werkloosheid andere processen aan de beurt te laten om zo de efficiëntie verder te verhogen. Een ander systeem dat naast multiprogramming werd ontwikkeld was spooling. Hierbij werden in- of uitvoeropdrachten in een wachtrij geplaatst en werden deze uitgevoerd van zodra er tijd voor was.

Timesharing systemen

Naast de multiprogramming systemen werden ook timesharing systemen ontwikkeld. Hierbij konden verschillende gebruikers tegelijk via een terminal een connectie maken naar eenzelfde computersysteem, dat zijn processortijd verdeelde tussen de verschillende gebruikers.

Personal computer

Dankzij de opkomst van LSI-schakelingen (Large Scale Integration) begonnen de prijzen voor computersystemen enorm te dalen. Daardoor was het mogelijk elke werknemer (of particulier) een aparte computer te geven. Doordat er veel meer eindgebruikers in contact kwamen met de computer, werd er veel meer aandacht besteed aan de gebruiksvriendelijkheid van de systemen. De belangrijkste besturingssystemen die werden ontwikkeld waren Unix, Microsoft DOS en Windows en Apple Macintosh.

Andere varianten

Sinds de jaren '80 zijn er nog andere belangrijke evoluties merkbaar:

- Real-time systemen: dit zijn systemen waarbij timing enorm van belang is. Hard real-time systemen moeten absoluut binnen een bepaald tijdsinterval reageren, soft real-time systemen kunnen al eens een deadline missen. Een typisch voorbeeld van een hard real-time systeem is software voor het monitoren van een kerncentrale. Hier is het belangrijk dat bij een forse stijging van de temperatuur (of andere gemeten waarden) zeer snel de juiste acties worden ondernomen, of er dreigt een kernramp.
- Embedded systemen: met de huidige markt van GSM's, MP3-spelers, ... zijn embedded systemen enorm belangrijk geworden. Ook bijvoorbeeld besturingssystemen in vliegtuigen vallen onder deze noemer. Het zijn besturingssystemen die op een speciaal apparaat (device) moeten draaien, dikwijls met speciale eisen (laag energieverbruik, ...)
- Parallelle systemen: dit zijn systemen die meerdere processoren bevatten om op die manier de gelijktijdige verwerking van verschillende taken te optimaliseren. Dit moet echter ondersteund worden door het besturingssysteem.
- Gedistribueerde systemen: systemen die verbonden zijn via een netwerkverbinding.

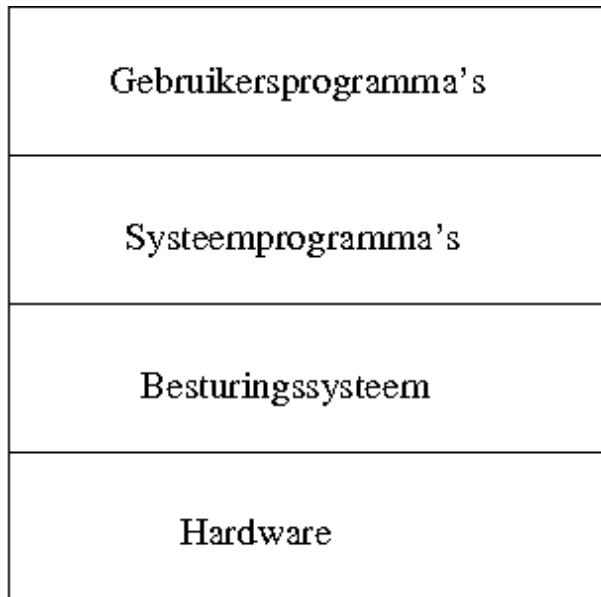
13.1.2 Functies

Een besturingssysteem is een programma dat de uitvoering van applicaties (gebruikersprogramma's) in goede banen leidt en een interface vormt tussen de gebruiker van een computer en de computerhardware. We kunnen dus stellen dat een besturingssysteem (Operating System of OS) drie belangrijke doelen heeft:

1. Gemak: een besturingssysteem zorgt ervoor dat een computer eenvoudig te gebruiken is.
2. Efficiëntie: de hardware die aangesloten is op een computersysteem zo efficiënt mogelijk gebruiken.
3. Mogelijkheid tot ontwikkeling: het aanbrengen en wijzigen van systeemfuncties moet kunnen gebeuren, zonder dat hierdoor andere diensten worden aangetast.

