Computerarchitectuur

Roel Van Steenberghe

Table of Contents

List of Figures

List of Tables

List of Examples

Over deze cursus

Deze cursus werd opgesteld doorheen de jaren, met wisselende auteurs. Elk van hen ben ik uiteraard dankbaar, specifiek Dhr Sven Sanders Dhr Johan Donne die de fundamenten van deze informatiebron reeds jaren geleden gelegd hebben.

Er werd in deze cursus gepoogd om steeds correct om te gaan met extern bronmateriaal. Mocht je toch een stukje materiaal zonder correcte bronvermelding, dan passen we dat uiteraard ook meteen aan. Stuur hiervoor zeker een mailtje[1].

Door deze cursus in bronvorm aan te bieden op Github is er ook de hoop dat er ook door anderen toevoegingen kunnen gebeuren. Hergebruik van het materiaal is dan ook toegelaten, maar wel onder voorwaarden:

- het materiaal mag niet commercieel beschikbaar gesteld worden zonder uitdrukkelijke en schriftelijke toestemming van de auteur
- het materiaal aanpassen mag, maar dan op voorwaarde dat de aanpassingen ook publiek beschikbaar worden gesteld. Bij voorkeur gebeurt dit via deze weg, zodat iedereen mee kan genieten van de verbeteringen.

Concreet betekent dit dat al het materiaal onder de Creative Commons Naamsvermelding-NietCommercieel-GelijkDelen 4.0 Internationaal-licentie[2] valt.

Wie correcties of aanvulling aanbrengt in deze cursus, zal een vermelding krijgen op deze pagina.

[1] mailto:roel.vansteenberghe@hubkaho.be
[2] http://creativecommons.org/licenses/by-nc-sa/4.0/deed.nl

Voorwoord

Als iemand van de buitenwereld je de komende jaren vraagt wat je precies gestudeerd hebt, of wat je doet als werk, dan zal je antwoord vaak 'iets met computers' zijn. Soms is dat nu eenmaal de makkelijkste manier om je er vanaf te maken. Voor veel mensen zijn computers, tablets, smartphones en andere devices tegenwoordig zo gewoon, dat ze bijna thuis horen in het rijtje van basisbehoeften als stromend water, elektriciteit en TV. Toch blijft het belangrijk om te weten wat er onder de motorkap van je devices schuilt. Die achtergrondkennis is onontbeerlijk om later efficiënt problemen op te lossen of producten (in de breedste zin van het woord) van goeie kwaliteit af te leveren. Zelfs wie zich later zal toespitsen op het ontwikkelen van software, zal efficiënter kunnen werken als hij ook snapt wat achter de schermen gebeurt. Deze cursus probeert je een overzicht te geven van de interne keuken van een moderne computer terwijl de cursus processorarchitectuur dan weer iets dieper ingaat op de werking van het kloppend hart ervan. Uiteraard zijn twaalf lessen veel te weinig om alle onderdelen tot op het bot uit te benen. Daarom kan ik enkele standaardwerken aanbevelen, die zeker een bron van inspiratie vormden voor deze cursus. De boeken van William Stallings [STALLINGS] en Umakishore Ramachandran [RAMA] verdienen zeker je aandacht. Een overzicht van de werkvorm die bij dit vak gebruikt wordt, vind je terug in de ECTS fiche en de studiewijzer. Beiden zijn te vinden op Toledo.

Veel succes met deze cursus!

Roel Van Steenberghe

Chapter 1. Computervoeding

1.1. Principe en werking

Een computer, of het nu een pc, laptop, server of smartphone is, kan enkel functioneren als de juiste elektrische spanningen aangevoerd worden. Elektronische componenten vragen een relatief lage, maar erg stabiele gelijkspanning om te functioneren. Deze wordt geleverd door een voeding, die de wisselspanning van het elektriciteitsnet omzet naar de nodige gelijkspanningsniveau's. De techniek die hierbij gebruikt wordt noemt men switched mode power supply of schakelende voeding. Het schema van de omzetting wordt weergegeven in onderstaande afbeelding

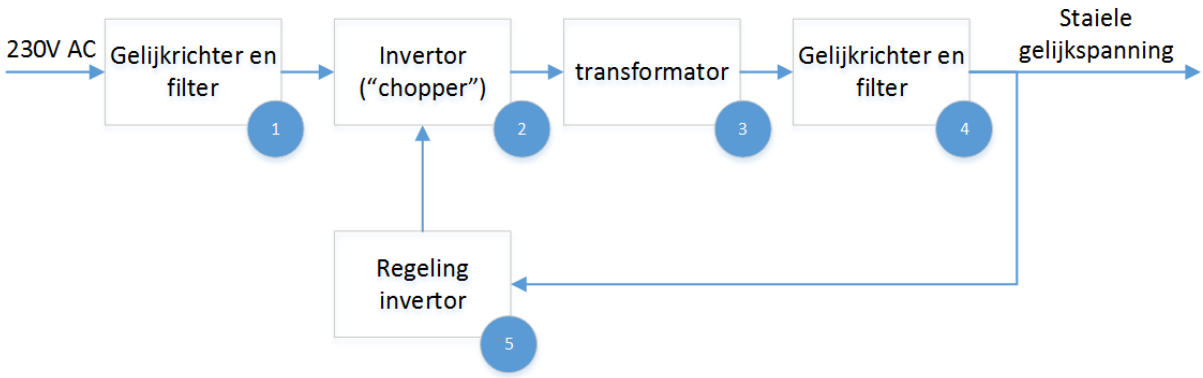

Figure 1.1. Principe schakelende voeding

Een eerste stap ① is de `gelijkrichter` aan de ingang, waarmee de wisselspanning wordt omgezet naar een gelijkspanning. Deze gelijkspanning wordt vervolgens gefilterd, zodat de variatie in het spanningsniveau beperkt wordt. De tweede stap ② is een stuk minder voor de hand liggend. Op basis van de gelijkgerichte en gefilterde ingangsspanning zal een `invertor` een blokgolf genereren. Deze wisselspanning wordt bekomen door het aan- en afschakelen van de ingangsspanning. Eigenlijk is de combinatie van de eerste twee stappen samen te vatten als een frequentieomvormer. Het ingangssignaal wordt omgezet naar een hoger-frequent signaal (van 50Hz naar frequenties boven 20kHz).

Het voordeel van deze omzetting is dat de `transformator` in de volgende stap een stuk kleiner en efficiënter kan zijn. Deze gelijkrichter en transformator ③ brengt de spanning naar het gewenste niveau, waarna het met de uitgangs-gelijkrichter en filter ④ wordt omgezet naar een stabiele gelijkspanning.

In de figuur is ook te zien dat de uitgangsspanning teruggekoppeld wordt ⑤ naar de invertor. Op die manier kan de uitgangsspanning nog geregeld worden. De invertor-stap heeft immers ook invloed op de amplitude van zijn uitgang. Deze `terugkoppeling` gebeurt meestal met optocouplers om een galvanische scheiding te bekomen.

> Om het verbruik van een CPU uit te drukken, wordt vaak gesproken over `TDP` (Thermal Design Power). Dat is een waarde die aanduidt hoeveel energie de CPU maximaal dissipeert onder een zware, maar realistische belasting gedurende een bepaalde tijd. Echter dient opgemerkt te worden dat *AMD* en *Intel* hiervoor verschillende berekeningsmethodes gebruiken. De TDP geeft dus een indicatie over het maximale verbruik, maar gedetailleerde benchmarks blijven nodig om exacte waarden te kennen. [1]

1.2. Eigenschappen

1.3. Vormfactor en connectoren

De eerste (IBM) PC beschikte over een moederbord met AT vormfactor, de bijhorende voeding was dan ook een AT voeding. In 1995 kwam er een opvolger, met name de ATX vormfactor. Ondertussen is deze specificatie ook geëvolueerd (ondertussen ATX 2.3).

De belangrijkste verschillen situeren zich op het vlak van de spanningen die de voeding kan afgeven en de connectoren die voorzien zijn op de voeding. In het bijzonder is er natuurlijk een verschil tussen de verschillende connectoren die op het moederbord worden aangesloten. Een AT voeding bood een connector van tweemaal zes aansluitingen, een ATX voeding biedt daarentegen een connector met 24 aansluitingen. (in de oorspronkelijke versie was dit 20, tegenwoordig is er ook steeds een vierpolige connector die twee keer twaalf volt levert aan de processor.)

[1] meer uitleg over de berekening van TDP bij Intel vind je in deze whitepaper [http://www.intel.com/content/www/us/en/benchmarks/resources-xeon-measuring-processor-power-paper.html]

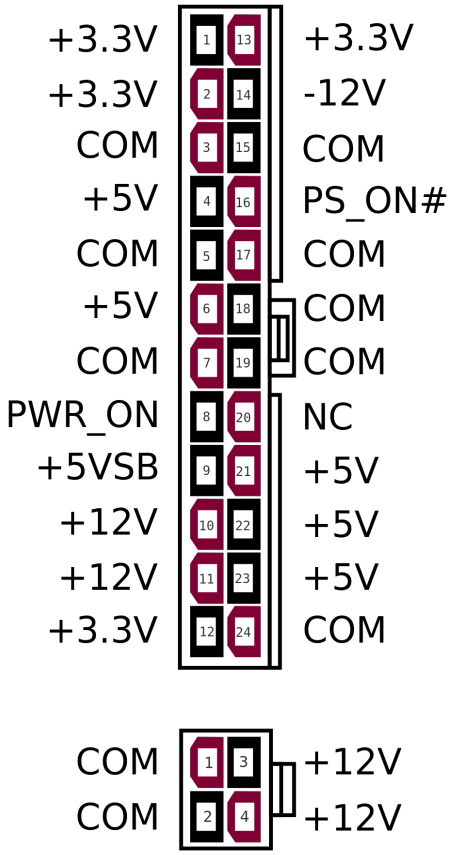

Figure 1.2. ATX 2.0 moederbord connector (CC Wikimedia commons)

Een belangrijk verschil tussen beiden is de aanwezigheid bij ATX van een 3.3V spanning, een +5V standby en power on signaal. Deze laatste twee maken het mogelijk dat de mechanische schakelaar van de AT voeding (die rechtstreeks de voeding aanstuurde), vervangen kon worden door een elektronisch signaal van het moederbord naar de voeding. In eerste instantie kan het moederbord dit signaal sturen als het zelf een input krijgt van een drukknop. Er zijn echter ook alternatieven mogelijk, zoals wake-on-lan, speciale toetsen op een toetsenbord, ... Hieruit kan je afleiden dat bij een computer die uitgeschakeld is, een deel van het moederbord nog steeds onder spanning staat. Deze spanning kan je alleen wegnemen door de voeding uit te schakelen (schakelaar op voeding, stekker uittrekken). Naast de verschillen in connectoren die op het moederbord worden aangesloten, is er ook onderscheid op het vlak van de andere connectoren. Afhankelijk van de andere apparaten (en hun voedingsaansluiting), moet je erop letten om een voeding te kiezen die de nodige connectoren aanbiedt.

