FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

INFORMATICA

BASISKENNIS

UNIVERSITEIT
GENT

Inhoudsopgave

Hoofdstuk 1

Inleiding

1.1 Algemeen

Informatica heeft te maken met informatie, met gegevens en met de verwerking ervan. Het Engels begrip voor gegevens is *data*, vandaar ook het begrip *data processing*, gegevensverwerking. Een EDP-manager is de beheerder van de afdeling *Electronic Data Processing*.

Een computer is dus een gegevensverwerkende machine. Deze te verwerken data kunnen variëren van signalen gegenereerd door auto's op de weg, tot lonen van arbeiders in een fabriek. Al deze gegevens worden volgens bepaalde formele regels vastgelegd, zodanig dat ze door computers kunnen verwerkt worden. Deze gegevens worden voor de mens pas informatie, als er een betekenis aan wordt toegekend volgens welbepaalde conventies.

Dit gegevensverwerkende apparaat is slechts een computer, als deze machine in staat is zelf, zonder menselijke tussenkomst, bepaalde taken uit te voeren. In tegenstelling met een eenvoudig rekentoestel, beschikt de computer hiertoe over een programma dat in zijn geheugen zit opgeslagen.

Het geheugen van een computer bevat dus niet alleen de data, maar ook het programma dat deze data verwerkt.

Een *programma* is een nauwkeurig gedefinieerde reeks van regels en voorschriften die bepalen hoe de machine een al dan niet lege verzameling invoergegevens (input data) zal verwerken tot een reeks uitvoergegevens (output data) op een eindige en eenduidige manier. Input-output wordt ook wel aangegeven door I/O.

Het is evident dat het de mens is, die deze programma's ontwerpt. Ze zijn een vertaling in computertaal van een of ander *algoritme*. Een algoritme is een meer algemene formulering van een verzameling regels voor het oplossen van een klasse van problemen in een eindig aantal stappen.

Alle programmatuur die hoort bij een informatiesysteem wordt omschreven als *software*. *Hardware* daarentegen is de verzameling van alle toestellen en elektronische en mechanische componenten. Beide elementen zijn onafscheidelijk met elkaar verbonden bij een bruikbare computer.

1.2 Soorten computertoepassingen

Er waren tot enkele decennia geleden twee grote categorieën computertoepassingen te onderscheiden: administratieve en technische.

De eerste categorie beslaat ruwweg geschat 80 à 90 percent van alle toepassingen. Daarin vinden we o.a. toepassingen van personeels-, loon-, en voorraadadministratie, van bankbeheer, registratie van autogegevens, bibliotheekbeheer, typen en verzenden van documenten, boekhouding, facturatie, enz....
Dit soort van applicaties wordt gekenmerkt door een massa te bewaren gegevens, waarbij evenwel nooit veel en ingewikkelde berekeningen moeten gemaakt worden.

Bij de tweede categorie is dit net omgekeerd.
In een vroeger stadium van de informatica vond men er uitsluitend technisch-wetenschappelijke berekeningen voor onderzoek op allerlei gebied, voor het ontwerpen van auto's, schepen, vliegtuigen, woningen, e.d. Meer recent zijn hier nog toepassingen bijgekomen als sturing van verkeerslichten, telefooncentrales, bank- en pompautomaten, sturing van processen (chemie-, staalnijverheid), robots, communicatie tussen computers.

Door de opkomst van de datacommunicatie vindt er een integratie plaats van beide soorten van toepassingen. Zo zal bijvoorbeeld de sturingscomputer van een hoogoven gekoppeld worden aan een centrale databank, om de opvolging mogelijk te maken van de bekomen kwaliteiten van het staal, en om aldus de hoogovenparameters te kunnen aanpassen. Ook zal de hoeveelheid geproduceerd staal een gevolg zijn van berekeningen op management niveau, die rekening houden met allerlei administratieve gegevens van de onderneming.

Een ander belangrijk voorbeeld van integratie is te vinden in de CAD/CAM branche. De term staat voor *Computer Aided Design/Computer Aided Manufacturing*. Soms spreekt men ook van CAE; de E staat voor *engineering*. Bij het ontwerp en de fabricatie met behulp van de computer is de gebruiker in staat om interactief op een grafisch scherm allerhande constructies te ontwerpen: vliegtuigen, auto's, bruggen, huizen,...
Maten en specificaties kunnen aangepast worden als het nodig is. Eens het ontwerp gemaakt, worden de werktekeningen automatisch getekend via plotters, worden de gegevens gegenereerd

voor de sturing van draaibanken, frezen, e.d., worden stuklijsten geproduceerd, voorraadberekeningen gemaakt om zo kostprijs, productiequota en winstmarges vast te leggen.

Sinds de opkomst van de PC, de tablet, de smart phone zijn er nog een heleboel andersoortige toepassingen bijgekomen. We vermelden onder andere computerspellen, het bekijken van video?s en beluisteren van muziek. De interconnectie van al deze apparaten via computernetwerken vereist bovendien een grote hoeveelheid software die niet in de twee bovenstaande categorieën is onder te brengen.

1.3 Digitale en analoge computers

Tot enkele decennia geleden bestonden er zowel digitale als analoge computers. Het onderscheid tussen beide heeft te maken met de wijze waarop de gegevens voorgesteld en verwerkt worden.

Een *digitale* computer werkt met discrete waarden, met cijfers en getallen. In digitaal vindt men het woord *digit* terug, wat cijfer betekent.
Alle informatie in een digitale computer wordt dus inwendig onder de vorm van cijfers voorgesteld. Het rekenen gebeurt echter met een beperkt aantal cijfers. Er ontstaan fouten door afronding. De uitkomsten van berekeningen op een digitale computer zijn evenwel volledig reproduceerbaar.
Een ander voorbeeld van een digitaal toestel is de kilometerteller van een auto: er is steeds onzekerheid over de aanduiding van het minst beduidende cijfer. De fout is evenwel steeds precies gekend.

Een *analoge* computer werkt met continue grootheden, meestal elektrische spanningen en/of stromen. De werking berust op het principe van meten. Daar een analoge meter (bijvoorbeeld een snelheidsmeter van een auto, een thermometer) steeds alle mogelijke waarden kan aanduiden binnen een bepaald interval, kan men stellen dat een analoge computer in theorie nauwkeuriger is dan een digitale.
Daar de grootheden evenwel fysisch worden voorgesteld door spanningen of stromen, kan het niet anders dan dat er fouten optreden. Hier is slechts hun grootte-orde bekend. Analoge berekeningen zijn maar binnen bepaalde grenzen reproduceerbaar.
In analoge computers zorgen elektronische schakelingen voor optellen, aftrekken, vermenigvuldigen, delen, integraties en differentiaties van spanningen en stromen. Het zijn vooral deze twee laatste berekeningen die de analoge machine interessant maakten. Bij de regeling van bepaalde processen moet men snel kunnen integreren en differentiëren. Dit kon vroeger met een digitale machine soms veel te lang duren. Het belang van analoge computers is door de evolutie in de digitale elektronica volledig verdwenen.

Analoge computers worden nagenoeg niet meer gebruikt, gezien de enorme evolutie van de digitale elektronica (prijs en snelheid). Voor processturing worden nu ook digitale computers

gebruikt; er zijn dan uiteraard analoog-digitaal omvormers nodig voor de invoer en digitaal-analoog convertors voor de uitvoer.

Het vervolg van deze informatica cursus heeft uitsluitend betrekking op digitale computers.

1.4 Bits en bytes - woordlengte

Zoals hiervoor werd besproken, worden alle gegevens in een digitale computer opgeslagen onder de vorm van getallen en cijfers. Gezien het feit dat de computer een concreet fysisch ding is, betekent dit, dat er een fysisch element moet bestaan dat de waarde van een cijfer kan voorstellen.

Als het zou gaan om decimale getallen, dan zou dit fysisch element voor elke mogelijke cijferwaarde een corresponderend "niveau" moeten kunnen aannemen, dat van alle andere niveaus kan onderscheiden worden.
Op de fysische voorstelling van elke cijferpositie in de computer, zouden er dus 10 verschillende niveaus moeten kunnen gezet en gedetecteerd worden. Zo'n oplossing is tot op heden nog niet praktisch verwezenlijkt.

In plaats van getallen in decimale vorm voor te stellen, slaat men ze op onder *binaire* of tweetallige gedaante. Hoe decimale getallen binair kunnen worden voorgesteld, wordt besproken in een van de volgende hoofdstukken.

Een binair cijfer (**BI**nary digi**T** = **BIT**) kan slechts 2 mogelijke waarden aannemen: 0 of 1. Mogelijke fysische voorstellingen van een bit liggen voor het grijpen: een deur (open of toe), een schakelaar (aan of uit), een elektronisch circuit (stroom of geen stroom, spanning of geen spanning).
Men kan zich het geheugen van een computer voorstellen als een reeks schakelaars die elk de waarde 0 of 1 voorstellen.
Zoals reeds hiervoor aangegeven, worden niet alleen de te verwerken gegevens op deze manier opgeslagen, maar ook de programma's die de opdrachten bevatten om deze gegevens te verwerken.

Een bit is de kleinste eenheid van informatie. Bits worden meestal gegroepeerd: een groep van 8 bits heet een *byte* (by eight).
De hoeveelheid informatie die in een computergeheugen kan opgeslagen worden, is meestal uitgedrukt in kilobyte (kbyte of kB), in megabyte (Mbyte of MB), in gigabyte (Gbyte of GB) of terabyte (TB).
Een kbyte is 1024 byte ($1024 = 2^{10}$), een Mbyte is 1024 kB of 2^{20} byte, een GB is 1024 MB, een TB 1024 GB.

Een begrip dat hier nauw mee samenhangt is de *woordlengte* waarmee een computer werkt: het is het aantal bits dat door de machine (processor) in één stap uit het geheugen kan gelezen en

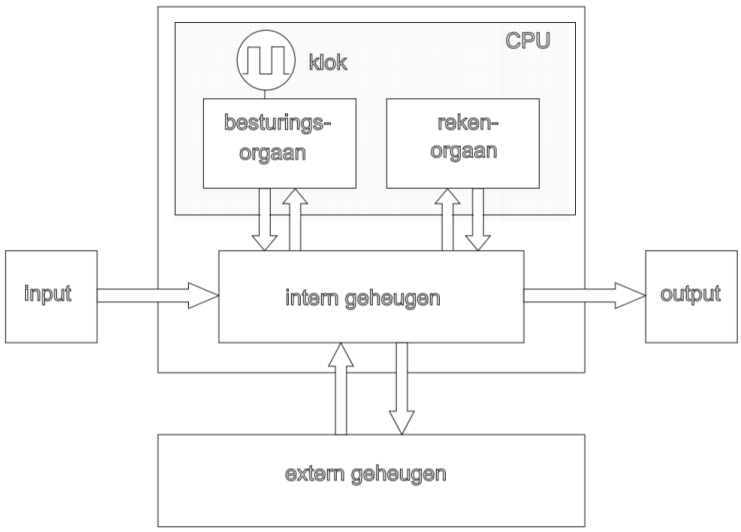
kan verwerkt worden. De woordlengte beslaat een veelvoud van 8 bits en wordt dan eventueel uitgedrukt in byte. Courante woordlengten zijn 1,2, 4 of 8 byte, ofwel 8,16, 32 of 64 bits. Vandaar komt een benaming als bijv. een 32-bit computer. Tegenwoordig beschikken hoe langer hoe meer computers over een woordlengte van 8 byte of 64 bits.

Hoofdstuk 2

Computercomponenten

2.1 Schema van een computer

Onderstaand schema geeft een overzicht van de logische componenten van een computer.



De pijlen duiden op een mogelijk gegevenstransport: we noemen ze *transportlijnen*.
Naast transportlijnen zijn er ook besturingslijnen. Voor de overzichtelijkheid van de figuur werden deze niet getekend. Ze verbinden het besturingsorgaan met alle componenten en vervoeren controlesignalen.

Alle transportlijnen zijn verbonden met het *intern geheugen*. Dit vervult een sleutelrol in alle gegevenstransport binnen de centrale machine: het is een buffer tussen alle systeemcomponenten.

Het *extern geheugen* is een ander type van geheugen, met een iets andere functie, zoals ook uit de figuur moge blijken.
De verschillende functies van beide geheugentypes worden in een volgende paragraaf verder besproken.

Het *input*-blok stelt de invoerapparatuur voor waarlangs de gegevens en de programma's worden omgezet in elektrisch digitale signalen; ze activeren bepaalde delen van het interne geheugen.
De aangevoerde informatie wordt via elektronische en softwarematige omzetting binaire vorm opgeslagen in het geheugen.

Het *output*-blok stelt de uitvoerapparatuur voor; ze geeft de resultaten van de verwerking weer in een voor de mens waarneembare vorm. De (elektrisch) binair gecodeerde gegevens worden op die manier omgezet tot zinvolle informatie.

De centrale verwerkingseenheid (CVE) of *Central Processing Unit* (CPU) bestaat uit 2 belangrijke componenten: het besturingsorgaan en het rekenorgaan.

In het *rekenorgaan* of ALU (*Arithmetic and Logical Unit*) gebeuren alle numerieke en logische bewerkingen; alle gegevens en resultaten zitten in het geheugen opgeborgen.
Logische bewerkingen laten het besturingsorgaan toe om beslissingen te nemen; ze bestaan veelal uit het controleren van een voorwaarde, waarvan de uitkomst VOLDAAN (true) of NIET-VOLDAAN (false) is.

Het *besturingsorgaan* haalt de uit te voeren opdrachten uit het intern geheugen. Alle opdrachten worden stuk voor stuk gedecodeerd en geïnterpreteerd; de benodigde systeemcomponenten worden telkens geactiveerd.
Om het besturingsorgaan autonoom te laten werken, moeten er dus in het interne geheugen twee soorten gegevens aanwezig zijn: data en programmaopdrachten.

Alle activiteiten van het besturingsorgaan gebeuren op het ritme van een klok. Dit is een elektronisch circuit dat pulsen genereert. Op de frequentie hiervan wordt de werking van alle systeemcomponenten gesynchroniseerd.

2.2 Geheugens

2.2.1 Kenmerken

We weten reeds dat het geheugen fungeert als opslagplaats, zowel voor programmaopdrachten als voor de gegevens die de programma's verwerken. Er is onderscheid gemaakt tussen intern en extern geheugen.
Deze opsplitsing naar geheugenhiërarchie gebeurt om verschillende redenen, die we hier willen bespreken zonder in te gaan op de gebruikte technologieën.

Het intern geheugen doet dienst als buffer voor alle gegevenstransport doorheen het computersysteem. In- en uitvoergegevens worden opgeslagen in het intern geheugen, samen met de programmaopdrachten en de te verwerken gegevens.
Deze laatste twee *moeten* in het interne geheugen zitten, anders kunnen ze niet worden gebruikt door besturingsorgaan en rekenorgaan.

Een van de kenmerken van geheugens is de *toegangstijd*: dit is de tijd nodig om een gegeven uit te lezen of weg te schrijven.

De *omvang* van het intern geheugen verschilt van computer tot computer: het kan variëren van meerdere tientallen MB tot enkele GB. Evenwel is het duidelijk dat gegevens niet onbeperkt kunnen opgeslagen blijven in het intern geheugen, hoe groot men de capaciteit ook maakt. Gezien de hoge toegangssnelheid (grootteorde ns) is intern geheugen ook relatief duur.

Hierbij komt dat de technologie maakt dat intern geheugen *volatiel* (vluchtig) is. Dit betekent dat als de spanning wegvalt of als de machine wordt uitgeschakeld, alle gegevens uit het intern geheugen "verdwijnen". Dat betekent dat de vroegere stand van de elektronische schakelaars (aan of uit) bij het opnieuw opstarten van de computer niet meer kan achterhaald worden; de schakelaars komen op een willekeurige manier in de aan- of de uit-stand te staan en stellen dus at-random een 0 of een 1 voor.

Het is evenwel ook niet nodig dat alle betreffende gegevens in het intern geheugen blijven staan. Immers, alleen de gegevens waarmee de CPU direct werkt, moeten zich in het inwendig geheugen bevinden.

Een illustratie uit het dagelijks studentenleven kan dit verduidelijken. Gegevens waar men mee werkt, heeft men direct bij de hand, bijvoorbeeld op zijn bureau. Een boek dat men niet direct nodig heeft, maar toch regelmatig raadpleegt, staat in een kast, op niet te grote afstand en gemakkelijk bereikbaar. Een cursus van enkele jaren voordien, die men slechts zeer uitzonderlijk raadpleegt, bevindt zich op zolder in een archiefkast.