Om de functionaliteiten van een besturingssysteem uit te leggen wordt meestal gebruik gemaakt van een lagenmodel. Dit lagenmodel wordt weergegeven in figuur 13.1. De onderste laag in dit model stelt de hardware voor die we in de vorige hoofdstukken in detail hebben besproken: processor, harde schijf, werkgeheugen, ... In het lagenmodel is de eerstvolgende laag het besturingssysteem, waaruit afgeleid kan worden dat het besturingssysteem zal moeten weten hoe deze hardware aan te spreken. Bovenop het besturingssysteem komen enkele systeemprogramma's, die eigenlijk losstaan van het besturingssysteem zelf maar toch handige tools kunnen zijn. Hieronder verstaan we bijvoorbeeld de compilers, shells en debug-tools. Merk dus op dat de shell (of command line interface) van Linux/Unix en DOS geen deel uitmaakt van het besturingssysteem zelf. Ook grafische omgevingen kunnen onder de noemer systeemprogramma's geplaatst worden. Bovenop de systeemprogramma's komt er nog een extra laag, die van de gebruikersprogramma's of applicaties. Dit zijn de dagelijks gebruikte software-programma's zoals een email-programma, een web-browser, tekstverwerker, fotobewerkingssoftware, ...

In een lagenmodel wordt steeds de nadruk gelegd op de onafhankelijkheid tussen de verschillende lagen. Zo zou men een bepaalde laag kunnen vervangen, zonder dat de andere lagen daar last van ondervinden. Zolang de onder- of bovenliggende laag dezelfde interface krijgt



Figuur 13.1: Het lagenmodel: functionaliteit van besturingssystemen

aangeboden blijft alles werken. Een mooi voorbeeld hiervan is een linux kernel, die de core bevat van alle code die de hardware-componenten aanspreekt. Men kan een nieuwe kernel (dus die bijvoorbeeld nieuwere hardware ondersteunt, en meestal op een snellere manier) plaatsen zonder dat de bovenliggende systeemprogramma's daar hinder van ondervinden.

Er is een belangrijk verschil tussen systeemfuncties en het besturingssysteem zelf doordat systeemfuncties in user mode draaien (een beschermd modus op moderne processoren) en het besturingssysteem in kernel mode. De kernel van een besturingssysteem is als het ware de kern, met daarin geheugenbeheer, beheren van beschikbare processortijd en de interface om de hardware aan te spreken. We hebben ook meteen de drie hoofdactiviteiten van een modern besturingssysteem opgesomd:

- Process management: het beheren van verschillende processen die tegelijk in het besturingssysteem draaien.
- Memory management: het beheren van het werkgeheugen. Verschillende processen vragen geheugen aan, en het besturingssysteem zorgt ervoor dat deze verschillende processen niet in elkaar's vaarwater komen.
- File management: het beheren van alle bestanden die op de harde schijf voorkomen, alsook op andere opslagmedia.

13.2 Process management

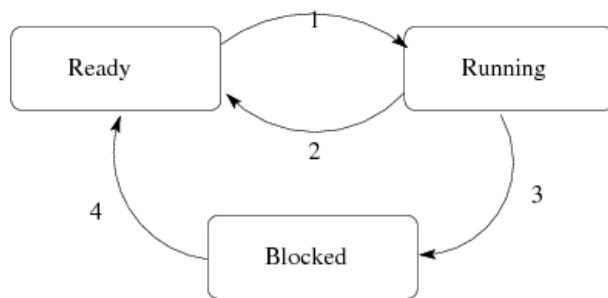
13.2.1 Processen

Vanuit de processor gezien voert de processor instructies uit zijn repertoire (het werkgeheugen) uit in een volgorde die wordt bepaald door de veranderende waarden in het register met de programmateller (de EIP of Extended Instruction Pointer). Tijdens deze uitvoering kan de EIP

verwijzen naar code in verschillende programma's die deel uitmaken van verschillende toepassingen. Vanuit een afzonderlijk programma gezien bestaat de uitvoering van een programma uit een opeenvolging van instructies binnen dat programma. De uitvoering van een afzonderlijk programma noemen we een proces of taak. Elk proces kan drie toestanden hebben:

1. Running: het proces wordt momenteel uitgevoerd op de processor.
2. Ready: het proces is klaar om uitgevoerd te worden, maar moet de processor momenteel afstaan aan een ander proces.
3. Blocked: het proces wacht op een externe gebeurtenis (zoals het voltooien van een in- of uitvoeroperatie)

De overgangen die tussen deze drie toestanden mogelijk zijn, worden aangegeven in figuur 13.2. Hierbij kan een proces van de ready naar de running state gaan (overgang 1) wanneer het gekozen wordt door de processor om uitgevoerd te worden. Wanneer het proces niet meer verder mag uitgevoerd worden (bv. omdat er even een ander proces tussenkomt) wordt het via overgang 2 terug naar de ready state gebracht. Overgangen 1 en 2 worden door de zogenaamde process scheduler uitgevoerd: deze zal beslissen welk proces er momenteel mag uitgevoerd worden en welke processen er moeten wachten in de ready queue. Een andere overgang is er bij 3: een proces kan van running naar blocked gaan doordat het op in- of uitvoer wacht, of omdat het geblokkeerd wordt door een ander proces. Overgang 4 kan tenslotte voorkomen wanneer de in- of uitvoer van een proces is afgelopen of omwille van een andere reden die ervoor zorgt dat het proces terug kan uitgevoerd worden op de processor.