Enkele belangrijke connectoren zijn:

- Moederbordconnector: afhankelijk van vormfactor
- 4-pin connector (molex): o.a. voor (ATA) schijven, optische drives
- SATA voedingsconnector
- Auxillary connectors: verschillende varianten van extra voedingsconnector en om extra vermogen te leveren

1.4. Vermogen

Een erg belangrijke eigenschap voor een voeding is het vermogen dat ze kan leveren. Uiteraard moet dit te leveren vermogen voldoende zijn om alle componenten in het systeem te voorzien van stroom. Het vergelijken van voedingen op dit vlak is iets complexer dan kijken naar de waarden die de fabrikant op zijn verpakking adverteert.

Belangrijker dan het getal is de betekenis ervan. Aangezien er geen voorschriften zijn voor de bepaling van die vermogenswaarde, kan 500W bij de ene fabrikant betekenen dat de voeding 500W piekvermogen kan leveren bij 10°C en bij een andere een continu vermogen van 500W bij 40°C.

Als het systeem continu 450W nodig heeft, zou de eerste voeding kunnen falen. Een tweede belangrijke opmerking is dat niet alleen het totale vermogen belangrijk is, maar ook het vermogen dat op elke voedingsspanning apart geleverd kan worden. Het is duidelijk dat een computervoeding meerdere eindtrappen moet bevatten voor de verschillende spanningen. Op elk van deze rails is er een maximale stroom die geleverd kan worden.

Als de maximale stroom op de 12V rail 5A is, kan je met een 500W voeding niet voorzien in de behoeften van een computer die een vermogen van 200W nodig heeft, maar wel 6A op de 12V rail. Dit kan een belangrijke reden voor prijsverschillen in voedingen zijn. Goedkopere voedingen kunnen typisch meer stroom leveren bij de lagere spanningen en minder bij 12V. Er moet nog worden opgemerkt dat sommige voedingen verschillende rails hebben voor eenzelfde voedingsspanning. Op elk van deze rails is dan een maximale stroom vastgelegd. Het zal wel duidelijk zijn dat je dan best de verbruikers op een zo evenwichtig mogelijke manier over deze rails moet verdelen.

Een laatste opmerking is dat het vermogen van de voeding zo goed mogelijk op het systeem moet worden afgestemd. Uiteraard betekent dit dat je voldoende

piekvermogen nodig hebt, maar zomaar een voeding van 1kW aanschaffen voor een systeem dat 200W nodig heeft is niet meteen een goede keuze.

1.5. Geluid

De geluidsproductie van een computer is in verschillende gebruiksomgevingen liefst zo klein mogelijk. Een belangrijke bron van lawaai wordt gevormd door de verschillende koelingen en in het bijzonder de ventilatoren die hierbij worden gebruikt. Hier blijkt alvast het belang van het rendement van een voeding. Hoe hoger het rendement, des te minder verlies er is. Dit verlies manifesteert zich steeds onder de vorm van warmte.

Meer warmteverlies betekent dus dat er nood is aan een groter koelvermogen. Naast het rendement is ook de grootte van de ventilator belangrijk. Een grotere ventilator zal bij lagere toerentallen voldoende kunnen koelen en daarbij minder lawaai produceren. Er bestaan ook voedingen die volledig passief (zonder ventilatoren) gekoeld worden. Deze produceren uiteraard geen lawaai, maar zijn typisch iets duurder.

1.6. Rendement

Het 'groene' aspect bij pc's komt steeds meer naar voor. Het rendement van de voeding is daarbij een belangrijke factor. Je wil natuurlijk voor elke 100 Watt die je uit het stroomnet haalt, ook 100W prestaties zien. Helaas is dit niet mogelijk: voedingen hebben een rendement dat een stuk lager ligt dan de ideale 100%. Dat verlies uit zich voornamelijk in warmte, die dan weer moet afgevoerd worden. Het spreekt voor zich dan een hoger rendement meestal ook een iets hoger prijskaartje met zich zal meebrengen. Toch is dit het overwegen waard als je een kleine rekenoefening maakt.

Example 1.1. rekenvoorbeeld stroomverbruik

Een computer met scherm die niet erg zwaar belast verbruikt ongeveer 200 Watt. Als je deze pc elke werkdag 10 uur gebruikt, dan komt het verbruik op

0,150 kW x 10 uur per dag x 250 werkdagen= 375 kWh per jaar

Als je daar de prijs tegenover zet die een gemiddeld gezin (bron: VREG, oktober 2012) betaalt per kWh, dan kost deze pc je 375 * 0,2€ = € 75. Een

voeding met een rendement dat 20% beter is zal je dus op jaarbasis makkelijk 15 Euro opleveren.

Het loont dus de moeite om bij de aankoop de voeding zorgvuldig te kiezen. In een bedrijf met honderden desktops begrijp je dat dit een verkoopsargument kan zijn.

Het 80-plus certificatieprogramma probeert voor de consument duidelijkheid te scheppen door voedingen een label te geven naargelang de efficiëntie. De certificatie is echter geen verplichting voor fabrikanten.

Table 1.1. 80 plus certificatie (bron: Wikipedia[2])

	standaard	brons	zilver	goud	platinum	titanium [a]
20% belast	>=80%	>=82%	>=85%	>=87%	>=90%	>=94%
50% belast	>=80%	>=85%	>=88%	>=90%	>=92%	>=96
100% belast	>=80%	>=82%	>=85%	>=87%	>=89%	>=94

[a] bij titanium worden ook nog extra eisen gesteld

Laptops hebben een verbruik dat typisch een flink stuk lager zit. Hoewel ze een voeding hebben die meestal een behoorlijk hoog wattage aankan om de accu op te laden, is het gemiddeld verbruik meestal slechts rond de 30Watt.

Het matige rendement van PSU's is voor een deel eigen aan de opbouw ervan. Omdat veel verschillende eindtrappen nodig zijn voor de verschillende spanningen, is het totale rendementsverlies een accumulatie van de kleinere verliezen bij de deeltrappen.

Ondertussen verlaten sommige grote spelers om die reden de ATX standaard om met eigen oplossingen hogere rendementen te behalen. Google ontwikkelt bijvoorbeeld z'n eigen servervoedingen die door hun eenvoud een veel hoger rendement halen. De eenvoud bestaat erin dat ze slechts 1 spanning aanbieden aan het moederbord: 12V. Als componenten een andere voedingsspanning

[2] http://en.wikipedia.org/wiki/80_Plus

vereisen, worden die waar nodig getransformeerd op het moederbord, wat veel efficiënter kan. Google research publiceerde een paper die schat dat de energiebesparing die je heermee kan behalen op een populatie van 100 miljoen computers 13 miljard kWh betreft op jaarbasis. Dat komt, om je een idee te geven, ongeveer overeen met de opbrengst van de helft van een kerncentrale zoals die in Doel (jaarproductie 22 miljard kWh)

1.7. Problemen

Problemen met voedingen hebben altijd gevolgen voor het volledige systeem, aangezien ze dit volledige systeem van stroom moeten voorzien. Een belangrijke oorzaak van problemen is een te klein vermogen voor het systeem of onvoldoende koeling. Dit probleem uit zich meestal niet in het niet opstarten van het systeem, maar eerder in het onverwacht afsluiten (of eventueel herstarten) ervan. Dit is dan nog het meest aangename gevolg van het probleem. Het is belangrijk om bij dergelijke problemen de voeding en de koeling ervan te controleren.

Minder aangename gevolgen kunnen zijn dat de voeding beschadigd raakt en in het meer dramatische geval dat er rook uit de computerkast komt. Deze kan dan afkomstig zijn van de voeding zelf, maar ook van andere componenten(moederbord, RAM, CPU). Een situatie die de meesten liever vermijden.

Een voeding kan ook slijtage vertonen. In het bijzonder op het vlak van de elektrolytische condensatoren kan er veel verschil zijn tussen voedingen. Minder kwalitatieve condensatoren kunnen uitdrogen (elektrolyt dat verdampt), waardoor ze hun functie minder tot niet meer vervullen en de voeding uiteindelijk rook in plaats van gelijkspanning produceert. Dit gebeurt uiteraard pas na verloop van tijd (afhankelijk van de belasting van de computer).

Sommige voedingen hebben een controlesysteem dat je door middel van geluidssignalen preventief waarschuwt als er problemen dreigen, zoals overbelasting of een gebrekkige koeling.

1.8. Uninterruptible Power Supply (UPS)

Een UPS is een toestel dat het wegvallen van de netspanning kan opvangen. Hiervoor bestaat een UPS uit een accu en een elektronische schakeling die de accuspanning kan omzetten naar een netspanning.