Dergelijke situatie treft men ook aan bij computers. Het intern geheugen is vergelijkbaar met het bureau: met de gegevens die zich daar bevinden, kan de CPU direct werken. Het boek in de kast is het extern geheugen: de gegevens die daar zijn opgeslagen, moeten eerst naar het intern geheugen getransfereerd worden, wil de CPU ze kunnen gebruiken. De gegevens in het archief bevinden zich, los van de computer, op een of andere gegevensdrager. Om met deze gegevens te kunnen werken, moet deze gegevensdrager geplaatst worden in de externe geheugeneenheid, vanwaar de gegevens naar het intern geheugen overgebracht worden. Voorbeelden van zo'n gegevensdrager zijn een usb-stick, een externe disk of eventueel een magneetband, die op een of andere manier met de computer moeten gekoppeld worden of, voor wat de magneetband betreft, in de magneetbandeenheid (bandrecorder) moet aangebracht worden.

Hierna volgt een vergelijkend overzicht van de verschillende kenmerken; de getalwaarden zijn slechts indicatief en gelden voor een typische PC. Vooral de prijzen van de geheugens zijn sterk marktafhankelijk en kunnen tamelijk sterk schommelen. De tendens is wel dat jaar na jaar de

omvang toeneemt en de prijs daalt.

	intern	extern
omvang	4 GB 16 GB	500 GB 1 à 2 TB
prijs	6 à 12 euro / GB	0,05 à 0,1 euro / GB
toegangstijd	< 10 ns	< 10 ms
bewaartijd	volatiel	permanent
medium	IC's	Disk, IC's

2.2.2 Interne geheugens

2.2.2.1 Adresseerbaarheid

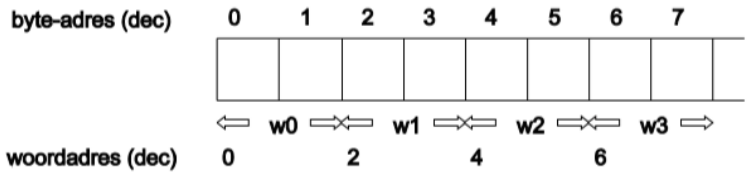
Naast de reeds aangehaalde kenmerken is de belangrijkste eigenschap van intern geheugen, dat elke locatie direct toegankelijk moet zijn. Dit gebeurt op de volgende manier.

Zo'n geheugen is best vergelijkbaar met postbussen in een postkantoor: zoals elke postbus een nummer krijgt toegewezen, zo heeft elke geheugenplaats een adres gekregen.
Elke geheugenlocatie bestaat uit 8 bits of een byte.
De nummering begint steeds bij nul.
De maximale hoeveelheid adresseerbare geheugenplaatsen is afhankelijk van het aantal bits dat voor de voorstelling van de adressen gebruikt wordt: met 16 adreslijnen bv., kan 64 kB geadresseerd worden ($2^{16} = 65536$), met 20 lijnen 1 MB ($2^{20} = 1048576$).

Vooraleer er in een bepaalde byte informatie kan geschreven worden, of vóór de inhoud ervan kan gelezen worden, moet eerst het adres van de desbetreffende byte aan het geheugen aangelegd te worden via de adreslijnen. Dit adres wordt gedecodeerd en de decoder activeert de juiste bytepositie. Pas dan kan door respectievelijk een schrijfpuls de inhoud van de databits in het geheugen op de juiste plaats aangebracht worden, of door een leespuls de geheugeninhoud van de locatie op de datalijnen geplaatst worden.

Niettegenstaande elk byte een adres krijgt, zullen de meeste computers meerdere bytes in één geheugentoegang kunnen verwerken. Men spreekt dan van een woord (zie Hoofdstuk 1). Elk woord heeft ook een adres: dit is steeds het adres van de eerste byte van het woord.
Is de woordlengte van de computer 4, 2 of 1 byte, dan is het woordadres deelbaar door 4, 2 of 1.

Voorbeeld: met een woordlengte van 16 bits (= 2 byte).



2.2.2.2 Halfgeleidergeheugen

RAM

Met de opkomst van de halfgeleidertechnologie is ferrietkerngeheugen stilaan verdrongen door goedkopere en betrouwbaardere geheugen-IC's. Op een oppervlak van ongeveer 1 cm^2 kan men tegenwoordig ongeveer 100 Gbit onderbrengen. De opslagplaatsen voor de bits, die we tot nu toe als schakelaars hebben voorgesteld, worden nu verwezenlijkt door transistoren in statische RAM's, en door condensatoren in dynamische RAM's.
RAM staat voor *Random Access Memory*: de geheugenbytes staan ook in matrix opgesteld in het IC, en zijn zo op een willekeurige plaats bereikbaar.
Bij *dynamisch*e RAM moet de lading van de condensatoren herhaaldelijk opgefrist worden, omdat die slechts gedurende korte tijd hun lading kunnen behouden.
Statische RAM is daardoor sneller toegankelijk dan dynamische RAM, maar is ook duurder. Deze technologie wordt o.a. gebruikt bij *cache* geheugens. Dit soort geheugen is relatief beperkt in omvang. De bedoeling is dat het de gegevens bevat die zeer frequent moeten gebruikt worden.

ROM

In sommige computertoepassingen heeft men behoefte aan een geheugen waar niet moet in geschreven worden, maar waarvan de inhoud uitsluitend moet kunnen gelezen worden. Dit kan natuurlijk alleen voor vaste programma's en gegevens (bv. BIOS, opstartcode, ...). Zo'n *Read Only Memory* wordt ook wel ROM genoemd.
Er bestaat ook **PROM** (*programmeerbare* ROM) en **EPROM** (*erasable* PROM). Bij PROM ligt de inhoud niet vast bij fabricatie, doch deze kan door de gebruiker zelf ingebracht worden. Bij een EPROM kan die inhoud dan ook nog gewist worden door UVlicht, waarna er opnieuw gegevens kunnen worden in aangebracht.
Bij **EEPROM** (*electrical* EPROM) kan het wissen en opnieuw aanbrengen van de gegevens gebeuren door elektrische signalen. Deze technologie wordt ook gebruikt in alle vormen van flash memory (zie hieronder).
Een ROM-geheugen is per byte adresseerbaar en komt dus strikt genomen in aanmerking om als intern geheugen te kunnen functioneren. Omdat het alleen maar leesbaar is, kan het evenwel

slechts dienen voor de opslag van vaste gegevens (zoals bv. het programma). In deze functie wordt het onder meer gebruikt bij microcontrollers en randapparatuur. In een "klassieke" computer is dit niet het geval, vooral omdat deze geheugenvorm daar veel te traag is.

FLASH MEMORY

Dit soort van geheugen gebruikt gelijkaardige technologie als die van EEPROM. Er bestaan twee vormen van flash-geheugen: NOR-flash en NAND-flash.
NOR-flash is byte-adresseerbaar maar kan evenwel niet per byte gewist worden. Het kan dus als geheugen gebruikt worden voor vaste gegevens (alleen lezen) die sporadisch gewijzigd kunnen worden door een aparte toepassing. We vinden het terug in bv. de BIOS of de firmware van een computer die onder meer dient om de machine op te starten.
NAND-flash is blok-adresseerbaar (een blok is dan bv. 512 bytes) en kan zich dan gedragen als een disk. Het kan niet fungeren als intern geheugen, maar is wel bruikbaar als extern geheugen. Het enige nadeel is dat de geheugenplaatsen geen oneindig aantal keer opnieuw kunnen beschreven worden, maar voor de huidige toepassingen is dat niet echt een probleem. NAND-flash bestaat in o.a. volgende vormen:

- USB-stick (USB = Universal Serial Bus), gebruikt als bv. back-upmedium;
- SD-card (secure digital memory card); het is een klein "kaartje" dat bestaat in verschillende vormen en dat gebruikt wordt o.a. als extra opslag voor fototoestellen, MP3-spelers, ...
- SSD (solid state disk), gebruikt als snellere (opstart)disk (zie verder).

2.2.3 Externe geheugens

Externe geheugens zijn bijna altijd schijfgeheugens.
Fungeert het geheugen echter alleen maar als hulpgeheugen voor de permanente opslag van data, dan is ook magneetband bruikbaar.
Als hulpgeheugen is ook optische technologie inzetbaar. Dit soort van geheugens is veel goedkoper, doch vereist altijd nog een toestel (drive) waarin de gegevensdrager kan geplaatst worden en dat de gegevens kan aanbrengen of lezen.
De toegangstijd voor schijfgeheugens ligt in de grootteorde van ms.

Informatie kan slechts in het extern geheugen geplaatst worden of er weer uitgelezen worden, op aanvraag van de CPU van de computer. De tussenopslag gebeurt steeds in het intern geheugen.

Er wordt onderscheid gemaakt tussen adresseerbaar en niet-adresseerbaar hulpgeheugen. Magneetschijf is een voorbeeld van de eerste soort: de informatie is naar analogie met het interne geheugen direct bereikbaar via een adres (nummer van spoor en sector). Magneetband behoort tot de tweede soort: de informatie is op de band slechts bereikbaar als alle ervoor staande informatie ook gelezen en bekeken is geweest; de informatie heeft geen adres.

2.2.3.1 Schijfgeheugen

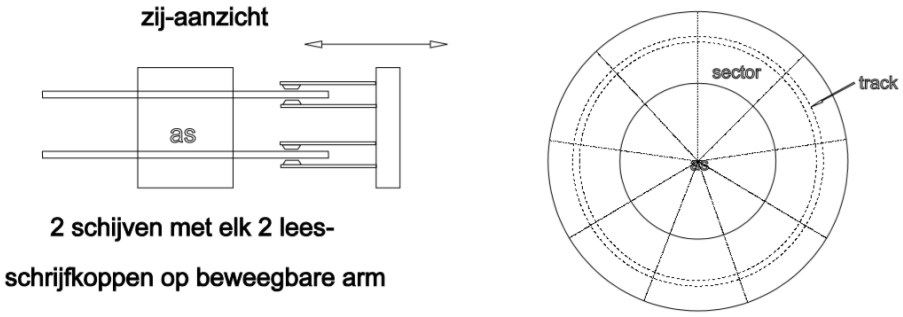
HARDDISK

Een *harddisk* of vaste of harde schijf is een magneetschijf die bestaat uit een dunne ronde aluminium of keramische plaat, met aan beide oppervlakken een laagje magnetiseerbaar materiaal: dit is een of ander metaaloxide, zoals het ook op audio- en videobanden te vinden is. De schijf bevindt zich op een verticale as die draait met een hoeksnelheid van enkele duizenden omwentelingen per minuut.

Courante kenmerken zijn:

- diameter: 3½ en 2½ inch;
- omwentelingssnelheid: 5400, 7200 en 10000 omwentelingen per minuut;
- opslagcapaciteit: 500GB tot 1 à 2 TB.

Meestal bevinden zich meerdere schijven boven elkaar op dezelfde as. Tussen de schijven is voldoende ruimte gelaten voor één lees/schrijfkop per oppervlak.
De koppen zijn zo gebouwd, dat ze door het luchtkussen dat ontstaat boven de draaiende schijf, op enkele micron boven de schijf worden gehouden. Stofdeeltjes verminderen de luchtdruk op de kop, waardoor het gevaar ontstaat dat de kop op de schijf terecht komt: men spreekt van een diskcrash. Door het rondvliegend magnetisch materiaal is dan meestal alle informatie op het schijvenpakket verloren. Gelukkig is de technologie zo ver gevorderd dat dit defect zich haast niet meer voordoet.



Op de boven- en onderkant van de schijf kunnen de bits als magnetische plekjes op concentrische sporen worden vastgelegd. Hiertoe dient de arm met de lees/schrijfkoppen zich volgens de straal van de schijven te bewegen. Een nauwkeurig en stabiel geleidingsmechanisme is hier vereist.
Afhankelijk van de gebruikte technologie is het mogelijk om per oppervlak enkele honderden sporen te beschrijven. Alle sporen op dezelfde afstand van de as vormen een cilinder. Elk spoor of track wordt verder in segmenten of sectoren verdeeld (aantal ongeveer 10).

Elke sector en elk spoor hebben een bepaald adres. Beide coördinaten bepalen de plaats van een *disk-cluster*. Deze bevat een hoeveelheid bytes die in één keer in het intern geheugen worden geladen voor verdere verwerking. Courante clustergrootten zijn 4 kB tot 32 kB.

De toegangstijd van een disk is onderverdeeld in de volgende onderdelen:

- de tijd nodig voor het positioneren van de lees/schrijfkop boven het gewenste spoor;
- de eventuele tijd nodig voor het selecteren van de juiste lees/schrijfkop;
- de wachttijd tot de gewenste informatie zich onder de kop bevindt;
- de tijd nodig om de informatie te lezen.

De eerste drie worden aangeduid met *accesstime*, de laatste wordt de *transferrate* genoemd.

Een vrij volledig overzicht met afbeeldingen en cijfermateriaal is te vinden op
`http://en.wikipedia.org/wiki/Hard_disk_drive`

SOLID STATE DISK

Een solid state drive (SSD) wordt ook vaak een solid state disk genoemd, maar bevat hoegenaamd geen schijf en ook geen bewegende delen. Het is een elektronisch toestel dat met behulp van geïntegreerde schakelingen in staat is om informatie op te slaan en te lezen. De elektronische interface zorgt er voor dat het device compatibel is met de traditionele blokgeoriënteerde I/O zoals die van harde schijven zodat het zich voor de computer dus gedraagt als een harddisk.

Er bestaan twee soorten technologieën.
De eerste maakt gebruik van dynamische RAM. Op deze manier kan een zeer snelle disk nagemaakt worden, alleen is die wel behoorlijk duur en is er een batterij nodig om de geheugencellen onder spanning te houden, zo niet "verdwijnt" de informatie.
De tweede en meest voorkomende technologie is die van flash geheugen. Die is een heel stuk goedkoper dan bij het gebruik van RAM-componenten, maar is vergeleken met de prijs van een gewone harddisk nog tamelijk duur ($\pm$ 1 Euro/GB tegenover 0,1 Euro/GB). Het voordeel is wel dat de toegangstijd van de orde van 100 μs is, wat wel 100 keer groter is dan die van RAM, maar ongeveer 10000 keer sneller is dan die van de modale harddisk. Een SSD wordt dan ook meestal gebruikt om er het besturingssysteem op te installeren, waardoor de computer een stuk sneller opstart dan bij het gebruik van een gewone harddisk.

2.2.3.2 Magneetbandgeheugen

Magneetband is een plastic band die bedekt is met een laagje magnetiseerbaar materiaal, met daaroverheen een slijtlaag. Hij is vergelijkbaar met de banden die bijvoorbeeld bij videotoepassingen worden gebruikt. Een magneetbandgeheugen kan slechts als dusdanig functioneren bij

aanwezigheid van een tapedrive, een soort van bandrecorder maar dan voor digitale gegevensopslag. Met behulp van de lees/schrijfkop kan er informatie op de band worden aangebracht en later weer worden gelezen. De banden zijn uiteraard verwisselbaar op de drive en dienen vooral voor back-up en archivering.

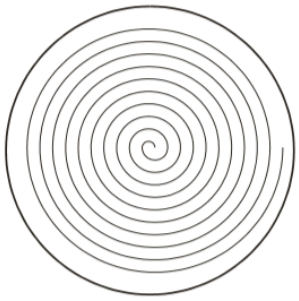
Vermits de geheugenplaatsen op een band geen fysische adressen hebben, zijn de informatieblokken ook niet adresseerbaar. Een band is slechts sequentieel toegankelijk: de informatie moet van de band gelezen worden in dezelfde volgorde als deze waarin ze erop geschreven werd.

De meeste soorten banden zitten in cassettes. Er zijn veel gebruikte types met verschillende standaarden. We vermelden hier de DAT (digital audio tape) die voldoet aan de DDS-standaard. De cassette heeft een afmeting van $\pm$ 5x3 cm, breedte van de band bedraagt 3.8 mm, de lengte is 170 m. Door de enorme dichtheid bedraagt de opslagcapaciteit 72 GB.

2.2.3.3 Optische geheugens

Er bestaan verschillende soorten optische geheugenvormen: de twee meest gangbare zijn cd-rom en dvd-rom. De cd (compact disc) kent zijn oorsprong in de audio-wereld. Het is pas later dat men hem is gaan gebruiken als read-only memory.

De gegevens op een cd staan op één enkel spiraalvormig spoor dat van het centrum naar de buitenkant van de schijf loopt en ongeveer zes kilometer lang is. Op dit spoor bevinden zich putjes (*pits*) die minder licht reflecteren dan het omliggende vlakke oppervlak (*lands*, basis). Een laserstraal tast het oppervlak af en wordt weerkaatst naar een sensor die de opeenvolging van putjes waarneemt. (Eigenlijk wordt het oppervlak afgetast langs de andere kant dan daar waar de putjes ingedrukt werden; de laser ziet dus eigenlijk de bulten.) Bijkomende sensoren sturen de straal bij zodat de laserstraal het spoor blijft volgen. De snelheid van de schijf wordt zodanig geregeld, dat het spoor met een constante lineaire snelheid (1.2 à 1.4 m/s) wordt afgetast.



