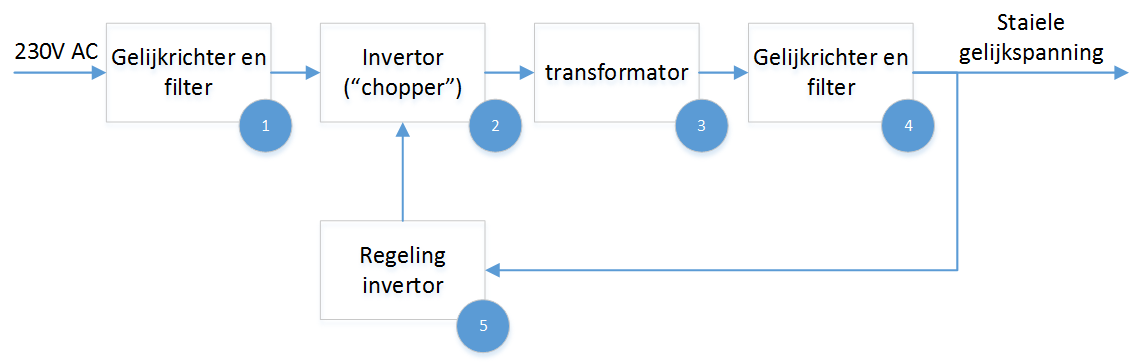
**Examen Computer Architectuur**

* + - * 1. Computervoeding

Switched mode power supply = Schakelende voeding.

* 1. Werking



1. **Gelijkrichter:** Wisselspanning => Gelijkspanning => Filter => Beperkt variatie spanningsniveau
2. **Invertor:** Genereren Blokgolf (Aan en afschakelen ingangsspanning) van 50Hz naar 20+Khz  
     
   **Reden Stap 1 & 2:** Principe frequentie omvormer => Transformator kleiner en efficiënter.  
    Menselijk gehoor neemt geen signalen boven 20Khz waar.
3. **Transformator:** Gewenste niveau spanning
4. **UitangsGelijkrichter en filter:** Omzetting stabiele gelijkspanning
5. **Terugkoppeling naar Invertor:** Om spanning te regelen (tegen spanningsdaling bij belasting), Invertor heeft invloed op amplitude.
   1. Eigenschappen:
      1. Vormfactor en connectoren

|  |  |  |
| --- | --- | --- |
| **Historiek** | **Vroeger** | **NU** |
| **Formfactor** | **AT** | **ATX 2.3** |
| **Aansluiting** | **2x6 aansluitingen** | **24-Polige en 4-Polige 2x12V** |

Nieuw bij ATX: **3.3V spanning** & **+5V stand-by/power on signaal**

* Mechanische schakelaar kan vervangen worden door elektronisch signaal van moederbord.
* Wanneer de computer is uitgeschakeld staat een deel v.h. moederbord nog onder spanning.

**Belangrijke connectoren:**

* ATX power Connector (Moederbord Connector)
* 4 – Pin Connector (Molex): Voor ATA, optische drives
* SATA voedingsconnector
* Auxiliary connectors (Verschillende varianten voor extra voeding & vermogen)
  + 1. Vermogen

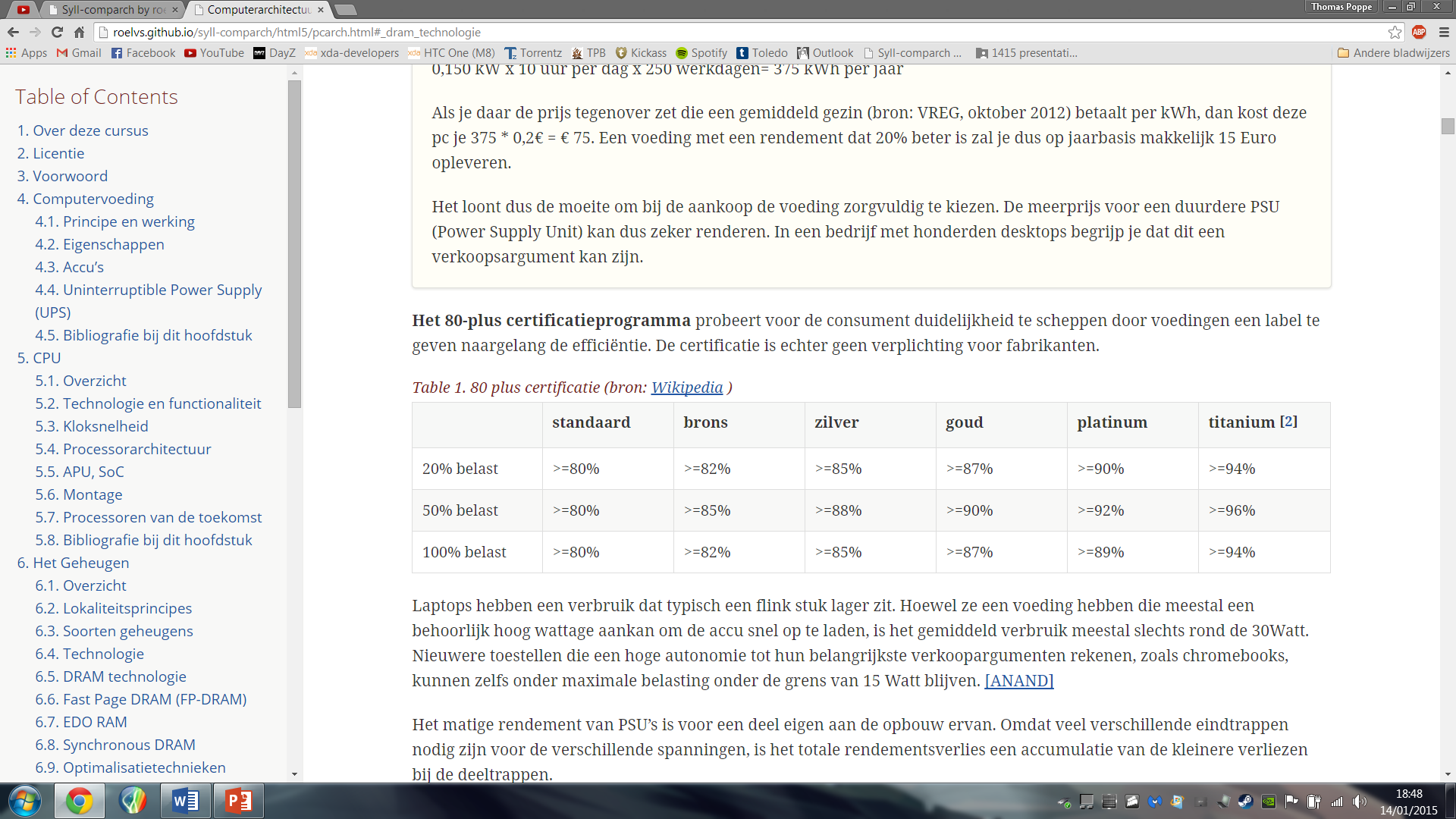
Voedingen komen in verschillende vermogens. Het is belangrijk dat men weet waarvoor het aangegeven vermogen staat. (Piekvermogen, continue vermogen, …)

**TDP:** Thermal Design Power: Warmte productie onder zware belasting gedurende een bepaalde tijd. (Max verbruik)

De voeding heeft verschillende rails voor verschillende spanningen, hierop is telkens de stroom gelimiteerd. Zo kan het zijn dat het vermogen van een rail te laag is.

* + 1. Rendement

Een voeding heeft altijd verliezen door o.a. warmte.  
Wanneer een voeding een goed rendement heeft krijgt deze het 80 plus label.



Het is op te merken dat een voeding een hoog rendement heeft wanneer deze matig belast is.  
Een onbelaste voeding is dus niet goed!

Het is ook belangrijk om op het sluimerverbruik te letten bij een voeding.

Het typisch matige rendement van de PSU is te wijten aan de verschillende eindtrappen voor verschillende spanningen.

* + 1. Geluid

Hoe groter de ventilator hoe minder lawaai, passief gekoeld heeft natuurlijk geen lawaai.

* + 1. Problemen met voedingen

|  |  |
| --- | --- |
| **Oorzaken:** | **Gevolgen** |
| * Te klein vermogen * Onvoldoende koeling * Uitgedroogde Elektrolytische condensatoren | * Onverwacht afsluiten systeem * Beschadiging voeding * Rook… |

* 1. Accu’s
     1. Eigenschappen
        + 1. Capaciteit
* **Ah(ampère/uur):** Gedurende 6 uur 1A leveren.
* **Wh(Watt/uur):** Vb. 5Wh, 5V 🡺 P = U x I 🡺 I = P / U 🡺 5 / 5 = 1 Ah
  + - * 1. Aantal Cellen

Een accu is opgebouwd uit verschillende cellen. (Bv. **Li-Ion accu’s** cellen leveren elk ongeveer 3V.)  
Hoe meer cellen hoe groter de totale capaciteit.

* + - * 1. Laadcurve

Een hogere stroom betekend niet noodzakelijk dat de batterij sneller zal opladen.  
Een juiste en kwalitatieve adapter is belangrijk.

* + - * 1. Memory-effect

Bepaalde types batterijen (vooral NiCd) verliezen capaciteit wanneer je ze halverwege hun ontlaadcyclus terug oplaadt.

* + - * 1. Li-ion accu’s
* Hebben geen Memory-effect & bij gebruik kleine degradatie
* Meest voorkomende bij mobiele toestellen
* Kent degradatie:
  + **Snelle degradatie:** Warme ruimte + Volledig opgeladen
  + **Trage degradatie:** Koele ruimte + Half opgeladen
    - * 1. Toekomst
* LiPo: Lithium polymeer
  + 50% efficiënter dan Li-Ion
* BrandstofCells
  1. UPS: Uninterruptable Power Supply

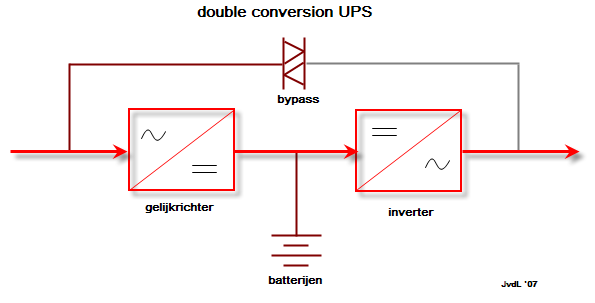
Een UPS vangt het wegvallen van de netspanning op. Het bestaat uit een accu en elektronische schakeling die de accuspanning omzet naar een netspanning.

Een UPS is meestal verbonden met de computers/servers om een melding te geven dat het net is afgevallen. Zo kunnen de computers proper afgesloten worden.

Verder heeft een UPS meestal ook een ingebouwde spanningsbeveiliging die de apparatuur kan beschermen tegen storingen. Er bestaan drie types UPS.

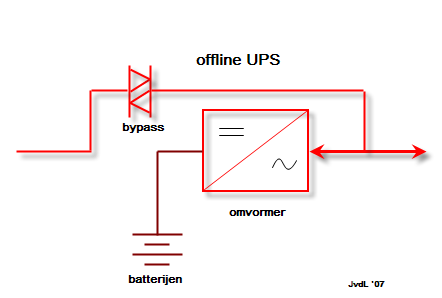
**Belangrijke kenmerken:**

* Vermogen
* Type
* Online-time
  + 1. Online-UPS



Ook wel double conversion UPS genoemd. Alle stroom die naar de apparatuur gaat, loopt door de UPS. Wanneer de stroom afvalt is het dus niet nodig om over te schakelen. Het nadeel is dat de elektronica constant onder belasting is, waardoor deze redelijk duur uitkomt.  
  
Soms wordt er gebruik gemaakt van een bypass, wanneer er onderhoud is voor de UPS.

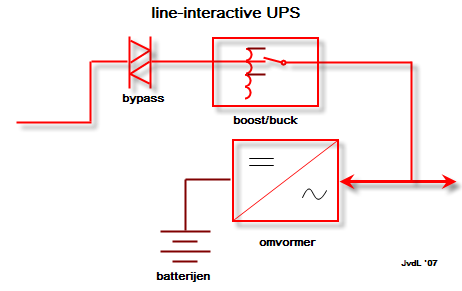
* + 1. Offline-UPS (Standby-type)



Is een goedkoper alternatief dat gebruik maakt van een constante bypass tot de spanning wegvalt. Dan wordt overgeschakeld naar de batterijen waardoor er enkele ms geen uitgangspanning is.

Deze UPS is niet zonder risico te testen en biedt geen beveiliging tegen storingen op het net.

* + 1. Line-Interactive UPS



Deze UPS is een hybride oplossing. Dit is een Offline-UPS waarbij er aanvullende filters zijn bijgeplaatst. Zo is de apparatuur toch beveiligd tegen storingen op het net.

1. CPU
   1. Overzicht

Eerste CPU: x86 8088 (8bits databus) eigenlijk een 8086 (16 bits databus), maar met minder bits want er waren geen andere 16 bits componenten.

* + 1. Wet van Moore

Volgens de wet van Moore zal het aantal transistors in een processor elke 2 jaar verdubbelen. (1965)  
Tot dusver klopt dit nog steeds.  
  
**Vanaf de 80286/80386:**

|  |  |
| --- | --- |
| Real Mode | Protected Mode |
| CPU gedraagt zich als snellere versie van 808  **Compatibiliteitsmodus** | CPU krijgt extra functionaliteit  (**Protected** = Extra geheugenbescherming) |
| **Voorbeelden**   * Opstarten * Compatibiliteit | **Voorbeelden**   * Multitasking * Virtueel Geheugen * Floating Point Unit * Cache geheugen * Extra Instructies |

**Note:** Tick/Tock-model Intel = Tick: Verbeterende Technologie | Tock: Nieuwe functionaliteit.

* 1. Kloksnelheid

De CPU kloksnelheid is enorm toegenomen t.o.v. vroeger, maar **de trend voor alsmaar hoger te gaan is het laatste decennium afgenomen.** Dit komt doordat het grootste probleem bij hoge kloksnelheden **warmte** is. Naast deze te moeten afvoeren zorgt deze ook voor een hoger verbruik.