Figure 1.3. Rack-mountable UPS (bron: www.apc.com[3])

Bij het wegvallen van de netspanning zal de UPS ogenblikkelijk de stroomvoorziening overnemen. Voor de aangesloten toestellen treedt er dus geen onderbreking op. Een UPS kan de stroom natuurlijk niet onbeperkt in de tijd overnemen. Hoe lang de UPS dit kan volhouden, hangt af van de accu's en het gevraagde vermogen. Om te vermijden dat apparatuur plotseling en ongecontroleerd stilvalt, heeft een UPS dikwijls ook een interface naar de computer. Deze laat toe dat de UPS de computer 'proper' afsluit op het ogenblik dat de accu-stroom een bepaalde ondergrens bereikt. Een alternatief kan erin bestaan dat de UPS gecombineerd wordt met een dieselgenerator.

De UPS zorgt dan voor de ogenblikkelijke overname van de stroomvoorziening en geeft de generator de nodige tijd om op te starten. Zodra de generator actief is (en de uitgangsspanning gestabiliseerd is), neemt deze de stroomvoorziening op zich.

Een UPS heeft meestal ook een spanningsbeveiliging aan boord die je apparatuur kan beschermen tegen storingen op het elektriciteitsnet. UPS'en vind je in alle prijsklasses, wat vaak te maken heeft met de inwendige opbouw ervan. Er onderscheiden zich enkele grote types.

1.8.1. Online UPS

De online UPS wordt ook wel "double conversion" ups genoemd. Alle stroom die naar de IT-apparatuur gaat, loopt door de UPS. Hierdoor is het niet nodig om te schakelen bij het uitvallen van de stroom. Met de bypass kan je evenwel de ups overbruggen. Dat kan bijvoorbeeld interessant zijn als er onderhoud nodig is. Omdat hierdoor veel gevraagd wordt van alle elektronica (die constant volledig belast wordt), is die een relatief duur concept.

[3] http://www.apc.com

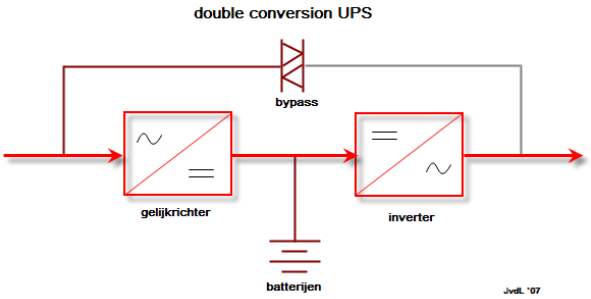

Figure 1.4. Online UPS (© GFDL Joslee 2007)

1.8.2. Offline UPS

Dit type UPS vind je voornamelijk terug bij particuliere ups'en waar kostprijs een belangrijk criterium is. Bij het wegvallen van de spanning, wordt een bypass ingeschakeld. Die procedure duurt enkele milliseconden waarbij je geen uitgangsspanning hebt, en dat moet opgevangen worden door de voeding van je computer of server. Een nadeel van dit type UPS is dat je hem ook niet zonder risico kan testen.

Een ander nadeel is dat in gewone omstandigheden de netspanning rechtstreeks gekoppeld is aan je IT-apparatuur. Als er storingen op het net zitten, zal je IT apparatuur daar hinder van ondervinden. De apparatuur is dus niet beveiligd.

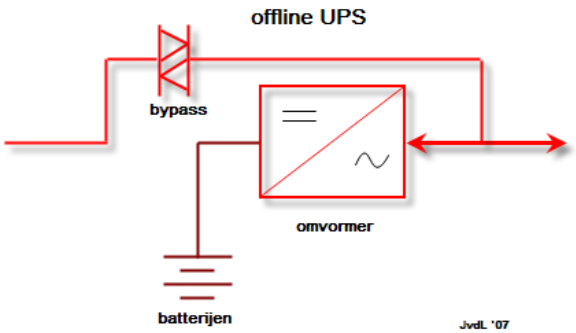

Figure 1.5. Offline UPS (© Joslee 2007 GFDL)

1.8.3. Line-interactive UPS

Deze vorm van UPS vormt een hybride oplossing. In feite gaat het om een off-line UPS waar de line-feed voorzien is van aanvullende filters. Zo ben je zeker dat de spanning die aan je servers aangelegd is, gezuiverd werd van pieken en storingen. In omgevingen waar veel storing optreedt is dat geen overbodige luxe. (bijvoorbeeld fabriekshallen, gebieden met gebrekkige stroomvoorziening)

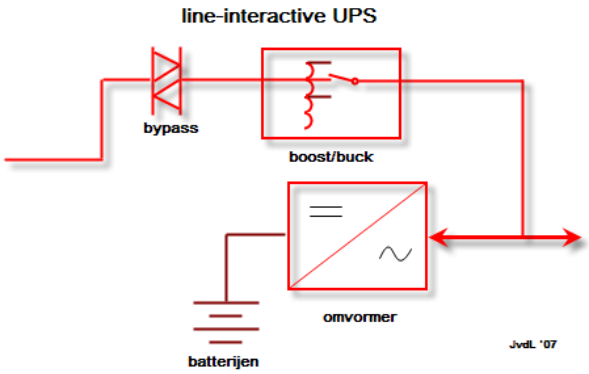

Figure 1.6. Line interactive UPS (© Joslee 2007 GFDL)

1.9. Accu's

Tegenwoordig kunnen we het niet meer hebben over computervoedingen zonder even uit te wijden over accu's. In de trend naar mobiliteit (laptops, tablets, smartphones), vormen die een onmisbare schakel.

1.9.1. Belangrijke parameters

Capaciteit

De capaciteit van batterijen wordt meestal uitgedrukt in Ah (ampère/uur) of mAh (milliampère/uur). Met die eenheid kan je makkelijk accu-packs vergelijken. Een batterij van 6Ah zal bijvoorbeeld in staat zijn om gedurende 6 uur een stroom af te leveren van 1 Ampère, of gedurende bijvoorbeeld 2 uur een stroom van 3 Ampère. Sommige fabrikanten verkiezen echter om hun capaciteiten uit te drukken in Wh, wat vergelijken moeilijk kan maken. Toch kan je eenvoudig omrekenen: Je weet immers dat

P=U*I (Vermogen=Spanning x Stroom)

Willen we dus de stroom I(A) kennen, dan moeten we het vermogen delen door de spanning. Nemen we onderstaand voorbeeld:

Figure 1.7. accu

We kunnen hier dus de capaciteit in Ah bepalen door

I_h = P_h/U = 2,4Wh/3,6V = 0,666Ah (of 666mAh)

Aantal cellen

Een accu wordt opgebouwd uit verschillende cellen. Bijvoorbeeld bij Li-ion accu's kunnen die elk ongeveer 3V leveren. Het spreekt voor zich dat een toename van het aantal cellen zal betekenen dat de totale capaciteit ook toeneemt.

Example 1.2. Oefening

Wat is de capaciteit van je eigen laptopaccu? Stel dat je deze accu gebruikt om een lamp te doen branden die 5Watt verbruikt. Hoe lang zal de lamp branden? Uit hoeveel cellen bestaat je accu-pack?

Laadcurve

Om de optimale kwaliteit van de accu te garanderen over langere termijn is het nodig om de juiste laadcurve te respecteren. Een batterij zal uiteraard stroom nodig hebben om zich op te laden, maar het is niet noodzakelijk zo dat een hogere stroom zal betekenen dat de batterij sneller oplaadt. Het gebruik van de juiste en kwalitatieve adapter is hierbij erg belangrijk.

Memory-effect

Het zogenaamd memory-effect is een term die vaak gebruikt wordt om aan te geven dat bepaalde types batterijen, met NiCd op kop, vaak een effect vertonen

waarbij het lijkt dat de batterijen snel hun capaciteit verliezen als je ze halverwege de ontlaadcyclus terug oplaadt. Dat fenomeen is eigenlijk de verzamelnaam van effecten die worden veroorzaakt door een combinatie van elektrische en chemische processen.

LI-ION accu's

Tegenwoordig is dit zowat het meest voorkomende type in hoogwaardige mobiele apparatuur. Dit type onderscheidt zich door een erg hoge energiedichtheid, en het 'memory-effect' is niet bestaande.

Toch zijn er enkele belangrijke eigenschappen aan dit type, die je beter kent.. Het zwakke punt van Li-Ion: degradatie

Wie een laptop of GSM gebruikt, kent het fenomeen: na enkele jaren is de cyapaciteit van de batterij slechts nog een fractie van wat ze was bij aankoop. Dit fenomeen kan je niet omkeren, maar het kan wel vertraagd worden als je de weet wat de factoren zijn die dit proces versnellen...

Een Li-ion-accu verliest zijn capaciteit het snelst als hij zich in een warme ruimte bevindt, en opgeladen is. Een volledig opgeladen Li-ion accu zal bijvoorbeeld na een jaar rusten in een ruimte waar het gemiddeld 20°C is, 20 procent van zijn capaciteit verliezen.

Bij een temperatuur van iets boven het vriespunt en een lading van ongeveer 40% zal dit type batterij de langste levensduur 'on the shelf' hebben.

Toekomstige ontwikkelingen

Gezien de enorme markt die ontstaan is voor accu's, is er enorm veel druk om betere modellen te ontwikkelen. Daarbij worden bestaande types geperfectioneerd, maar ook nieuwe types ontwikkeld. Zo zijn er de `LiPo` (Lithium polymeer) batterijen die ongeveer 50% efficiënter zijn dan klassieke Li-Ion equivalenten, en ook de brandstofcellen (fuel cells) die mogelijks een oplossing kunnen vormen voor de steeds grotere autonomie-behoefte van toestellen. Omdat veel van deze technieken gebruik maken van erg zeldzame delfstoffen, komen ook geavanceerde technieken met courante materialen in het vizier ter optimalisatie of vervanging, zoals nanostructuren met koolstof. Deze blijven echter toekomstmuziek voor consumentenelektronica...