De pits en de lands stellen zelf niet direct de binaire enen en nullen voor. Dat komt omdat er geen absolute detectie mogelijk is van het soort van reflectie. Men kan wel de overgangen detecteren. Dat betekent dat er dus een zekere code moet gebruikt worden die afspreekt hoe

een binaire 0 of 1 op de schijf wordt voorgesteld. Een overgang van pit naar land of omgekeerd betekent een 1, terwijl geen overgang een nul voorstelt. Tussen beide moeten er zich minstens twee en niet meer dan tien nullen bevinden. Dit kan gedetecteerd worden door rekening te houden met de grootte van de pits en afspeelsnelheid.
Daarnaast wordt nog een extra codering toegepast die het mogelijk moet maken om leesfouten te detecteren en ze ook te corrigeren. Een cd is inherent een zeer onbetrouwbaar medium: bij het lezen van één cd kunnen gemakkelijk een miljoen fouten optreden. Vandaar dat het grootste deel van de gegevens (2/3 op een audio-cd; 3/4 op een cd-rom) controlebits zijn: deze laten toe leesfouten op te sporen, en de meeste ook te herstellen.
Meer informatie is te vinden op
`http://en.wikipedia.org/wiki/Compact_Disc#Physical_details`
en
`http://electronics.howstuffworks.com/cd.htm`

Op een **cd-rom** staan bestanden en moet er dus een mogelijkheid zijn om de gegevens te adresseren en logisch te ordenen; hiervoor zijn uiteraard nog extra controlebits nodig. Uiteindelijk blijven er nog zowat 650MB (of 700MB) aan nuttige data over.

Naast de klassieke cd-rom bestaan er nog andere formaten, waaronder de **dvd** (*digital versatile disk*), die meer gegevens kunnen bevatten. Doordat er gebruik wordt gemaakt van een laser met een hogere frequentie, kunnen de putjes kleiner en kan het spiraalspoor compacter zijn. Op een dvd kunnen ook beide zijden gebruikt worden en tot twee lagen per zijde. Dit resulteert dan in een capaciteit van 15,9 GB.

Normale cd's worden in grote aantallen geperst uitgaande van een moeder-cd. Moeten gegevens slechts op één of op enkele cd's gekopieerd worden, dan gebruikt men andere technieken.
Bij de **cd-r** (*recordable*) worden door een laserstraal (met een intensiteit die veel groter is dan deze van een leeslaser) donkere plekken gebrand in een laklaag. Deze plekken kunnen niet terug gewist worden: elke cd-r kan slechts één enkele keer beschreven worden.
Bij de **cd-rw** (*read/write*) is deze laag vervangen door een metaallegering. Deze wordt door de laser verhit (500 à 700 °C) en smelt. Afhankelijk van de temperatuur stolt de laag traag of snel. Bij trage stolling kristalliseert de laag op die plaats en zorgt dan voor een hoge reflectiegraad; bij snelle stolling blijft het materiaal amorf met als gevolg een lage reflectiegraad. Op deze manier wordt hetzelfde effect bekomen als dat van de pits en lands op een gewone cd. Deze metaallegering kan door gepaste verhitting meerdere keren van toestand veranderen, waardoor gegevens weer kunnen gewist en overschreven worden.

Hoofdstuk 3

Talstelsels

3.1 Overzicht

3.1.1 Het decimaal of tientallig stelsel

Dit is het talstelsel waar wij gewoonlijk in rekenen. Puur mathematisch bekeken zouden we echter evengoed een ander talstelsel kunnen gebruiken.

In het decimaal stelsel worden er 10 verschillende symbolen gebruikt om de cijfers voor te stellen: 0,1,2,3,4,5,6,7,8,9.
De *basis*, *grondtal* of *radix* is immers 10. In getallen met één cijfer kan men bijgevolg 10 verschillende waarden noteren.

Om grotere waarden te kunnen optekenen gebruikmakend van dezelfde 10 symbolen, laat men de positie van het cijfer in het getal bepalen, welk gewicht het cijfer krijgt toegewezen.
De waarde van het cijfer 5 in de getallen 571 en 325 is duidelijk verschillend. Men noemt dit een *positioneel stelsel*, waar elke plaats een *orde* krijgt toegewezen. De waarde die het cijfer voorstelt, is dan $cijfer \times 10^{orde}$.

Er moet opgemerkt worden dat we in het vervolg de decimale komma zullen vervangen door een punt. Het is die notatie die in de Engelse taal gehanteerd wordt en die direct aansluit bij het gebruik in informatica.

Voorbeeld

Het getal 173.84 kan ontleed worden als volgt:

cijfer 1 is van orde 2,	waarde 1x10^2	= 100
cijfer 7 is van orde 1,	waarde 7x10^1	= 70
cijfer 3 is van orde 0,	waarde 3x10^0	= 3
cijfer 8 is van orde -1,	waarde 8x10^{-1}	= 0.8
cijfer 4 is van orde -2,	waarde 4x10^{-2}	= 0.04

ofwel $173.84 = 1 \times 10^2 + 7 \times 10^1 + 3 \times 10^0 + 8 \times 10^{-1} + 4 \times 10^{-2}$

Algemeen kan gesteld worden dat het cijfer volgende waarde krijgt, afhankelijk van de plaats: $cijfer \times grondtal^{orde}$.

Alle getallen in een ander talstelsel dan het decimale, worden als volgt genoteerd:
(getal in basis $t)_t$

3.1.2 Het binair of tweetallig stelsel

Zoals reeds vroeger vermeld, beantwoordt dit stelsel aan de binaire structuur van de elektronische gegevensverwerking.
De radix is 2 en er zijn dus slechts twee mogelijke cijfersymbolen: 0 en 1.
Ook dit stelsel is een positioneel stelsel.
Merk op dat $(2)_{10} = (10)_2$.

Algemeen geldt ten andere dat een grondtal van een talstelsel in het stelsel zelf voorgestel wordt door 10:

$$(grondtal)_{10} = (10)_{grondtal}$$

Dit is immers steeds het kleinste getal dat niet meer met één maar met twee cijfers moet genoteerd worden.

Voorbeeld

$(1011.01)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$
De decimale uitdrukking in het rechterlid bepaalt meteen ook de decimale waarde van het binaire getal: 11.25

Hierbij stippen we nogmaals aan dat als getallen in andere talstelsels zijn geschreven dan in het decimale, dit expliciet moet vermeld worden: het binaire getal 1011.01 heeft niet dezelfde waarde als het decimale getal 1011.01.

Hoewel het tweetallig stelsel direct aansluit bij de basisopbouw van de computer, is het voor de mens niet erg praktisch om dit stelsel te gebruiken. Een kleiner grondtal resulteert automatisch

in meer cijferposities om een getal te noteren.
Talstelsels met een grondtal groter dan 2, waarbij er evenwel een relatie is met het binair stelsel, kunnen gebruikt worden om binaire getallen op een verkorte manier te noteren.
Stelsels met als basis gehele positieve machten van 2 voldoen aan deze vereiste.

Niettegenstaande er analoge dingen als hierboven kunnen vermeld worden over het 3-, 4-, 5-, 6-,...tallig stelsel, zullen we het in deze cursus nog slechts hebben over het 8- en 16-tallig stelsel.

3.1.3 Het octaal of achttallig stelsel

Dit stelsel heeft als grondtal 8 en gebruikt 8 cijfers: 0,1,2,3,4,5,6,7.
Analoog aan wat vooraf gaat, geldt bijvoorbeeld:

$$\begin{aligned}(1437.56)_8 &= 1 \times 8^3 + 4 \times 8^2 + 3 \times 8^1 + 7 \times 8^0 + 5 \times 8^{-1} + 6 \times 8^{-2} \\ &= 512 + 256 + 24 + 7 + 0.625 + 0.09375 \\ &= 799.71875\end{aligned}$$

3.1.4 Het hexadecimaal of zestientallig stelsel

De radix van dit talstelsel is 16.
Er zijn bijgevolg zestien cijfersymbolen nodig: er is geopteerd voor de 10 decimale cijfers, aangevuld met de eerste 6 letters van het alfabet. Onderstaande tabel geeft een overzicht van de hexadecimale cijfers en hun bijhorende decimale waarde terug:

hex cijfer	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
dec waarde	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Volgend voorbeeld geldt naar analogie met de hiervoor vermelde talstelsels:

$$\begin{aligned}(3D9.A6)_{16} &= 3 \times 16^2 + 13 \times 16^1 + 9 \times 16^0 + 10 \times 16^{-1} + 6 \times 16^{-2} \\ &= 768 + 208 + 9 + 0.625 + 0.0234375 \\ &= 985.6484375\end{aligned}$$

3.2 Conversies

3.2.1 Van B-tallig naar decimaal stelsel

Voor dit soort omzetting volstaat het om de intrinsieke definitie van het positioneel talstelsel met basis B toe te passen.

Voor gehele getallen bestaat er nog een praktische regel, die gemakkelijk met de zakrekenmachine kan uitgevoerd worden of kan geprogrammeerd worden. Startend aan de linkerkant nemen we het eerste cijfer, vermenigvuldigen met B, tellen het tweede cijfer op, vermenigvuldigen opnieuw met B, enz...
Voorbeeld: conversie van $(6175)_8$ naar decimaal:

$$(6175)_8 = ((6 \times 8 + 1) \times 8 + 7) \times 8 + 5 = 3197$$

3.2.2 Van decimaal naar B-tallig stelsel

Om willekeurige decimale getallen om te zetten naar een ander talstelsel, moet men afzonderlijk het gehele en gebroken gedeelte converteren.
Daarna worden beide omgezette delen bij elkaar gevoegd.

3.2.2.1 Gehele getallen

Het decimale gehele getal wordt gedeeld door het grondtal B, quotiënt en rest worden bepaald:

$$\text{decimaal getal} = \text{B} \times \text{quotiënt} + \text{rest}$$

De gevonden rest is steeds kleiner dan het grondtal B (=deler) en geeft een eerste cijfer van het gezochte getal in het stelsel met grondtal B.
Het quotiënt van deze eerste deling wordt opnieuw gedeeld; de nieuwe rest is een volgend cijfer, het quotiënt van deze deling wordt opnieuw gedeeld.
Dit proces gaat verder tot een quotiënt wordt gevonden dat nul is.

Voorbeeld 1: naar **binair**: $89 = (?)_2$

Er moet steeds gedeeld worden door 2.

$$\begin{aligned}
89 : 2 &\rightarrow q = 44, & r = 1 \\
44 : 2 &\rightarrow q = 22, & r = 0 \\
22 : 2 &\rightarrow q = 11, & r = 0 \\
11 : 2 &\rightarrow q = 5, & r = 1 \\
5 : 2 &\rightarrow q = 2, & r = 1 \\
2 : 2 &\rightarrow q = 1, & r = 0 \\
1 : 2 &\rightarrow q = 0, & r = 1
\end{aligned}$$

Er wordt gedeeld tot het quotiënt nul wordt.
Opgelet: de laatst gevonden rest mag niet vergeten worden.

Vermits de eerst gevonden resten de minst beduidende cijfers voorstellen, is het interessant resten en quotiënten van rechts naar links te noteren, zodat het gezochte getal direct verschijnt. Dit levert dan de volgende praktische notatie op:

quotiënt	0	1	2	5	11	22	44	89
rest	1	0	1	1	0	0	1	

Dit betekent dat $89 = (1011001)_2$.

Voorbeeld 2: naar **octaal**: $2379 = (?)_8$

Er moet steeds gedeeld worden door 8.

$$\begin{aligned} 2379 : 8 &\rightarrow q = 297, \quad r = 3 \\ 297 : 8 &\rightarrow q = 37, \quad r = 1 \\ 37 : 8 &\rightarrow q = 4, \quad r = 5 \\ 4 : 8 &\rightarrow q = 0, \quad r = 4 \end{aligned}$$

Praktisch:

quotiënt	0	4	37	297	2379
rest	4	5	1	3	

Dus: $2379 = (4513)_8$.

Voorbeeld 3: naar **hexadecimaal**: $32159 = (?)_{16}$

Er moet steeds gedeeld worden door 16.
Direct praktisch:

quotiënt	0	7	125	2009	32158
rest	7	13	9	14	

De resten moeten evenwel nog als hexadecimale cijfers worden genoteerd.
Dit geeft: $32158 = (7D9E)_{16}$.

3.2.2.2 Decimale fracties

Voor het omzetten van decimale fracties naar een stelsel met grondtal B wordt de decimale fractie vermenigvuldigd met grondtal B. Het geheel deel van het product is het meest beduidende cijfer van het gezochte getal, de fractie van dit product wordt opnieuw met B vermenigvuldigd, enz...
Dit gaat in principe verder tot de fractie van het nieuwe product nul wordt. Uit de voorbeelden zal duidelijk blijken dat dit zich slechts in zeer uitzonderlijke situaties na een eindig aantal stappen voordoet. Meer in het bijzonder geldt het voor decimale fracties die exact kunnen geschreven worden als een som van negatieve machten van het nieuwe grondtal.

Als algemene regel moet dus gesteld worden, dat decimale fracties niet exact in een ander talstelsel kunnen geschreven worden met een eindig aantal cijfers. Dit heeft verregaande gevolgen voor de voorstelling van decimale getallen in een computer: de gebruiker moet zich steeds bewust zijn van mogelijke afrondingsfouten.

Voorbeeld 1: naar **binair**: vermenigvuldigen met 2

1. $0.125 = (?)_2$

   $$\begin{array}{lll} 0.125 & \times 2 = & 0.25 \\ 0.25 & \times 2 = & 0.5 \\ 0.5 & \times 2 = & 1 \end{array}$$

   Dus: $0.125 = (0.001)_2$
   Hier stopt men bij een fractie = 0: $(0.125)_{10}$ kan immers exact geschreven worden als een som van negatieve machten van 2.

2. $0.653 = (?)_2$

   $$\begin{array}{lll} 0.653 & \times 2 = & 1.306 \\ 0.306 & \times 2 = & 0.612 \\ 0.612 & \times 2 = & 1.224 \\ 0.224 & \times 2 = & 0.448 \\ 0.448 & \times 2 = & 0.896 \\ 0.896 & \times 2 = & 1.792 \\ 0.792 & \times 2 = & 1.584 \\ 0.584 & \times 2 = & 1.168 \\ \ldots & & \end{array}$$

   Dus: $0.653 = (0.10100111\ldots)_2$

Voorbeeld 2: naar **octaal**: vermenigvuldigen met 8

$0.653 = (?)_8$

$$\begin{array}{lll} 0.653 & \times 8 = & 5.224 \\ 0.224 & \times 8 = & 1.792 \\ 0.792 & \times 8 = & 6.336 \\ 0.336 & \times 8 = & 2.688 \\ 0.688 & \times 8 = & 5.504 \\ \ldots & & \end{array}$$

Dus: $0.653 = (0.51625\ldots)_8$

Voorbeeld 3: naar **hexadecimaal**: vermenigvuldigen met 16

$0.653 = (?)_{16}$

$$\begin{array}{lll} 0.653 & \times 16 = & 10.448 \\ 0.448 & \times 16 = & 7.168 \\ 0.168 & \times 16 = & 2.688 \\ 0.688 & \times 16 = & 11.008 \\ 0.008 & \times 16 = & 0.128 \\ 0.128 & \times 16 = & 2.048 \\ \ldots & & \end{array}$$

Na omzetting van de gehele delen in hexadecimale cijfers geldt dus: $0.653 = (0.A72B02\ldots)_{16}$

3.2.3 Opmerkingen

In het tientallig stelsel komt de verplaatsing van de punt over één rang naar links of naar rechts overeen met respectievelijk een deling door 10 of een vermenigvuldiging met 10.
Analoog geldt in een willekeurig stelsel met grondtal B dat de verplaatsing van de punt naar links of naar rechts een deling door of een vermenigvuldiging met B betekent.
Voorbeeld binair:

$$\begin{array}{lcl} (1111)_2 & = & 15 \\ (111.1)_2 & = & 7.5 \\ (11.11)_2 & = & 3.75 \\ (11110)_2 & = & 30 \\ (111100)_2 & = & 60 \end{array}$$

Om echte binaire breuken naar decimale te converteren, kan men dus ook als volgt te werk gaan:

$$(0.001101)_2 = 1101 \times 2^{-6} = 13/64 = 0.203125$$

Men kan zich ook nog afvragen hoeveel bits er nodig zijn om het grootste decimaal geheel getal met n cijfers in het tweetallig stelsel weer te geven.
Is a het gezochte aantal bits, dan geldt: $10^n = 2^a$.
Dus geldt: $n = a log 2 = 0.30103a$ ofwel $a = 3.3219281n$

3.2.4 Onderling tussen het 2-, 8- en 16-tallig stelsel

Door het exponentieel verband tussen de drie grondtallen kan zeer eenvoudig van het ene naar het andere stelsel worden geconverteerd.
Immers: $8 = 2^3$, $16 = 2^4$.