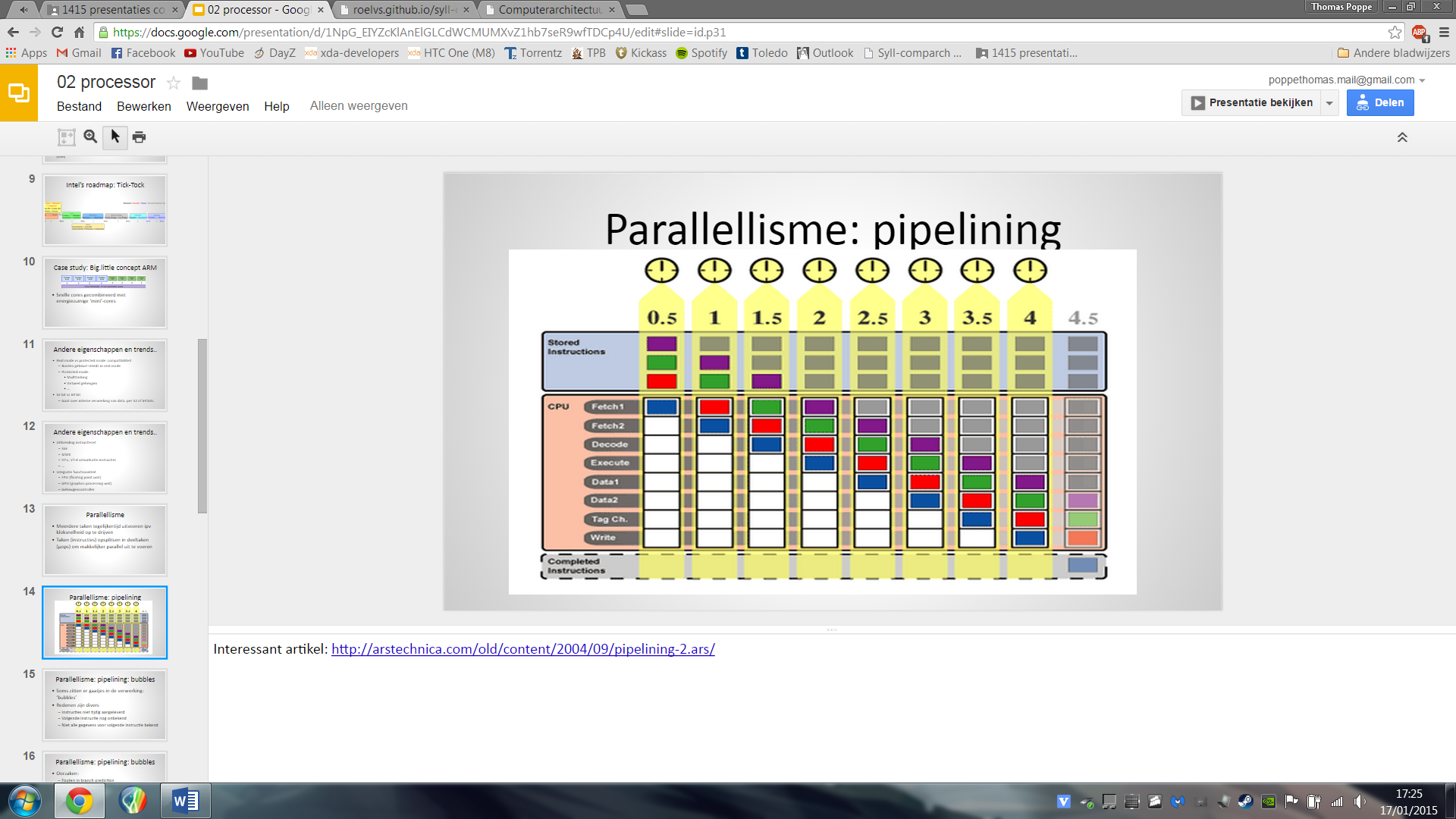
Er zijn wel aanvullende technieken bijgekomen, zoals een dynamische kloksnelheid (CPU throttling & i.f.v. load). Beter gekend als Turboboost (Intel) & speedstep (AMD).

**Verder gaat men voor eerder voor parallellisme:**

* Meerdere taken tegelijk uitvoeren
* Taken (instructies) opsplitten in deeltaken (µops) om makkelijker parallel uit te voeren.
* Pipelining
* Superscalaire processoren
* Hyperthreading
* Multicore
* Multiprocessor
* Computer Clusters
  1. ProcessorArchitectuur
     1. Pipelining

Processoren werken instructies niet na elkaar af maar gedeeltelijk tegelijkertijd. Dit gebeurd in een pipeline. Echter soms is dit niet mogelijk en treed er een tijdspanne op waarbij de processor moet wachten. (= bubble)

**Reden Bubbles:** Fouten in Branch prediction, niet tijdig een instructie, niet alle gegevens



* + 1. Superscalaire processor

Stappen die veel tijd in beslag nemen kunnen dubbel uitgevoerd worden. Er zijn dan verschillende eenheden die berekeningen kunnen maken, waardoor verschillende instructies tegelijkertijd uitgevoerd kunnen worden.

**Problemen:** De software is niet altijd klaar voor een parallelle verwerking.

* + 1. Multi-core Processor

Verdere vorm van een Superscalaire processor, in plaats van een deel van de processor te dubbelen wordt een volledige processor gedubbeld. Het probleem hier is om de caches op elkaar af te stemmen.

Het grootste nadeel is hier dat het voordeel van een Multi-core processor maar naar boven komt, wanneer er meerdere threads gebruikt worden. De software moet dus optimaal geschreven zijn voor een Multi-core of er moeten meerdere programma’s draaien.

* + 1. Cache Geheugen

Neemt alsmaar toe, zeer snel geheugen op de processor zelf.

* **Level 1:** Enorm snel
* **Level 2:** Snel
* **Level 3:** Matig Snel

**Actueel Cache desktop PC:** 8MB

**Gescheiden Instructie en data cache:**

* Omdat I-cache enkel dient te lezen = eenvoudiger
* Rechtstreekse aansluiting op controle logica
* Mogelijkheden: Branch prediction, …
* L1 cache is meestal gescheiden, L2 & L3 unified (omdat verdeling niet flexibel kan gebeuren)
  1. APU = Advanced Processing Unit, SoC = System On Chip

Men probeert steeds meer functionaliteit te verzamelen op een chip, redenen:

* Aantal verschillende chips (= kostprijs) kan teruggedrongen worden
* Alle componenten = geen trage bussen
* Kleinere nodige oppervlakte om het systeem te bouwen

Bij recente processoren zit er vaak een Grafische Chip ingebouwd, dan spreken we van een APU.  
Soms is er ook al een SoC (= CPU, GPU, IO). Hierdoor verkleint de rol van de chipset.

* 1. Montage
* De processor moet compatibel zijn met het moederbord. (Socket!)
* ZIF: Zero Insertion Force (= Zonder kracht de processor plaatsen)
* Mobiele processoren zijn vaak op het moederbord gemonteerd, vervangen is onmogelijk.
  1. Processoren van de toekomst

Alles zal nog gewoon verdergaan, maar op de PC-markt komt ARM als nieuwe speler t.o.v. x86 (Cisc).

* Ontworpen voor toestellen met een laag verbruik
* Succes op de mobiele markt
* Architectuur is in licentie bij meeste chipbakkers
* RISC-architectuur = Van nature efficiënt (Reduced Instruction Set Computing)

1. Het Geheugen
   1. Overzicht

Volgens **Von Neumann architectuur:** Uit te voeren instructies & data samen => Geheugen  
(Volgens **Harvard:** Apart)

* **Snelheid:** Geheugen moet processor voorzien van uit te voeren instructies.
* **Grootte:** Programma’s + te bewerken data zijn enorm toegenomen.

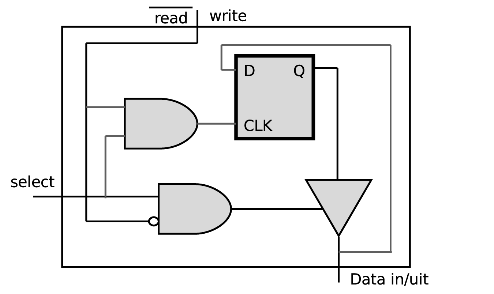
**GeheugenPiramide:** Groote geheugen neemt af, snelheid neemt toe, naarmate je hoger gaat.

* 1. Lokaliteitsprincipes
* **Tijdsgebonden:** Gebruik je iets? Dan gebruik je het binnenkort waarschijnlijk terug.
* **Plaatsgebonden:** Gebruik je iets? Dan gebruik je iets uit de buurt waarschijnlijk ook.
  1. Soorten Geheugens
     1. Behuizing

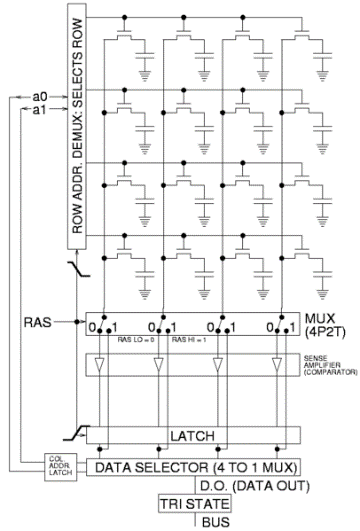
**Geheugen Module:** Vroeger waren er discrete chips op het moederbord geplaatst, dit werd echter te duur en toen schakelde men over naar geheugenmodules.

* Bepaald aantal contactpunten
* Bepaalde werkspanning
* Soort geheugenchips
* **SIMM:** Single Inline Memory Module: Contactpunten voor en achter met elkaar verbonden
* **DIMM:** Dual Inline Memory Module: Contactpunten zijn afzonderlijk (168 of 184 pins)
* **so-DIMM:** Small outline: Speciaal voor laptops

**Geheugenbanken:** Een of meerdere sockets of geheugenvoeten. Het aantal sockets per geheugenbank hangt af van de breedte van de databus. (en het geheugen…).

* 1. Technologie
     1. Statisch

Opgebouwd uit actieve geheugencellen (flipflop schakelingen).

* Complex => Duur
* Zeer snel
* Verliest waarde bij stroomuitval
* Gebruik: Snelle cache geheugens
  + 1. Dynamisch

Opgebouwd uit condensatoren.

* **Nadelen:**
  + Leesoperatie ontlaad de condensator
  + Geheugen gaat verloren => Moet ververst worden
  + Trager t.o.v. Statisch
  + Kan geen stroom sturen
  + Verliest waarde bij stroomuitval
* **Voordelen:**
  + Goedkoper
  + Gebruik: Ram, Dram
  1. DRAM technologie
     1. Gemultiplexte adresklemmen

Dynamische RAMs hebben een grote densiteit en capaciteit. Hierdoor zijn er veel adressignalen nodig om een bepaalde geheugencel te selecteren. Daarom wordt dit opgesplitst in een **Column Address** en een **Row Address.**  
  
Deze worden na elkaar aangeboden aan het IC: RAS- en CAS-klem om de twee adresgedeelten te latchen. De snelheid wordt bepaald door de toegangstijd Tacc.

* + 1. Destructieve Leescyclus

Om dit op te vangen zal er op de geheugenchips zelf een rij van statische geheugencellen aanwezig zijn. Wanneer een rij-adres wordt aangelegd wordt de dynamische rij gekopieerd naar de statische rij. Nadat de gewenste cel is gelezen wordt de inhoud terug naar de matrix gekopieerd.

* + 1. Refresh

Op regelmatige tijdstippen zal een RAS-only cyclus uitgevoerd worden. Een hele rij wordt gekopieerd naar de statische rij en dan terug weggeschreven.

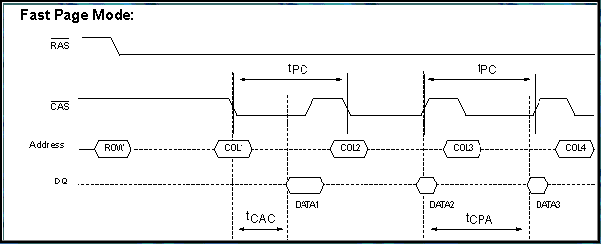
* + 1. Bandbreedte bij DRAM

De bandbreedte is het eigenlijke geheugendebiet. DRAM werkt met cyclussen en we komen dus op volgende formule:

**Getransfereerde bytes per cyclus / tijdsduur cyclus**

* 1. Fast Page DRAM (FP-DRAM)

Door gemultiplexte adresklemmen zijn er minder adresklemmen nodig, maar de lees – en schrijfcyclus wordt langer. FP-DRAM verbeterd dit door cycli te combineren.



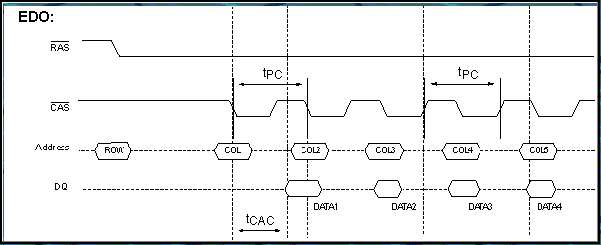
Opeenvolgende cycli gaan meestal door bij naburige cyclusadressen. Wanneer een byte gelezen wordt is de kans groot dat op dezelfde rij er nog een gelezen wordt. FP-DRAM zal dus eenmaal een rij-adress opgeven, en hierna verschillende kolommen lezen en schrijven alvorens een nieuwe rij te selecteren.

**Praktijk:** Burst van 4 leescycli met aanduiding hoeveel klokcycli er nodig zijn per transfert. Bv. 5-3-3-3  
FP-DRAM werd gebruikt met bus frequenties tot 66Mhz

**Rekenvoorbeeld:** 486 processor (32-bit databus (32/8 = 4)) |FP-DRAM 5-3-3-3 (= 14) van 66Mhz.   
  
 = 75MB/s

* 1. Extended Data Out RAM (EDO RAM)

Aanpassing van FP, waarbij de memorycontroller moest wachten met het aanbieden van een nieuw kolom adres tot de vorige data volledig gelezen waren. Bij EDO RAM blijven de data op de uitgangen van het geheugen beschikbaar tot na het aanbieden van het volgend kolomadres. **Hierdoor kan men terwijl de data gelezen wordt al een volgend kolomadres aanleggen.**



EDO-RAM kon gebruikt worden met een databus tot 75Mhz met een timing van 5-2-2-2 klokcycli.

**Rekenvoorbeeld:** 64-bit databus (64/8 = 8)) |EDO-RAM 5-2-2-2 (= 11) van 75Mhz.  
  
 = 218MB/s

* 1. Synchronous DRAM (SDRAM)

SDRAM gaat nog verder, niet alleen naburige kolommen zullen uitgelezen worden, maar opeenvolgende kolommen! Bij elke klokpuls wordt dan een kolom naar buiten gebracht.  
  
SDRAM is geschikt voor een databus tot 133Mhz en een timing van 2-1-1-1 klokcycli.

**Rekenvoorbeeld:** 64-bit databus (64/8 = 8)) |SDRAM 2-1-1-1 (= 5) van 133Mhz.  
  
 = 851MB/s

* + 1. DDR SDRAM – DDR2 – DDR3 (Double Data Rate)

DDR werkt op dezelfde manier als SDRAM, maar verschilt bij het tempo. Bij Double Data Rate wordt date naar buiten gebracht op stijgende & dalende flank van de klok. Om dit te kunnen bereiken wordt gebruik gemaakt van een prefetch buffer. In elke cyclus worden 2 bits getransfereerd naar de prefetch buffer.