Figuur 13.2: Proces status

13.2.2 Procestabel

Per proces wordt er informatie bijgehouden in de zogenaamde procestabel. Deze informatie is nodig om een proces te kunnen herstellen wanneer het eerder werd gestopt (in de blocked of ready queue gezet) en terug moet kunnen worden uitgevoerd. De belangrijkste gegevens die per proces worden bijgehouden zijn:

- De program counter: de waarde van de programmateller, of ook wel EIP of Extended Instruction Pointer. Dit om ervoor te zorgen dat wanneer het proces terug op de processor wordt uitgevoerd, er op de juiste positie in het programma wordt verdergegaan. (waar tervoren de uitvoering werd gestopt)

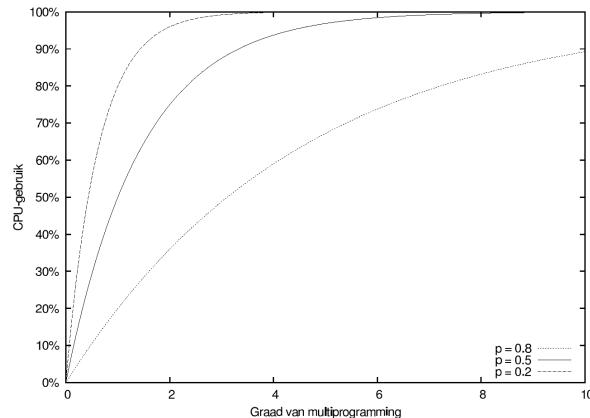
- Het ouderproces: indien dit proces wordt opgeroepen door een ander proces, aangeven wie de oproeper is.
- Starttijd van het proces: wanneer werd het proces opgestart?
- Gebruikte CPU-tijd: hoe lang heeft het proces al op de processor kunnen uitvoeren sinds het opstarten ervan.
- Eigenaar-informatie: wie heeft het proces opgestart?
- Sheduling-informatie: afhankelijk van het gebruikte scheduling-algoritme. Een scheduling-algoritme zal bepalen welk proces er op een bepaalde ogenblik op de processor mag uitgevoerd worden.
- Lijst met open bestanden: lijst met alle bestanden die door het programma worden gebruikt.

13.2.3 Multiprogramming

Een modern computersysteem laat toe dat meerdere processen zich tegelijk in het geheugen bevinden. Dit noemt men ook wel multiprogramming. De CPU wordt dan afwisselend toegewezen aan de verschillende aanwezige processen. Het bepalen van welk proces de CPU krijgt toegewezen noemt men scheduling. Het gebruik van multiprogramming kan het CPU-gebruik verhogen. Indien een proces gemiddeld een fractie p van de tijd aan het wachten is op een I/O-operatie, betekent dit dat de CPU een fractie $1 - p$ van de tijd niet wordt gebruikt. Bij gebruik van monoprogrammatie en $p = 0.8$ wordt de processor gedurende 20% van de tijd gebruikt. In een systeem met multiprogrammatie wordt dit $1 - p^n$, met n het aantal gelijktijdige processen die worden uitgevoerd. n wordt ook wel de graad van multiprogramming genoemd. Wanneer p hoog is (en het proces dus een groot deel van de tijd moet wachten op bijvoorbeeld in- of uitvoer) kan door multiprogramming het gebruik van de processor sterk verhoogd worden. Figuur 13.3 geeft aan hoe het processorgebruik verhoogd wanneer er multiprogramming wordt toegepast. Ook wordt in deze figuur duidelijk dat indien p niet zo hoog is, we al snel aan de 100% processorgebruik zitten en er dus in principe niet veel processen meer hoeven bij te komen. In verband met processorgebruik wordt meestal gestreefd naar 100%, zonder dat daardoor te veel processen moeten wachten om uitgevoerd te worden.

13.2.4 Interproces communicatie

Dikwijls moeten verschillende processen met elkaar kunnen communiceren, bijvoorbeeld door het doorgeven van berichten aan elkaar. Ook moeten soms verschillende processen op dezelfde variabele(n) (gedeelde variabelen dus) werken. Men spreekt van zogenaamde race condities indien het resultaat van een reeks processen afhangt van de volgorde waarin de opdrachten werden ingegeven. Om te vermijden dat meerdere processen tegelijk bepaalde gegevens veranderen, moeten we gebruik maken van een zogenaamde kritische sectie. Deze sectie is een gedeelte van de code dat onafgebroken moet kunnen worden uitgevoerd om een juiste werking te garanderen. Er zijn een heleboel manieren ontwikkeld om kritische secties uit te voeren, zoals busy waiting, het algoritme van Peterson, Test en Set, Sleep en Wakeup, semaforen en monitors. Deze onderwerpen bespreken zou ons te ver leiden voor deze inleidende cursus.