3.2.4.1 Binair naar octaal

Voor de omzetting van binair naar octaal worden in het binair getal de bits gegroepeerd in groepjes van drie bits, beginnend bij de punt. Eventueel moeten vooraan of achteraan nullen bijgevoegd worden.
Elke 3 bits vervangt men door de overeenstemmende octale waarde die meteen ook de decimale waarde voorstelt. Dit is mogelijk, omdat de maximale waarde die met 3 bits kan genoteerd worden $2^3 - 1 = 7$ bedraagt.

Voorbeeld:

$$\begin{array}{lcl} (1010111.11011)_2 & \rightarrow & 001\ 010\ 111\ .\ 110\ 110 \\ & \rightarrow & 1\quad 2\quad 7\quad .\quad 6\quad 6 \\ & = & (127.66)_8 \end{array}$$

3.2.4.2 Binair naar hexadecimaal

Het analoge gebeurt voor de omzetting van binair naar hexadecimaal: de groepering gebeurt dan per 4 bits. Elke 4 bits worden door het overeenstemmende hexadecimale cijfer vervangen.
Hierbij kan eventueel voor de duidelijkheid eerst volgende tabel worden opgesteld. Merk op dat het eerste deel ook gebruikt kan worden voor omzetting naar het octale stelsel.

binair	hexadecimaal (octaal)	binair	hexadecimaal
0 000	0	1 000	8
0 001	1	1 001	9
0 010	2	1 010	A
0 011	3	1 011	B
0 100	4	1 100	C
0 101	5	1 101	D
0 110	6	1 110	E
0 111	7	1 111	F

Voorbeeld:

$$\begin{array}{rcl}(1010111.11011)_2 & \rightarrow & 0101\ 0111\ .\ 1101\ 1000 \\ & \rightarrow & 5 \quad 7 \quad . \quad D \quad 8 \\ & = & (57.D8)_{16}\end{array}$$

3.2.4.3 Octaal of hexadecimaal naar binair

Van hexadecimaal of octaal naar binair gaat even eenvoudig: elk hexadecimaal of octaal cijfer wordt vervangen door 4, respectievelijk 3 bits.

Voorbeelden:

$$\begin{array}{rcl}(523.7)_8 & \rightarrow & 101\ 010\ 011\ .\ 111 \\ & = & (101010011.111)_2\end{array}$$

$$\begin{array}{rcl}(802.AC)_{16} & \rightarrow & 1000\ 0000\ 0010\ .\ 1010\ 1100 \\ & = & (100000000010.101011)_2\end{array}$$

3.2.4.4 Octaal naar hexadecimaal en vice versa

Conversies tussen het octaal en het hexadecimaal stelsel zal men steeds het gemakkelijkst uitvoeren via het binair stelsel.

Voorbeeld:

$$\begin{array}{rcl}(C2F.A)_{16} & = & (\ 1100\ 0010\ 1111\ .\ 1010\ )_2 \\ & = & (\ 110\ 000\ 101\ 111\ .\ 101\ )_2 \\ & = & (6057.5)_8\end{array}$$

3.3 Bewerkingen

Alle bewerkingen in andere talstelsels gebeuren in principe op dezelfde manier als in het decimaal stelsel. Men moet er echter rekening mee houden dat als in een welbepaalde cijferpositie een waarde zou moeten genoteerd worden die groter is dan radix-1 (dit is het grootste cijfer in het talstelsel), er een overdracht naar de volgende cijferpositie moet doorgevoerd worden.

We beperken ons hier tot een korte bespreking van de optelling en de aftrekking aan de hand van enkele voorbeelden.

3.3.1 Optelling

3.3.1.1 Binair

Men kan volgende basissommen gebruiken om binaire optellingen uit te voeren:

$$0 + 0 = 0 \qquad 0 + 1 = 1 \qquad 1 + 0 = 1 \qquad 1 + 1 = 10$$

Praktisch betekent de laatste som, dat er een nul wordt geschreven en dat de 1 wordt overgedragen naar de volgende rang.

```
 101          10100          11.01
+110          +1111         +101.11
 ---         ------         ------
1011         100011         1001.00
```

3.3.1.2 Octaal

Vermits er slechts 8 cijfers zijn, betekent dit dat 7 + 1 = 10. Praktisch gezien kan men decimaal optellen; als het resultaat in een welbepaalde positie evenwel groter is dan 7, behoudt men wat er meer is dan 8 en telt men 1 op bij de volgende positie.

```
 135          165.34          617.3
 +74          +36.12         +1253.5
 ---          ------         ------
 231          223.46          2073.0
```

3.3.1.3 Hexadecimaal

Hier geldt: F + 1 = 10. Wat overblijft na aftrekking van 16, blijft in de desbetreffende positie staan en er wordt 1 overgedragen.

```
  3A2          3DC6          4A.BC
 +B1          +A8C4         +20.F9
 ---          -----         ------
  453          E68A          6B.B5
```

3.3.2 Aftrekking

3.3.2.1 Binair

Er geldt:

0 - 0 = 0 1 - 0 = 1 1 - 1 = 0 10 - 1 = 1

Het laatste betekent dat men 1 schrijft en 1 aftrekt van een hogere rang in het aftrektal.

```
 1001          11011          10000
 -101          -1010            -11
 ----          -----          -----
  100          10001           1101
```

Men kan steeds weer de som maken van verschil en aftrekker, om het resultaat te verifiëren.

3.3.2.2 Octaal

Praktisch kan men om $15 - 7$ te berekenen als volgt te werk gaan:

$$15 - 7 \Rightarrow (5 + 8) - 7 = 6$$

Het grondtal wordt op die positie dus bijgeteld en er wordt 1 afgetrokken van de naasthogere rang.

```
  165          17133          215.3
 -47          -4625          -26.7
 ---          -----          -----
  116          12306          166.4
```

3.3.2.3 Hexadecimaal

Hier moet men 16 bijtellen bij een cijfer uit het aftrektal als dit kleiner is dan het overeenstemmende cijfer uit de aftrekker.

```
 168          5A4          7E.F9
−75          −3F8         −11.32
 ──           ────         ──────
  F3           1AC          6D.C7
```

3.4 Complementen

In een computer wil men alle bewerkingen herleiden tot optellingen, zo ook de aftrekking. Hiertoe maakt men gebruik van *complementen*. Er kan slechts één complement van een getal gevormd worden als het aantal cijfers vermeld is waarin er dient gecomplementeerd te worden. Men vormt hiertoe een nieuw getal met evenveel cijfers als het oorspronkelijke getal.

3.4.1 Radix−1-complement

Het (radix-1)-complement in n cijfers van een getal wordt bekomen door van het grootste getal met n cijfers het te complementeren getal af te trekken. Beide getallen hebben dus evenveel cijfers.

Voorbeeld 1: **decimaal**: **9-complement**

a) 9-complement van 823 in 3 cijfers is $999 - 823 = 176$
Er geldt natuurlijk ook: 9-complement van 176 is 823

b) Het 9-complement in 4 cijfers van 823 is $9999 - 0823 = 9176$

Voorbeeld 2: **binair**: **1-complement**

Het 1-complement van 101011 in 6 bits is $111111 - 101011 = 010100$

In praktijk vervangt men elke 0 door een 1 en elke 1 door een 0: elk bit wordt geïnverteerd. Dit is in een elektronische schakeling makkelijk te implementeren.

3.4.2 Radix-complement

Het radix-complement van een getal wordt bekomen door bij het (radix-1)-complement 1 op te tellen.

Voorbeeld 1: **decimaal**: **10-complement**

Het 10-complement in 3 cijfers van 823 is gelijk aan 177, want:
9-complement in 3 cijfers van 823 is 176 en $176 + 1 = 177$

Voorbeeld 2: **binair**: **2-complement**

Het 2-complement van 101011 in 6 cijfers is gelijk aan 010101, want:
1-complement in 6 cijfers van 101011 is 010100 en $010100 + 1 = 010101$

Reken zelf na (binair): in 6 bits is het 2-complement van 100100 gelijk aan 011100.

Hieruit kan volgende praktische regel afgeleid worden om voor de mens makkelijk het 2-complement te bepalen: van rechts naar links laat men alle bits onveranderd tot en met de eerste 1, daarna inverteert men elk bit.
In een computer is ook de bepaling van het 2-complement te realiseren met een eenvoudige logische elektronische schakeling.

3.4.3 De aftrekking als optelling

Een aftrekking kan via de optelling worden berekend, gebruikmakend van complementen.
Volgende regels kunnen gehanteerd worden bij de berekening van aftrektal minus aftrekker:

1. Aftrektal en aftrekker worden in evenveel cijfers genoteerd; eventueel moet de aftrekker links worden aangevuld met nullen, tot hij evenveel cijfers bevat als het aftrektal.

2. Bereken het radix-complement van de aftrekker.

3. Bepaal de som van het aftrektal en het radix-complement van de aftrekker (zie punt 2).

4. (a) Is aftrektal groter dan aftrekker: laat de meest linkse 1 weg van de som om het resultaat te bekomen.

   (b) Is aftrektal kleiner dan aftrekker: het verschil is het radix-complement van de som, met een min-teken ervoor.

3.4.3.1 Decimale voorbeelden

(i) $725 - 81 = ?$

1. 81 uitbreiden tot 3 cijfers: 081
2. 10-complement van 081 is 919
3. $725 + 919 = 1644$
4. $725 > 81$ (geval 4.a)), dus meest beduidende (= meest linkse) 1 weglaten
   $\Rightarrow$ resultaat = 644

$725 - 81$ is dus 644.

(ii) $76 - 1764 = ?$

1. 1764 moet niet uitgebreid worden
2. 10-complement van 1764 is 8236
3. $76 + 8236 = 8312$
4. $76 < 1764$ (geval 4.b)), dus 10-complement van 8312 geeft 1688
   $\Rightarrow$ resultaat = -1688

$76 - 1764$ is dus -1688.

3.4.3.2 Binaire voorbeelden

(i) $100010 - 1101 = ?$

1. 1101 uitbreiden tot 6 bits: 001101
2. 2-complement van 001101 is 110011
3. $100010 + 110011 = 1010101$
4. $100010 > 1101$ (geval 4.a)), dus meest beduidende (= meest linkse) 1 weglaten
   $\Rightarrow$ resultaat = 10101

$100010 - 1101$ is dus 10101.

(ii) $1101 - 100010 = ?$

1. 100010 moet niet uitgebreid worden
2. 10-complement van 100010 is 011110
3. $1101 + 011110 = 101011$
4. $1101 < 100010$ (geval 4.b)), dus 2-complement van 101011 geeft 010101
   $\Rightarrow$ resultaat = -10101

$1101 - 100010$ is dus -10101.

3.4.3.3 In de praktijk

Het zal de lezer duidelijk zijn dat deze berekeningsmethode in computers het grote voordeel biedt dat er in de ALU slechts één aritmetische schakeling nodig is, namelijk een optelschakeling. Vermenigvuldigen en delen worden immers meestal herleid tot herhaaldelijk optellen en aftrekken. Het is alleen in de huidige complexere processoren dat dit ook in hardware is geïmplementeerd.

Laat ons de regels van §3.4.3 vanuit een praktisch oogpunt bekijken:
Regel 1.:

- "Aftrektal en aftrekker worden in evenveel cijfers genoteerd; eventueel moet de aftrekker links worden aangevuld met nullen, tot hij evenveel cijfers bevat als het aftrektal."
- Deze regel vervalt: alle getallen worden immers gecodeerd in een vast aantal bits.

Regel 2.:

- "Bereken het radix-complement van de aftrekker."
- Het 2-complement wordt door een triviale logische schakeling verwezenlijkt.

Regel 3.:

- "Bepaal de som van het aftrektal en het radix-complement van de aftrekker."
- De som wordt berekend door de optelschakeling.

Regel 4.(a):

- "Is aftrektal groter dan aftrekker: laat de meest linkse 1 weg van de som om het resultaat te bekomen."
- Deze regel vervalt doordat alle getallen in een vast aantal bits worden gecodeerd en de meest linkse 1 van de som dus niet kan worden opgeslagen.

Regel 4.(b):

- "Is aftrektal kleiner dan aftrekker: het verschil is het radix-complement van de som, met een min-teken ervoor."
- Deze regel vervalt indien de computercode gebruikmaakt van een complementvoorstelling voor negatieve getallen; het bekomen resultaat is dan immers de voorstelling van het gezochte negatieve getal, van het verschil.

Een en ander wordt verder verduidelijkt in het volgende hoofdstuk.

Hoofdstuk 4

Gegevenscodering

4.1 Inleiding

Er is reeds herhaaldelijk op gewezen dat het intern geheugen op elk moment de gegevens bevat die de CPU op dat ogenblik nodig heeft. Deze data zijn onder meer de opdrachten die deel uitmaken van het uitgevoerde programma samen met de gegevens die door dat programma verwerkt worden. Dit hoofdstuk behandelt uitsluitend deze te verwerken gegevens, zonder evenwel volledigheid na te streven.

De conventie die gebruikt wordt om gegevens intern in een computer voor te stellen, heet een code. De gegevens kunnen letters, cijfers of speciale tekens zijn. Afhankelijk van de gebruikte code en van de toepassing wordt elk teken ofwel afzonderlijk ofwel als een groep van tekens (bv. een groep van cijfers is een getal) gecodeerd. In een eerste paragraaf komt foutdetectie kort aan bod. Daarna wordt de voorstelling van numerieke en van alfanumerieke gegevens besproken. Tot slot wordt kort aandacht besteed aan de manier waarop beeld en geluid kunnen worden voorgesteld en hoe ze kunnen worden gecomprimeerd.

4.2 Foutdetectie

Om de intern opgeslagen gegevens te beveiligen tegen fouten kunnen allerlei technieken aangewend worden. De meest gebruikte en goedkoopste manier is, dat bij elke byte een extra bit wordt gevoegd, *pariteitsbit* genoemd. Dit betekent dat bijvoorbeeld elke geheugenbyte eigenlijk uit 9 bits bestaat, waarvan er slechts 8 effectief voor gegevensopslag worden gebruikt.

Bij het wegschrijven van de gegevens zal de computer voor elke groep van 8 gegevensbits zelf een waarde geven (0 of 1) aan de pariteitsbit, zodat er in die negen bits een even of een oneven

aantal bits op 1 staan, afhankelijk van het feit of er gewerkt wordt met even of met oneven pariteit. Deze werkwijze gebeurt buiten de gebruiker om, deze het negende bit zelf niet kan beïnvloeden.

Bij elke gegevensoverdracht wordt er een pariteitscontrole uitgevoerd door de computer. Als de pariteit niet meer in orde is, dan betekent dit dat er bij de overdracht van deze byte een fout is opgetreden.
Het ontvangen van één foutief bit in een byte kan op die manier ontdekt worden. Twee fouten detecteren is evenwel onmogelijk: de kans dat die tegelijkertijd optreden is evenwel veel kleiner.

Merk op dat pariteitscontrole alleen toelaat een fout te melden; verbeteren is onmogelijk omdat niet de plaats van de foutieve bit kan aangeduid worden. Wil men toch een *error-correcting memory*, dan moeten er aan elk byte 3 extra bits worden toegevoegd om de plaats aan te duiden van de opgetreden fout. Met drie bits kunnen immers waarden van 0 tot 7 worden aangeduid. ECC-geheugen wordt meestal alleen toegepast in duurdere servers waar men streeft naar een foutvrije werking de klok rond.

4.3 Codes voor numerieke informatie

De codes die hier eerst besproken worden bekijken een reeks cijfers als een getal en coderen dit getal in zijn geheel. Ze laten op die manier rekenkundige bewerkingen toe met de voorgestelde (gecodeerde) getallen. Hierbij moet onderscheid gemaakt worden tussen enerzijds gehele en anderzijds kommagetallen.

4.3.1 Gehele getallen: fixed point codering

Wanneer alleen positieve getallen moeten worden voorgesteld, bijvoorbeeld adressen in een geheugen, worden gewone binaire getallen gebruikt. Wanneer zowel positieve als negatieve waarden moeten worden voorgesteld, kunnen verschillende werkwijzen gebruikt worden.

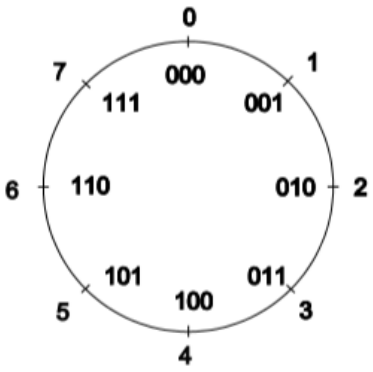
Vermits bij al deze coderingen een vast aantal bits wordt gebruikt, zullen telkens de cyclische eigenschappen optreden eigen aan een nummering met een vast aantal cijfers per getal. Indien we bijvoorbeeld slechts 4 cijfers gebruiken in het decimaal stelsel, volgt op 9999 het getal 0000. Hetzelfde geldt in het binair stelsel met 4 bits: op 1111 volgt 0000. We zullen bijgevolg een circulaire schaal gebruiken om deze codes voor te stellen.