**Voordeel:** Hogere maximale bandbreedte  
**Nadeel:** Hogere latentietijd (Tussen aanleggen adressen <> buiten brengen data)  
  
**Aanduiding snelheid heeft twee vormen, maar zijn gelijk:**

* DDR-400 = 2x200Mhz (Kloksnelheid)
* PC3200 = 3200 MB/s (Maximale transfersnelheid)

Tussen de modules kan er echter nog veel verschil zijn d.m.v. de totale latentietijd.  
Deze wordt opgegeven door:

* TCL= CAS Latency Time: Tijd tussen Cas en beschikbaar worden van data
* Trcd = DRAM RAS to CAS Delay: tijd tussen RAS en CAS (ook tijd tussen active en read/write-commando )
* Trp = DRAM RAS Precharge: tijd tussen selecteren van twee rijen
* Tras = Precharge delay: minimale tijd tussen actief worden en precharge van volgende rij.

**Rekenvoorbeeld:** PC3200 geheugen met parameters (timing) 2-2-2-6 , 4 transfers van 8 bytes

Dit geeft een snelheid van 1067 MB/s. Voor twee dergelijke opeenvolgende transfers zijn:

Dit geeft een snelheid van 914 MB/s.

Bij DDR kunnen het aantal cellen in een burst variëren. Als er zo inplaats van 2 burst, 1 dubbel zo grote burst gedaan wordt, kan Tras (min = Trcd + Tcl + 1) vermeden worden.

**DDR2 & DDR3:**

* Hogere kloksnelheid
* Lagere werkspanning (kloksnelheid = Meer warmte <> minder Warmte = Lagere spanning)
* Grotere Prefetchbuffer (4bits & 8bits)

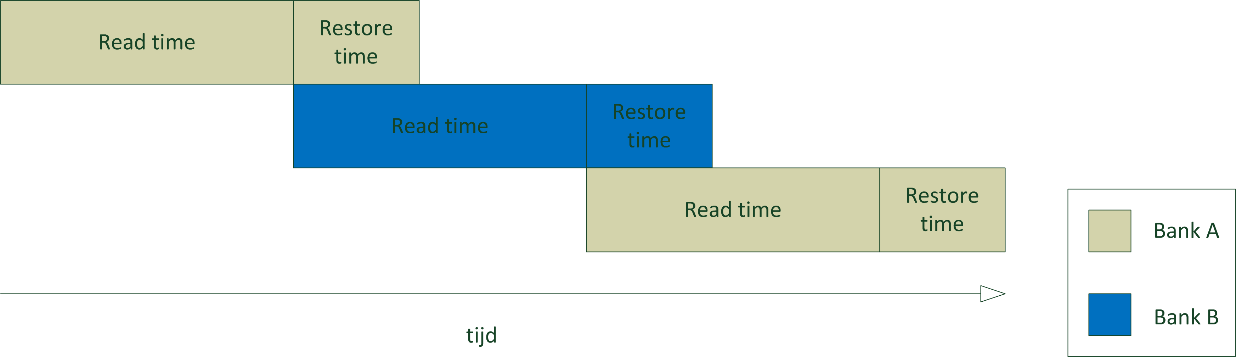
**DDR4:**

* Hogere klokfrequenties
* Geoptimaliseerde signalering
* 3D geheugen (Meer lagen geheugen op elkaar in chip)
  1. Optimalisatie Technieken

Bij de evolutie om de bandbreedte alsmaar te doen toenemen doet het probleem zich voor dat de latentie tijd ook toeneemt. Na elke rijtoegang is er een hersteltijd nodig om de buffer vrij te maken.

* + 1. Interleaving

Met interleaving wordt de dode tijd tussen twee rijen vermeden, dit door de volgende operatie te laten doorgaan op een andere geheugenmodule. Om dit te bereiken worden geheugenblokken verdeeld over verschillende geheugenbanken, die onafhankelijk van elkaar aangesproken kunnen worden. Het is belangrijk dat de memory controller weet of er gewisseld kan worden tussen banken.



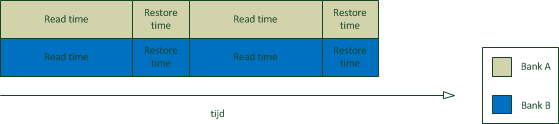
* + 1. Dual Channel

Een andere techniek om de snelheid te vergroten is door het vergroten van de databus.

* Extra aansluitingen op geheugenmodule en moederbord (+ signaal lijnen)
* **Probleem:** Tijdverschil tussen verschillende datalijnen

**Dual Channel** is een techniek die dit doet zonder de databus te vergroten. Dit gebeurt weer door verschillende geheugenbanken te gebruiken. Elke geheugenbank heeft een databus van 64 bit => 128bit databus naar memory controller. De DIMM sockets op een moederbord zijn dus fysiek verbonden met één van de twee 64 bit kanalen.

In theorie neemt de bandbreedte met 100% toe, in praktijk is dit echter maar 10%.  
Wanneer er een grote hoeveelheid cache geheugen aanwezig is, is dit zelfs nog minder.



Hierdoor is het belangrijk dat geheugenmodules in de juiste sockets gestoken worden. (Zelfde kleur)  
Verder moeten beide modules identiek zijn in capaciteit.

* + 1. Buffered/registered RAM en foutdetectie

Dit betekend dat de geheugenmodule een buffer chip bevat, zoals bij de statische bufferrij.  
Deze doet dienst als extra buffer/versterker tussen de geheugen IC’s en de rest van het systeem.

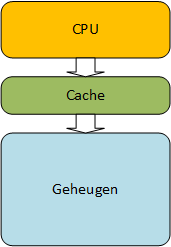
**Andere termen:**

* **Parity:** Per byte wordt een pariteit bit berekend. De ontvanger berekent deze opnieuw en vergelijkt deze. Als ze niet overeenkomt moet de transfer opnieuw gebeuren.
  + Enkel detectie enkelvoudige bitfouten
* **ECC:** Hash functie
  + Meervoudige bitfouten detectie of enkelvoudige bitfouten correctie
  + Meer rekenkracht en tijd nodig
  1. Cache Geheugen

Een processor kan instructies niet sneller afwerken dan ze door het geheugen aangeboden worden.  
Sneller geheugen is dus nodig. We weten dat statisch geheugen veel sneller is dan dynamisch geheugen, dit is echter wel duurder per byte. Het komt er op neer deze zeer efficiënt te gebruiken.

De snelheid van de interface naar het geheugen vormt ook een probleem, om dit op te lossen was er vroeger een snelle back-side bus. Tegenwoordig zijn de verschillende niveaus van cache geheugen in de processor geïntegreerd.

* + 1. Werking

Het cache geheugen vormt een buffer die het snelheidsverschil tussen het geheugen en de CPU moet opvangen. Het komt er op neer de nodige gegevens klaar te hebben zitten in het cache geheugen.

Om dit te realiseren gebruiken we weer het lokaliteitsprincipe:

* **De Cachelines kunnen een kopie bevatten van een blok uit het hoofdgeheugen.**

**Verloop leescyclus:**

* Processor vraagt adres
* **Adres zit in een blok van cache:** Cache-hit => snelle leescyclus
* **Adres zit niet in een blok van cache:** Cache-miss => trage leescyclus + blok hoofdgeheugen wordt naar cachegeheugen gekopieerd.

**Mogelijkheden schrijfcyclus:**

* **Write-back cache:** Inhoud cache geheugen wordt aangepast, bij verwijderen cacheline wordt dit teruggeschreven naar het hoofdgeheugen.
  + **Nadeel:** Cache verschillend van hoofdgeheugen => Problemen bij DMA
* **Write through cache:**
  + **Nadeel:** Gebruik trager hoofdgeheugen
* **Write-allocate:** Indien cache miss => Geheugenblok in cache laden dan schrijven
* **Write-No-allocate:** Indien cache miss => Geheugenblok wordt direct bewerkt
  + 1. Soorten caches

Er zijn verschillende “levels” cache geheugen. Level 1 cache staat het dichtst bij de processor en is dan ook het snelste geheugen. Level 2 cache is trager dan Level 1, omdat dit niet rechtstreeks moet voorzien van gegevens. Dit is echter wel een stuk groter.

Dikwijls wordt L1 cache geheugen opgedeeld in data cache en instructie cache.

* **Pipelining:** Instruction fetch unit van de processor haalt instructies op en tegelijkertijd zal de operand fetch gegevens ophalen om de bewerking mee uit te voeren.
* **Optimalisatie:** Onafhankelijk van elkaar operaties uitvoeren + Instructies vragen geen wijzigingen. M.a.w. heeft geen voorziening nodig voor schrijfoperaties.
  + 1. Overschrijfstrategieën

Bij een cache mis zal er een cacheline vervangen worden door een geheugenblok. De cacheline die het langst niet gebruikt zal worden wordt verwijdert en deze wordt bepaald door verschillende strategieën:

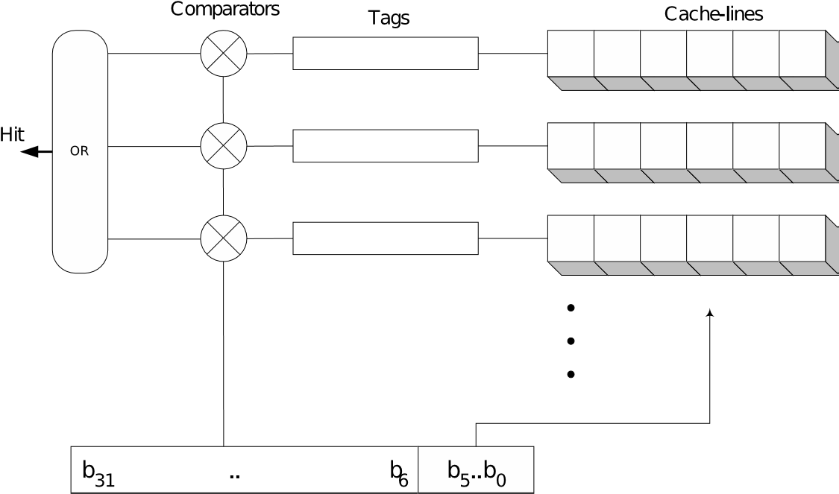
* **First In First Out (FIFO):** Weinig Efficiënt, eenvoudig, de oudste lijn wordt verwijderd.
* **Least Recently Used (LRU):** Langst niet gebruikte lijn wordt verwijderd. Duur om te implementeren en tijdrovend om uit te voeren.
* **Least Frequently Used(LFU):** Minst frequent gebruikte lijn verwijderen. Zelfde nadelen LRU.
* **Adaptive Replacement Cache (ARC):** Combinatie van LRU & LFU.
* **Random:** Eenvoudig en matige resultaten. Soms gecombineerd met LRU. Als een lijn gebruikt is geeft een bit dit aan. Wanneer alle lijnen gebruikt zijn, worden deze gereset.
  1. Associativiteit

De associativiteit bepaalt op welke cachlines bepaalde geheugenblokken terecht kunnen komen.  
Het bepaalt ook het aantal lijnen die geselecteerd kunnen worden en welke overschrijfstrategie toegepast wordt. Het heeft een grote invloed op de hitrate.

* + 1. Fully Associative cache

Een blok uit het geheugen kan in gelijk welke cacheline terecht komen. Aan elke cacheline wordt een tag gekoppeld. Deze is nodig om te weten welke blok in de cacheline aanwezig is en om eventueel de juiste te kunnen selecteren. Het grote nadeel hier is dat de tags een deel van het cache geheugen innemen.

Er zijn comperatoren voorzien naast de cachelines en de tags, bij het begin van een cyclus vergelijken deze de bovenste 26 bits van het adres met de tags. Er is dan een Cache-hit of Cache-miss.



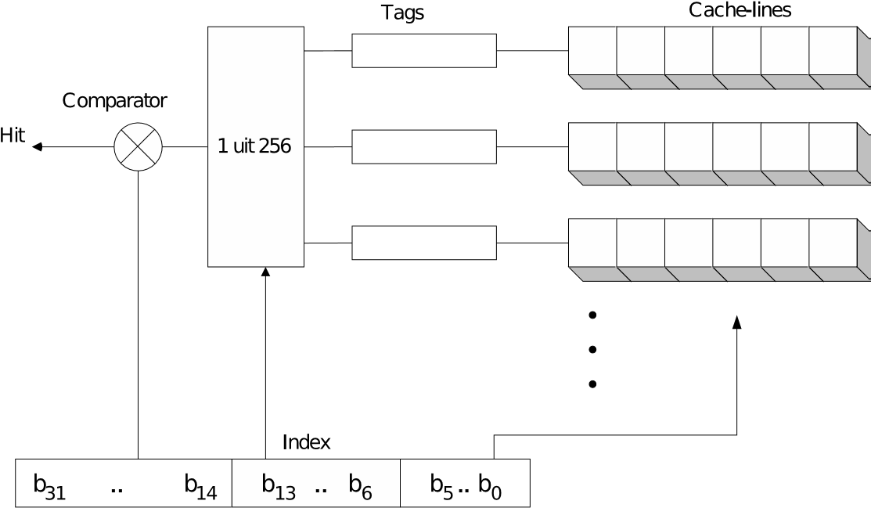
**Voordeel:**

* Bij cache hit, is de juiste cacheline direct geselecteerd.
* Elke pagina uit het geheugen kan in elke cacheline geplaatst worden => optimaal benut
* Optimale overschrijf strategie kan gebruikt worden

**Nadeel:**

* Tags nemen een deel van het cache geheugen in
* Complex: Comparator moet binnen klokcyclus werken (Hoog aantal bits)
* Duur
  + 1. Direct Mapped Cache

Elk geheugenblok kan slechts in 1 cacheline terecht komen. Dit wil zeggen dat overschrijfstrategieën overbodig zijn.



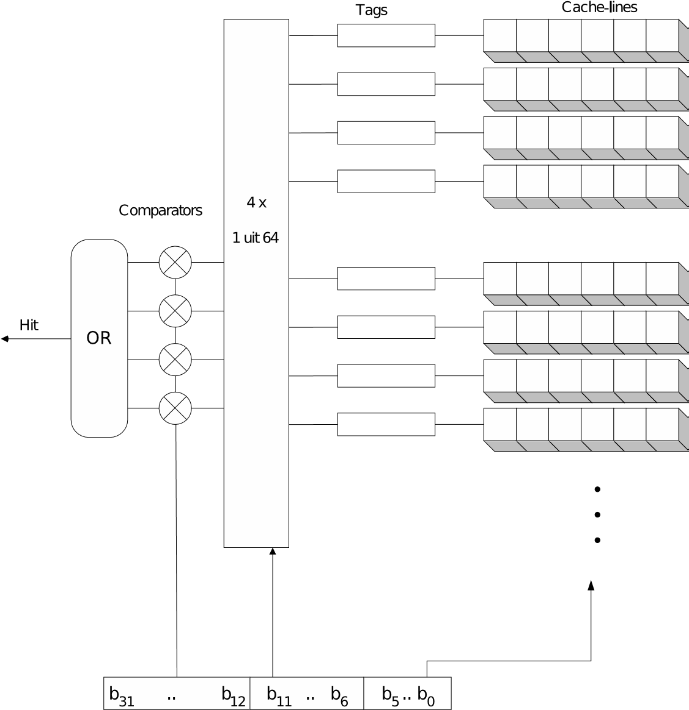
**Voordeel:**

* Geen overschrijfstrategieën nodig
* Kleinere tags
* 1 simpelere comparator.
* Goedkoper

**Nadeel:**

* Negatief effect hitrate.
* Mogelijkheid dat cache vrijgemaakt wordt wanneer er nog ongebruikte plaats is.
  + 1. Set-Associative cache

Tussenvorm van Direct Mapped en Associative cache. Alle cachelines worden gegroepeerd in sets. Elke geheugenblok is toegewezen tot een set, maar binnen die set kan het in elke cacheline terecht.  
Elke set heeft een bepaald aantal –way. Bijvoorbeeld 4-way, heeft 4 cachelines in elke set.