1.10. Bibliografie bij dit hoofdstuk

CPU

1. Overzicht

In onderstaande tabel worden een aantal processoren van de x86 familie weergegeven met hun belangrijkste eigenschappen. De processor die in de eerste (IBM-)PC werd gebruikt was een 8088. Eigenlijk was dit een 8086 waarvan de databus beperkt werd tot 8 bits in plaats van 16 bits. De enige reden hiervoor was dat op dat ogenblik er geen andere 16bit componenten beschikbaar waren. Deze processorfamilie komt uitgebreid aan bod binnen het vak processorarchitectuur. Het is niet de bedoeling om hier terug te komen op programmeermodel, segmentering, ...

Wel zullen we de evolutie van een aantal eigenschappen bekijken.

Table 2. processoroverzicht

type	jaar	data/ adres-bus	L1 cache (kB)	FSB (Mhz)	Clock(Mhz)	transistoren (miljoen)	technologie (nm)
8088	1979	8/20	-	4,77..8	4,77..8	0.029	3000
8086	1978	16/20	-	4,77..8	4,77..8	0.029	3000
80286	1980	16/24	-	6..20	6..20	0.134	1500
80386DX	1985	32/32	-	16..33	16..33	0.275	1500
80486DX/ SX	1989	32/32	8	25..50	25..50	1.2	1000
80486DX2	1992	32/32	8	25..40	25..80	1.2	800
80486DX4	1994	32/32	8+8	25..40	75..120	1.2	600
Pentium	1993	64/32	8+8	60..66	60..200	3	600
Pentium MMX	1997	64/32	16+16	66	166..233	4.5	350
Pentium Pro	1995	64/36	8+8	66	150..200	5.5	350
Pentium II	1997	64/36	16+16	66/100	300..450	7.5	250

type	jaar	data/ adres-bus	L1 cache (kB)	FSB (Mhz)	Clock(Mhz)	transistoren (miljoen)	technologie (nm)
Pentium III	1999	64/36	16+16	100/133	450..1300	28	130
AMD Athlon	1999	64/36	64+64	200/266	500..2200	37	130
Pentium IV	2001	64/36	8+96	400/533	1400..2800	42	130
AMD 64	2005	64/36	2*512k L2	2000	2,4GHz	233	102
Core duo	2006	64/36	2*2M L2	800	3,6GHz	376	65
Intel Nehalem	2008	64/36	32+32/ core	-	3,2 Ghz	731 (QC)	45/32
Intel Sandy Bridge	2011	64/36	32+32/ core	-	3,8 Ghz.[a]	995 (QC)	32
Intel Ivy bridge	2012	64/36	32+32/ core	-	3,9 Ghz.[a]	1400 (QC)	22
Intel Haswell	2013	64/36	32+32/ core	-	3,9 Ghz.[a]	1400	22

[a] deze waarden zijn niet continue en kunnen pas tijdelijk gehaald worden

2. Technologie en functionaliteit

Een eerste duidelijke evolutie is de toename van het aantal transistors. Volgens de wet van Moore verloopt deze stijging zelfs exponentieel. Elke vierentwintig maanden zou het aantal transistors in een processor verdubbelen . Die toename is uiteraard enkel mogelijk als de transistordichtheid kan toenemen. In het verleden werd hierbij vaak gedacht dat er technische beperkingen zouden opduiken, maar tot dusver blijven fabrikanten slagen om vast te houden aan de ontwikkelsnelheid die geponeerd werd door Gordon More, één van de oprichters van Intel.

> The number of transistors incorporated in a chip will approximately double every 24 months

— Gordon Moore *Electronics Magazine 1965*

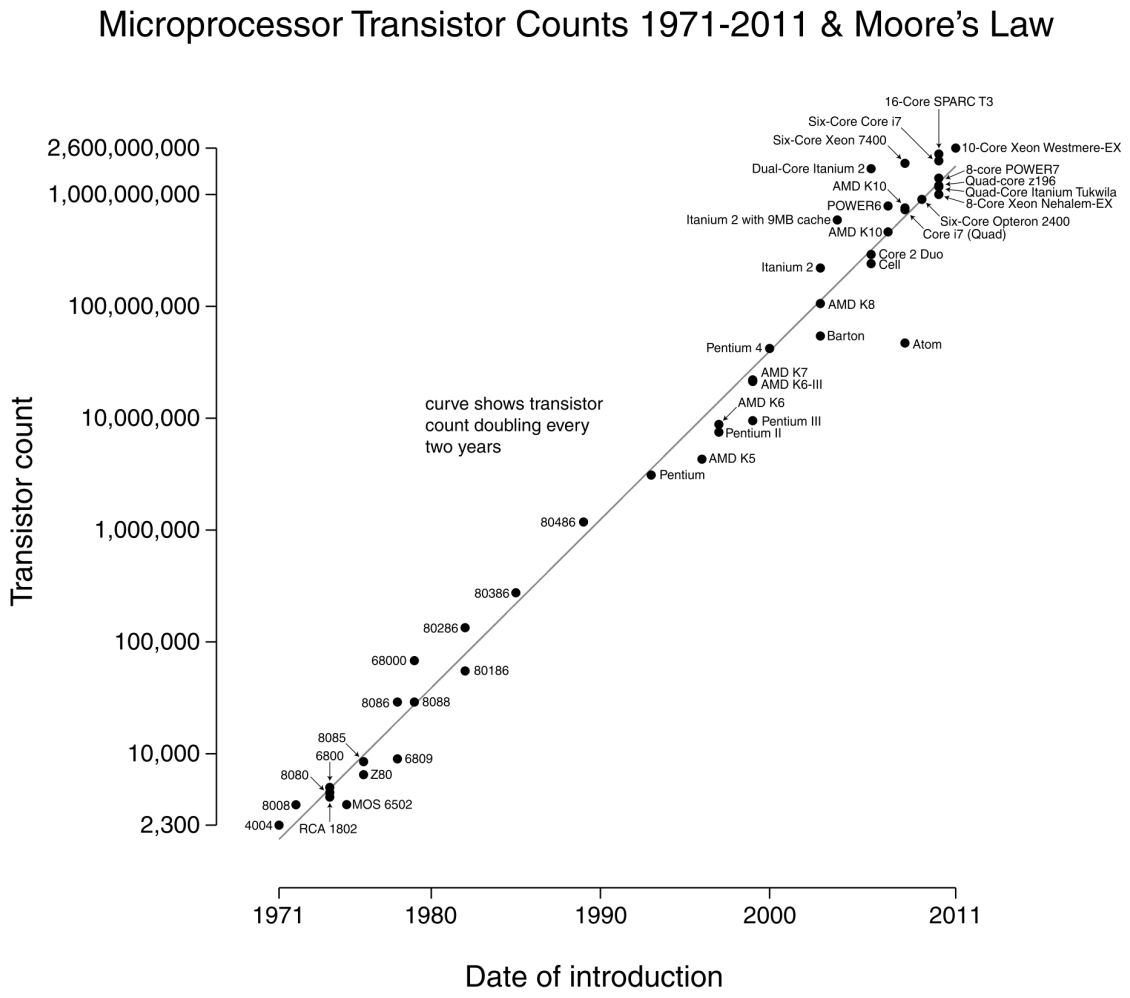

Figure 8. Wet van Moore (CC, Wikimedia Commons)

Met deze stijging van het aantal transistoren gaat uiteraard ook een toename in functionaliteit gepaard. Zo kent een x86 processor vanaf de 80286/80386 (in principe vanaf de 286, praktisch vanaf de 386) twee werkingsmodi: real mode en protected mode. In real mode heeft de cpu dezelfde functionaliteit als een 8086. Onder andere multitasking en virtueel geheugen zijn enkel mogelijk met een protected mode processor. Hier moet nog opgemerkt worden dat processoren nog steeds opstarten in real mode. Het is de taak van het besturingssysteem (of beter de loader ervan) om de processor om te schakelen naar protected mode.
Intel, de grootste producent van x86 processoren, hanteert voor de ontwikkeling een model dat het tick/tock-model genoemd wordt. Afwisselend worden nieuwe modellen uitgebracht met nieuwe funcitonaliteit (tock) en verbeterde technologie (tick). Dit wordt duidelijk in volgende intel roadmap.

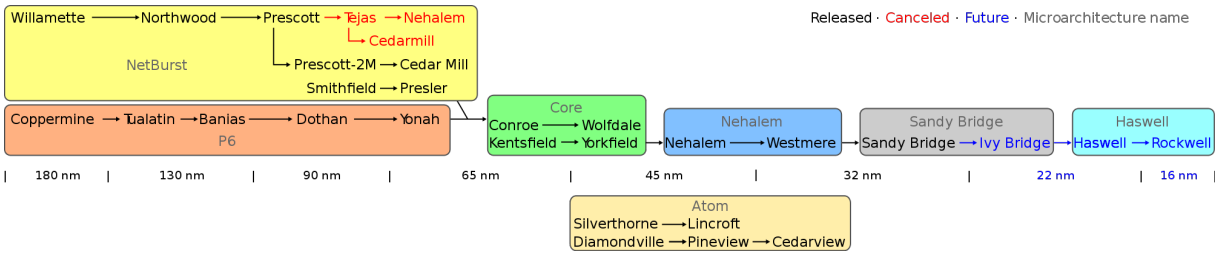

Figure 9. Intel roadmap (copyright 2008-2012 WhiteTimberwolf GFDL)

3. Kloksnelheid

In het bijzonder bij laptops zijn dit twee vervelende problemen: de warmteafvoer vraagt grotere (en dus zwaardere) koellichamen en ventilatoren. Extra verbruik verkleint uiteraard de autonomie van een draagbaar toestel (tijd dat op accu gewerkt kan worden).