In de volgende paragrafen die deze codes beschrijven, veronderstellen we voor de eenvoud van voorstellen dat elk getal wordt gecodeerd in een 3-bit-woord. Het is evident dat dit in praktijk nooit zal gebeuren, omdat men hiermee slechts 8 verschillende gehele getallen kan coderen. Maar wat geldt voor 3 bits, kan zonder meer veralgemeend worden tot 16, 32 of 64 bits, de meest courante woordlengten voor *fixed point codering* of vastekommacodering.

In de onderstaande cirkelfiguren staan aan de binnenkant telkens de inwendige bitpatronen, aan de buitenkant het voorgestelde getal. Het aantal bits zal meer algemeen als n genoteerd worden.

Er bestaan meerdere codes, die elk zowel voor- als nadelen hebben, afhankelijk van de toepassing waarin de codering wordt toegepast. Voor de voorstelling van gehele getallen in programma's op PC's is de 2-complement notatie wellicht de meest gebruikte.
Hierna volgt een overzicht.

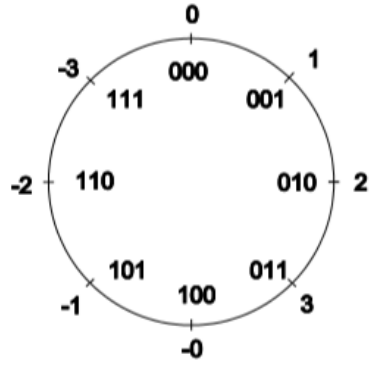
4.3.1.1 Binair



De *binaire* voorstelling is de meest voor de hand liggende codering. Ze stelt een geheel getal voor als zijn binaire equivalent. Deze n bits kunnen $2n$ verschillende bitpatronen vormen.
Deze codering kan evenwel slechts uitsluitend gebruikt worden voor de voorstelling van positieve getallen in het interval $[0, 2^n - 1]$.

Het grootste nadeel is dat er met deze voorstelling geen negatieve getallen kunnen gecodeerd worden. Wil men dit wel mogelijk maken, dan opteert men uiteraard voor een code die de helft van de patronen toewijst aan negatieve getallen en de andere helft aan negatieve.

4.3.1.2 Absolute value plus sign

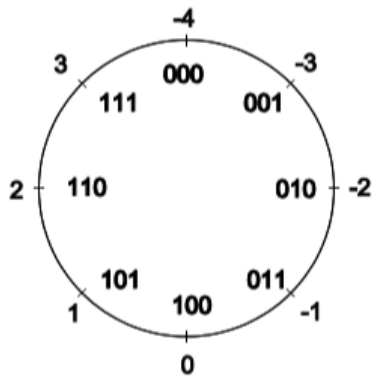


De *absolute waarde en teken* voorstelling gebruikt het uiterst linkse bit als tekenbit: een 0 voor positieve en een 1 voor negatieve getallen. De overige bits stellen de absolute waarde voor in binaire vorm.

Vermits n bits $2n$ verschillende bitpatronen kunnen aannemen, waarvan de helft gebruikt wordt voor de voorstelling van negatieve getallen en de andere helft voor die van de positieve getallen, kunnen de voorgestelde getallen waarden in het interval $[-(2^{n-1}-1), 2^{n-1}-1]$ aannemen.

Het voordeel van deze voorstelling is dat vermenigvuldigen met -1 evident is.
De nadelen zijn echter dat er 2 discontinuïteiten zijn op de schaal en 2 voorstellingen voor nul.

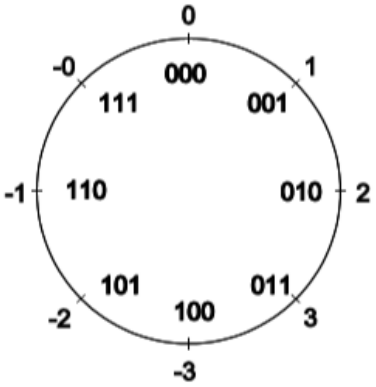
4.3.1.3 Binary offset



We bespreken deze voorstelling omdat ze gebruikt wordt bij het coderen van de exponent in de floating point codering (zie verder).
Hierbij wordt de interne voorstelling bekomen door het binair equivalent te nemen van het voor te stellen getal vermeerderd met 2^{n-1}. Er grijpt als het ware een verschuiving plaats over 2^{n-1}.
Het tekenbit is 0 voor negatieve en 1 voor positieve getallen.
De voor te stellen waarden behoren tot het interval $[-2^{n-1}, 2^{n-1} - 1]$.

Het voordeel van deze voorstelling is dat er slechts 1 discontinuïteit.
Het nadeel is echter dat de voorstelling van nul geen getal is met alle bits gelijk aan nul.

4.3.1.4 1-complement codering



Positieve getallen, nul inbegrepen, worden gecodeerd door hun binair equivalent, negatieve getallen door het 1-complement van de binair geconverteerde absolute waarde.
Het tekenbit is dus nul voor positieve en 1 voor negatieve getallen.
De voor te stellen waarden behoren tot het interval $[-2^{n-1} + 1, 2^{n-1} - 1]$.

Voordelen van deze voorstelling zijn:

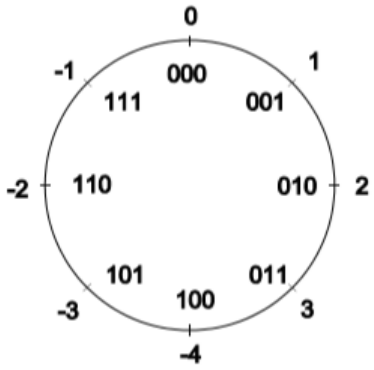
- Een getal vermenigvuldigen met -1 gebeurt door alle bits te inverteren (d.i. het 1-complement nemen).
- De voorstelling van het resultaat van een optelling of van een vermenigvuldiging kan bekomen worden door de bewerkingen uit te voeren met de voorstellingen, en de n minst betekenisvolle bits op te tellen bij de n meest betekenisvolle bits.

  Voorbeeld 1
  $3 + (-2) =?$
  met de gecodeerde waarden: $011 + 101 = 1000$
  1000 opsplitsen in 2 keer 3 bits en optellen: $001 + 000 = 001$
  001 is de voorstelling van 1, dus: $3 + (-2) = 1$

Voorbeeld 2
$3 + 2 =?$
met de gecodeerde waarden: $011 + 010 = 101$
101 is de voorstelling van -2, dus $3 + 2 = -2$ (5 is immers niet voor te stellen met 3 bits)

Het nadeel van deze voorstelling is dat er 2 voorstellingen zijn van het getal nul.
Dit maakt dat in recente computers men liever de onderstaande code gebruikt.

4.3.1.5 2-complement codering



Voor positieve getallen (nul inbegrepen) wordt het binair equivalent genomen, voor negatieve getallen het 2-complement van de binair geconverteerde absolute waarde.
Het tekenbit is dus 0 voor positieve en 1 voor negatieve getallen.
De voor te stellen waarden liggen in het interval $[-2^{n-1}, 2^{n-1} - 1]$.

Voordelen van deze voorstelling zijn:

- Vermenigvuldigen met -1 gebeurt door het 2-complement te nemen..
- Berekeningen kunnen gebeuren met de gecodeerde waarden en geven ogenblikkelijk de code van het resultaat. Hierbij worden uitsluitend de n minst beduidende bits behouden.

  Voorbeeld 1
  $3 + (-2) =?$
  met de gecodeerde waarden: $011 + 110 = 1001$
  001 is de voorstelling van 1, dus: $3 + (-2) = 1$

  Voorbeeld 2
  $3 + 2 =?$
  met de gecodeerde waarden: $011 + 010 = 101$
  101 is de voorstelling van -3, dus $3 + 2 = -3$ (5 is immers niet voor te stellen met 3 bits)

Deze voorstelling is de meest gebruikte om gehele getallen inwendig te coderen. Het courante aantal bits is dan
- ofwel 16 (bereik [-32768 .. 32767])
- ofwel 32 (bereik [-2147483648 .. 2147483647]).

Het gegevenstype dat met deze codering overeenstemt is `integer` of `int`.

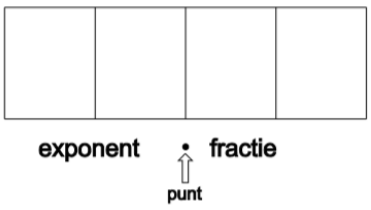
Er moet opgemerkt worden dat bij de meeste computers en talen die deze methode gebruiken, er GEEN foutmelding wordt gegeven als dit bereik wordt overschreden; er zal een foutief resultaat worden verkregen, want de voorstelling blijft immers beperkt tot de getallen op de cirkel.
Dit betekent bv. ook dat in 32 bits zal gelden: $2147483647 + 1 = -2147483648$.

4.3.2 Rationale getallen: floating point codering

Met de fixed point codering is het in principe mogelijk ook niet gehele getallen voor te stellen. Het enige wat zou moeten afgesproken worden, is de plaats van de punt tussen twee van de gebruikte bits. Dit impliceert dan evenwel dat voor alle gecodeerde getallen deze plaats dezelfde is. Getallen die in grootteorde zeer sterk afwijken van elkaar, kunnen dan evenwel met dit soort van codering niet worden voorgesteld.

Floating point codering of vlottendekommavoorstelling gelijkt sterk op de wetenschappelijke notatie van een calculator, d.w.z. dat het getal wordt geschreven als $fractie \times radix^{exponent}$. Hierbij is $fractie$ kleiner dan 1. In een computer is $radix$ impliciet, zodat uitsluitend $fractie$ en $exponent$ expliciet moeten worden opgeslagen.

4.3.2.1 Een decimaal voorbeeld



Om de algemene principes en de gevolgen hiervan te beschrijven, houden we eerst een decimale denkoefening. Onderstel dat er vier decimale cijfers gereserveerd zijn voor de notatie van getallen en dat alleen positieve getallen en positieve exponenten zijn toegelaten. Onderstel ook dat we twee cijfers voorzien voor de exponent van 10 en twee cijfers na de punt voor de fractie.

De grenzen van deze gekozen voorstelling worden als volgt bepaald:

- het kleinst voorstelbare getal (verschillend van nul) wordt bekomen door de kleinste fractie te combineren met de kleinste exponent: $0.01 \times 10^0 = 0.01$
- het grootst voorstelbare getal wordt bekomen door de grootste fractie te combineren met de grootste exponent: $0.99 \times 10^{99} \approx 10^{99}$

Hieruit kan het dynamisch bereik bepaald worden: dit is per definitie de verhouding van het grootste tot het kleinste ($\neq 0$) voor te stellen getal.
Dit geeft hier $10^{99}/0.01 = 10^{101}$.
Als we dezelfde vier cijfers zouden gebruiken in een fixed point codering, dan zou het dynamisch bereik zijn $9999/0001 \approx 10^4$, wat vele malen kleiner is dan in de floating point voorstelling in hetzelfde aantal cijfers.

4.3.2.2 De kenmerken

In praktijk wil men natuurlijk ook negatieve fracties en exponenten voorstellen. Er zijn veel formaten mogelijk, maar ze hebben meestal de volgende kenmerken.

TEKENBIT

Om het teken van het getal te kunnen voorstellen voorziet men steeds een extra cijferpositie. In de voorstelling hierboven met 4 decimale cijfers zal er dan een extra cijfer moeten toegevoegd worden. In een binaire computercode zal het teken van het getal worden voorgesteld door het meest beduidende bit.

GENORMALISEERDE MANTISSE

Vermits het teken apart wordt beschouwd, is de fractie zelf dus een getal zonder teken dat kleiner is dan 1. Van deze fractie noteert men evenwel alleen dbf e cijfers na de punt (komma). Deze cijfers noemt men de **mantisse**. Er geldt dus: $fractie = 0.mantisse$.

In de floating point codering van de getallen zoals die tot nu toe is beschreven, bestaat er geen eenduidige voorstelling voor die getallen: het getal 9 kan zowel geschreven worden als 0.9×10^1 of als 0.09×10^2.
Deze dubbelzinnigheid kan evenwel niet worden getolereerd: het maakt immers het vergelijken van getallen zeer moeilijk.

Om dit te verhelpen worden vlottendekommagetallen voorgesteld in een **genormaliseerde** vorm, waarbij het meest beduidende cijfer van de mantisse steeds verschillend van nul moet zijn.
Deze regel zorgt er ook voor, dat steeds een zo groot mogelijk aantal cijfers van de fractie kan genoteerd worden, wat de nauwkeurigheid ten goede komt.

Het is evident, dat deze regel niet opgaat voor het getal 0. Daarom wordt meestal per definitie gesteld, dat het getal 0 in genormaliseerde vorm wordt voorgesteld door alle cijfers, ook die van de exponent en van het teken, gelijk aan nul te stellen. Dit vereenvoudigt de nuldetectie en maakt de voorstelling ervan gelijk aan die in fixed point codering.

EXPONENT IN EXCESS-CODE

Om zowel positieve als negatieve exponenten te kunnen noteren, wordt de exponent voorgesteld in een **excess-codering** of **binary offset codering**. Deze voorstelling wordt bekomen door bij de exponent een afwijking (offset, bias of excess) Δ bij te tellen:

$$\Delta = \frac{1}{2} radix^s$$

waarbij s het aantal cijfers is van de exponentvoorstelling. Het aantal mogelijke exponenten

wordt immers gegeven door $radix^s$.

Er geldt dus:
exponentvoorstelling = exponentwaarde + excess Δ

Voor het formaat uit het decimale voorbeeld is de excess $1/2 \times 10^2 = 50$.
Een exponent = 2 wordt in het voorbeeld dus genoteerd als 52, een exponent = -2 wordt genoteerd als 48.

4.3.2.3 De gevolgen

GRENZEN, OVERFLOW, UNDERFLOW

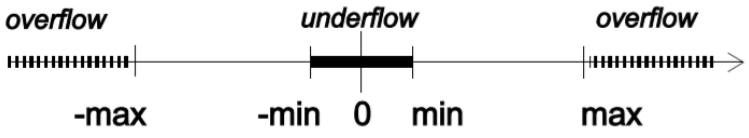
De grenzen zijn een getallenpaar (kleinste, grootste) die binnen een bepaald systeem alle voorstelbare getallen omsluiten. In een genormaliseerd floating point systeem kunnen volgende vergelijkingen worden genoteerd:

$$max = 0.M_{max} \times radix^{E_{max}} \qquad min = 0.M_{min} \times radix^{E_{min}}$$

Hierbij zijn

- max en min het grootste en kleinste voorstelbare getal in absolute waarde;
- E_{max} en E_{min} de grootste en kleinste voorstelbare exponentwaarde;
- M_{max} en M_{min} de grootste en kleinste voorstelbare mantisse.

Noteer dat er grenzen zijn aan positieve zijde, doch ook aan de negatieve kant. In de veronderstelling dat van de mantisse de absolute waarde wordt genoteerd, kunnen volgende grenzen en gebieden op een getallenas worden aangeduid:



De grenzen zijn dan immers volledig symmetrisch t.o.v. nul.

Overflow is het gebied aangeduid met lichte arcering. Er is sprake van overflow als ten gevolge van een berekening een getal moet voorgesteld worden in dit gebied: de absolute waarde van dit getal is te groot.
Underflow is het gebied aangeduid met dichte arcering. Men spreekt van underflow als de absolute waarde van het voor te stellen getal te klein is. Nul maakt geen deel uit van dit gebied

want wordt hiervan uitgesloten door de vroeger gemaakte afspraak over de genormaliseerde mantisse.
Beide situaties geven aanleiding tot foutmeldingen en vaak wordt het programma onderbroken.

Indien we deze principes toepassen in een binaire situatie, moeten we er ons bovendien van bewust zijn dat zelfs in de gebieden waar wel getallen kunnen voorgesteld worden, niet elk reëel getal een voorstelling heeft. Zoals vroeger is besproken, heeft namelijk niet elke decimale fractie een exact binair equivalent.

NAUWKEURIGHEID

De nauwkeurigheid waarmee rationale getallen worden voorgesteld, is afhankelijk van het aantal bits waarin de mantisse wordt gecodeerd.
Is dit m bits, dan is de kleinst mogelijke aangroei van de mantisse 2^{-m}. Als $2^{-m} = 10^{-d}$, dan stelt d de decimale nauwkeurigheid voor van de voorstelling.
Het aantal decimale cijfers waarmee in de voorstelling wordt rekening gehouden is dus:

$$d = mlog2$$

GEVOLGEN VOOR BEREKENINGEN

Optellingen en aftrekkingen vereisen dat de exponenten gelijk zijn. Dit impliceert veelal dat één van de operanden niet meer genormaliseerd is, waardoor aan nauwkeurigheid ingeboet wordt. Hoe de afrondingen gebeuren is afhankelijk van de constructeur en/of de onderliggende software.