Fully associative cache is eigenlijk set-associative met maar 1 set, direct-mapped is eigenlijk set-associative met slechts 1 lijn per set.

**Voordeel:**

* Tussen beide soorten in
* Doenbaar qua prijs en complexiteit
* Door beperking aantal cachelines in een set valt het zeer goed te combineren met een overschrijfstrategie.
  + 1. Snelheid van de cache

De snelheid van het cache geheugen wordt door een aantal factors bepaald. Zo is er eerst de hitrate. We zien dat de hitrate bepaald wordt door:

* Grote van het cachegeheugen
* Associativiteit
* Aantal Coldstart misses (Onvermijdbare misses, bij eerste keer toegang tot een blok)

Verder spelen deze factoren een rol in de snelheid:

* **Hit Time:** Tijd om bij een hit de gegevens op te halen
* **Miss time:** Tijd om bij een miss een nieuwe geheugenblok te laden en de gevraagde gegevens beschikbaar te maken.
* **Miss penalty:** extra tijd nodig bij een cache miss (tijd tussen miss time en hit time)
* **Hit rate:** Percentage toegangen die rechtstreeks lang de cache gaan
* Grotere cachelines: Geven eerst een betere hitrate, omdat meer naburige gegevens kunnen gekopieerd worden. Maar de miss time wordt groter. (Wanneer heel groot, daalt hitrate)

**Formule:**

**Voorbeeld Oefening:** L1: t=1ns HR=96% | L2: t=2ns HR=88% | Externe: t=8ns

* Zonder L2 geheugen zou: Het ram geheugen meer invloed hebben
* Als de toegangstijd tot het ramgeheugen vergroot heeft dit klein effect op de tijd
  1. Virtueel geheugen

Is de oplossing om een tekort aan fysiek geheugen op te lossen.

* + 1. Werking

Virtueel geheugen zal gebruik maken van beschikbare ruimte op een ander medium voor opslag, om het geheugen groter te laten lijken.  
  
Op dat medium zal ruimte voorzien worden als swapruimte. Het kan een swap-file of swap-partitie zijn. Wanneer er gegevens uit het geheugen wordt opgevraagd, maar deze blijken niet in het fysieke geheugen te zitten dan treed er een page fault op. De processor wordt onderbroken en zal een exceptieroutine uitvoeren. De exception handler zal plaats maken en het stuk uit de pagefile verplaatsen naar het fysieke geheugen.

Het is belangrijk dat bepaalde delen nooit uit het fysieke geheugen gaan, zoals de handler. Anders is er geen mogelijkheid meer om stukken naar het fysieke geheugen te verplaatsen.

* + 1. Paging – segmenting

**Paging:**

De meest gebruikte manier om te werken met virtueel geheugen is door zowel de swap als het fysieke geheugen te verdelen in stukken van dezelfde grote. Zo’n blok wordt een pagina genoemd.

Tabel: Elke pagina is opgeslagen in een tabel, waar een bit aangeeft of het fysiek of virtueel aanwezig is en andere bits duiden de locatie aan. Deze tabel is altijd in het fysieke geheugen beschikbaar. Door de tabellen in meerdere niveaus te steken is het mogelijk de tabel klein te houden. De andere niveaus kunnen zich dan in het swap geheugen bevinden.

* Interne fragmentatie: Bestand wordt verdeeld over pagina’s, maar de laatste pagina is niet volledig gevuld. Zo is er ongebruikte ruimte.

**Segmenting:**

Hierbij worden de gegevens opgedeeld in segmenten die verschillen van grootte, hierdoor krijgt men te make met geheugen fragmentatie. Bij het vervangen van de geheugenpagina’s is er nu extra tijd nodig. Ook hier wordt er in een tabel aangeduid waar alles zich bevindt.

* Externe Fragmentatie: Stuk wordt vervangen door een kleiner stuk, er is nu lege ruimte.
  + 1. Snelheid en virtueel geheugen

Virtueel geheugen heeft de reputatie de computer te vertragen, dit is echter eigenlijk geen waar aangezien het de mogelijkheid biedt om verder te werken zonder afsluiten van andere programma’s.

**Translation Lookaside Buffer:** Snel cachegeheugen in de processor dat de meeste fysieke adressen opslaat. Lost probleem op dat bij gebruik van fysiek geheugen er 3 geheugentoegangen nodig zijn.

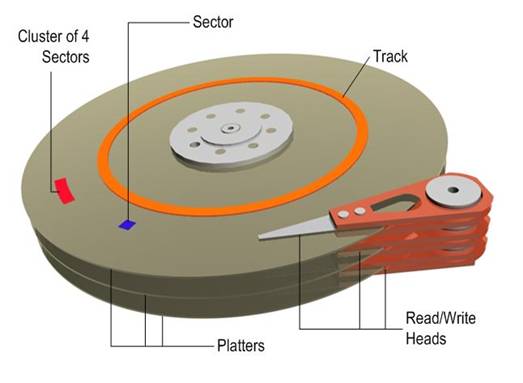
Vaak probeert een besturingssysteem een voorbereiding te treffen om een toekomstige swap-operatie te versnellen. Zo zal het op de achtergrond ongebruikte pagina’s zoeken die niet gebruikt zijn en naar het swap verplaatsen. Hierdoor werkt de PC weer even iets trager.

1. Opslagmedia
   1. HardeSchijf
      1. Fysieke opbouw

**Platters:** Aluminium schijven (Licht en stijf, maar niet magnetisch). Hierop is een dun laagje magnetisch materiaal aangebracht.

**Schijfkop:** Boven – en meestal ook onder – de platter zweeft er een schijfkop. Doordat de Platters ronddraaien bewegen de schijfkoppen over het oppervlak. Het is belangrijk dat de lucht tussen de platters en de schijfkop zuiver is, daarom worden deze bij de productie luchtdicht verpakt.

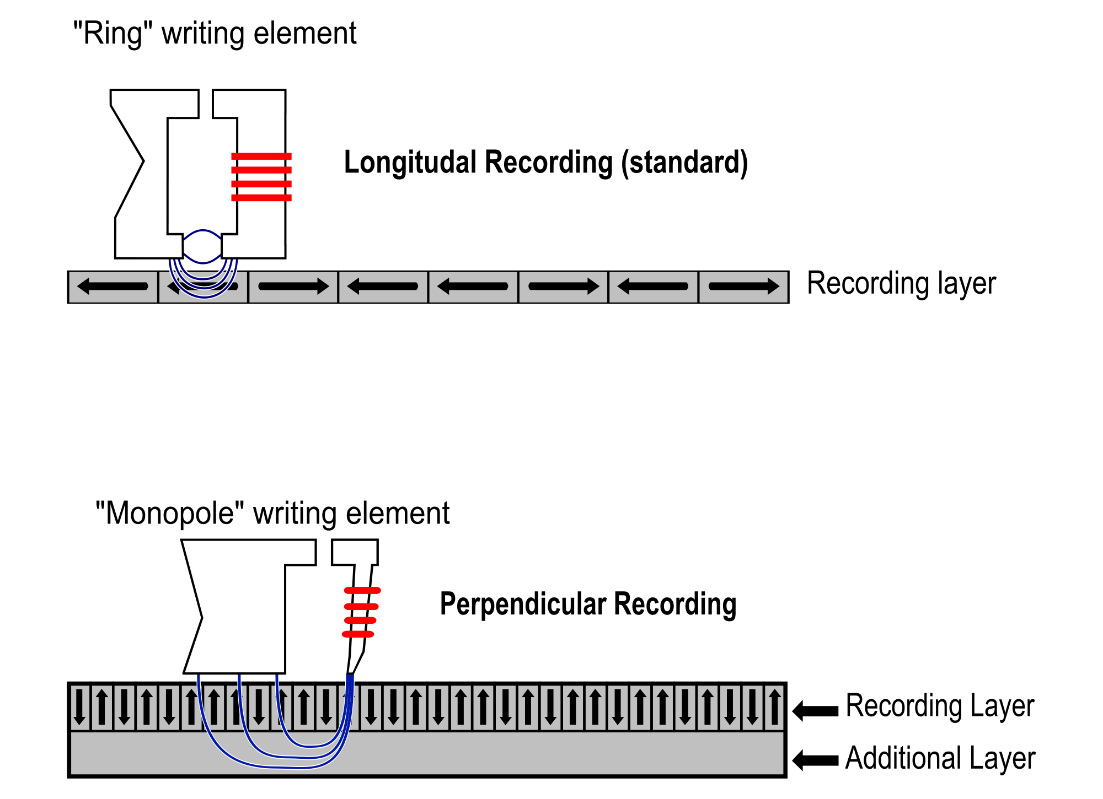
**Werking:**

* Positieve stroom door kop: Magnetische deeltjes op schijf georiënteerd in bepaalde zin
* Negatieve stroom door kop: Magnetische deeltjes georiënteerd in omgekeerde zin
* Kop beweegt over schijfopp.: Er wordt een positieve of negatieve stroom geïnduceerd.

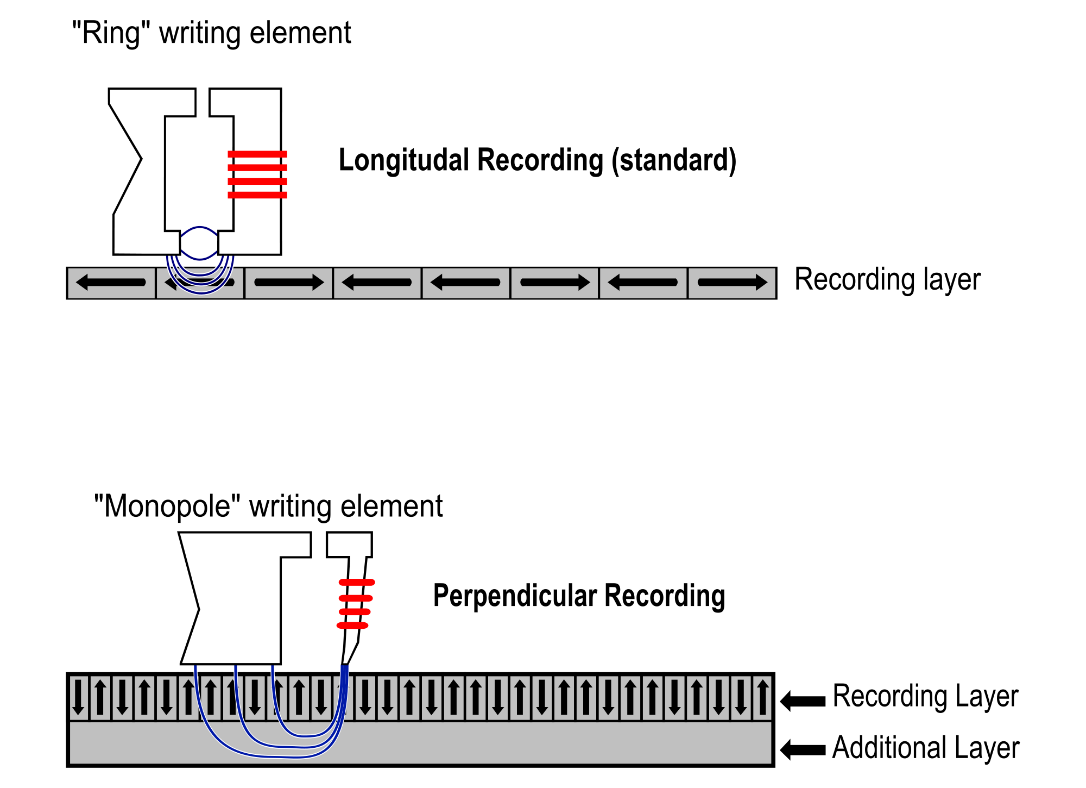
**Sector:** Kleinste hoeveelheid data die bestaat uit een bepaald aantal bits die de kop in één keer kan verwerken. Begint met een preamble (patroon om begin te herkennen), gevolgd door een aantal databits. (Meestal 4096 of 512 bytes). Hierna volgt een fout corrigerende code, waarmee databits kunnen hersteld worden. Na een sector volgt een lege tussenruimte.

Deze indeling gebeurt tijdens het formateren, er gaat dus een groot stuk ruimte verloren aan tussenruimtes, preambles en ECC-bits.  
De schijf kan zo met 15% capaciteit afnemen.  
  
De bitdichtheid op een spoort van de platter is tevens hoger dan die in de axiale richting.

**Tangentiële Richting:** Bij magnetisatie worden dipool moleculen in een bepaalde richting georiënteerd. De kleinste groep moleculen die gemagnetiseerd worden is een magnetisch domein, hierdoor kan de bitdichtheid met deze techniek niet meer toenemen.

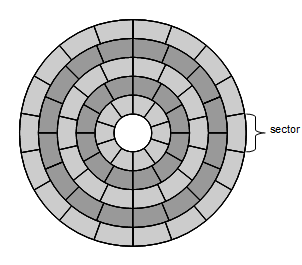


**Perpendiculair Recording:** Een nieuwe techniek slaag er echter wel in te magnetiseren in diepte.



* + - 1. Toegangstijd

De snelheid is afhankelijk van verschillende factoren:

* **Seek:** Schijfkop moet boven juiste spoor gebracht worden + in radiale richting gepositioneerd worden. (= Track-To-Track Seek time (~1ms) & Avarage Seek time (~10ms))
* **Average Latency:** Gemiddelde tijd nodig om de kop boven de gewenste locatie te laten komen. ~Tijd voor halve toer. Afhankelijk van rotatiesnelheid. (Snelheden: 5400, 7200, 10000 of 15000 toeren/minuut).
* **Overdrachtssnelheid:** Moderne schijf gemiddeld 60Mb/s. Seeks time en average Latency domineren de tijd.
  + - 1. Invloed Fysieke geometrie op de performantie
* **Snelheid schijf:** **Hoger:** Meer batterijverbruik, kleinere average latency.
* **Aantal Platters:** **Meer:** Meer beschikbare gegevens zonder radiale verplaatsing, sterkere motoren nodig, trager op gang komen schijven, meer trillingen, meer lawaai.
* **Opdeling in zones:**
  + Cilinder aan de buitenkant van de schijf hebben een grotere omtrek dan aan de binnenkant van de schijf.
  + Schijf is in zones verdeeld, zodat er meer sectoren liggen aan de buitenkant van de schijf.
    - * 1. Schijfcontroller
* Stuurt de motoren en leeskoppen aan
* Verzorgt communicatie met de buitenwereld
  + Omzetting LBA adressen naar fysieke kop, cilinder en sector.
  + Houdt mapping bij van slechte sectoren (op elk spoor zijn er reserve sectoren)
  + Sectoren bufferen in cache,, zodat operaties sneller verlopen
    1. Adressering

**Originele adressering:**

* **Werking:** CHS: Cillinder (10 bits), Head (4 bits), Sector (6 bits)
  + **Voordelen:**
    - Bij opeenvolgende locaties kan alles met minimale latency verwerkt worden.
  + **Nadelen:**
    - Slechts 20 bits beschikbaar voor adres sector.
    - Door fout in bios routines: Start nummering sector bij 1 i.p.v. 0.
    - Maximaal 504Mb op schijf kon worden aangesproken hierdoor.