Figuur 13.3: Graad van multiprogramming

13.2.5 Scheduling

CPU- en I/O-bursts

Het typische verloop van een proces is een afwisseling van CPU-bursts en I/O-bursts. Tijdens de CPU-bursts worden gedurende korte tijd CPU-intensieve berekeningen gemaakt. Tijdens een I/O-burst wordt er dan weer vooral aan in- of uitvoer gedaan. Van deze typische afwisseling tussen CPU- en I/O-bursts kunnen we gebruik maken bij multiprogrammatie. Wanneer een bepaald proces aan zijn I/O-burst begint, kan een ander proces op de processor toegelaten worden om zijn CPU-burst gedeelte te laten uitvoeren. In realiteit is de tijd tussen CPU- en I/O-burst niet mooi 50-50 verdeeld, maar men kan toch de efficiënt gebruikte tijd van de CPU verhogen door multiprogrammatie.

Process scheduling

Om te beslissen welk proces op de processor mag uitvoeren, of welk proces gedurende een tijdje moet wachten, wordt gebruik gemaakt van process scheduling algoritmen. Er zijn verschillende algoritmen, en meestal worden de algoritmen op volgende factoren vergeleken:

- Rechtvaardigheid: elk proces krijgt een vergelijkbaar aandeel van de CPU-tijd toegeewezen.
- Efficiëntie: zorg dat de processor steeds 100% van de tijd aan het werk is.
- Responstijd: minimaliseer de responstijd voor interactieve gebruikers, zodat ze hun systeem vooruit zien gaan en geen noemenswaardige vertragingen merken.
- Turnaround: minimaliseer de wachttijd voor gebruikers van batch processen.
- Throughput: verwerk een zo hoog mogelijk aantal processen per uur.

Afhankelijk van de situatie kan een van bovenstaande factoren belangrijker worden.

Preemptive vs. non-preemptive scheduling

Er bestaan twee belangrijke soorten van scheduling: de preemptieve en niet-preemptieve scheduling algoritmen. De oudste soort is niet-preemptieve scheduling. Hier wordt slechts overgeschaald naar een ander proces wanneer het huidige proces op de CPU zelf naar de waiting-queue

gaat (bv. door een lock op een variabele) of naar de blocked-queue door I/O. Bij preëmptieve scheduling algoritmen kan een proces op elk willekeurig moment onderbroken worden om plaats te ruimen voor een ander proces. MS Windows gebruikte tot versie 3.11 de niet-preëmptieve scheduling, vanaf Windows 95 en Unix werkte men met preëmptieve scheduling algoritmen.

13.2.6 Enkele scheduling algoritmen

We bekijken nu achtereenvolgens een aantal scheduling-algoritmen die vandaag de dag worden toegepast in enkele besturingssystemen.

Round Robin

Het Round Robin algoritme is het meeste eenvoudige scheduling algoritme. Hierbij houdt het besturingssysteem een lijst bij van alle processen die klaar zijn om uitgevoerd te worden. Elk proces krijgt een zelfde quantum CPU-tijd toebedeeld. Na de verwerking van elk proces wordt het proces achteraan in de lijst geplaatst en wordt begonnen met de uitvoering van het volgende proces. Een probleem bij Round Robin is het bepalen van het tijdsquantum. Als we dit zeer klein nemen, zal elk proces voortgang maken maar hebben we last van overhead door de zogenaamde context switch. Alle informatie van het lopende proces moet namelijk opgeslagen worden, en opnieuw ingeladen wanneer het terug mag worden uitgevoerd. Als aan de andere kant het tijdsquantum te hoog wordt ingesteld, kan het zijn dat de responsied voor interactieve processen te lang is.

Shortest Job First

Bij Round Robin kreeg elk proces eenzelfde tijdsquantum CPU-tijd toebedeeld. Bij het shortest job first algoritme wordt eerst het proces met de kortste uitvoeringstijd op de processor toegelaten, daarna het proces met de tweede kortste uitvoeringstijd, ... Natuurlijk moeten we hier op een of andere manier aan gegevens geraken van een proces: wat is nu juist de uitvoeringstijd van zo'n proces? Dit kan bijvoorbeeld op basis van voorgaande uitvoeringen van datzelfde proces berekend worden.