Vermenigvuldigen en delen is eenvoudiger: de mantissen worden vermenigvuldigd of gedeeld, de exponenten opgeteld of afgetrokken. Meestal verricht men deze bewerkingen uitgaande van de genormaliseerde getallen in een uitgebreid aantal bits. Achteraf wordt het resultaat opnieuw genormaliseerd; dit resulteert uiteraard opnieuw in afrondingsfouten.

4.3.2.4 Voorbeelden

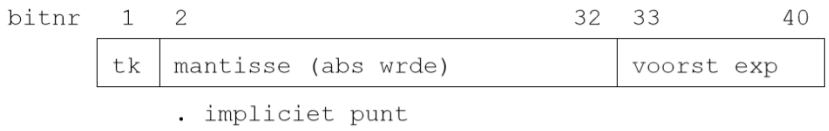
De volgende voorbeelden maken allemaal gebruik van de hiervoor geschetste algemene regels, evenwel in het binaire stelsel. We geven telkens de gebruikte conventies aan en leiden de grenzen en de nauwkeurigheid af.
Deze berekeningen moeten als oefening worden opgevat. Het is zeker niet de bedoeling om de verschillende coderingsafspraken uit het hoofd te leren, wel moet de student op basis van gegeven afspraken de grenzen en de nauwkeurigheid kunnen berekenen.

Vermits er binair wordt gewerkt, zal het grondtal van de opgeslagen exponent 2 of een macht van 2 zijn. IBM heeft ooit een codering uitgewerkt met als grondtal 16, maar hier gaan we

verder niet op in. In alle onderstaande voorbeelden is de grondtal van de exponent 2, een keuze die ook het meest voor de hand ligt.

APPLE-II

Dit is een zeer eenvoudige en voor de handliggende voorstellingswijze: de radix is 2 en de mantisse wordt gewoon binair genoteerd in absolute waarde. Het feit dat er in totaal 5 bytes gebruikt worden, heeft vooral historische redenen. De redering zou evengoed met minder of meer bytes kunnen gevoerd worden.



Bit nummer 1 is het tekenbit (0 voor pos en 1 voor neg).
Radix = 2.
Exponent: 8 bits, $\Delta = 2^7 = 128$, $E_{max} = 127$, $E_{min} = -128$.
Mantisse: 31 bits

$$0.M_{max} = (0.111...11)_2 \approx 1$$
$$0.M_{min} = (0.100...00)_2 = 0.5$$

Grenzen: max $\approx 1 \times 2^{127} \approx 1.7 \times 10^{38}$
min = $0.5 \times 2^{-128} \approx 1.47 \times 10^{-39}$
Voor negatieve getallen zijn de grenzen -min en -max.

Voorbeeld:

$$\begin{aligned} 36.79 &= (\ 24.CA3D70A3D70A3D70\ldots\ )_{16} \\ &= (\ 10\ 0100\ .\ 1100\ 1010\ 0011\ 1101\ 0111\ 0000\ 1010\ \ldots)_2 \\ &= (\ 0.10010011001010001111010111000001\ )_2 \times 2^6 \end{aligned}$$

Exponent: $6 + 128 = 134 = (86)_{16} = (10000110)_2$.

IEEE-FORMAAT

IEEE is een afkorting voor *Institute of Electrical and Electronical Engineers*. Deze organisatie houdt zich ondermeer bezig met het uitwerken van standaarden op allerlei domeinen van ICT.
Dit formaat van floating point codering wordt zeer veel gebruikt, o.a. ook omdat ze geïmplementeerd is in de numerieke rekeneenheden van de meeste recente processoren.
Er bestaat een codering in 32 bits (enkele precisie) en een codering in 64 bits (dubbele precisie); ze gaan beide wel uit van dezelfde principes.

Ten overstaan van het hiervoor beschreven model werden twee wijzigingen aangebracht, éé aan de voorstelling van de mantisse en als gevolg hiervan ook één aan die van de exponent.

Het feit dat de mantisse genormaliseerd moet zijn, maakt dat die steeds begint met een 1. Men kan zich dan wel afvragen waarom dat bit dan moet opgeslagen worden, vermits men toch weet dat het de waarde 1 heeft. In het voorstel van IEEE wordt dat bit dus niet opgeslagen. Men spreekt van een **mantisse met een verborgen meest beduidende bit**. Het gevolg is dat een extra bit ter beschikking is voor de mantisse, waardoor de nauwkeurigheid met een bit verhoogd is.
Vermits de waarde die overeenkomt met de opgeslagen mantisse nu niet meer $0.M$ is, maar wel $1.M$, spreekt men niet meer van een fractie, maar van een **significand**.
Hierdoor geldt: $1 \leq \text{significand} < 2$.

Door deze keuze treedt er evenwel een dubbelzinnigheid op. In de hiervoor beschreven algemene situatie wordt 0 voorgesteld door alle bits nul te maken: 0 voor het tekenbit, allemaal nullen voor de mantisse en evenzo voor de exponent. Een echte mantisse kon immers worden onderscheiden van die 0, omdat er minstens één bit 1 moest zijn.
Indien dat bit echter niet wordt opgeslagen, is dit onderscheid niet meer te maken. Om deze onduidelijkheid weg te werken heeft men aan de normale grenzen van de exponentcodering 0 en E_{max} een speciale betekenis gehecht. Buiten de normale genormaliseerde situatie, levert dit in combinatie met het al dan niet nul zijn van de mantissebits vier extra situaties op.
Hieronder volgt een schematisch overzicht; in de figuren staan van links naar rechts telkens het tekenbit, de exponentbits en de mantissebits.

(a) normaal genormaliseerd

$\pm$	$0 < E < E_{max}$	M met willekeurig bitpatroon

Dit is de normale situatie waarbij E een waarde heeft die niet een grenswaarde is. De significand heeft als waarde $1.M$.

(b) voorstelling van 0

$\pm$	$0000\ldots0$	$0000\ldots0$

Is $E = 0$ (alle exponentbits 0) en geldt bovendien dat $M = 0$ (alle mantissebits 0), dan is dat de voorstelling van de waarde nul. Rekening houdend met het tekenbit kan zelfs +0 en -0 genoteerd worden, moest dat van belang zijn.

(c) gedenormaliseerd

$\pm$	$0000\ldots0$	minstens 1 bit verschillend van 0

Is $E = 0$ (alle exponentbits 0) en geldt bovendien dat $M \neq 0$ dan is dat de voorstelling van een niet-genormaliseerd getal. Dit laat toe om nog kleinere waarden op te slaan

dan genormaliseerd mogelijk is, evenwel met een kleinere nauwkeurigheid. Hierdoor kan men indien men dat wil de underflow-fout omzeilen en zeer kleine waarden noteren vooraleer men bij 0 uitkomt.

(d) voorstelling van oneindig

$\pm$	$1111\ldots1$	$0000\ldots0$

Is $E = E_{max}$ (alle exponentbits 1) en geldt bovendien dat $M = 0$ (alle mantissebits 0), dan is dat de voorstelling van oneindig. Het laat toe de fout overflow niet te moeten laten genereren en een "waarde" te noteren waar zelfs mee verder "gerekend" kan worden.

(e) voorstelling van NaN

$\pm$	$1111\ldots1$	minstens 1 bit verschillend van 0

Is $E = E_{max}$ (alle exponentbits 1) en geldt bovendien dat $M \neq 0$, dan is dit is de voorstelling NaN (Not a Number). Hiermee wordt een non-getal aangeduid, een resultaat wat men bijvoorbeeld bekomt door nul te delen door nul.

Het feit dat de grenzen van de exponentvoorstelling een speciale betekenis hebben gekregen, heeft tot gevolg dat de verschuiving (excess Δ) met 1 moet verminderd worden, vermits het aantal mogelijke exponenten met 2 is verminderd. Bovendien liggen de gebruikte waarden niet tussen 0 en E_{max}, maar wel tussen 1 en $E_{max} - 1$.
Dit resulteert in de grenzen die worden berekend in de volgende voorbeelden.

A) ENKELE PRECISIE (SHORT REAL, SINGLE, FLOAT) IN 32 BITS OF 4 BYTES

T	E 8(nrs 2-9)	M 23 (nrs 10-32)

De 32 bits worden in 3 velden onderverdeeld van links naar rechts:
T(eken) : 1 bit; 0 voor positieve, 1 voor negatieve getallen
E(xponent) : 8 bits, bit nr 2 meest beduidend, nr 9 minst beduidend
M(antisse) : 23 bits, nr 10 meest beduidend, nr 32 minst beduidend; punt vóór nr 10.

De basis is 2. Deze keuze leidt tot volgend overzicht:

Interne code	Voorgestelde waarde
$0 < E < 255$	$(-1)^T \times 2^{(E-127)} \times 1.M$
$E = 0$ en $M \neq 0$	$(-1)^T \times 2^{(-126)} \times 0.M$ niet genormaliseerd getal
$E = 0$ en $M = 0$	$(-1)^T \times 0.$
$E = 255$ en $M \neq 0$	NaN, Not a Number
$E = 255$ en $M = 0$	$(-1)^T \times \infty$

Voor de exponent kan volgende omrekeningstabel worden opgesteld:

E	exponent (exp), voorgestelde waarden	
	normaal excess-128	excess-127 vanaf 1
0	-128	niet gebruikt
1	-127	-126
...	...	...
126	-2	-1
127	-1	0
128	0	1
...	...	...
254	126	127
255	127	niet gebruikt

Het is de laatste kolom die gebruikt wordt in deze codering. Merk ook op dat voor deze kolom blijft gelden: $E = exp + \Delta$, met $\Delta = 127$. Zie ook algemene bespreking van het IEEE-formaat.

Dat geeft voor de grenzen in het gewone genormaliseerde geval:

max: $E_{max} = 254 = (11111110)_2$ in 8 bits
$M_{max} = (11111111111111111111111)_2$ in 23 bits
$\Rightarrow$ max = $(0\ 11111110\ 11111111111111111111111)_2$ in 32 bits
$\Rightarrow$ max = $2^{127} \times (2 - 2^{-23}) = 3.402823 \times 10^{38}$

min: $E_{min} = 1 = (00000001)_2$ in 8 bits
$M_{min} = (00000000000000000000000)_2$ in 23 bits
$\Rightarrow$ min = $(0\ 00000001\ 00000000000000000000000)_2$ in 32 bits
$\Rightarrow$ min = $2^{-126} \times 1.0 = 1.175494 \times 10^{-38}$

Deze ondergrens kan nog verlaagd worden in het niet-genormaliseerde geval (zie tabel interne code / voorgestelde waarde):
$E = 0$ en $M = (00000000000000000000001)_2$ ofwel $2^{-126} \times 2^{-23} = 1.4 \times 10^{-45}$

Nauwkeurigheid: er zijn 24 mantissebits (waarvan 1 verborgen)
$\Rightarrow$ kleinste aangroei: $2^{-24} = 10^{-d}$
$\Rightarrow d = 24log2 = 7.22$ beduidende decimale cijfers

Voorbeeld 1: wat is de interne voorstelling van 0.028?

$$\begin{aligned} 0.028 &= (0.\ 0000\ 0111\ 0010\ 1011\ 0000\ 0010\ 0000\ 1100\ 0100\ 1001)_2 \\ &= (1.11\ 0010\ 1011\ 0000\ 0010\ 0000\ \underline{11}00\ 0100\ 1001)_2 \times 2^{-6} \end{aligned}$$

Hieruit volgt:
M = $(110\ 0101\ 0110\ 0000\ 0100\ 0001 + 1)_2$ (om af te ronden op 23 bits)
= $(110\ 0101\ 0110\ 0000\ 0100\ 0010)_2$
E = -6 + 127 = 121 = $(01111001)_2$
T = 0

$\Rightarrow$ interne voorstelling van 0.028 = $(\ 0\ 0111\ 1001\ 110\ 0101\ 0110\ 0000\ 0100\ 0010\ )_2$

Voorbeeld 2: wat is de interne voorstelling van 36.79?

$$\begin{aligned} 36.79 &= (10\ 0100\ .\ 1100\ 1010\ 0011\ 1101\ 0111\ 0000\ 1010)_2 \\ &= (1.00100\ 1100\ 1010\ 0011\ 1101\ 0\underline{11}1\ 0000\ 1010)_2 \times 2^5 \end{aligned}$$

Hieruit volgt:
M = $(0\ 0100\ 1100\ 1010\ 0011\ 1101\ 01 + 1)_2$ (om af te ronden op 23 bits)
= $(001\ 0011\ 0010\ 1000\ 1111\ 0110)_2$
E = 5 + 127 = 132 = $(10000100)_2$
T = 0

$\Rightarrow$ interne voorstelling van 36.79 = $(0\ 1000\ 0100\ 001\ 0011\ 0010\ 1000\ 1111\ 0110)_2$

B) DUBBELE PRECISIE (LONG REAL, DOUBLE) IN 64 BITS OF 8 BYTES

T	E 11(nrs 2-12)	M 12 (nrs 13-64)

De 64 bits worden in 3 velden onderverdeeld van links naar rechts:
T(eken) : 1 bit; 0 voor positieve, 1 voor negatieve getallen
E(xponent) : 11 bits, bit nr 2 meest beduidend, nr 12 minst beduidend
M(antisse) : 52 bits, nr 13 meest beduidend, nr 64 minst beduidend; punt vóór nr 13.

De basis is 2. Deze keuze leidt tot volgend overzicht:

Interne codering	**Voorgestelde waarde**
$0 < E < 2047$	$(-1)^T \times 2^{(E-1023)} \times 1.M$
$E = 0$ en $M \neq 0$	$(-1)^T \times 2^{(-1022)} \times 0.M$ niet genormaliseerd getal
$E = 0$ en $M = 0$	$(-1)^T \times 0.$
$E = 2047$ en $M \neq 0$	NaN, Not a Number
$E = 2047$ en $M = 0$	$(-1)^T \times \infty$

Voor de exponent kan volgende omrekeningstabel worden opgesteld:

E	**exponent (exp), voorgestelde waarden**	
	normaal excess-1024	excess-1023 vanaf 1
0	-1024	niet gebruikt
1	-1023	-1022
...	...	...
1022	-2	-1
1023	-1	0
1024	0	1
...	...	...
2046	1022	1023
2047	1023	niet gebruikt

Het is de laatste kolom die gebruikt wordt in deze codering. Merk ook op dat voor deze kolom blijft gelden: E = exp + Δ, met Δ = 1023. Zie ook algemene bespreking van het IEEE-formaat.

Dat geeft voor de grenzen in het gewone genormaliseerde geval:

max: $E_{max} = 2046 = (11111111110)_2$ in 11 bits
$M_{max} = (1111111111111111111111111111111111111111111111111111)_2$ in 52 bits
$\Rightarrow$ max = $(0\ 11111111110\ 1111111111111111111\ldots1)_2$ in 64 bits
$\Rightarrow$ max = $2^{1023} \times (2 - 2^{-52}) = 1.79769313 \times 10^{308}$

min: $E_{min} = 1 = (00000000001)_2$ in 11 bits
$M_{min} = (0000000000000000000000000000000000000000000000000000)_2$ in 52 bits
$\Rightarrow$ min = $(0\ 00000000001\ 0000000000000000000\ldots0)_2$ in 64 bits
$\Rightarrow$ min = $2^{-1022} \times 1.0 = 2.22507386 \times 10^{-308}$

Deze ondergrens kan nog verlaagd worden in het niet-genormaliseerde geval (zie tabel interne code / voorgestelde waarde):
$E = 0$ en $M = (0000000000000000000\ldots01)_2$ ofwel $2^{-1022} \times 2^{-52} = 4.94 \times 10^{-324}$

Nauwkeurigheid: er zijn 53 mantissebits (waarvan 1 verborgen)
$\Rightarrow$ kleinste aangroei: $2^{-53} = 10^{-d}$
$\Rightarrow d = 53log2 = 15.95$ beduidende decimale cijfers

4.3.3 Decimale getallen

Bij administratieve toepassingen is het volume aan berekeningen meestal zeer klein, doch ze moeten met een volledige nauwkeurigheid gebeuren. In dit geval kan men geen gebruikmaken van een binaire voorstelling van de numerieke informatie zoals in de vorige paragrafen is beschreven.

De individuele cijfers moeten dan elk afzonderlijk als een geheel binair getal gecodeerd worden, en elk decimaal getal zal dan voorgesteld worden door een verzameling van gehele binaire getallen.
Elk cijfer wordt gecodeerd in 4 bits; de 6 overblijvende combinaties van deze 4-bit voorstelling worden dan gebruikt om teken, decimaal punt en einde van het getal voor te stellen.