**Huidige adressering:**

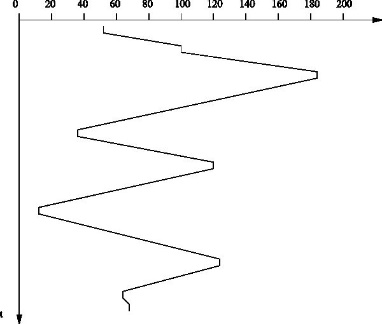
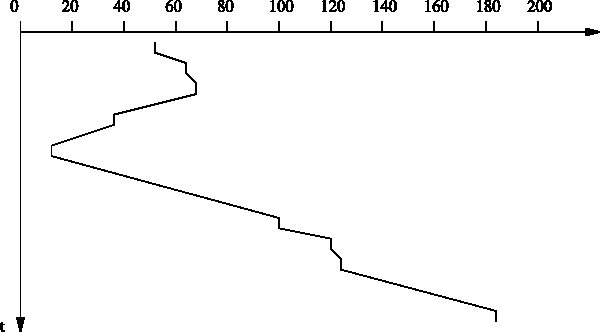
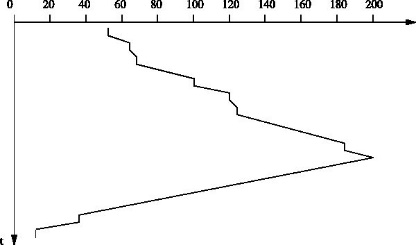
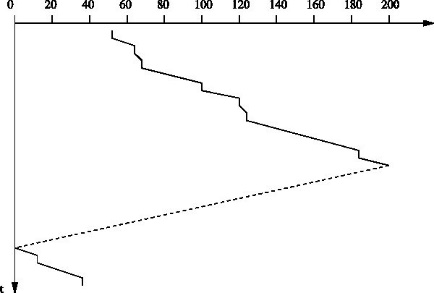
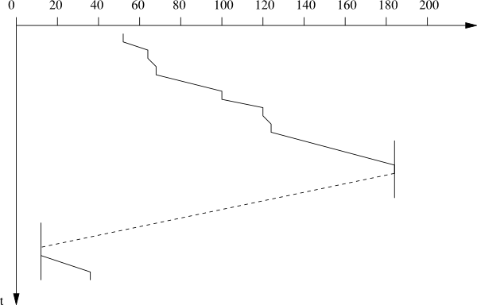
* **Werking:** LBA: Logical Block Adressing, elke sector krijgt een volgnummer
  + **Voordelen:**
    - Op schijf is er een omzetting naar fysiek CHS adres en enkel daar gebruikt
    - Adressering van 48 bits => 128 PB (Petabyte) mogelijk
    1. Scheduling

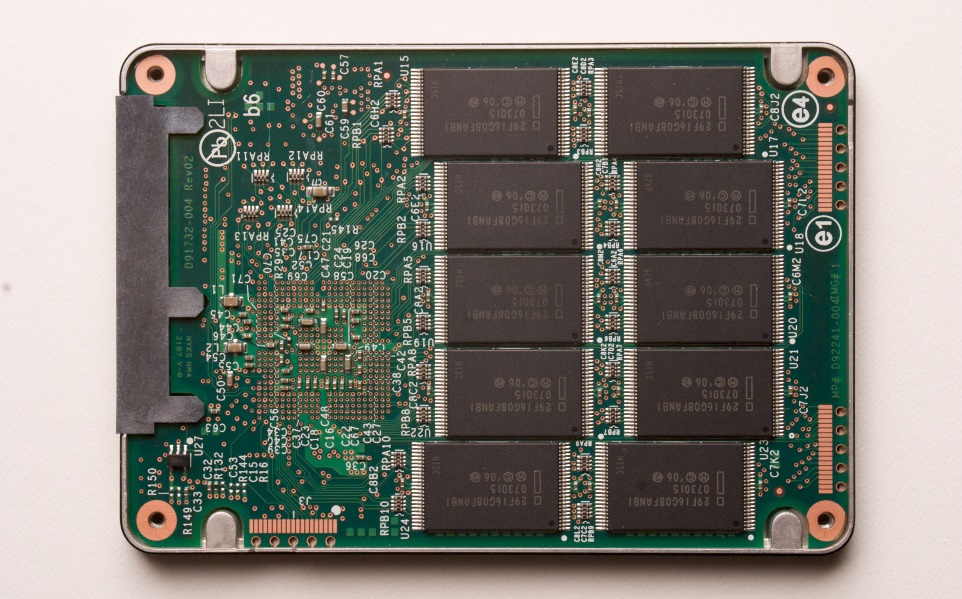
Tijd nodig om een bepaalde sector te lezen bestaat uit drie componenten:

* Seek time
* Latency time
* Transfer time

Om de totale tijd te beperken zullen sectoren in blokken of clusters gelezen worden. Vaak zal een cluster 4 tot 8kB zijn. Hoe groter de blokken hoe groter de efficiëntie. Echter komt er wel interne fragmentatie voor.

Wanneer er verschillende schijfoperaties moeten gebeuren kan de performance verbeterd worden, d.m.v. de volgorde van de operaties aan te passen. Dit is of gedaan door het besturingssysteem of door de regelaar van de schijf (= NCQ = Native Command Queuing)

* + - 1. FCFS: First-Come First-Serve
* Eenvoudigste strategie
* Schijfoperaties worden niet oneindig lang uitgesteld
* Schijfoperaties kunnen op totaal andere plaatsen van de schijf plaatsvinden
* Niet de beste prestaties.
  + - 1. SSTF: Shortest Seek Time First
* Tegen wild heen en weer bewegen schrijfkop
* Voorkeur nabijheid huidige koppositie
* Kortere zoektijden
* Zware belasting = Bepaald deel kan even niet bediend worden
  + - 1. SCAN: Scannen
* Eerlijk gedrag
* Loopt de schijf af en omgekeerd, voert alle operaties uit wanneer hij voorbijkomt
* Gegarandeerd alle schijfoperaties finaal zullen afgewerkt worden
  + - 1. C-SCAN: Cirkulair Scannen
* Verbetering op SCAN strategie (eerlijker)
* Loopt schijf af, maar op terugweg voert hij geen operaties uit
* Slechtere doorvoercapaciteit dan SCAN, doordat terugweg onbenut wordt.
  + - 1. (C-)LOOK: Aangepaste SCAN + C-SCAN
* Schijf laten bewegen tussen twee uiterste sporen van schijfaanvragen i.p.v. het einde van de schijf.
  + Look Disk Scheduling
  + Cirkular Look Disk Scheduling
  1. Solid State Drive



Dram of Flash geheugen om gegevens op te slaan:

* **Indien DRAM**:
  + Uitgerust met batterij om bij stroom uitval gegevens naar een backupschijf te kunnen schrijven.
  + Geschikt voor opslagmedium voor swap
  + Beperking snelheid door interface naar systeem
  + Enkel nuttig wanneer intern geheugen niet meer uitbreidbaar is.
* **Indien Flash geheugen:**
  + Goedkoper dan Dram
  + Verliest geen geheugen bij uitvallen van stroom

**Voordelen SSD:**

* Geen schijf die moet draaien, opstarten of kop die bewogen moet worden = Snellere toegang
* Fragmentatie bestanden is minder belangrijk
* Geen lawaai van bewegende onderdelen
* Lager stroom verbruik
* Beter bestand tegen schokken, temperatuur

**Nadelen:**

* Kostprijs t.o.v. hardeschijf
* Beperkt aantal keer beschrijfbaar flash
  + **Recente ontwikkelingen:** flash 1 000 000 keer beschrijfbaar
    - De controller zorgt ervoor dat elke deel van de hardeschijf ongeveer evenveel beschreven wordt.
* Gebruik Erase blocks: Meer gegevens moeten aangepast worden, trager random schrijven
  + 1. Flash Technologie

Flash geheugen bestaat uit cellen die in staat zijn om een spanningsniveau te onthouden.

* + - 1. SLC (Single Level Cell)
* Elke cel bevat 1 bit
* Eenvoudigste en snelste maar ook duurste manier om geheugen te produceren
  + Grote capaciteit = Veel cellen
    - 1. MLC (Multi Level Cell) (TLC en QLC)
* Elke cell kan meerdere bits onthouden.
  + Iedere cell kan bijvoorbeeld 4 spanningsniveau’s bewaren.
* Gevoeliger voor fouten
* Nood aan precisie: Trager lezen en beschrijven
* Gebruik: Grote capaciteit & betrouwbaarheid of snelheid.
  + 1. SchrijfCyclus Flash Memory

Lezen gebeurd in 1 beweging, schrijven is echter een heel ander verhaal.

* Huidige data wordt in snelle cache ingelezen
* Waardeloos aangeduide cellen worden effectief leeggemaakt
* Huidige data en de te schrijven data worden samengevoegd **in de cache**
* Cache wordt terug weggeschreven naar een **leeg** block in de flash.
  + 1. Optimalisatie van Flash

SSD’s hebben nog vaak problemen. Het besturingssysteem vervult hier een belangrijke rol.

* + - 1. Buffering

Voor het schrijven moeten de benodigde blokken eerste gewist worden, deze werkwijze kost flink wat tijd. Daarom worden schrijfacties gebufferd. Als de cache te klein is zal na een periode van continue schrijven de prestatie sterk degraderen.

* + - 1. Schijfverlamming

Wanneer er geschreven wordt naar de schijf, moeten blokken eerst leeggemaakt worden, tenzij deze al leeg zijn. Maar naarmate de tijd vordert, zijn er steeds minder lege blokken. Waardoor de performance ineenzakt.  
  
Oplossing: **Garbage Collection:** Op rustige momenten op schijfniveau defragmenteren. Informatie in blokken wordt samengevoegd, zodat er terug lege blokken zijn, die veel sneller te beschrijven zijn.  
Reserve space: Om efficiënte garbage collection te hebben bij bijna volle schijven

Bijkomende manier: **ATA-Trim:** Besturingssysteem verteld met dit commando aan de schijf welke data is verwijderd, zodat verwijderede sectors niet meer verplaatst worden op de schijf.

* + - 1. Alignment

Het is belangrijk dat de grenzen van de partities exact overeenkomen met de grenzen van SSD-blocks. Anders zal elke keer wanneer er een blok moet worden geschreven, 2 blokken aangepast moeten worden.  
Voor deze reden is het best dat er geen gebruik gemaakt wordt van sector-based copy tools zoals Ghost of dd (linux commando)

* 1. Logische structuur van een opslagmedium

Hiermee wordt er bedoeld hoe het systeem de schijf ziet. De Hardeschijf wordt verdeeld in één of meerdere partities (= Fysieke formattering) en elke partitie wordt dan georganiseerd volgens de regels van een welbepaald bestandssysteem (=logische formattering).

Sommige toepassingen die intensief gebruik maken van het secundair geheugen vereisen dat een schijf partitie niet logisch geformatteerd wordt. Zij gebruiken de “RAW disk” en hun eigen logische formatering. 🡺 Om overhead van bestandsysteem te omzeilen.

* + 1. Boot Blok

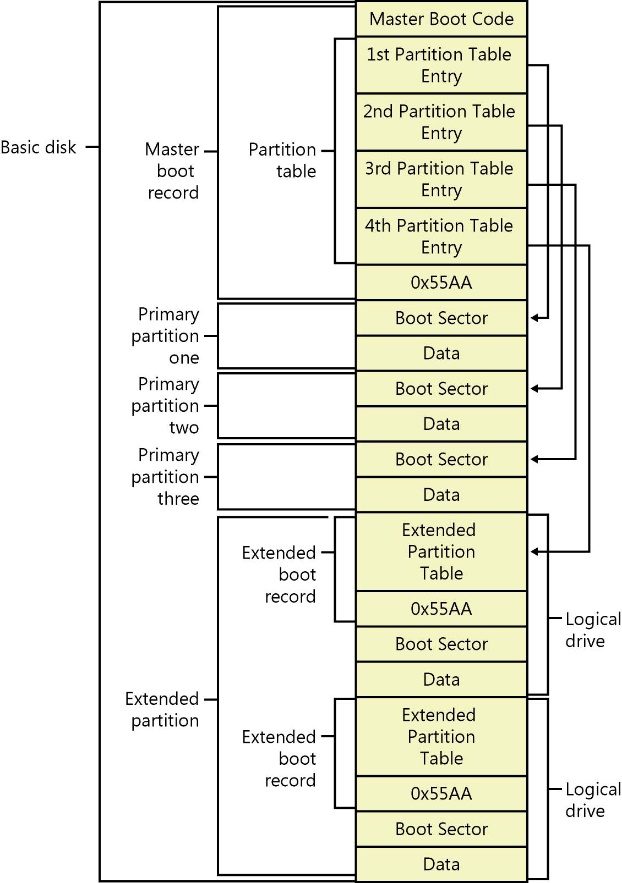
Inhoud van de “Systeemschijf” bepaalt welke besturingssysteem opgestart wordt.  
**Bootstrappen:** Opstarten van een computersysteem

* Processor voert code uit op vaste plaats in het geheugen
* Programma gaat na of hardware zich gedraagt zoals het hoort
* Programma zoekt naar systeemschijf en leest de bootsectoren
* De code in de bootsectoren zal het volledige besturingssysteem inladen, config bestanden lezen en controle overleveren aan het besturingssysteem.
  + 1. MBR (Master Boot Record) layout

|  |  |  |
| --- | --- | --- |
| **offset** | **lengte (bytes)** | **inhoud** |
| 0 | 446 | MBR programmacode |
| 446 (1BEh) | 16 | eerste partitie-record |
| 462 (1CEh) | 16 | tweede partitie-record |
| 478 (1DEh) | 16 | derde partitie-record |
| 494 (1EEh) | 16 | vierde partitie-record |
| 510 (1FEh) | 2 | 55 AA (einde markering) |

* **MBR:** Eerste fysieke sector van de hardeschijf.
  + **Primaire Partitietabel**
    - Hoe is schijf onderverdeel in partities
    - Voor elke partitie een record van 16 bytes
  + **MBR-Programmacode**
    - Instructies voor beslissing welke primaire partitie opgestart wordt om de juiste bootsector in het geheugen te laden.
    - Eenvoudigste vorm: Zoekt in partitietabel naar de actieve partitie, laadt de bootsector van die partitie in het geheugen en voert die code uit.
    - Complexere vorm: Interactie met de gebruiker (Zoals bij LILO, Grub, Bootmagic)
* Partitionering:
  + Onderverdelen van de hardeschijf in verschillende blokken
  + Interresant om bepaalde soorten gegevens te groeperen
  + Swap of multiboot
    - 1. Partitietabel-Record

|  |  |  |
| --- | --- | --- |
| **offset** | **lengte (bytes)** | **inhoud** |
| 0 | 1 | 80h (actieve partitie) of 00h (niet actief) |
| 1 | 3 | CHS adres eerste sector van de partitie |
| 4 | 1 | type partitie |
| 5 | 3 | CHS adres laatste sector van partitie |
| 8 | 4 | LBA adres eerste sector |
| 12 | 4 | aantal sectoren in partitietabel |



* **Eerste byte:** Is de partitie actief of niet
* **Aantal Parameters:** locatie, grootte, ..
* **Type partitie:** Welk bestandssysteem   
  (FAT16, FAT32, EXFAT, NTFS, EXT4, ZFS, …)

Er zijn slechts 4 records in de primaire tabel, waardoor er maar 4 primaire  
partities kunnen gedefinieerd worden. Als er meer nodig zijn moet men  
werken met extended partities.