Scheduling met prioriteiten

Indien men prioriteiten toekent aan processen is het mogelijk dat processen met een hogere prioriteit meer CPU-tijd krijgen toebedeeld. Historisch groeide dit systeem door processen van het management een hogere prioriteit toe te kennen dan deze van de andere werknemers. In moderne computersystemen kunnen prioriteiten echter nog een belangrijkere rol spelen: zo kan het proces dat de gebruiker momenteel gebruikt (actief op de voorgrond in de grafische user interface) een hogere prioriteit krijgen. Zo zorgen we ervoor dat we een reactief systeem hebben. Bij gebruik van prioriteiten wordt er voor elke prioriteitsklasse een aparte wachtrij aangemaakt. De scheduler zal eerst kijken of er zich processen in de wachtrij van de hoogste prioriteit bevinden. Indien dit het geval is, worden de processen in deze wachtrij uitgevoerd (hiervoor kunnen de andere scheduling algoritmen gebruikt worden). Indien er zich geen processen meer bevinden in de hoogste prioriteitsklasse, wordt de volgende klasse onderzocht. Er moet echter vermeden worden dat processen met een lagere prioriteit nooit worden uitgevoerd. Vandaar dat

men meestal een systeem voorziet waarbij een proces nadat het een aantal keer is uitgevoerd, toegewezen wordt aan een klasse met een lagere prioriteit.

Fair share scheduling

Dit scheduling algoritme zorgt ervoor dat alle gebruikers een gelijke portie CPU-tijd krijgen toebedeeld. Als er n gebruikers zijn aangelogd, zal elke gebruiker $1/n$ van de CPU-tijd toegewezen krijgen.

Real-time scheduling

Bij real-time systemen is het belangrijk dat bepaalde processen voor een deadline worden afgewerkt. Er zijn twee soorten deadlines: soft deadlines (mogen eventueel gemist worden, kan niet zo veel kwaad) en hard deadlines (mogen nooit gemist worden). Bij real-time systemen kent men vaak periodieke events: dit wil zeggen dat een proces elke x microseconden een bepaalde taak laat uitvoeren op de CPU. Indien er n processen zijn die elk met een interval P_i voorkomen en C_i CPU-tijd opvragen, is het systeem *schedulable* indien:

$$\sum_{i=1}^n \frac{C_i}{P_i} \leq 1 \quad (13.1)$$

Een mogelijk algoritme voor real-time scheduling is het rate monotonic algoritme. Hierbij krijgt elk proces een prioriteit die omgekeerd evenredig is met zijn frequentie van voorkomen. De scheduler kiest telkens voor het proces met de hoogste prioriteit. Onderzoek heeft aangetoond dat dit algoritme optimaal werkt.

Andere algoritmes zijn earliest deadline first, waarbij het proces wordt gekozen waarvan de deadline het kortst is genaderd en least laxity. In deze laatste wordt eerst de *laxity* berekend als het verschil tussen de tijd resterend tot de deadline en de benodigde CPU-tijd. Het proces met de kleinste laxity wordt geselecteerd.

13.2.7 Deadlocks

Indien er zich meerdere processen in het geheugen bevinden die gebruik willen maken van de computer resources (randapparaten, bestanden, ...), kunnen er deadlocks optreden. Een **deadlock** treedt op wanneer een groep processen wacht op een bepaalde gebeurtenis die enkel kan veroorzaakt worden door een ander proces in de groep. Een typisch voorbeeld van een deadlock is het volgende: proces A en proces B hebben beide zowel de printer als de scanner nodig. Stel dat proces A eerst de printer alloceert, en deze toegewezen krijgt. Stel nu tevens dat proces B eerst de scanner alloceert en deze ook toegewezen krijgt. Wanneer proces A de scanner wil aanvragen, zal deze benomen zijn door B, en vice versa. We zitten in een zogenaamde deadlock, aangezien de twee processen geblokkeerd zullen blijven tot een van de processen een apparaat weer vrijgeeft.

Volgende voorwaarden moeten gelijktijdig waar zijn voor het verkrijgen van een deadlock:

1. **Wederzijdse uitsluiting:** een resource is steeds aan exact één proces toegewezen, of is vrij beschikbaar.
2. **Hold and wait:** een proces dat reeds een resource heeft toegewezen gekregen mag bijkomende resources aanvragen.

3. **Non-preemptive:** een toegewezen resource kan enkel worden vrijgegeven door het proces dat de resource bezit. De resource is dus niet-afneembaar.
4. **Cyclisch wachten:** twee of meer processen wachten op een resource die is toegewezen aan het volgende proces in de keten.

Er zijn verschillende oplossingen voor deadlock-problemen:

- Negeren van het probleem (deadlocks kunnen nog steeds voorkomen)
- Detectie en opheffing: detecteren wanneer een deadlock is opgetreden, en deze oplossen.
- Vermijden: door middel van de juiste scheduling algoritmen ervoor zorgen dat geen deadlock-situatie optreedt.
- Onmogelijk maken: een van bovenstaande voorwaarden voor een deadlock niet toelaten.