4. Processorarchitectuur

4.1. Pipelining

Naast de kloksnelheid werd ook aan de interne opbouw van de processor gesleuteld om hem sneller bepaalde taken te laten uitvoeren. Zo werken processoren instructies niet na elkaar af, maar gedeeltelijk tegelijkertijd. Dit gebeurt in een zogenaamde pipeline. Het is eenvoudig om in te beelden dat terwijl de ene instructie uit het geheugen wordt opgehaald, een andere gedecodeerd kan worden en van nog een andere het resultaat berekend kan worden. Hieronder wordt dit principe grafisch voorgesteld. Helaas is dit principe niet zaligmakend: soms zijn instructies van andere, waardoor er een ‘bubble’ optreedt: een tijdspanne waarin de processor verplicht moet wachten.

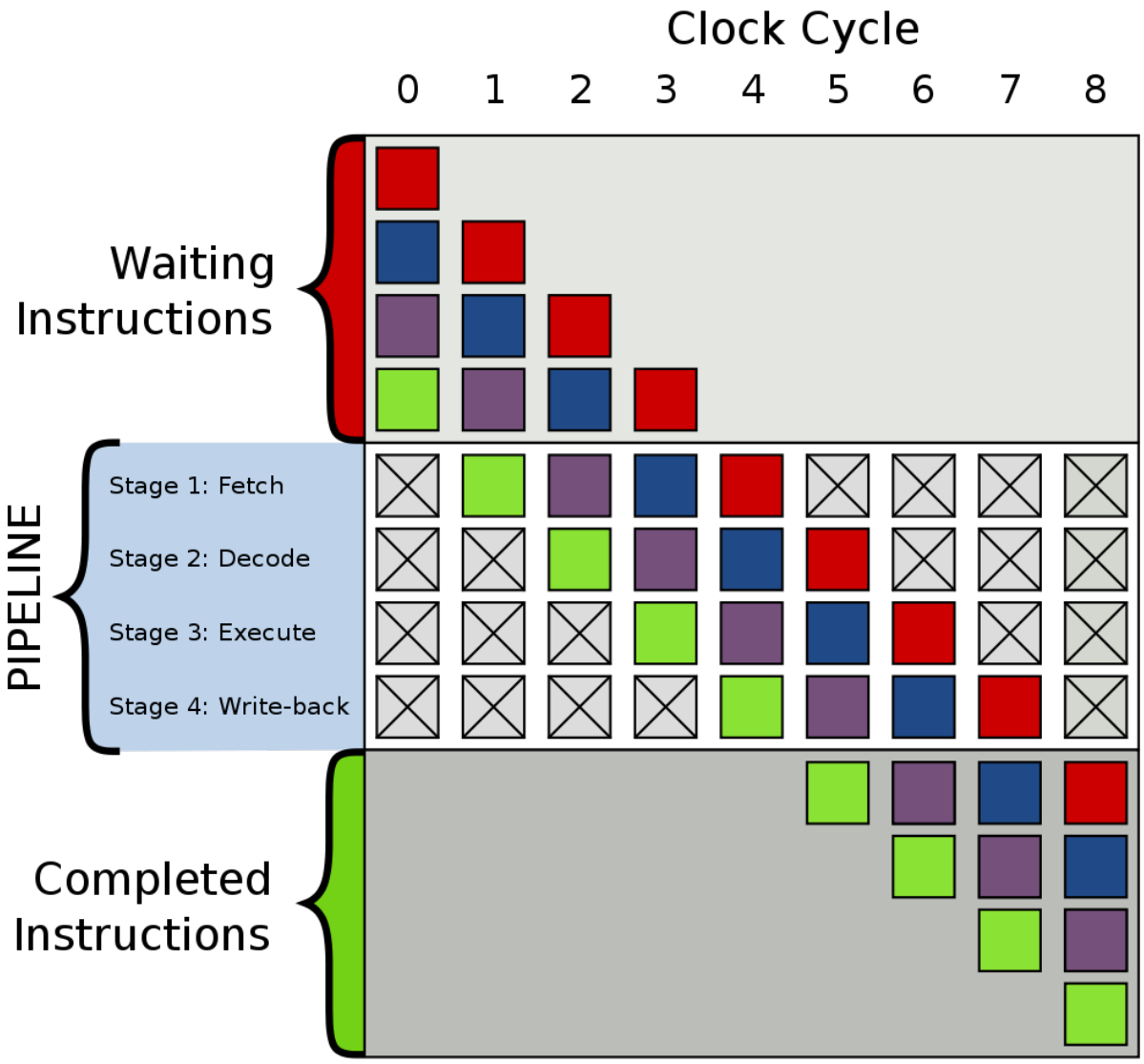

Figure 10. processor pipeline (CC mediawiki)

4.2. Superscalaire processoren

Als dit principe verder gedreven wordt, kunnen stappen die veel tijd in beslag nemen dubbel uitgevoerd worden. Men spreekt dan over een superscalaire processor. In afbeelding 8 en afbeelding 9 worden de blokschema's getoond van de Ivy bridge en de Athlon Bulldozer. In deze blokschema's is duidelijk te zien hoe er verschilllende eenheden zijn die berekeningen kunnen maken, waardoor verschillende instructies tegelijkertijd uitgevoerd kunnen worden. Een belangrijke uitdaging hierbij vormen voorwaardelijke sprong instructies. Aangezien pas bij de uitvoering van de instructie geweten is of de sprong uitgevoerd wordt of dat gewoon de volgende instructie wordt uitgevoerd. In het schema zijn hiervoor branch prediction eenheden voorzien.

Meer details over hun werking en de principes van pipelining en superscalaire architecturen krijg je in het vak "microprocessoren".