**Extended partitie:**

* Logische partitie, niet primair
* EMBR (Extended master boot record) met 2 partitie tabellen.
* 1 tabel kan gebruikt worden voor een nieuwe extended partitie
* Niet alle info mag op een extended partitie terecht komen.   
  (Zoals Windows installatie)
  + 1. GPT layout

**Nadelen disk-layout met MBR:**

* Maximale partitiegrootte: 2TB
* Partitietabel is erg belangrijk, maar is niet beschermd

**GPT-layout** **zoekt hier een oplossing voor:**

* Partitietabel op het einde van de schijf i.p.v. begin
* Valse MBR in het begin om partitioneringstools te misleiden
* Maximaal 128 partities
* Geen limiet grootte van de partitie

**Gegevens partitie entry GPT:**

|  |  |  |
| --- | --- | --- |
| **#bytes** | **Naam** | **Beschrijving** |
| 16 bytes | Partition Type GUID | Bevat een GUID die zegt over welk soort partitie het gaat. Vb Linux Swap Partition, Windows Basic Data Partition, Apple HFS+ partitie, … |
| 16 bytes | Unique Partition GUID | Unieke GUID voor deze partitie |
| 8 bytes | Starting LBA | Begin van de partitie |
| 8 bytes | Ending LBA | Einde van de partitie |
| 8 bytes | Attribute bits | Extra info over de partitie |
| 72 bytes | Partition name | Leesbare naam voor de partitie |

* 1. Bestandssystemen
     1. Algemeen

Een bestandssysteem bestaat uit verschillende lagen die elk van de diensten van de onderliggende lagen gebruik maken.

1. **Logisch Bestandssysteem:** Niveau waarmee de gebruiker in contact komt. Hier wordt er met bestandsnamen, protectie, enz. gewerkt. Tevens beheert dit ook de directories de als speciale gegevensbestanden beschouwd worden.
2. **Bestandsorganisatie:**
   1. **Voor logische bestandssysteem:** Vormt de verbinding tussen het logische en het fysieke niveau. Hier wordt aan het beheer van de vrije schijfruimte gedaan, bestandsnamen omgezet naar logische schijfadressen.
   2. **Voor fysieke bestandssysteem:** BO geeft logische schijfadressen door die dan naar fysieke schijfadressen worden vertaalt voor de desbetreffende schijven.
3. **IO-controle:** Dit zijn de drivers voor de schijven. Deze verbergen alle details van de schijfhardware voor de bovenliggende lagen.

Een bestandssysteem zal de bestanden op een partitie organiseren en zorgen dat het besturingssystemen over genoeg informatie beschikt om elk bestand terug te vinden.

Elke partitie begint met een PBS (Partition Boot Sector). Dit is weer de fysieke sector die bij een bepaalde partitie hoort. Hij bestaat uit twee delen.

* SprongInstructie naar eventueel aanwezige boot routine
* BIOS parameter block: Aantal parameters belangrijk om toegang te krijgen tot het bestandsysteem.
  + 1. Bestanden

Hoe plaats je de clusters waaruit een bestand bestaat op een schijf? De keuze van de methode hangt af van de manier een schijf zal gebruikt worden: sequentieel, direct of geïndexeerd en van de prestaties die men verwacht. Doordat de processor veel sneller is dan de harde schijf, zal men proberen bepaalde tabellen in het geheugen te houden en data te verspreiden over schijven of via een complex algoritme de data te lezen zodat het schijfgebruik kan verminderen.

* + - 1. Contigue Allocatie
* Eenvoudigste allocatiemethode
* Een bestand bestaand uit n blokken zal n opeenvolgende fysieke blokken op de schijf innemen.
* Minimaal verplaatsing lees/schrijf kop
* Snelste allocatiemethode
* **Nadelen:**
  + Externe fragmentatie
  + Voor creatie bestand moet men weten hoe groot het zal worden
    - 1. Gelinkte Allocatie
* Los alle externe-fragmentatieproblemen op.
* Alle blokken op de schijf worden aan elkaar gelinkt
* **Nadelen**
  + Geen directe toegang mogelijk, eerst met de lijst met blokken afgelopen worden
  + Sequentieel lezen van een bestand kan traag worden
    - Oplossing: Trachten zo veel mogelijk blokken sequentieel op een schijf te hebben, regelmatig compacteren
  + Blokken zullen iets kleiner zijn, omdat de link naar het volgend blok moet worden opgenomen
    - Oplossing: Grotere blokgrootte
  + Niet betrouwbaar: Vanaf 1 blok corrupt wordt is heel de schijf onbetrouwbaar
    - 1. AllocatieTabel
* Problemen oplossen door: FAT (File Allocation Table):   
  Alle wijzers naar blokken bij houden in een afzonderlijk blok
* De tabel bevat een wijze per blok
* **Voordelen:**
  + Om een bestand te zoeken moet men maar een paar blokken van de schijf lezen
  + Geen wijzers in gegevensblokken
  + Men kan de Fat meerdere keren opslaan om schijfdefecten te voorkomen.
* Vrije blokken kunnen worden teruggevonden door een speciaal teken op de plaats van de wijzer.
* Normaal wordt de Fat tabel in het geheugen bijgehouden om schijfgebruik te verminderen.
  + - 1. IndexTabel
* Allocatietabellen geven geen efficiënte directe toegang tot bestanden, omdat links sequentieel moeten afgelopen worden.
* Geïndexeerde locatie: Alle wijzers naar de gealloceerde blokken zijn sequentieel opgenomen in een indexblok. Dit kan gebruikt worden om het gegevensblok rechtstreeks terug te vinden.
* Om niet beperkt te zijn tot een maximale lengte kunnen blokken gelinkt worden of verschillende niveaus indexblokken gecreëerd worden
* **Nadelen:**
  + Minder efficiënt gebruik schijfruimte, meeste indexblokken halfvol
    - Voor kleine blokken moet een volledige indexblok gealloceerd worden
    1. Mappen(Directories)

Tegenwoordig hebben meeste bestandssystemen een boomvormige structuur. Hierdoor hebben bestanden niet enkele een naam maar ook een Pad.

* **Absoluut pad:** t.o.v. de root-directory
  + Verwijst altijd naar correcte directory in alle omstandigheden
  + Kan lang worden
* **Relatief:** t.o.v. de huidige directory
  + Positie-onafhankelijk (binnen bepaalde boomstructuur geldig)
  + Korter
* **Nadelen Boomstructuren:**
  + Concept “huidige directory nodig”
  + Om een bestand terug te vinden moet men in verscheidene directory’s kijken. (PATH)
  + Voorziening nodig om bepaalde bestanden in meerdere directory’s te hebben zonder te kopiëren.
    - Symbolische Link: Verwijzing naar ander bestand, bevat enkel de naam
    - Harde link: 2 gelijke frame adressen in de tabellen, er wordt een teller bijgehouden hoeveel verwijzingen er zijn.
    - **Acyclische graaf:** Boomstructuur met links, zonder lussen
      * **Nadeel:** Bij zoeken moet men opletten dat niet 2 keer hetzelfde doorzocht wordt.
    - **Cyclische graaf:** Boomstructuur met links, met lussen
      * **Nadeel:** Bij zoeken kan met in een oneindige lus terecht komen.
    1. SwapRuimte

Het kan zijn dat de blokken van een bestandssysteem niet vereenkomen met de pagina’s en frames van de swapruimte. Meestal wordt gekozen om de swapruimte onder te brengen in een afzonderlijke partitie.

* **Voordeel:** Schrijftoegangen minimaal bij het pagineren.
* **Nadeel:** Bij generatie besturingssysteem moet grootte vastgelegd worden.

Sommige besturingssystemen zoals windows alloceren de swapruimte als een contigu bestand.

* **Voordeel:** Eenvoud en de mogelijkheid tot constant uitbreiden
* **Nadeel:** Traagheid: elke toegang tot swap gaat via bestandssysteem
  + 1. Vrije ruimte

De vrije ruimte op een schijf moet efficiënt kunnen bijgehouden worden, zodat een vrij blok snel kan terug gevonden worden.

* **Bitmap:** Bitrij zegt of blok leeg of vol is.
* **Gelinkte Lijst:** Vrije blokken worden niet gebruikt, dus kunnen ze gebruikt worden om elkaar te linken. **Nadeel:** Als er 100 blokken moeten gealloceerd worden, zullen er ook 100 schrijftoegangen nodig zijn.
* **Wijzerblok:** Vrije blokken worden in indexblokken opgenomen. Meestal gelinkte indexblokken, zodat men ineens kan alloceren.
  + 1. Fragmentatie

Behalve bij contigue allocatie zullen na verloop van tijd bestanden versnipperd geraken over de gehele schijf. Dit fenomeen noemt fragmentatie.

Om dit probleem op te lossen kunnen de bestanden gedefragmenteerd en gecompacteerd worden.  
De bestanden zullen contigu gemaakt worden en de lege ruimte zal samen staan. Wanneer een nieuw pakket wordt geïnstalleerd zal dit aan één stuk in de vrije ruimte terecht komen.

* + 1. Bestandssysteem ‘mounten’

Het secundair geheugen van een computer kan aangeboden worden op twee manieren:

* A: B: C: … Hierdoor lijven ze expliciet zichtbaar.
* Opgenomen in een subdirectory: Transparant voor de gebruiker.
  + 1. Types: journaling file systems

Bestandssystemen zijn grootte datastructuren, wanneer er een verandering is heeft dit meerdere schrijfoperaties nodig. Wanneer het systeem crasht, kan het zijn dat maar de helft is uitgevoerd en dit kan leiden tot een inconsistente toestand.

**Manieren tot herstel:**

* **Complete datastructuren Bestandssysteem doorlopen:** Tijdrovend en traag
* **Logbestand bijhouden (journaal):** Houd bij welke veranderingen er moeten gebeuren.
  + **Voorbeelden:** JFS, EXT3/4, ReiserFS en NTFS
    1. FAT file system

Bekendste varianten FAT: FAT16 & FAT32 (Afgedaan als primair bestandssysteem)



**Opbouw:**

* **Gereserveerde sectoren:**
  + PBS met nodige parameters (bytes per sector, sectors per clusters, …)
  + Wat andere boel
* **File Allocation Tables:** Meestal 2
* **Root folder**
* **Data clusters:** Opslag bits die de bestanden vormen. (In clusters!)

**Voordelen FAT32 vs. FAT 16:**

* MAX 2TB vs. MAX 2GB
* Onbeperkt aantal entries (hoofddirectory als gewone) vs. 512 entries in hoofddirectoy
* Kleinere clusters, minder slack, waardoor je wel 10 tot 15% meer ruimte hebt.
* Beide fat-tabellen staan niet meer in begin van schijf => Minder snel beschadigd

**Root folder:** Alle bestanden en mappen opgeslagen in de root folder.

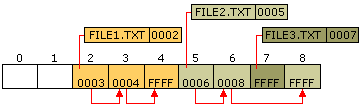
* Namen voldoen aan 8.3 formaat
* Element = 32 bytes
* Naam = eerste 11 bytes
* Vanuit de rootfolder kan het volledige pad gevolgd worden als een bestand is ondergebracht.

**Zoeken van een bestand in een FAT-systeem:**

1. **Zoeken root-folder:** Uit het partition boot sector kan je de structuur van de partitie afleiden:
   * het aantal gereserveerde sectoren
   * het aantal FAT’s
   * het aantal sectoren per fat
   * aantal bytes per sector
2. **Start root = aantal gereserveerde sectoren + aantal FAT’s \* aantal sectoren per FAT**

**Byte adres = sectornummer \* aantal bytes per sector**

1. **Zoeken naam gevraagde bestand**
2. **Overige informatie om bestand terug te vinden:** 
   * **Laatste 4 bytes:** lengte van het bestand (in bytes)
   * **16 bits daarvoor:** Eerste clusternummer waar er data terug te vinden is
3. **A.d.h.v. clusternummer in FAT kijken of bestand bestaat uit 1 of meer clusters:**
   * FAT-tabellen zijn kopieën van elkaar.
   * Bij FAT16 bestaat elk element in de FAT tabel uit 16 bits. Elk element is gelinkt met een welbepaald clusternummer (element 2 hangt samen met cluster 2, element 3 hangt samen met cluster 3, …​).
   * De inhoud van de twee bytes bepalen hoe een cluster samenhangt met de andere:
     + 0000h: Available Cluster
     + 0002h-FFEFh used, Next Cluster in File
     + FFF0h-FFF6h reserved Cluster
     + FFF7h BAD Cluster
     + FFF8h-FFFF Used, Last Cluster in File
4. **Via eerste clusternummer, alle clusters doorlopen en uitlezen. De laatste gedeeltelijk uitlezen om aan de in de rootfolder gevonden grootte te voldoen.**



* + 1. NTFS

Standaard bestandssysteem voor windows machines.

* Andere informatie in de PBS
* Geen FATs, maar MFT (Master File Table) met alle revelante informatie van een bestand
  + Stuk uitgebreider dan de informatie in de rootfolder van een FAT systeem
  + Mogelijkheid dat een klein bestand direct in MFT wordt opgeslagen
  + Naam en runlist zijn belangrijkste informatie voor een zoeking.