Het is evident dat deze voorstelling veel geheugenruimte in beslag neemt. Ook is het wenselijk dat speciale voorzieningen zijn getroffen in de rekeneenheid om dit soort van getallen op een efficiënte wijze te manipuleren. Deze voorstellingswijze wordt op bijna alle business-computers toegepast. Ook is het een courante voorstelling bij de meeste zakrekenmachines.

4.4 Codes voor alfanumerieke informatie

Alfanumerieke informatie is gewone tekst. Elk tekensymbool (letter, cijfer, leesteken, spatie,...) in de tekst moet apart gecodeerd worden. De volledige tekst wordt dan voorgesteld door de aaneenschakeling van alle bits van de codes van alle tekens afzonderlijk.

Bij het ontstaan van computers werden er door verschillende constructeurs meerdere dergelijke codes opgesteld mede in functie van de toestellen waarvoor ze bestemd waren. Alfanumerieke codes moeten allerlei soorten tekens kunnen voorstellen. Deze karakters kunnen in twee categorieën worden opgesplitst: afdrukbare en niet-afdrukbare karakters.

Met afdrukbare karakters wordt bedoeld: alle karakters die, zoals de naam het zegt, visueel weergegeven kunnen worden bijvoorbeeld op papier of op een computerscherm. Het zijn de hoofdletters, de kleine letters, de cijfers, de spatie of blanco, lees- en andere tekens, kortom, minstens alle tekens die op een normaal toetsenbord van een computer staan.

Niet-afdrukbare tekens kunnen niet visueel worden weergegeven, maar ze moeten wel inwendig in de computer kunnen opgeslagen worden. Het zijn controlekarakters die gebruikt worden bij de sturing van allerlei rand- en communicatie-apparatuur (printer, terminal, modem, ...).
Volgende tekens vindt men o.a. steeds terug:

- CR, carriage-return: plaatst de cursor of de kop van de printer aan het begin van de lijn;
- LF, line-feed: plaatst de cursor of de kop van de printer op volgende lijn;
- BS, backspace: plaatst kop 1 positie naar links;
- FF, formfeed: plaatst kop bovenaan nieuw blad;
- BELL: geeft een hoorbaar alarm.

Elk teken wordt steeds voorgesteld door een bitpatroon, dat bestaat uit 6,7, 8 of 16 bits, afhankelijk van de gebruikte code en van hoeveel tekens men in de code wil opnemen.
De meest gebruikte codering is de ASCII-code (7 en 8 bits).
Met de sterke internationalisering van de informatica wordt recent ook UNICODE hoe langer hoe meer gebruikt: men voorziet 16 (of 24) bits voor een teken, waardoor ongeveer elk denkbaar teken codeerbaar is. Bovendien komen de eerste 256 tekens overeen met de ASCII-code.
Het is alleen de ASCII-code die we hier kort bespreken, zodat de lezer zich een beeld kan vormen van hoe dergelijke codering er uit ziet.

ASCII staat voor *American Standard Code for Information Interchange*. Het is een 7-bit code, die tot stand kwam door samenwerking tussen constructeurs van computerapparatuur en communicatiemateriaal. Met deze 7 bits kunnen 128 tekens gecodeerd worden, wat voldoende is om alle noodzakelijke karakters (letters van het Engels alfabet, cijfers, leestekens en controletekens) te kunnen opslaan.

Daar de meeste computers bytegericht werken, zal men veelal de 7 bits uitbreiden met een achtste bit dat meestal de waarde 0 krijgt. In alle afgeleide coderingen zijn de eerste 128 tekens steeds dezelfde (codes 0 t/m 127); de volgende 128 tekens kunnen op verschillende manieren worden toegewezen.

De volgende tabel bevat de standaard ASCII-codes (32-127). De eerste 32 karakters zijn niet-afdrukbare tekens en zijn niet weergegeven in de tabel.

Afdrukbare karakters standaard set

Code	Kar	Code	Kar	Code	Kar	Code	Kar	Code	Kar	Code	Kar
32		48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	DEL

Verklaring bij code xx

32 spatie, blanco
39 enkel aanhalingsteken, *single quote*
44 komma
45 minteken, *hyphen*
95 onderstreepteken, *underscore or back-arrow*
96 accent grave
126 tilde
127 *delete*-karakter, niet afdrukbaar

4.5 De voorstelling van beelden

Er zijn fundamenteel twee manieren waarop beelden kunnen voorgesteld worden in een computer: als bitmapafbeelding en als vectortekening. Het is de gebruikte software die uiteindelijk beslist op welke manier de figuur zal bewaard worden.

4.5.1 Bitmapafbeelding

Wordt een beeld voorgesteld als bitmapafbeelding, dan wordt ze geïnterpreteerd als een verzameling puntjes. Elk dergelijk punt wordt een **pixel** genoemd; deze term is een samentrekking van *picture element*. De manier waarop zo'n pixel zich voordoet in de afbeelding wordt dan gecodeerd. De hele afbeelding is dan de verzameling van dergelijke gecodeerde pixels en wordt een bitmap genoemd.

Deze benadering is populair omdat veel displayapparaten zoals printers en schermen ook werken met het pixelconcept en de opgeslagen afbeeldingen op die manier gemakkelijk weer te geven zijn.

De manier waarop de pixels in de bitmap gecodeerd worden, verschilt van toepassing tot toepassing.
In het geval van een simpel zwart-witbeeld kan elk pixel voorgesteld worden als 1 bit waarvan de waarde afhankelijk is van het feit of de pixel wit is dan wel zwart.
Moeten er ook grijswaarden worden weergegeven, zoals bijvoorbeeld in een zwart-witfoto, dan wordt elk pixel voorgesteld door meerdere bits (meestal 8), zodat een heel gamma aan grijsintensiteiten kan worden gecodeerd.
Gaat het om een kleurenafbeelding, dan wordt elk pixel gecodeerd door een meer complex systeem. Er zijn twee veel gebruikte technieken: RGB-codering en codering van helderheid samen met twee kleurcomponenten. Op deze laatste techniek gaan we niet dieper in; ze wordt vooral gebruikt omdat ze TV-beelden toelaat die ook kunnen getoond worden op de vroegere zwart-wittoestellen.
Bij de RGB-codering wordt elk pixel voorgesteld door 3 componenten, een rode (R), een groene (G) en een blauwe (B). Deze componenten komen overeen met de drie primaire kleuren van licht. Normaal wordt er één byte gebruikt voor elke kleur; die byte geeft de waarde weer van de intensiteit van de corresponderende kleur in de pixel. Dat betekent wel dat er nu voor elk pixel van het beeld 3 bytes nodig zijn in de voorstelling.

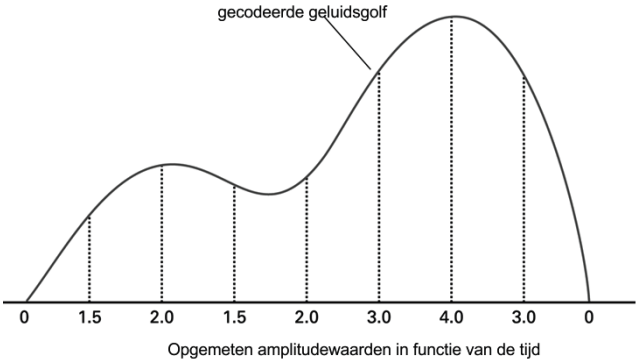
Een nadeel van het voorstellen van een beeld als bitmap is dat het niet zonder meer in een ander formaat kan getoond worden. Om een beeld bijvoorbeeld te vergroten moeten de pixels groter gemaakt worden, wat voor een meer korrelig effect zorgt. Deze techniek wordt onder meer gebruikt bij de digitale zoom van een fototoestel.

4.5.2 Vectortekening

Er bestaat een alternatieve manier om beelden voor te stellen. Ze vermijdt meteen ook het probleem van de aanpassing van de schaalgrootte. De figuur wordt hierbij beschreven als een verzameling geometrische structuren zoals lijnen en krommen. De codering van deze structuren gebruikt de technieken van de analytische meetkunde. Dergelijke benadering laat toe dat het toestel dat het beeld uiteindelijk moet tonen, zelf kan beslissen hoe deze deelfiguren moeten afgebeeld worden, eerder dan dat het een vast pixelpatroon wordt opgelegd.

Deze aanpak wordt onder meer gebruikt bij de schaalbare lettertypes die gebruikt worden in de hedendaagse tekstverwerkingssystemen; TrueType en Postscript zijn manieren om tekstsymbolen (en extra informatie er over) te beschrijven. Deze geometrische manier om beelden voor te stellen wordt ook toegepast in software voor computer-aided design (CAD) waarin ook driedimensionale figuren kunnen gemanipuleerd worden.

4.6 De voorstelling van geluid



De meest algemene manier om audio-informatie op te slaan in een computer en om die nadien ook te kunnen manipuleren, is de amplitude van de geluidsgolf te bemonsteren op regelmatige tijdsintervallen en de bekomen waarden op te slaan. Zo zou bijvoorbeeld de opeenvolging van de waarden 0, 1.5, 2.0, 1.5, 2.0, 3.0, 4.0, 3.0, 0 een geluidsgolf voorstellen waarvan de amplitude stijgt, weer een beetje daalt en nadien weer stijgt om dan weer volledig stil te worden (zie bovenstaande figuur).

De vraag is natuurlijk hoe groot de bemonsteringsfrequentie moet zijn om een aanvaardbaar resultaat te verkrijgen. In telefooncommunicatie is lang 8000 samples per seconde de standaard geweest. De opgemeten waarden werden dan over de communicatieverbinding doorgestuurd en aan de andere kan van de lijn werden die dan gebruikt om de opgenomen stem te reproduceren. Deze frequentie is evenwel veel te laag om aan de hedendaagse normen van high-fidelity kwaliteit van muziekopnames te kunnen voldoen. Bij muziekcd's is de samplefrequentie 44.100 Hz. Elke opgemeten waarde wordt dan in 16 bits gecodeerd (32 bits bij stereo), wat maakt dat elke opgenomen seconde muziek meer ongeveer 1.4 miljoen bits vereist.

Een alternatieve codeermethode is gekend onder de naam van *Musical Instrument Digital Interface* (**MIDI**). Ze wordt veel gebruikt in de muzieksynthesizers van bijvoorbeeld elektronische

keyboards en geluid bij computerspellen. In plaats van de muziek zelf te coderen, wat veel opslagruimte in beslag neemt (zie hiervoor), slaat men instructies op om de muziek te produceren. Meer in het bijzonder codeert MIDI welk instrument er gespeeld moet worden gedurende welke tijdsduur. Indien bijvoorbeeld een klarinet gedurende twee seconden een re moet spelen, kan dit gecodeerd worden in 3 bytes in plaats van meer dan twee miljoen bits aan een samplefrequentie van 44100 Hz. MIDI is dus een codeervorm die doet denken aan de opslag van een muziekpartituur.

4.7 Gegevenscompressie

Wil men gegevens opslaan of overdragen, dan is het soms aangewezen (in sommige gevallen zelf verplicht) om de omvang van de data te reduceren, zonder evenwel aan de inhoud van de eigenlijke informatie te raken. Men noemt dit gegevenscompressie. Deze paragraaf wil vooral een algemeen beeld schetsen en kort inzoomen op sommige bestandsformaten, zonder evenwel in detail te treden.

4.7.1 Mogelijke technieken

Er zijn twee soorten compressieschema's: compressie zonder verlies van informatie en compressie waarbij wel gegevensverlies optreedt. In het tweede geval kan er meestal sterker gecomprimeerd worden, maar uiteraard is dit alleen maar bruikbaar in situaties waar een beperkt aantal fouten kan getolereerd worden, zoals bijvoorbeeld bij beelden en audio.

Als de informatie lange sequenties van dezelfde waarde bevat, dan kan **run-length encoding** worden gebruikt. Hierbij wordt de waarde die meerdere keren voorkomt genoteerd samen met het aantal keer dat ze voorkomt. Zo heeft men bijvoorbeeld veel minder plaats nodig om aan te geven dat een bitpatroon bestaat uit "118 enen gevolgd door 294 nullen gevolgd door 58 enen" eerder dan door het volledige patroon van 470 bits te noteren.

Een andere techniek zonder gegevensverlies heet **frequency-dependent encoding**. In plaats van voor elk gegevensitem evenveel bits te voorzien (zoals bijvoorbeeld bij ASCII of Unicode), maakt men de lengte van de gebruikte code omgekeerd evenredig met de frequentie van voorkomen van dat item. In een Nederlandse tekst komen bijvoorbeeld de letters e, n en a veel frequenter voor dan x, y en q. Indien men voor de eerste drie een codewoord neemt dat veel korter is dan voor de laatste drie, dan zal men in het algemeen veel plaats sparen. Het enige nadeel is dat men op die manier een code creëert van variabele lengte. Dergelijke codes worden *Huffman*-codes genoemd naar de man die een algoritme heeft opgesteld om frequentieafhankelijke codes te ontwikkelen.

In sommige gevallen bestaat de gegevensstroom uit items die slechts een beetje verschillen van

het vorige item. Een typisch voorbeeld is een film die bestaat uit beeldframes; de verschillende mekaar opvolgende beelden wijken in de meeste gevallen slechts zeer minimaal van mekaar af. De techniek om dit soort gegevens te comprimeren noemt men **relative encoding** of **differential encoding**. Hierbij worden de verschillen tussen de items opgeslagen eerder dan de items zelf. Relatieve codering kan zowel zonder als met gegevensverlies, afhankelijk van het feit of men de verschillen precies of bij benadering noteert.

Een andere compressiemethode heet **dictionary encoding**. Hier verwijst de term "woordenboek" naar een verzameling bouwblokken waaruit de gecomprimeerde boodschap bestaat. In plaats van de blokken zelf op te slaan, noteert men een verwijzing naar de plaats in het "woordenboek". Men zou verwachten dat dit een verliesloze methode is. Evenwel zal blijken uit de discussie over beeldcompressie dat als het "woordenboek" bestaat uit benaderingen van de correcte dataelementen, er toch een zeker informatieverlies optreedt.
Dictionary encoding kan gebruikt worden door tekstverwerkers die kunnen gebruik maken van de ingebouwde woordenboeken gebruikt door de spellingcontrole. Een woord kan dan gecodeerd worden door een verwijzing naar de plaats in het woordenboek eerder dan door de verschillende tekens waaruit het woord bestaat. Bestaat een woordenboek bijvoorbeeld uit 25000 woorden, dan kan elke plaats aangeduid worden door een geheel getal tussen 0 en 24999, wat perfect in 15 bits past. Stel dat men een woord wil opslaan dat bestaat uit 6 letters, dan heeft men daar 6x8=48 bits voor nodig als men de ASCII code zou noteren; gecomprimeerd volgens deze methode heeft men echter slechts 15 bits nodig.

Een variatie op dictionary encoding is **adaptive dictionary encoding** (ook wel **dynamic dictionary encoding** genoemd), waarbij het woordenboek wordt bijgewerkt tijdens het codeerproces. Een populair voorbeeld is de *LZW-codeermethode* (Lempel-Ziv-Welsh, genoemd naar de opstellers van de code). Men start met een eenvoudig woordenboek dat uit elementaire bouwblokken bestaat. Naarmate er echter grotere blokken worden gedetecteerd, worden die aan het woordenboek toegevoegd. Het grote voordeel is dat bij de gecomprimeerde informatie alleen het eenvoudig woordenboek bewaard moet worden; bij de decompressie kan immers het uitgebreide woordenboek opnieuw worden samengesteld. Zonder verder in details te willen treden, bespreken we hieronder een eenvoudig voorbeeld om de werking te verduidelijken.
Veronderstel dat we de boodschap xyx xyx xyx xyx willen coderen met LZW.
Men start met een woordenboek met 3 elementen, namelijk de 3 letters x, y en een spatie. xyx zou gecodeerd worden door 121, wat betekent dat de boodschap begint met een x (eerste woord), gevolgd door een y (woord 2) gevolgd door een x (woord1). Dan wordt de spatie gecodeerd wat een 3 oplevert. De code wordt dus: 1213. Maar omdat een spatie een woord afbakent, kan het patroon xyx opgenomen worden in het woordenboek als vierde element. Dat betekent dat de hele boodschap nu kan gecodeerd worden als 121343434.
Stel dat we deze boodschap willen decoderen vertrekkend van het oorspronkelijke eenvoudige woordenboek met 3 elementen (x, y en spatie). 1213 levert xyx op gevolgd door de spatie. Op dit moment kan evenwel xyx als bouwblok gedetecteerd worden en aan het woordenboek worden toegevoegd als vierde element. Vanaf dat moment kan de 4 in de code omgezet worden door het hele patroon xyx. Op deze manier kan op basis van 43434 de tekst xyx xyx xyx aan de eerst gevonden xyx worden toegevoegd, wat uiteraard in de oorspronkelijke tekst resulteert.