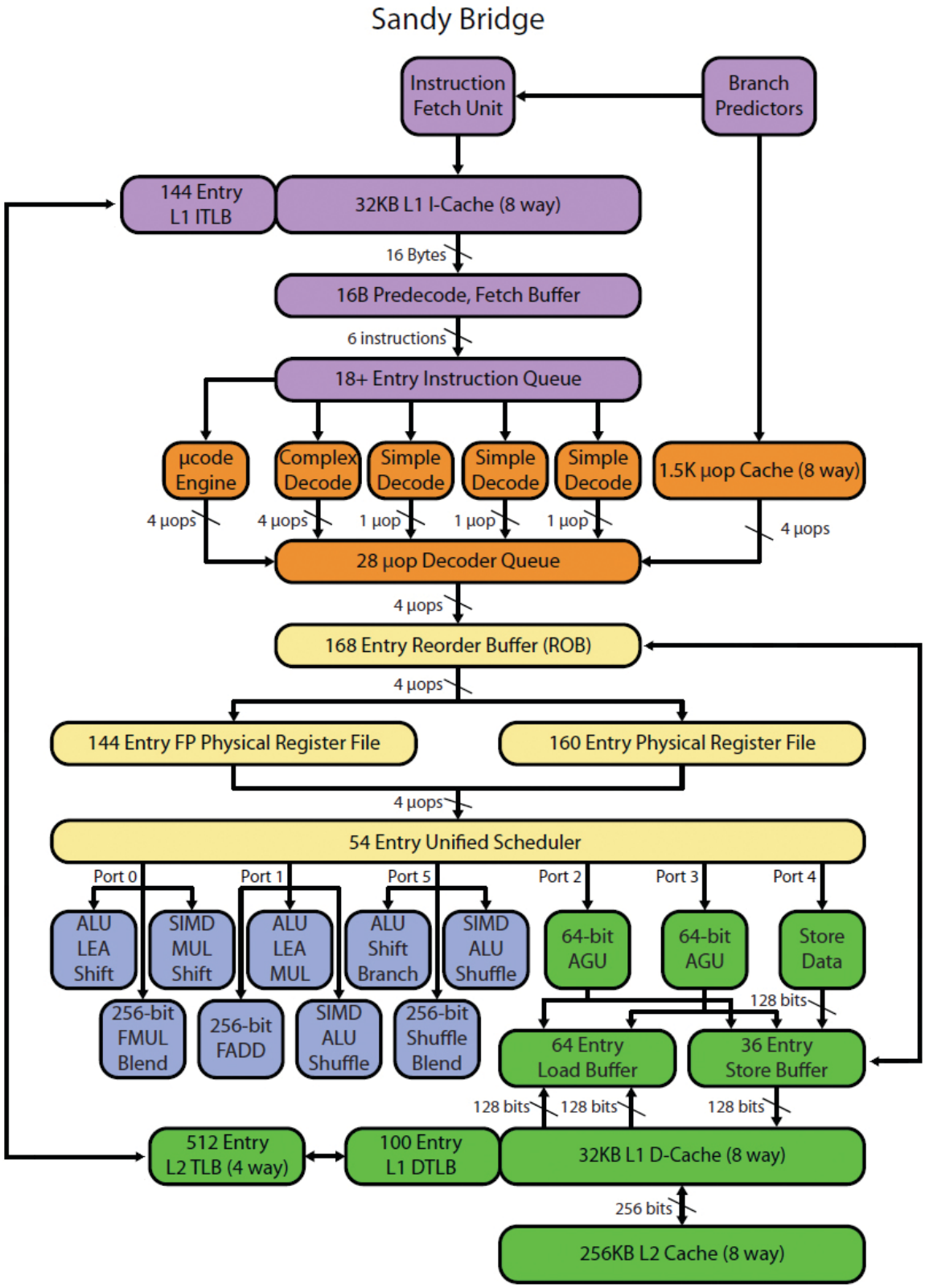

Figure 11. Sandy bridge microarchitectuur

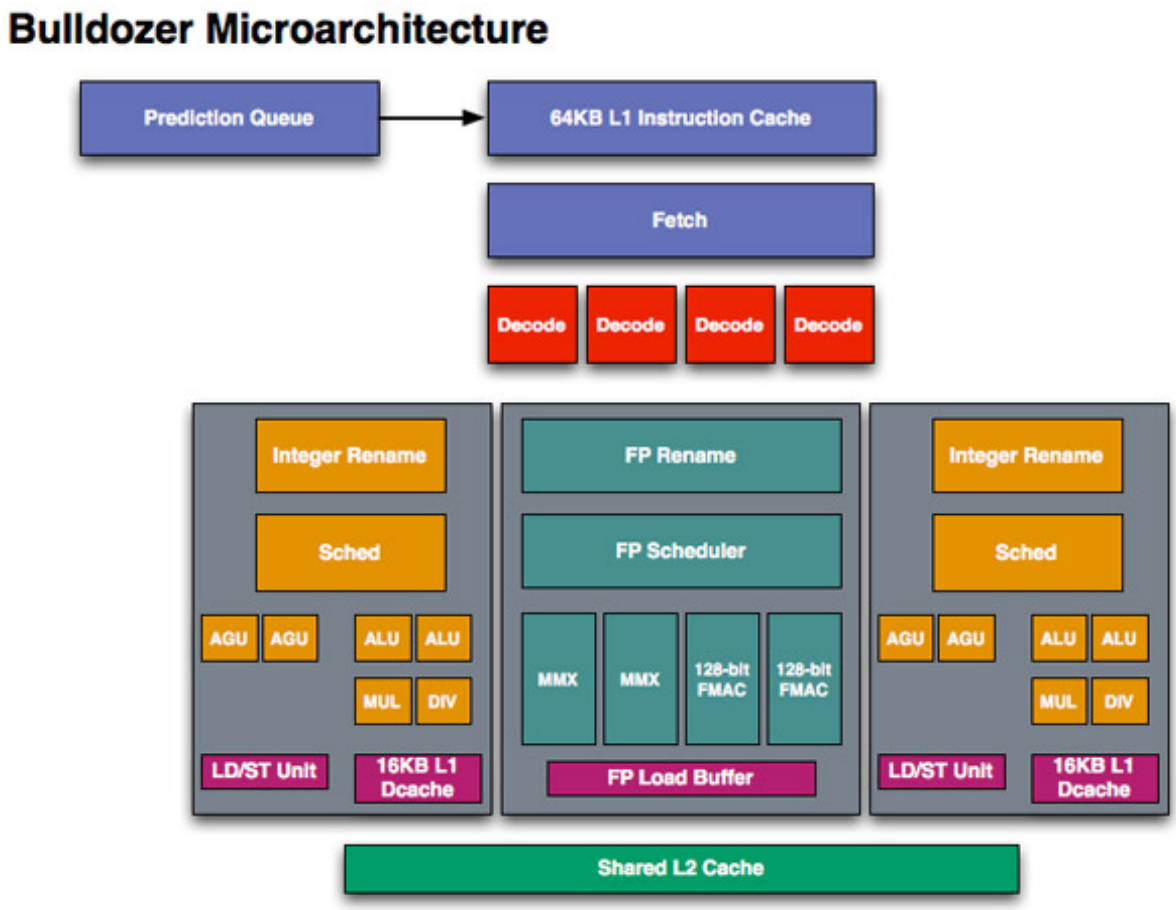

Figure 12. AMD bulldozer architectuur (copyright AnandTech)

Multicore processoren zijn al geruime tijd niet meer weg te denken. Hexacores en octocores zullen de komende jaren eerder regel dan uitzondering worden. Je zou dit een verder doorgedreven vorm van een superscalaire architectuur kunnen noemen. In plaats van delen van de processor te ontdubbelen, wordt een volledige processor ontdubbeld. Een grote moeilijkheid bij deze werkwijzes is om de caches op elkaar af te stemmen. Een probleem dat duidelijker zal worden in het volgende hoofdstuk.

Net zoals een superscalaire architectuur pas voordeel geeft als de verschillende eenheden tegelijk gebruikt worden, zal een dual core pas voordeel geven als meerdere cores tegelijk werk verrichten. Dit kan als er bijvoorbeeld verschillende programma's tegelijk actief zijn of als de software zodanig geschreven is, dat ze bestaat uit verschillende threads die naast elkaar (en dus tegelijk door verschillende processorkernen) kunnen uitgevoerd worden.

Een simpel voorbeeldje om de beperkingen van een multicore processor aan te tonen: als je een eenvoudige toepassing een rekenintensieve opdracht laat

uitvoeren, dan zal een multicore processor slechts een deel belast worden. Eén processorkern verricht namelijk al het werk.

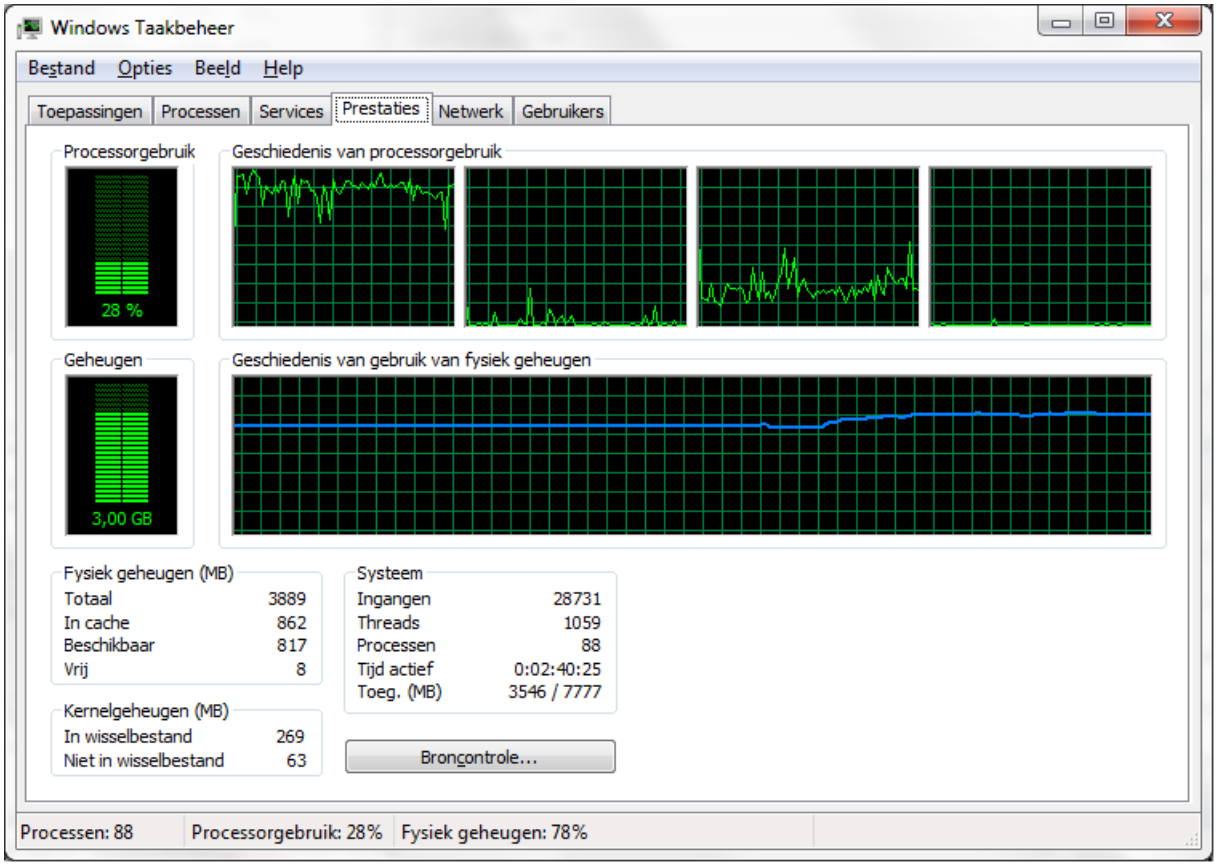

Figure 13. single threaded applicatie op multicore processor

Het voordeel van de multi-core merk je pas als je tegelijk nog een ander programma probeert te gebruiken. Dat zal met een multi-core vlot lukken, in tegenstelling tot een single core. Het OS zal op zoek gaan naar de minst belastte core, en het nieuwe proces daarop uitvoeren.

4.3. Cache

Een andere eigenschap die plots opduikt en doorheen de processorgeschiedenis steeds toeneemt is het cache geheugen. De toename van het cache volgt de trend van alle soorten geheugens die in een pc te vinden zijn. Dit is een gevolg van de eerder opgemerkte trend dat de processor veruit het snelste onderdeel is in het systeem, dat zo optimaal mogelijk benut moet worden. Naarmate data en programma's steeds groter werden, werd ook het belang van geheugen groter. Tot de intrede van de grafische interface was de belangrijkste parameter in het systeem de kloksnelheid van de processor. Met de intrede van de grafische interface was een groter geheugen soms te verkiezen boven een hogere

kloksnelheid. Het belang van cache geheugen is ook duidelijk als je de budget- en performance-processoren van fabrikanten met elkaar gaat vergelijken.

In onderstaand lijstje staan enkele desktop en serverprocessoren opgelijst. Je merkt dat ze qua kloksnelheid niet voor elkaar moeten onderdoen, maar dat de hoeveelheden cache wel verschillen.

De werking van de cache wordt verder in detail besproken in het derde hoofdstuk.

Table 3. cache in desktop en serverprocessoren (actuele topmodellen, feb 2014)

CPU	doel	cache	maxCPU	#cores/ threads
Atom 7560	mobile	512KB	2.13 Ghz	1/2
i7-4771	desktop	8MB	3.50 Ghz	4/8
E7-8893v2	server	37.5MB	3.40 Ghz	15/30

5. APU, SoC

De wet van Moore impliceert dat steeds meer mogelijk is op eenzelfde oppervlakte substraat. Die ruimte wordt ingenomen door bijvoorbeeld meerdere cores te huisvesten op eenzelfde processor, maar dat is maar een deel van het verhaal.