**Zoeken van een bestand in een NTFS-systeem:**

1. **Terugvinden MFT:** Clusternummer staat in PBS opgegeven & aantal sectoren per cluster
2. **Naam gevraagde bestand zoeken in de MFT**
3. **Runlist zoeken in het record:** Onderdeel van $DATA waarvan de start gekenmerkt wordt door “80H”. Hier is ook de werkelijke grootte van het bestand terug te vinden.
   1. **Opbouw runlist:** Eerste byte wordt opgesplitst in twee nibbles
      1. **Meest significante Nibble:** Bytes voor offset (stel K)
      2. **Minst significante Nibble:** Bytes voor lengte (stel N)
   2. **Volgende N bytes in little endian:** Aantal clusters na elkaar voor deze stream
   3. **Volgende K bytes in little endian:** Op welke clusteroffset begint de stream terug
   4. **Volgende byte: 0x00** -> Runlist eindig hier, anders zelfde manier verder werken

**Interpretatie van een runlist**

..

Eerste byte: Het eerst volgende bepaalt het aantal clusters

Drie volgende bytes zijn de offset.

Geen data meer

Besluit: Data staat in 10 clusters, te beginnen van 0x052321

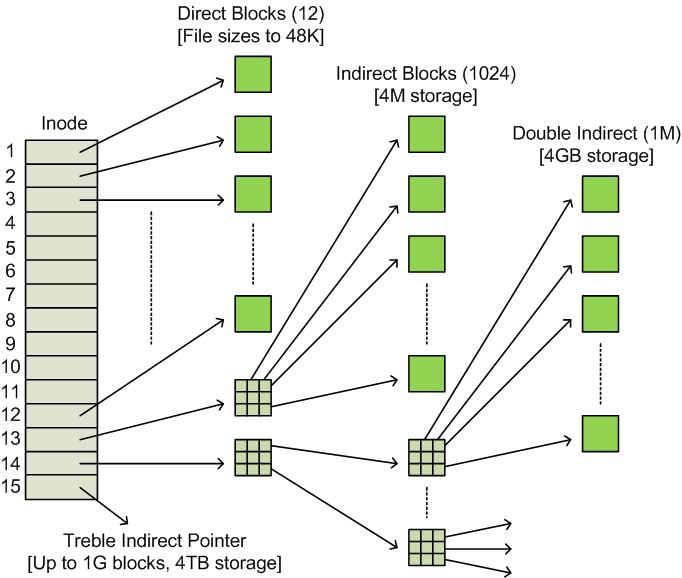
* + 1. Ext4 (Extended File System 4)

Brug tussen EXT3 en meer geavanceerdere bestandssystemen. Het is de standaard op zowat alle linux distributies.

* + - 1. Inodes

In Unix maakt men gebruik van een gecombineerd systeem met inodes (=Gegevens structuur tussen directory-entry en bestand. Elk fysiek bestand heeft slechts 1 inode, waarvoor er verschillende directory entries kunnen zijn).

1. **Directory Inodes:** Bevatten info over een bepaalde map (O.a. subdirectory’s, bestanden en namen.)
2. **Data file inodes:** Bevatten info over waar een bepaald bestand kan gevonden worden. Deze bevatten de bestandsnaam niet. Zo kunnen verschillende bestandsnamen gebruikt worden voor 1 fysiek bestand. (In linux heeft dit HARD LINKING).

In de inode zijn een vast aantal indices opgenomen als attributen van een bestand. (bv 12)  
Deze volstaan voor bestanden kleiner dan bv. 48Kb.  
Er is ook nog een wijzer naar niveau 1, niveau2 en niveau 3.  
  
Deze oplossing laat toe om snel directe toegang te hebben tot kleine bestanden, redelijk snel tot middelgrote en aanvaardbaar voor echt grote bestanden.  
  
De grootte van de bestanden is niet beperkt. In praktijk zijn minder dan 5% van de bestanden groter dan 48KB en een indexblok nodig hebben.

* + - 1. Andere bestandssystemen
      2. BTFRS

Opvolger van EXT4 in Linux. Nog niet volledig stabiel, biedt ondersteuning voor snapshots en quota’s.

* + - 1. ZFS

Oorspronkelijk ontwikkeld door SUN, met rijke featureset die BTFRS en EXT4 ruim achter zich laat. Echter door conflicterende licenties is het niet standaard te vinden in linux-distributies, maar het kan meestal achteraf wel geconfigureerd worden.

* + - 1. RE-Fs

Enkel ondersteund door Windows server 2012, ervan booten in nog niet ondersteund. Het bevat enkele features die vooral bij kritieke data belangrijk zijn.

* + - 1. F2FS

Nieuw bestandssysteem speciaal opgebouwd om een hoog rendement te halen bij flash disks.

* + 1. Bestanden wissen

**FAT16:** Speciaal teken om naam in root folder mee te beginnen en alle clusters die gebruikt werden worden in de FAT tabel als ongebruikt gemarkeerd. Het is mogelijk om dit bestand terug te vinden, tenzij het gefragmenteerd was. Dan is de tabel nodig om de juiste clusters te vinden.

**NTFS:** Enkel MFT record wordt aangepast, waarin een parameter aangeeft dat het bestand is gewist. Aangezien dit record niet gewist is, is het mogelijk om dit bestand terug te vinden, zelfs als het gefragmenteerd is. – De vrijgekomen ruimte mag vanzelfsprekend nog niet herbeschreven zijn –

* 1. Opslagmedia combineren: RAID

Techniek waarbij meerdere HDD’s gecombineerd worden.

**Doel:** Data veiligstellen & performantie

**Hoe:**

* **Hardwarematig:** Ingebouwde RAID-controller of RAID-Kaartje
* **Softwarematig:** Door OS
  + 1. Types RAID

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **(schrijf)snelheid** | **Databescherming** | **Minimum # schijven** |
| RAID 0 | ++ | -- | 2 |
| RAID 1 | - | ++ | 2 |
| RAID 5 | -- | + | 3 |
| RAID 10 | - | ++ | 4 |

* + - 1. RAID 0

Wordt ook stripping genoemd. Data wordt uitgesplitst over meerdere schijven. De snelheid is de som van de schijven. **Nadeel:** Als 1 schijf faalt, dan is alle data onbruikbaar.

* + - 1. RAID 1

Data wordt parallel op 2 plaatsen geschreven, opslagcapaciteit wordt gehalveerd.

* + - 1. RAID 5

Data wordt gestripped over meerdere schijven, per data wordt ook pariteit bewaard. Er is herstel mogelijk als er 1 schijf uitvalt. Schrijven is traag (pariteit moet berekend en geschreven worden.)

* + - 1. RAID 10

Is een RAID 0 van RAID 1’s. Snelheid van RAID0 maar door RAID 1 is uitval geen probleem.

* 1. Snelheid bij disks: IOPS (=Input Output per seconde)

Manier om snelheid van opslagmedia aan te duiden. Komt overeen met gemiddeld aantal bewerkingen dat het medium per seconde kan afwerken.

Worst case IOPS: (Random acces & kleine datablokken)

|  |  |
| --- | --- |
| **Type schijf** | **Voorbeeld IOPS** |
| 7200 RPM SATA | 75 IOPS |
| 10000 RPM SAS | 140 IOPS |
| 15000 RPM SAS | 180 IOPS |
| SSD | 6000 IOPS (zelfs stuk hoger bij nieuwe modellen) |

Rekenen met IOPS:

**Algemene formule**

Aantalschijven \* IOPS/schijf = reëleIOPS \* ( %READ \* 1 + %WRITE \* Writepenalty )

**WritePenality:**  
 1 voor raid 0

2 voor raid 1

2 voor raid 10

4 voor raid 5

Rekenvoorbeeld:

**Gegevens:**   
HD-stream: Lezen 30 IOPS & Schrijven 3 IOPS. Tien films moeten tegelijk kunnen worden bekeken.

**Gevraagd:**  
Welke raid voor: 300 IOPS lezen + 30 IOPS Schrijven = 330 IOPS

**Oplossing:**Aantalschijven \* IOPS/schijf = reëleIOPS \* ( %READ \* 1 + %WRITE \* Writepenalty )

6 \* 75= 330 \*( (91% \* 1) +(9% \* X)  
450 = 300 + 30x  
x = 4.54 => Raid 5

1. Opstartroutine
   1. Geheugens voor opstartroutine

Bij het opstarten kan een processor enkel spreken met het geheugen, daarom is er een tussenstap nodig om het besturingssysteem te laden van de hardeschijf. Hiervoor hebben we verschilellende ROM geheugens die nog relevant zijn.

* **MROM:** Masked-programmed ROM
  + Echt Read-only geheugen
  + Inhoud bij fabricage, niet meer te veranderen
* **PROM:** Programmable ROM
  + Echt Read-only geheugen
  + Inhoud met speciaal programmeertoestel, niet meer te veranderen
* **EPROM:** Erasable PROM
  + Inhoud kan gewist worden met UV-licht
  + Te programmeren met speciaal toestel
* **EEPROM:** Electrically Erasable PROM
  + Inhoud kan gewist worden met Elektrische puls
  + Te programmeren met speciaal toestel
  + Duurder dan EPROM, voor zelfde densiteit is het een pak duurder.
* **Flash:**
  + Te wissen en opnieuw te beschrijven met een elektrisch signaal.
  + Geen speciaal programmeertoestel
  + Vergelijkbare densiteit als met EPROM
  + Minder dikwijls herbeschrijfbaar dan EPROM   
    ( gewoon 300-500.000 tot 2.000.000 High endurance)

Bios routines zijn nu meestal opgeslagen op flashgeheugens. Dit laat ons toe de routines te upgraden.

* 1. Opstartroutine met BIOS

Bij het aanschakelen van de spanning worden er een reeds stappen doorlopen, volgorde is afhankelijk van de fabrikant.

* + 1. Systeemstart

Eerst start de voeding op, het duurt even voor alle lijnen een stabiele spanning afgeven. Wanneer dit het geval is geeft de voeding een power-good signaal. De chipset, gevoed door het standby 5V signaal neemt het reset-signaal pas weg wanneer het power-good signaal er is. Dan start de computer op en de processor begint op adres FFFOh waar de uitvoering van de bios opstartroutine begint.

* + 1. Power On Self Test (POST)

Wanneer er fatale fouten ontdekt worden, geeft het een foutmelding, indien om de een of andere redden de grafische kaart niet bruikbaar is enkele BEEP-signalen, en stopt het startproces.

* + 1. Zoeken naar BIOS-uitbreidingen

System-bios gaat onmiddellijk op zoek naar de biossen van andere apparaten en voert die uit. Hierdoor verschijnt er kort info over op het scher.  
  
Hierna voert de computer enkele tests uit op het systeem. O.a. het geheugen wordt onderzocht en de geheugenplaatsen worden geteld. Bij een defect zal er een foutmelding gegeven worden.

Vervolgens wordt er een inventaris opgemaakt voor alle randapparatuur. Een recent bios zal zelfs automatisch het soort geheugen herkennen en de juiste parameters voor de HDD instellen.

* 1. Plug and play configuratie

Wanneer er een P&P ondersteunend bussysteem (zoals PCI) aanwezig is, zal deze fase doorlopen worden.

* **Enumeration:** Zoeken naar P&P-devices, bepalen welke resources ervoor nodig zijn.
  + De vorige inventaris en lijst met resources wordt geladen uit het ESCD (Extended System Configuration Data) en vergeleken. Hierin wordt de P&P config. opgeslagen.

Bij het opstarten wordt dit door de bios gecontrolleerd, in geval van wijzigingen wordt er een herconfiguratie gedaan.

Wanneer het bestuuringssysteem is opgestart heeft het de mogelijkheid de P&P config. Te controleren en eventueel te wijzigen.

* 1. Extensible Firmware Interface (EFI)
     1. BIOS op pensioen

De BIOS is verouderd, deels in machinetaal, maximum grootte opstartpartitie 2TB…

Intel deed een poging om BIOS overbodig te maken: EFI (Extensible Firmware Interface)

* Gecodeerd in breed gekende taal (C++)
* Volledig modulair
* Compatibel met de oude BIOS om de overgang mogelijk te maken.
* Drivers kunnen opgenomen worden in deze EFI
* Geen opstartpartitie grootte grens

EFI biedt enkele nieuwe mogelijkheden, het is een mini bestuuringssysteem:

* Driver kunnen geïntegreerd worden = Makkelijker nieuw bestuuringssysteem installeren
* Makkelijk uit te breiden met kleine toepassingen. Vb. Mediaplayer
* Zou kunnen voorzien worden van virtualisatielaag, om meerdere besturingsystemen tegelijk te kunnen gebruiken?
* De eerste EFI’s zullen niet meer functies hebben dan de vroegere BIOS’en.
  + 1. (u)EFI verspreiding

Sinds 2002 wordt aan EFI gewerkt door enkele grote firma’s.

* Apple: Standaard op alle recente computers (met bios support)
* Microsoft: Standaard vanaf vista X64 sp1
* Linux: Reeds lange tijd standaard geïmplementeerd

Moederborden bieden tegenwoordig ook ondersteuning, al zullen ze vaak manueel moeten worden ingesteld.

* 1. Coreboot

Tweede alternatief voor BIOS. Het omgekeerde van EFI, zo simpel mogelijk zijn is het doel.  
Het is de bedoeling zo snel mogelijk de controle door te geven aan het besturingssysteem.  
Het zal enkel de elementaire hardware initialiseren.  
  
Doordat het geen GUI bevat is de code erg compact en snel. Momenteel wordt deze technologie gebruikt in chromebooks.

1. Internal I/O

Verschillende manieren waarop data uitgewisseld wordt tussen moederbord en systeemapparaten.

* 1. I/O transfers

Naast processor en geheugen, is I/O het derde fundamentele onderdeel van een computersysteem.

* + 1. Pollen

Voortdurend bevragen van de I/O om te kijken of er een actie van de CPU gewenst is.

**Voordeel:** I/O apparaat wordt onmiddellijk opgemerkt, waardoor de CPU direct kan reageren

**Nadeel:** CPU intensief, zelfs wanneer het niet nodig is.

* + 1. Interrupts

De I/O zal de processor onderbreken, m.a.w. de I/O zal om aandacht vragen en hoeft niet voortdurend gecontroleerd te worden. We bekijken de uitleg voor een 8086 processor, het wijkt een beetje af van het geheel omwille van de afscherming van de I/O door een buscontroller.

* + - 1. Interrupt hardware

Processor beschikt over een aparte INT-ingang, langs waar de I/O een interrupt signaal kan doorgeven. Om meerdere signalen toe te laten zou de processor moeten beschikken over meerdere ingangen, liefst hebben deze ook een prioriteit.  
  