Negeren van deadlocks

Het negeren van deadlocks wordt ook wel het *struisvogelgoritme* genoemd. Hoewel dit op het eerste zicht een onacceptabele oplossing is, is dit vaak een goede oplossing. Indien de frequentie van het voorkomen van een deadlock veel lager ligt dan de gemiddelde falingsfrequentie van het systeem ten gevolge van andere bugs of hardwareproblemen, wegen de nadelen van de andere strategieën (een lagere performantie en gebruiksgemak) niet op tegen de potentiële voordelen. Diverse besturingssystemen, waaronder Unix, gebruiken deze strategie.

Detectie en opheffing van deadlocks

Volgens deze strategie worden deadlocks toegelaten. Wanneer een deadlock plaatsvindt, wordt deze door het besturingssysteem gedetecteerd en opgelost. De algoritmen die gebruikt worden voor detectie van deadlocks, zijn meestal gebaseerd op de grafentheorie. Wanneer in een graaf een lus wordt gedetecteerd, is er kans op deadlock.

Vermijden van deadlocks

Processen vragen zelden van in het begin alle resources aan die ze tijdens hun verloop zullen nodig hebben. Indien op voorhand voor elk proces is geweten hoeveel resources het zal gebruiken, kan met een juist scheduling algoritme deadlock vermeden worden.

Onmogelijk maken

In het begin van deze sectie hebben we de 4 voorwaarden besproken waaraan tegelijk moet voldaan zijn opdat een deadlock zou kunnen optreden. Indien we deadlocks willen vermijden, moeten we zorgen dat aan minstens één van deze voorwaarden is voldaan. Dit kan als volgt:

- Wederzijdse uitsluiting: Om dit te vermijden zouden we elke resource exclusief aan één proces moeten toekennen. In het geval van een printer kan dit gebeuren door gebruik te maken van een zogenaamde printer daemon. Enkel dit proces kan effectief iets sturen naar de printer, andere processen kunnen gebruik maken van deze daemon. Dit systeem noemt men spooling.

- Hold and wait: Indien we processen verplichten om helemaal in het begin van hun uitvoering op te geven welke resources het proces zal gebruiken, kan vermeden worden dat er tijdens de uitvoering van het programma te weinig resources beschikbaar zijn. Dit is echter geen realistische veronderstelling, aangezien resources soms wel en dan weer niet worden gebruikt door een proces. Men kan echter een variant hierop ontwikkelen waarbij een proces eerst tijdelijk alle resources dat het momenteel heeft moet afgeven vooraleer een nieuwe resource wordt aangevraagd.
- Non-preemptive: Het afnemen van resources tijdens de uitvoering van een proces is niet echt een evidentie oplossing: voor sommige apparaten (printers, scanner) kan dit leiden tot grote problemen.
- Cyclisch wachten: Indien resources in een door het besturingssysteem opgelegde volgorde moeten worden aangevraagd kan vermeden worden dat er cyclisch gewacht wordt. Door een bepaalde volgorde op te leggen kan er namelijk nooit een lus in de aanvragen ontstaan. Bij complexe systemen is het echter niet evident om zo een volgorde te vinden, waarbij alle situaties haalbaar zijn.

13.3 Memory management

13.3.1 Swapping

Op een besturingssysteem met zeer veel draaiende processen (denk aan multi-user omgevingen of zelfs single-user) is het dikwijls niet mogelijk om alle processen op dezelfde moment in het werkgeheugen te laden. Omdat een proces kan uitgevoerd worden, moet het zich echter in het werkgeheugen bevinden. Om dit op te lossen wordt gebruik gemaakt van zogenaamde swapping, waarbij sommige processen voor bepaalde tijd worden stilgelegd en even naar de harde schijf worden weggeschreven. Dit gebeurt op een zogenaamde swap file of swap partitie op de harde schijf.

Bij gebruik van swapping wordt het geheugen onderverdeeld in partities. Processen kunnen dan toegewezen worden aan een partitie. Doorgaans worden partities van variabele grootte gebruikt, om niet te veel geheugen te verspillen. Men spreekt dan ook van variabele partities.

Bij het toewijzen (alloceren) van geheugenruimte aan processen moet ermee rekening gehouden worden dat de grootte van het programma kan wijzigen gedurende de uitvoering van het programma. De geheugenruimte van een programma wordt doorgaans in 3 delen opgesplitst:

- Text segment: deze bevat de werkelijke code en de globale variabelen.
- Stack segment: deze bevat de stapel van het programma, een dynamische geheugenstructuur die belangrijk is voor het oproepen van functies. Immers, bij het oproepen van een functie worden argumenten, lokale variabelen en andere belangrijke waarden op de stapel gezet.
- Heap segment: bij het aanmaken van dynamisch gealloceerde variabelen (met behulp van bv. `malloc()`) wordt de heap gebruikt. Ook om bijvoorbeeld aangemaakte objecten (in een object-geïndiceerde programmeertaal) op te slaan wordt de heap gebruikt.