Het is namelijk ook zo dat men probeert om steeds meer functionaliteit die voorheen op andere plaatsen op het moederbord te vinden was, te verzamelen op eenzelfde chip.

Daar zijn een aantal goede redenen voor te bedenken:

- het aantal verschillende chips (en dus kostprijs) op een moederbord kan zo teruggedrongen worden
- als alle componenten dicht bij elkaar zitten, zijn geen *trage* bussen nodig tussen deze onderdelen
- de oppervlakte die nodig is om het systeem te bouwen verkleint zo, een belangrijk argument bij de ontwikkeling van mobile devices.

testje in sidebar

Bij recente processoren zit bijvoorbeeld steeds vaker een grafische chip ingebouwd. Dan spreekt men niet meer over CPU, maar over `APU` (=advanced processing unit) om dit verschil in de verf te zetten.

Het integratieproces gaat soms zo ver dat je kan spreken van een *System On A Chip*: alle belangrijke onderdelen (cpu, gpu, IO) zitten dan verzameld op één enkele chip. De rol van secundaire chips ("de chipset") wordt dus steeds kleiner.

> Op welke SoC is jouw telefoon gebaseerd?

6. Montage

Bij de montage van een processor moet je enkele zaken in acht nemen.

- De processor moet compatibel zijn met het moederbord. Meestal kom je dit te weten door de socket van de processor te vergelijken met die van het moederbord.
- De processor plaatsen moet gebeuren zonder het uitoefenen van kracht: de processor valt normaalgezien in z'n socket (ZIF: zero insertion force), waarna je hem kan inklemmen.
- Mobiele processoren zijn vaak vast op het moederbord gemonteerd, vervangen is dan onmogelijk.

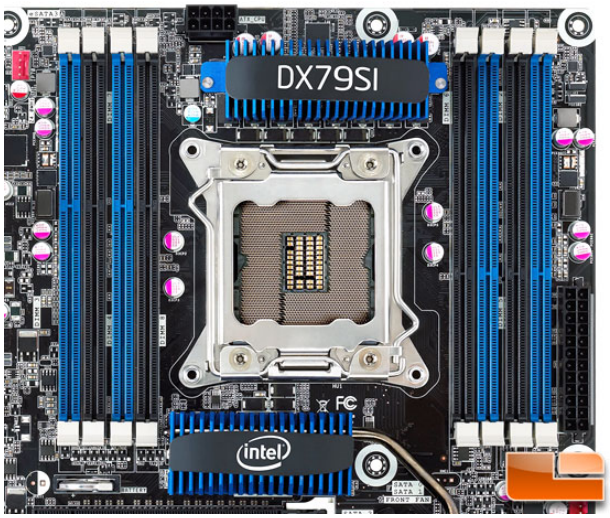

Figure 14. LGA2011 socket zonder processor

Tweede belangrijk aandachtspunt bij de installatie van een processor is dat gezorgd moet worden voor voldoende koeling. Dit betekent dat gezorgd moet

worden voor een voldoende grote koelvin en ventilator en dat er goed contact is tussen de chip en de koelvin. Hiervoor moet eventueel koelpasta aangebracht worden. Een slecht gekoelde processor kan aanleiding geven tot een instabiel werkende computer en in het meest dramatische geval tot een beschadigde processor.

7. Processoren van de toekomst

Voorspellingen maken is geen sinecure. De trends die ingezet zijn, zullen vermoedelijk nog een hele poos verder gaan, met een verdere miniaturisatie en toename van efficiëntie tot gevolg. Een kaper op de kust voor de x86 technologie die momenteel monopolist is op de PC-markt, is de ARM architectuur. Hoewel deze absoluut niet nieuw is (eerste ontwerpen midden jaren 80), biedt deze processorfamilie grote voordelen:

- Deze architectuur is steeds ontworpen voor toestellen met een laag verbruik. Het succes op de mobiele markt (iPAD2,3, nagenoeg alle android smartphones, consumer elektronica, ...)
- Deze architectuur is in licentie bij de meeste chipbakkers
- Door een RISC (Reduced instruction set computing) architectuur van nature efficient

De kans dat de RISC architectuur op korte termijn succesvol wordt op de desktopmarkt is gering, en ook het omgekeerde kan gezegd worden over CISC (x86) op mobiele devices. Voor specifieke servertoepassingen zijn er wel aankondigingen gebeurd[1] door bijvoorbeeld AMD, die zich hier sterk wil in specialiseren en profileren.

Toch lijken deze twee werelden van gespecialiseerde ARM en meer universele x86 naar elkaar toe te groeien, en zullen de grenzen ongetwijfeld snel vervagen. Microsoft heeft bijvoorbeeld eind 2012 z'n RT tablet vrijgegeven, met ARM SOC. Uiteraard zal software die gecompileerd werd voor x86 op dit soort toestellen niet werken.

8. Bibliografie bij dit hoofdstuk

[1] http://www.anandtech.com/show/7724/it-begins-amd-announces-its-first-arm-based-server-soc-64bit8core-opteron-a1100

[INTEL] Intel. http://www.intel.com/content/www/us/en/history/museum-gordon-moore-law.html.

Chapter 2. Het Geheugen

2.1. Overzicht

Het belang van geheugen is tijdens de cursus computertechniek al voldoende gebleken. Zoals bekend worden zowel uit te voeren instructies als data opgeslagen in dit geheugen. (cfr de *von Neumann architectuur*) Dit legt twee belangrijke behoeften bloot: snelheid en grootte.

Snelheid is belangrijk omdat het geheugen de processor moet voorzien van uit te voeren instructies. Een snelle processor met traag geheugen geeft een traag systeem.

Grootte is dan weer belangrijk omdat zowel programma's als de te bewerken data aanzienlijk zijn toegenomen. Bovendien moeten in een multitasking omgeving meerdere programma's en dus ook grotere hoeveelheden data opgeslagen worden in het geheugen.

Als de verschillende soorten geheugens bekeken worden, wordt ook wel gesproken van de geheugenpiramide. Deze term wordt gebruikt omdat de hoeveelheid geheugen afneemt naarmate je stijgt in de piramide. De reden hiervoor is dat ook de prijs per byte toeneemt naarmate je hoger gaat in de piramide.

Figure 2.1. Geheugenpiramide

Daarnaast neemt de snelheid van het geheugen toe in de richting van de top van de piramide. In deze voorstelling is het begrip geheugen vrij ruim geïnterpreteerd. We zullen het in dit hoofdstuk niet hebben over registers of opslagmedia, wel over de technieken die toegepast worden om het geheugen voldoende snel en groot te maken.

2.2. Lokaliteitsprincipe

Bij het bestuderen van de technieken die gebruikt worden om het geheugen groot en snel te maken, zullen we regelmatig beroep doen op het zogenaamde lokaliteitsprincipe. Dit principe zegt dat als het data op een bepaalde locatie wordt aangesproken, de kans groot is dat volgende geheugentoegangen op naburige adressen doorgaan.

Een paar voorbeeldjes die deze stelling moeten ondersteunen: * Sequentiële uitvoering van instructies in een programma * Lussen in code, er is een regel die zegt dat programma's 90% van hun tijd spenderen met het uitvoeren van 10% van de code, dus worden regelmatig dezelfde stukken geheugen aangesproken * Bestanden op een schijf worden, indien mogelijk, in opeenvolgende clusters opgeslagen.

2.3. Soorten geheugens

Tussen de verschillende soorten geheugens kan een onderscheid gemaakt worden op een aantal vlakken.

2.3.1. Behuizing

Naarmate de capaciteit van het geheugen steeg, werd dit te duur en ging men over op geheugenmodules. Daarnaast spreekt men soms van het gebruik van geheugenbanken. Een geheugenbank op een moederbord bestaat uit een of meerdere sockets of geheugenvoeten (insteekplaatsen voor geheugenchips). Het aantal sockets per geheugenbank hangt af van de uitvoeringsvorm van het gebruikte geheugen en van de breedte van de databus. Een geheugenbank heeft dezelfde breedte als de databus die voor de aansluiting op de datalijnen zorgt.

Niet alle geheugenbanken moeten gevuld zijn maar iedere geheugenbank waar geheugen in geplaatst werd, moet volledig gevuld zijn. In moderne systemen vult een module een volledige geheugenbank en is deze dus automatisch vervuld. Een geheugenmodule wordt gekenmerkt door het aantal contactpunten (pins), de werkspanning en het soort geheugenchip. Het is de geheugencapaciteit van alle chips samen die de capaciteit van de module bepalen.

Langs beide zijden van een geheugenmodule bevinden zich contactpunten. Indien deze contactpunten inwendig verbonden zijn spreekt men van een SIMM (Single Inline Memory Module). Indien deze contactpunten afzonderlijk werken en niet verbonden zijn spreekt men over een DIMM (Dual Inline Memory Module). Een DIMM biedt op dezelfde afstand veel meer contactpunten en wordt dan ook toegepast in de moderne modules, die onder andere voor de steeds breder wordende databussen extra contactpunten nodig hebben.