Om dit mogelijk te maken zijn er twee interrupt controllers:

* **PIC (Programmable Interrupt Controller):**
  + Beschikt over 8 ingangen
  + Uitgang eerste controller wordt verbonden met de INT-ingang processor
  + Uitgang controller 2 zit op de 2de ingang van de eerste.
    - 1. Interrupt transfer

**Verloop:**

1. **I/O geeft interrupt signaal aan CPU**
2. **Signaal komt toe op interruptcontroller:** Indien geen interrupts met hogere prioriteit actief zijn, geeft de controller het signaal door aan de processor.
3. **CPU werkt huidige instructie af en slaat zijn context op in instruction pointer.**
4. **CPU geeft INTA (Interrupt acknowledge) signaal door aan de interrupt controller**
5. **Controller reageert door over databus een vectornummer door te geven aan de CPU.**
6. **Op basis van het vectornummer start de CPU de juiste interrupthandler en werkt de Interrupt af**
7. **CPU laadt de instruction pointer in wanneer hij gedaan heeft met de interrupt.**

We kunnen afleiden dat de reactie van de CPU trager is dan bij polling. Hiertegenover staat dat de processor alleen tijd moet spenderen aan de I/O als dat nodig is. Het maakt interrupts dan ook uitermate geschikt voor invoerapparaten.

* + - 1. Interrupts bussystemen

Bij de PCI bus wordt er gebruik gemaakt van een PCI controller om deze af te schermen van de CPU. Om op PCI een interrupt te genereren wordt er een speciaal interrupt bericht met de nodige informatie over het type en de bron van de interrupt verstuurd naar de bus controller. De buscontroller vertaalt dit dan naar een IRQ voor de processor.

Op de PCI bus kunnen er ook een aantal interrupt lijnen zijn. Deze worden gedeeld tussen verschillende apparaten. Een signaal op deze lijn wordt dan door de controller vertaald, naar een signaal dat doorgegeven wordt aan de processor. Op deze manier kunnen apparaten hetzelfde IRQ nummer delen.

* + 1. DMA (Direct Memory Acces)

Het I/O apparaat wisselt rechtstreeks data uit, zonder dat de processor moet tussenkomen. Hiervoor is er een DMA controller nodig.

**Verloop:**

1. DMA transfer begint met processor die opdracht doorgeeft aan DMA controller (aantal bytes en geheugenadres)
2. I/O controller krijg nodige instructies.
3. I/O apparaat onderneemt nodige acties om data beschikbaar te maken in buffer
4. Wanneer data beschikbaar is, geeft I/O controller een DRQ (DMA request) signaal door aan DMA-controller
5. DMA controller probeert nu controle te krijgen over de bus, d.m.v. een HOLD signaal aan de processor
6. De processor werkt lopende transfers af en geeft de bus dan vrij met een HOLDA signaal
7. DMA controller heeft nu controle over databus, het juiste geheugenadres wordt op de bus geplaats en met een DACK-signaal wordt aan de hardeschijf duidelijk gemaakt dat er data op de bus mag geplaatst worden.
8. Wanneer alles verplaatst is worden alle stuursignalen inactief gemaakt en de DMA cyclus beëindigt.

**Voordelen:**

* Processor niet betrokken bij verplaatsing van de gegevens.
* Processor kan ondertussen verderwerken

**Nadelen:**

* Cycle Stealing: Tijdens overdracht is de bus bezet en kan de processor er geen gebruik van maken
* Opstarten DMA transfer kost wat extra processor cycli
  + Hierdoor is DMA nuttig wanneer grote hoeveelheden moeten verplaatst worden
  1. Bussystemen
     1. PCI (Periphal Component Interconnect)

Ontwikkeld door Intel dat de patenten in een publiek domein plaatste, zodat alle fabrikanten randapparatuur kon bouwen, zonder hiervoor rechten te hoeven betalen.

**Eisen PCI:**

* Zorgen voor voldoende bandbreedte, ook naar het geheugen
* Backward compatibility verzekeren
* Kleinere connector dan ISA slots
* Ondersteuning voor P&P, power management (laag stroomverbruik)
* Hiërarchie van bussen

De oplossing ligt in een I/O waarin verschillende bussen aanwezig zijn, waarbij over snelle bussen data uitgewisseld kan worden zonder rekening te moeten houden met tragere onderdelen van het computersysteem die op een tragere bus zijn aangesloten.

De processor was verbonden met een back-side-bus met het cachegeheugen en met een front-side-bus met de PCI-bridge. Deze kon dan enerzijds verbinding maken met het hoofdgeheugen en met de PCI bus.

Door het gebruik van bridges wordt het processor onafhankelijk. Hierdoor kon het met andere processoren gebruikt worden.

* + - 1. Transfers

PCI is een gedeeld bussysteem.

* Datalijnen worden gebruikt door alle aangesloten apparaten
  + Nood aan adressering, welk apparaat moet data accepteren
  + Nood aan toegangscontrole, twee apparaten mogen niet tegelijk toegang hebben
* Adreslijnen en datalijnen worden gemultiplext
  + Eerst is een adrescyclus nodig
  + Daarna wordt data getransfereerd
* Elk apparaat kan op elk ogenblik een transfer starten (**initiator** zijn)
* Een arbitrator zorgt er voor dat een apparaat niet de bus gaat monopoliseren.
  + - 1. Snelheid

**Originele PCI:** 32 bits databus, 33Mhz klok = Max bandbreedte van 133MB/s.

**PCI-X:** 64-bits databus, 66Mhz klok = Max bandbreedte van 533Mb/s (x4)

* + 1. AGP (Accelerated Graphics Port)

Om aan de bandbreedte eisen voor video te voldoen wordt het bussysteem uitgebreid met AGP. De functionaliteit om deze interface te besturen werd geïntegreerd in de chipset. Deze werden initieel de North en South bridge genoemd.

**AGP 1x:** 32bit bij 66Mhz = 266Mb/s. Punt to punt verbinding => Bandbreedte niet gedeeld worden.

**Graphics Adressing Remapping Table:** AGP voorziet de mogelijkheid om direct uit het geheugen te lezen.

**Evolutie:**

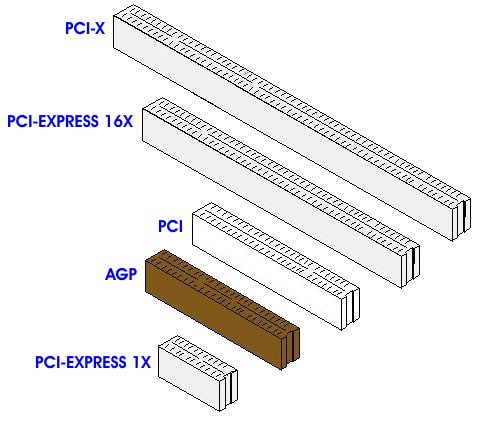
* Kloksnelheid: 66 -> 133 -> 265 -> 528 Mhz
* Spanning: 3.V, 1.5V, 0.8V
  + 1. PCI-Express

Er is nood aan een hogere snelheid.

* + - 1. Architectuur

Seriële punt to punt verbinding i.p.v. breedte databus. Seriële verbindingen hebben als voordeel dat looptijdverschillen en overspraak vermeden kunnen worden, waardoor een veel hogere kloksnelheid kan gebruikt worden.

Centraal in het concept staat een PCIe switch, die met een aantal gepaarde seriële links verbonden is met de aanwezige hardware.



* + - 1. Snelheid

De verbinding tussen de PCIe-switch en de I/O bestaat uit één of meer paren van eenrichtings-links (lanes). Elke lane heeft een snelheid van 2.5Gbps. Vergelijken met PCI is de transferrate per lane 2.3 keer sneller.

* + - 1. Protocol Stack
* Informatie wordt tussen twee gekende vlaggen verzonden, die start en einde van de data aanduiden.
* Om te synchroniseren: 8b/10b codering. Om acht bits te versturen worden er eigenlijk tien verstuurd. De brutosnelheid verminderd zo wel naar 2Gbps.
* Flow Control: Zorgt ervoor dat zender data niet sneller stuurt dan de ontvanger het kan verwerken.
* Error Control: Foutdetecterende code wordt toegevoegd aan het pakket. In het geval van een fout wordt er een hertransmissie aangevraagd.
  + - 1. Vergelijking met PCI
* Overgang naar seriële transmissie
* Hogere kloksnelheden
* Foutcorrectie
* Geen gedeelde bandbreedte op punt-tot-punt verbinding
* Kleinere connectoren
  1. Bussystemen voor harde schijven
     1. SATA

Van PATA naar SATA: Van Parallel naar serieel.

* + - 1. Compatibiliteit SATA VS PATA

PATA en SATA zijn enkel software compatibel.

* + - 1. Signalering en bandbreedte

Dataverbinding schijf-moederbord bestaat uit 7 geleiders. 3 zijn ground en 4 zijn twee paren voor data. Er wordt 8b/10b toegepast. De kloksnelheid is 1500Mhz.

Sata is hotplugable.

* + - 1. NCQ

Sata ondersteund Native Command Queuing.

* + 1. eSATA

External SATA, met iets andere signalering zodat langere kabels kunnen gebruikt worden.

* + 1. Serial Attached SCSI (SAS)

Lijkt enorm op SATA, gemaakt voor hogere snelheid. Biedt iets grotere betrouwbaarheid, gericht op servers.

1. Externe I/O
   1. PS/2

Aansluiten toetsenborden en muizen. Het is niet hotplugable. De pinnen van de microcontroller kunnen beschadigd worden, wanneer onder spanning geswaped wordt.

* 1. USB
     1. Overzicht

Doelstellingen:

* Eenvoudige en betrouwbare aansluiting geschikt voor alle soorten randapparatuur
* Immuun voor problemen met systeemresources of interrupt- of DMA-conflict
* Automatische detectie en configuratie van aangesloten apparaten
* Goedkoop te realiseren
* Laag stroomverbruik
* Hot-pluggable

Soorten snelheden:

* **Low Speed:** 1.5Mbps
* **Full Speed:** 12Mbps (USB2.0)
* **High Speed:** 480Mbps (USB 3.0)
* **Super Speed:** 4.8Gbps

Maximaal mogen 4 hubs aangesloten worden en in totaal 127 apparaten op een USB bus.

* + 1. Mechanische en elektrische opbouw

Kabels zijn niet langer dan 5m. De USB2-kabel bevat vier geleiders.

* Rood: +5V
* Zwart: GND
* Wit: Data
* Groen: Data+

Kan stromen voeden tot 500mA, bij een spanning van 5V.

De USB-bus verstuurd de data serieel met NRZI-codering (Non Return On Zero Inverted) via een differentieel signaal. Een overgang in het datasignaal stelt een 0 voor, terwijl een ongewijzigd signaal een 1 voorstelt.

**Bit-Stuffing:** Wanneer er zes opeenvolgende bits hetzelfde zijn, wordt er een nul ingevoerd. Dit wordt gedaan voor een betrouwbare gegevensoverdracht. Tevens is er ook een CRC foutcontrole per pakket.

* + 1. Communicatie over USB

Er is shared bandwidth en er wordt gebruik gemaakt van het master/slave principe, waardoor de host-controller de volledige controle heeft over de communicatie met de USB-bus.

* + - 1. Frames, pakketten

De host verdeelt de tijd in frames met een vaste lengte. Dit wordt op zijn beurt ingedeeld in een aantal pakketten, waarvan het eerste een Start Of Frame (SOF) pakket is.  
Binnen een pakket zijn er enkele belangrijke velden:

* **De sync-vlag:** Bedoeld voor bit- en byte-synchronisatie
* **Packet-ID:** Geeft het type van het pakket weer
* **Data-veld:** Bevat naast eventuele gegevens ook het adres van het apparaat waaraan het pakket is gekoppeld en het endpoint.

Het einde van een pakket wordt gekenmerkt door een laag signaal voor een bepaalde tijd.

Er zijn vier types van pakketen:

* **Start-of-frame:** geeft het begin aan van een nieuw frame inclusief een 11 bit frame nummer.
* **Token**: het token geeft de richting aan van de pakketten.
  + IN:: informeert het toestel dat de host iets wil ontvangen
  + OUT:: informeert het toestel dat de host iets wil versturen
  + SETUP:: wordt gebruikt om een transfer te starten.
* **Handshake**
  + ACK:: geeft een bevestiging dat de info correct werd ontvangen
  + NAK:: geeft aan dat het toestel eventjes geen data kan zenden of ontvangen, of dat er geen data beschikbaar is op dit moment.
  + STALL:: het toestel bevindt zich in een staat waarin de host actie moet ondernemen.
* **Data:** bevat maximum 1024 bytes aan data. Er zijn ook verschillende subtypes:
  + Data0
  + Data1
  + Data2 (enkel bij high-speed)
  + MDATA (enkel bij high-speed)
    1. Endpoints

Elk USB-device beschikt over enkele databuffers (endpoints), deze worden gebruikt om de informatie uiteen pakket op te slaan.

* + - 1. Types van transfers:
* **Control Transfers:** Voor beheer USB, voor informatie over apparaat
* **Bulk Transfers:** Voor situaties waarbij gegevens moeten verstuurd worden, maar waar de snelheid niet kritisch is.
* **Interrupt Transfers:** Wanneer een apparaat op regelmatige tijdstippen de gelegenheid moet krijgen om gegevens te verzenden of te ontvangen. Gebaseerd op regelmatige polling, want interrupts is onmogelijk voor een USB slave.
* **ISOCHRONE transfers:** Garanderen van een gevraagde bandbreedte, maar zonder foutcorrectie.
  + - 1. Verloop transfers
* De host-controller zegt in welke richting de data zal gaan. (in-token & out-token)
* De Data wordt verzonden
* Bevestiging van ontvangen
  + Bij iscochrone transfers is er geen bevestiging.
    1. Enumeratie P&P

Om de juiste drivers te laden bij het aansluiten & een goede P&P te hebben zijn er enkele zaken nodig.

* Kant van HOST:
  + 2 datalijnen via pull-down weerstanden (15K) verbonden met de massa.
  + De host ziet een laag niveau op beide datalijnen
* Een low-speed device:
  + Pull-up weerstand van 1.5k tussen de voedingsspanning en D- (bij full & high speed naar D+)
  + Wanneer een apparaat wordt aangesloten merkt de host dat beide datalijnen naar 1 worden getrokken en dat er dus een nieuw apparaat is en of dit Low speed is.

**Enumeratie:** Bij de opstartprocedure krijg elke apparaat een uniek adres toegewezen, het besturingssysteem vraagt informatie op over het apparaat en weet welke drivers er eventueel ingeladen moeten worden.

Wanneer een nieuw apparaat gedetecteerd wordt er een aanvraag voor informatie verstuurd naar endpoint 0 (controle-endpoint). Dan geeft de host een adres aan het USB apparaat.

* + 1. Device classes, drivers
* Audio devices
* Mass Storage devices
* Monitors
* Communication devices (modems)
* Printer still image capture devices (scanner, camera)
* HID (Human Interface Devices)

Een modern besturingssysteem bevat drivers voor een aantal veel voorkomende classes. Het belangrijkste is de HID-classe. Deze zijn zo simpel dat er daarvoor geen drivers moeten geschreven worden, deze zijn al in het bestuuringssysteem aanwezig.

* + 1. USB3.0

Nieuwe connector, backwards compatible. In een kabel is de USB2.0 nog steeds aanwezig, de USB3.0 zijn twee schielded differential Pairs. Er kan nu 900mA geleverd worden. 8b/10b codering.  
Minder polling, full duplex, complexere protocol stack, link layer, power management, max. 3m