Merk op dat zowel de stack als de heap tijdens de uitvoering van een programma kunnen vergroten. Daarnaast is het belangrijk te weten dat we met segmenten hier niet verwijzen naar de segmentatie die voorkwam in de 8086 architectuur, die eerdere besproken werd. Verwar beide dus niet met elkaar!

Allocatie

Bij het inswappen van een proces in het geheugen moet het besturingssysteem bepalen op welke plaats het proces zal ingeladen worden. Het beschikbare geheugen wordt hiervoor in gelijke blokken van n bytes opgedeeld. Elk blok is ofwel ongebruikt, ofwel toegewezen aan een proces. Het geheugen kan dan worden voorgesteld als een aaneenschakeling van procesruimten (P) en gaps of gaten (G). Het besturingssysteem houdt dan een gelinkte lijst bij van alle processen en de gaps.

Wanneer een nieuw proces wordt ingeswapt, kan de gelinkte lijst worden gebruikt om ongebruikte geheugenruimte te vinden. Hiervoor bestaan een aantal mogelijke werkwijzen:

- First fit: Het proces krijgt het eerste gap toegewezen dat groot genoeg is. Er wordt telkens gestart vanaf het eerste element in de lijst.
- Next fit: Een variant op first fit waarbij het zoeken wordt gestart vanaf het punt waarop men de vorige keer is gestopt.
- Best fit: De volledige lijst van gaps wordt onderzocht en het blok dat net groot genoeg is wordt toegewezen.
- Worst fit: Het omgekeerde van best fit: hier wordt de grootste beschikbare gap toegewezen.

Hoewel men zou denken dat best fit de beste resultaten zou opleveren, heeft onderzoek uitgewezen dat first fit het beste algoritme is. Het nadeel van best fit is dat er allemaal kleine gaps overblijven waarmee niets meer kan gedaan worden.

Statistische analyses hebben aangetoond dat bij het gebruik van first fit het gemiddeld aantal gaps in het geheugen gelijk is aan de helft van het gemiddeld aantal processen. Na verloop van tijd kunnen er zeer veel kleine gaps zijn, waardoor er geen ruimte meer kan toegeewezen worden aan nieuwe processen, hoewel het totaal beschikbare geheugen (tel alle kleine gaps op) wel zou volstaan. Hier spreekt men van fragmentatie. Een oplossing hiervoor is de zogenaamde compactie, waarbij de verschillende gebruikte geheugenruimtes worden samengevoegd, zodat weer één grote vrije geheugenruimte ontstaat.

13.3.2 Paging

Het probleem van fragmentatie kan worden opgelost door gebruik te maken van paging. Hierbij is het niet vereist dat elk proces een aaneensluitend blok in het geheugen inneemt. In plaats daarvan wordt de fysieke geheugenruimte verdeeld in een aantal frames van vaste grootte (meestal 4 KB). Het logische geheugen wordt ingedeeld in pages van dezelfde grootte als de frames. Wanneer een programma toegang vraagt tot een bepaald geheugenadres, moet er een vertaling plaatsvinden van het logische geheugenadres naar het fysieke geheugenadres. Deze taak wordt uitgevoerd door de Memory Management Unit (MMU). De CPU geeft het logische

adres door aan deze eenheid, die het vertaalt naar een fysiek adres en dit doorstuurt naar het geheugen.

13.3.3 Virtueel geheugen

Bij het gebruik van virtueel geheugen is het mogelijk dat een programma meer geheugen gebruikt dan er fysiek beschikbaar is. Men steunt op het principe dat een programma nooit volledig in het geheugen moet aanwezig zijn: bepaalde delen van het geheugen kunnen (voorlopig) weggeschreven worden naar de harde schijf en later terug ingeladen worden wanneer deze nodig zijn. Hetzelfde geldt wanneer er meerdere processen tegelijk actief zijn. Wanneer een gevraagd paginanummer niet in het geheugen zit treedt er een paginafout (page fault) op. Het wegschrijven van een pagina naar de harde schijf is een redelijk dure (lees: tijdrovende) operatie. Vandaar dat er meestal een extra bit wordt opgeslagen per pagina die aangeeft of deze gewijzigd werd. Indien dit niet het geval is, kan zo'n pagina gewoon overschreven worden zonder informatieverlies.

13.4 File management

13.4.1 Blokgrootte

Het beheer van een bestandssysteem vertoont veel overeenkomsten met het beheer van geheugen. Een eerste vraag die moet gesteld worden is hoe bestanden op de schijf worden opgeslagen: als een aaneensluitende reeks van bytes of wordt het bestand opgedeeld in blokken van gelijke grootte? In het eerste geval zou het bestand telkens verplaatst moeten worden van zodra het groter wordt. De tweede oplossing is ook niet ideaal: er is wat overhead verbonden aan het bishouden van de boekhouding van de schijf. Toch krijgt deze laatste methode de voorkeur, omwille van de serieuze inefficiëntie van de eerste.