4.7.2 Comprimeren van beelden

Zoals we vroeger hebben besproken, wordt een beeld voorgesteld door een aantal pixels. Wil men dat beeld opslaan in een bestand of over het netwerk versturen, dan is de omvang van cruciaal belang. Past men geen compressie toe dan kan men stellen dat

bestandsgrootte (in bytes) = breedte (in pixels) * hoogte (in pixels) * aantal bytes per pixel

Een niet-gecomprimeerd bestandsformaat is bijvoorbeeld **BMP** (afkorting van bitmap en gebruikt o.a. in Windows). In functie van de kwaliteit van de afbeelding gebruikt men dan voor 1 pixel 1 bit (2 kleuren), 4 bits (16 kleuren), 8 bits (256 kleuren) of true-color (RGB, 8 bits per kleur dus 24 bits of 16777216 kleuren). De bestandsgrootte van een typische afbeelding van 1280x1024 pixels in true-color-BMP-indeling (exclusief header en andere overhead) beslaat dan bijna 4 MB. Hierdoor zijn afbeeldingen in BMP-indeling meestal ongeschikt om te worden verstuurd via het internet.
Omdat beelden onder de vorm van een bitmap dus vaak heel groot uitvallen, kunnen verschillende compressiemethoden worden toegepast om deze omvang te reduceren.

4.7.2.1 GIF en PNG

Het eerste formaat dat we bespreken heet **GIF** (*Graphic Interchange Format*). Het is een dictionary encoding system oorspronkelijk ontwikkeld door CompuServe om grafische gegevens over een netwerk te versturen. Vandaag wordt het ook gebruikt om beelden in een bestand op te slaan.
In plaats van aan een pixel een kleurwaarde te hechten bestaand uit 3 bytes (en wat dus 16 miljoen kleurwaarden toelaat per pixel) beperkt GIF het aantal kleuren tot 256 (8 bits). Op basis van de kleuren in de oorspronkelijke afbeelding, worden er 256 kleuren opgesteld (geselecteerd) die in een palet worden opgeslagen; men kan zich dat voorstellen als een tabel die dan fungeert als “woordenboek”. Elke entry kan dan worden aangeduid door een enkele byte. Indien de oorspronkelijke afbeelding ook 256-kleuren pixels bevat, wordt hierdoor geen winst gehaald. Gaat het om een truecolor afbeelding, dan is deze manier van comprimeren verlieslatend, omdat de kleuren van de oorspronkelijke afbeelding slechts benaderd worden. GIF gebruikt ook LZW-technieken met adaptive dictionary encoding. GIF laat ook toe dat de achtergrond transparant is, waardoor bijvoorbeeld tekst doorheen de figuur kan getoond worden. GIF geeft goede resultaten bij relatief eenvoudige tekeningen zoals bijvoorbeeld cartoons, maar is minder geschikt om foto’s kwalitatief weer te geven.

In 1995 werd het **PNG**-formaat ontwikkeld. PNG staat voor *Portable Network Graphics* en werd gecreëerd als een vervanger voor GIF, gaf betere resultaten en was vooral niet gepatenteerd. Het is op dit moment het meest gebruikte niet-verlieslatende compressieformaat voor het world wide web. Het laat 24 bits per pixel toe en geeft betere compressieresultaten op grote beelden, doch in tegenstelling tot GIF laat het standaard geen geanimeerde beelden toe.

4.7.2.2 JPEG

Een ander populair compressiesysteem voor beelden is **JPEG**. Het is een standaard die ontwikkeld is door de *Joint Photographic Experts Group* (vandaar komt de naam) binnen ISO (International Standard Organisation). Het is een standaard die veel gebruikt wordt binnen de fotoindustrie. Als bewijs hiervoor geldt het feit dat JPEG door de meeste digitale camera's als standaard compressietechniek is ingesteld.

De JPEG-standaard kent verschillende compressieformaten met elk hun specifieke doelstellingen. Hij kent zelfs een verliesloze modus die uiteraard een hoge precisie in het beeld oplevert maar nauwelijks comprimeert. Deze wordt dan ook niet veel gebruikt in vergelijking met de andere modi.

De meest populaire is wat men de baseline standaard noemt, ook wel sequentiële modus. Hij comprimeert in verschillende stappen en profiteert van de beperkingen van het menselijk oog. Meer in het bijzonder is het menselijk oog meer gevoelig voor veranderingen in helderheid, eerder dan voor veranderingen in kleur. Men vertrekt van een beeld dat bestaat uit helderheidscomponenten en kleurcomponenten.
De eerste stap neemt een gemiddelde van de kleurcomponenten van 2 bij 2 pixels. Hierdoor wordt de kleurinformatie met een factor 4 verkleind, zonder dat er aan de helderheidsinformatie geraakt wordt, waardoor er nauwelijks aan zichtbare beeldkwaliteit wordt ingeboet.
De volgende stap verdeelt het beeld in blokken van 8 bij 8 pixels. Elk blok wordt in zijn geheel getransformeerd op zo'n manier dat het oorspronkelijk blok vervangen wordt door een nieuw blok dat weergeeft hoe de pixels zich onderling verhouden eerder dan de pixelwaarden zelf. In dit nieuwe blok worden waarden beneden een bepaalde ondergrens door nullen vervangen, in het idee dat ze te klein zijn om door het menselijk oog te worden waargenomen. Het zou ons evenwel veel te ver leiden om deze techniek in detail uit de doeken te doen.
Vanaf dit punt worden de klassieke compressietechnieken (run-length encoding, relative encoding en variable-length encoding) toegepast om bijkomende verkleining te bekomen.
Dat resulteert voor kleurafbeeldingen in een minimale compressieverhouding van een factor 10:1 zonder merkbaar kwaliteitsverlies. Bij lijntekeningen echter valt dat kwaliteitsverlies wel op; JPEG is hiervoor dan ook minder geschikt.

4.7.2.3 TIFF

Een andere manier om beelden voor te stellen is **TIFF** (*Tagged Image File Format*). Het is niet zozeer populair omwille van zijn compressie-eigenschappen, maar eerder als een standaardformaat om foto's op te slaan samen met gerelateerde informatie zoals datum, tijd en camera-instellingen. De standaard bevat ook compressietechnieken die vooral hun nut kennen in beelden afkomstig van tekstdocumenten. Het feit dat er vele witte pixels zijn maakt dit zeer geschikt voor het toepassen van run-length encoding.

4.7.3 Comprimeren van geluid en video

De meest gangbare standaarden om audio en video te comprimeren werden ontwikkeld door de *Motion Picture Experts Group* (MPEG) onder auspiciën van ISO. De standaarden zelf worden dan ook aangeduid door **MPEG**.
MPEG behelst een groot aantal standaarden voor verschillende toepassingen. Er worden immers andere eisen gesteld aan uitzendingen van hogedefinitie-televisie (HDTV) dan aan een videoconferentie waarvoor de informatie over communicatiekanalen gevoerd wordt met beperkte capaciteiten. Hun beider eisen zijn ook verschillend aan die voor de opslag van een video die verdeeld wordt in secties die opnieuw kunnen gespeeld worden of die kunnen overgeslagen worden.
De technieken die door MPEG gebruikt worden vallen ver buiten het kader van deze nota's. Algemeen kan wel gesteld worden dat een video is samengesteld uit een opeenvolging van beelden, net zoals dat bij een film gebeurt. Om deze sequenties te comprimeren worden slechts bepaalde beelden (*I-frames* genoemd) in hun totaliteit gecodeerd. Voor de beelden tussen deze I-frames worden relatieve codeertechnieken gebruikt waarbij de afwijking t.o.v. het vorige beeld wordt opgeslagen. De I-frames zelf worden meestal gecomprimeerd met technieken die vergelijkbaar zijn met die van JPEG.

Het best gekende systeem om audio te comprimeren is **MP3**. Deze standaard is ook ontwikkeld door MPEG en is in feite een afkorting voor MPEG layer 3. Het "misbruikt" de eigenschappen van het menselijk oor en verwijdert die details uit de muziek die wij niet kunnen waarnemen.
Een van de technieken heet *temporal masking*: voor een korte periode vlak na een luide passage kan het menselijk oor zeer zacht geluid niet waarnemen (wat het normaliter wel zou doen).
Een andere techniek is *frequency masking*: geluid op een bepaalde frequentie verbergt stiller geluid op frequenties die in de buurt liggen.
Deze technieken resulteren in een significante compressie terwijl de kwaliteit van CD-geluid nog steeds benaderd wordt.

Hoofdstuk 5

Computernetwerken

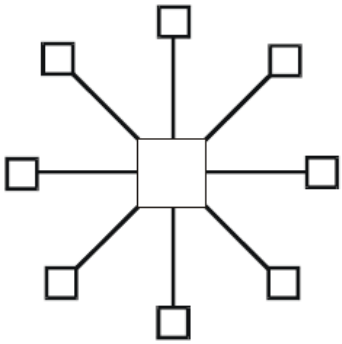
Een belangrijke toepassing van computers bestaat uit datacommunicatie. Hierbij worden verschillende vormen van informatie (teksten, programma's, beelden, geluid) van de ene computer naar de andere gestuurd. Op de computer aan het eindpunt zorgt de nodige programmatuur ervoor dat deze informatie in een passende vorm aan de eindgebruiker wordt voorgeschoteld. In dit hoofdstuk bespreken we enkele elementen van de technologie die hierbij gebruikt wordt.

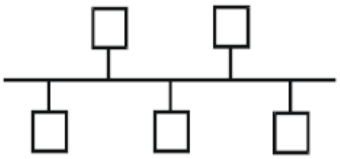
5.1 Soorten netwerken

Netwerkverbindingen bestaan op verschillende niveau's. Zo kunnen we onderscheid maken tussen een lokaal netwerk, een gesegmenteerd netwerk, een internet, hét internet, een intranet en een virtueel privaat netwerk.

5.1.1 Lokaal netwerk

Een lokaal netwerk wordt ook wel een **LAN** (*Local Area Network*) genoemd. Dit kan enkele tot enkele tientallen computers met elkaar verbinden. Bij een dergelijk klein netwerk zijn alle computers rechtstreeks aan elkaar gekoppeld.
Dit kan fysisch op enkele manieren: ofwel zijn alle computers afzonderlijk via een UTP-kabel (*Unshielded Twisted Pair*, een niet afgeschermd paar in mekaar gevlochten geleiders) of draadloos via radiosignalen verbonden met een hub (het Engels voor wielnaaf), ofwel zijn de computers onderling verbonden via een coaxkabel (een centrale geleider met er omheen een isolatielaag en een gevlochten draadmantel die als tweede geleider dienst doet). De laatste technologie wordt niet vaak meer gebruikt voor gewone netwerken, maar is nog steeds de basistechnologie van Telenet om de huizen in een bepaalde wijk met mekaar te linken.
De volgende figuren tonen de logische opbouw van beide mogelijkheden.





Een computer die informatie wil doorsturen naar een andere computer deelt deze op in pakketten. Elk pakket wordt op de kabel verzonden naar alle andere computers, ook degene voor wie de informatie niet bestemd is. Het is evident dat dit een veiligheidsprobleem kan opleveren. Elk datapakket is echter voorzien van een adres dat aangeeft voor welke computer de informatie bestemd is. In een dergelijk netwerk kunnen geen twee pakketten tegelijkertijd verstuurd worden, wat de capaciteit beperkt houdt.

Soms heeft een netwerk ook nog inbelpunten, waarmee een verbinding kan tot stand gebracht worden via een telefoonlijn. Op deze manier kunnen computers die niet fysisch met het netwerk verbonden zijn er toch gebruik van maken.

5.1.2 Een gesegmenteerd netwerk

Als het netwerk groter wordt, moet het in segmenten verdeeld worden. Elk segment is een LAN op zichzelf. Deze segmenten worden met elkaar verbonden door middel van een **repeater** of een **bridge** of een **switch**.
Een repeater versterkt het signaal als de kabel te lang wordt en stuurt elk pakket dat het krijgt naar alle segmenten waarmee het verbonden is.
Een bridge daarentegen verbindt twee segmenten en stuurt een pakket alleen door naar dát segment waar het naar toe moet. Hiertoe onthoudt de bridge de adressen van de apparaten die in een bepaald segment zitten. Hierdoor vergroot de capaciteit, want op verschillende segmenten kunnen verschillende pakketten onderweg zijn.
Tegenwoordig wordt vaak een switch gebuikt. Dit is een bridge met verschillende poorten. Een switch wordt ook vaak ingezet als HUB.

5.1.3 Een internet en hét internet

Verschillende netwerken kunnen met elkaar verbonden worden op uiteenlopende manieren. We beperken ons hier tot de internettechnologie. Een **internet** is een verzameling netwerken die verbonden is door **routers**. Hierbij wordt elk netwerk in de verzameling verbonden met een of

meerdere - maar niet noodzakelijk met alle - andere netwerken van het internet.
Elk toestel op een bepaald internet heeft een uniek internetadres, het **IP-adres**. IP staat hier voor *Internet Protocol*. Wie een bericht naar een toestel wil versturen hoeft enkel dit internetadres te kennen en dit op te nemen in de pakketten die het op het eigen netwerk verstuurt. Als het pakket naar een ander netwerk moet, zorgen de routers ervoor dat dit, eventueel langs andere netwerken, uiteindelijk op het juiste netwerk bij de juiste computer terecht komt.
Zeer dikwijls zijn er verschillende wegen mogelijk tussen twee netwerken. Dit verhoogt de betrouwbaarheid: als er een netwerk of een router wegvalt hindert dit niet noodzakelijk de verbinding tussen andere netwerken.

Hét internet (in het Nederlands met een kleine letter, in het Engels met hoofdletter) wordt soms ook wel aangeduid als **het Net**. Het is een wereldomvattend internet met duizenden routers, die hierarchisch met elkaar verbonden zijn.

5.1.4 Intranet en VPN

Een **intranet** is een internet dat volledig tot één enkele organisatie behoort en afgeschermd is van de buitenwereld. Het is vaak verbonden met het internet door middel van een **firewall**. Dit toestel beveiligt het intranet tegen indringers en laat selectief de nodige informatiepakketten door.

VPN staat voor *virtual private network*. Het is een technologie die toelaat dat verschillende computers op fysisch verschillende locaties met elkaar en met het intranet van het bedrijf zijn verbonden. De communicatie verloopt niet over speciale lijnen maar wel over het onveilige internet. Voor de afgelegen werkposten is het net alsof ze achter de firewall zitten zoals de computers binnen de organisatie. Om dit te realiseren worden de pakketten speciaal beveiligd vooraleer ze verstuurd worden; deze beveiliging wordt aan de andere kant van het internet opnieuw weggenomen en verder verstuurd over het intranet naar de juiste computer.

5.2 Internettoepassing: DNS

In een van de vorige paragrafen hebben we reeds vermeld dat elke computer op een internet een uniek adres moet hebben. Dit is een binaire code die uit 32 bits of 128 bits bestaat, afhankelijk van de versie van IP (IPv4 of IPv6). Dit adres heet het IP-adres.
Bekijken we een adres van versie 4. De 32 bits worden typisch voorgesteld als 4 bytes gescheiden door een punt, waarbij de bytewaarde decimaal wordt weergegeven (0..255): bv. 193.190.173.1

Op het internet is er ook een alfanumerieke codering, die voor mensen veel gebruiksvriendelijker is. Dit zijn de DNS-namen; DNS staat voor *Domain Name System*. Een DNS-naam kan een

domein aanduiden maar ook een individuele computer. Verschillende DNS-namen kunnen ook één en dezelfde computer aanduiden.
Een domein is een logische groep computers. In dit geval zal afhankelijk van de toepassing een vooraf aangeduide computer uit deze groep gepikt worden: zo zal bijvoorbeeld in het geval van elektronische post de mailserver worden aangeduid.

DNS legt de link tussen de naam en het IP-adres. De informatie over de relatie tussen de DNS-naam en het IP-adres wordt evenwel niet op een enkele plaats bewaard maar zit verspreid over het hele internet. We gaan hier niet in op de manier waarop die twee adressen worden gekoppeld. Het is wel belangrijk om te beseffen dat als een of andere toepassing een pakket moet sturen naar een computer op het internet, dat het IP-adres moet gekend zijn. Geeft de gebruiker de DNS-naam op, dan moet de toepassing aan een **DNS-server** via de naam het IP-adres opvragen.

Een DNS-naam bestaat uit meerdere stukken, door een punt van mekaar gescheiden. Het laatste deel in de naam noemt men een top-level-domein; voorbeelden zijn be, nl, uk, eu, com, edu, org, net. Per top-level is er een organisatie die instaat voor de toekenning van domeinnamen binnen dat domein. Voorbeelden van Belgische domeinnamen zijn telenet.be, scarlet.be, belgacom.be, ugent.be, hogent.be. Per domein kan er verder een onderverdeling komen in subdomeinen: binnen ugent zijn dat bv. intec.ugent.be en tiwi.ugent.be. Het eerste deel van een domeinnaam verwijst echter meestal naar een computer: www.ugent.be, www.hogent.be, mail.hogent.be, smtp.telenet.be.