De grootte van de blokken is een belangrijke parameter, die meestal een afweging is tussen datasnelheid en gebruiksgraad van de schijf. Indien de blokgrootte te groot genomen wordt, blijft er te veel vrije ruimte (gaps) over. Indien een kleine blokgrootte wordt genomen, is er een grote overhead voor het opzoeken en combineren van alle kleine blokken. Meestal wordt voor een compromis gekozen: een blokgrootte tussen 512 en 4 KB is courant.

13.4.2 Allocatie

Er bestaan 3 belangrijke manieren om de vrije ruimte op een schijf toe te kennen aan een bestand:

- Aaneensluitende allocatie: hierbij worden alle blokken van één bestand aaneensluitend op de schijf bewaard. Dit is natuurlijk de eenvoudigste methode, waarbij per bestand in een directory wordt bijgehouden op welk adres het begint en eindigt. Wanneer een nieuw bestand wordt opgeslagen kunnen algoritmes zoals best fit en first fit gebruikt worden. Er zijn echter problemen wanneer het bestand te groot wordt voor de voorziene ruimte: in dit geval moet het hele bestand verplaatst worden naar een andere locatie. Dit is een tijdsrovende actie.

- Linked list: bij een linked list worden alle blokken met verwijzingen naar het volgende blok op de schijf opgeslagen. In het eerste blok staat een verwijzing naar de positie van het tweede blok, daarin wordt verwezen naar het derde blok, ... tot het laatste blok bereikt is. Aangezien elk blok een pointer bevat is er een verspilling van schijfruimte, daarom worden de blokken meestal gegroepeerd in clusters. Zo'n cluster bestaat dan bijvoorbeeld uit 4 blokken. Deze clusters worden als geheel naar schijf geschreven, zodat de verspilling wordt gereduceerd. Een andere zwakte van dit systeem is dat indien een pointer verloren gaat, de rest van het bestand niet meer of moeilijk gevonden kan worden. Tenslotte moet nog gezegd worden dat random toegang tot een bestand niet mogelijk is met deze methode. Men moet namelijk het bestand sequentieel (lees: blok per blok) volgen om op een bepaalde plaats in het bestand terecht te komen.
- Geïndexeerde allocatie: deze methode biedt een oplossing voor het hiervoor vermelde probleem van zogenaamde random access in een bestand. Bij geïndexeerde allocatie wordt bijgehouden welke blokken bij een bepaald bestand horen. Zo kan zeer snel het *i*-de blok van een bestand ingelezen worden. Het nadeel is dat het index blok (blok waarin de indexen worden bijgehouden) meer plaats inneemt dan het systeem van gelinkte lijsten. Een ander nadeel is dat indien het index blok te klein is, men niet alle pointers van een bestand kan opslaan.

De laatste methode is duidelijk de meest interessante, ondanks de beschreven nadelen. Er werden oplossingen bedacht voor het probleem van een te klein index blok, waarvan de i-nodes in Unix een bekend voorbeeld zijn. Elke i-node bevat een lijst van pointers naar data blokken. Indien het gaat om een groot bestand wordt een i-node gebruikt als pointer naar een andere i-node die op haar beurt de pointers naar de data blokken bevat. Dit kan nog worden uitgebreid met een tweede en derde niveau, zodat er steeds voldoende i-nodes zijn om zeer grote bestanden te ondersteunen.

13.4.3 Beheer van vrije ruimte

Om ruimte aan bestanden toe te kennen, moet het besturingssysteem bijhouden welke blokken ongebruikt zijn. Hiervoor worden twee methodes gebruikt: de eerste gebruikt een bitmap waarin voor elk blok een bit wordt voorzien. Is de bit 1, dan betekent dit dat het blok reeds in gebruik is. Dit systeem kan enkel voor kleinere schijven gebruikt worden, omdat anders de bitmap niet in het geheugen past. Een andere methode maakt opnieuw gebruik van een gelinkte lijst. Hierbij wordt een vrij blok op de schijf gebruikt om de pointers naar andere vrije schrijfruimtes op te slaan. Het laatste element in zo'n blok verwijst dan naar een ander blok dat opnieuw een lijst met vrije blokken bevat.

13.4.4 Veiligheid

Een besturingssysteem moet in vele gevallen meerdere gebruikers ondersteunen (zogenaamde multi-user omgevingen). Er komt dan een belangrijk aspect bij, namelijk de beveiliging van bestanden. Niet iedereen mag immers een bestand lezen van een andere gebruiker. Veelgebruikte systemen zijn de Read, Write en Execute rechten die kunnen toegekend worden op een Unix systeem en de Access Control Lists die vanaf Windows 2000 werden ondersteund. Bij

zo'n ACL's is het mogelijk om op een bestand te specifiëren wat de rechten voor een bepaalde gebruiker zijn. Dit laat zeer gedetailleerde en geavanceerde rechtenspecificaties toe.