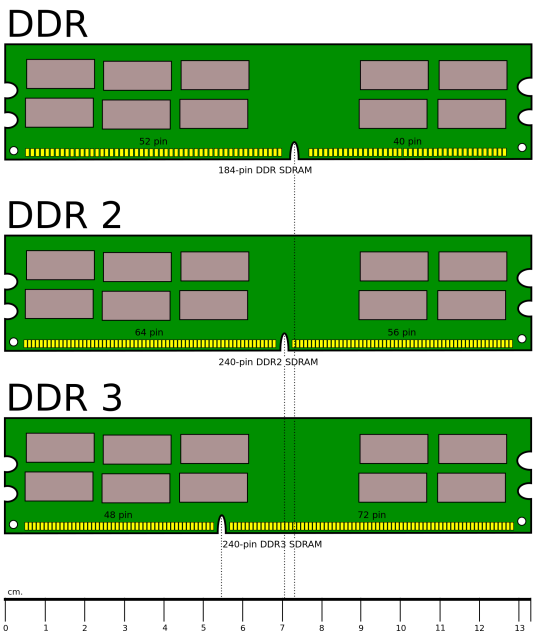

Figure 2.2. Courante DDR-dimm modules (Wikimedia public domain)

Een variante van DIMM is so-DIMM (small outline), een miniatuurversie van DIMM, specifiek geschikt voor mobiele apparatuur als laptops.

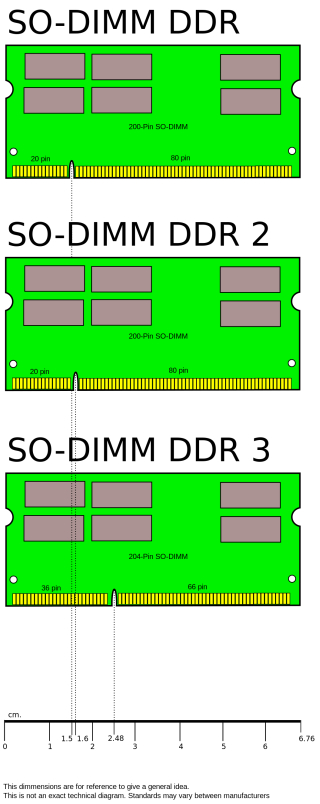

Figure 2.3. SO-DIMM modules (Wikimedia public domain)

2.4. Technologie

Een zeer belangrijk onderscheid tussen geheugens kan gemaakt worden op het vlak van de technologie die gebruikt werd om de geheugencellen te bouwen. Er zijn twee soorten: statisch en dynamisch geheugen. Statisch geheugen is opgebouwd uit actieve geheugencellen (flipflop schakelingen). Deze vragen een grotere complexiteit bij het IC ontwerp en zijn dus duur. Anderzijds zijn ze zeer snel. Uit deze eigenschappen kan je afleiden dat ze hoog in de piramide worden toegepast. Meer bepaald gebeurt dit bij de snelle cache geheugens.

Figure 2.4. voorstelling statisch geheugen

Dynamisch geheugen bestaat in essentie eigenlijk gewoon uit een condensator. Als je over een condensator een gelijkspanning aanbrengt en die vervolgens wegneemt, kan je achteraf nog meten welk spanning erop stond. Dit is een vorm van geheugen. Deze geheugens hebben twee belangrijke nadelen. Ten eerste zal een leesoperatie de condensator ontladen.

Figure 2.5. Voorstelling leescyclus dynamisch geheugen

Een leesoperatie is met andere woorden destructief. Ten tweede bestaan er geen perfecte condensatoren en vertoont dit soort geheugen dus ook een lek. Dit betekent dat mettertijd de inhoud van het geheugen verloren gaat, tenzij die ververst wordt. Bij dit soort van geheugen is dan ook een regelmatige refresh noodzakelijk. Vergeleken met statisch geheugen is dynamisch geheugen trager. Het statisch geheugen is een actieve schakeling en kan dus stroom sturen of opnemen als het gelezen wordt. Dynamisch geheugen kan niet echt stroom sturen, het is de condensator die ontladen of opgeladen wordt. Anderzijds is dynamisch geheugen dan weer goedkoper, waardoor het toegepast wordt op een lager niveau in de piramide. Meer bepaald is dit de technologie die in geheugenmodules wordt gebruikt. Deze zijn ook gangbaar bekend onder de term RAM of DRAM.

2.5. DRAM technologie

2.5.1. Gemultiplexte adresklemmen

Dynamische RAMs hebben vanwege de grote densiteit meestal ook een grote capaciteit op de chip (tegenwoordig tot 16 Gbit per chip). Een dergelijke capaciteit betekent ook dat er heel wat adressignalen noodzakelijk zijn om een welbepaalde geheugencel te selecteren. Wanneer elk signaal op een aparte pin zou aangesloten worden, zou het noodzakelijk zijn om zeer grote behuizingen te gebruiken. Om dit probleem te omzeilen, wordt gebruik gemaakt van gemultiplexte adreslijnen: het volledige adres wordt opgesplitst in een Column Address een Row Address.

Deze twee adresgedeeltes worden de een na de ander aangeboden aan de adresklemmen van het IC. Hierbij wordt gebruik gemaakt van de RAS- en CAS-klem om de twee adresgedeelten te latchen. De snelheid van het geheugen wordt in grote mate bepaald door de toegangstijd tacc.

2.5.2. Destructieve leescyclus

Eerder werd al aangegeven hoe een leescyclus de data op de condensatoren zal vernietigen. Dit is uiteraard een onaanvaardbare situatie. De oplossing ligt voor de hand. Als data gelezen is, wordt dezelfde data nadien terug weggeschreven, zodat de originele toestand hersteld wordt. Uiteraard is het niet verstandig

deze taak aan de processor toe te wijzen, het is iets wat in de geheugenchips zelf geregeld moet worden. Het geheugen is, zoals in vorige paragraaf werd aangegeven, opgebouwd als een matrix van rijen met een welbepaald aantal kolommen. Naast deze matrix van dynamische geheugencellen is er ook een rij van statische geheugencellen. Op het ogenblik dat een rij-adres aangelegd wordt, zal de dynamische rij gekopieerd worden naar de statische rij. Hierbij verliest de dynamische rij dus haar inhoud. Vervolgens kan de gewenste cel gelezen worden en daarna wordt de inhoud van de statische rij weer naar de matrix gekopieerd. Hierdoor wordt de inhoud van het geheugen hersteld.

2.5.3. Refresh

Met deze kennis wordt ook duidelijk hoe een refresh georganiseerd kan worden. Op regelmatige tijdstippen zal een zogenaamde RAS-only cylcus uitgevoerd worden. Hierbij wordt eigenlijk elke rij geselecteerd, gekopieerd naar de statische rij en weer weggeschreven. Hierdoor is de originele inhoud weer op peil gebracht, op voorwaarde dat deze cyclus voldoende regelmatig herhaald wordt. Met deze manier moet niet elke cel afzonderlijk gerefreshed worden, maar wordt een volledige rij ineens hersteld.

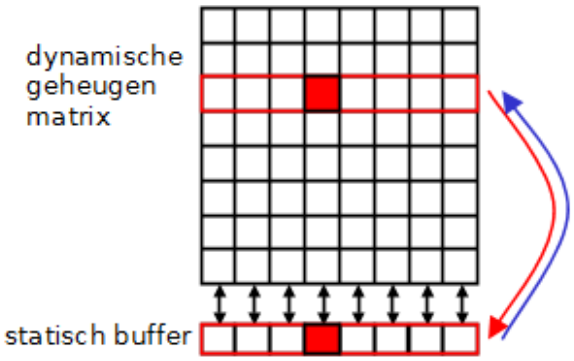

Figure 2.6. dynamisch geheugen met statische buffer

2.5.4. Bandbreedte bij DRAM

DRAM heeft, zoals je verder zal lezen, de eigenschap te werken met cyclussen. Om te berekenen wat de effectieve bandbreedte is (=geheugendebiet) hoor je steeds dezelfde benadering te maken:

"Aantal bytes die getransfereerd worden bij een cyclus"//"tijdsduur van een cyclus" = "bandbreedte"

Deze erg eenvoudige benadering wordt bij de verschillende types geheugen die volgen telkens toegepast.

2.6. Fast Page DRAM (FP-DRAM)

Het nadeel is dat een lees- of schrijfcyclus langer wordt. Het kost immers extra tijd om de adressen na elkaar door te geven. FP-DRAM verbetert de snelheid door cycli te combineren. Zoals aangehaald bij het lokaliteitsprincipe gaan opeenvolgende cycli meestal door op naburige cyclusadressen. De kans dat meer dan een byte gelezen wordt in dezelfde rij, is dus vrij groot. FP-DRAM maakt hiervan gebruik door eenmaal een rij-adres op te geven en vervolgens een kolom te selecteren en deze te lezen of te schrijven. Onmiddellijk hierna wordt een tweede kolom geselecteerd en wordt deze gelezen of beschreven, vervolgens kan een derde kolom geselecteerd worden...Op die manier worden een aantal cycli vermeden. Het zal relatief lang duren vooraleer het eerste geheugenwoord gelezen kan worden, terwijl de volgende minder tijd vragen.

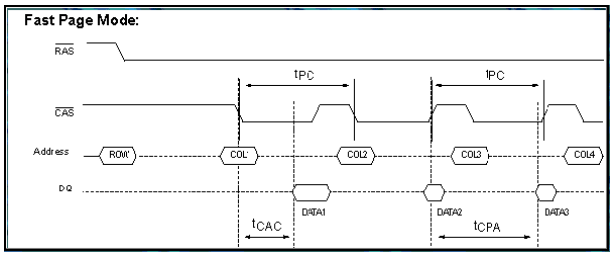

Figure 2.7. Fast page DRAM

FP-DRAM werd gebruikt tot busfrequenties van 66 MHz.

Op een computersysteem met een 486 processor (32-bit databus) met 5-3-3-3 FP-DRAM geheugen aan 66Mhz betekent dit dat het maximale geheugendebiet gelijk is aan:

(4 "bytes"/"transfer" xx 4 "transfers"/"burst" xx 66 "Mcycli"/"sec")/(14 "cycli"/"burst")= 75 "MB"/"sec"

2.7. EDO RAM

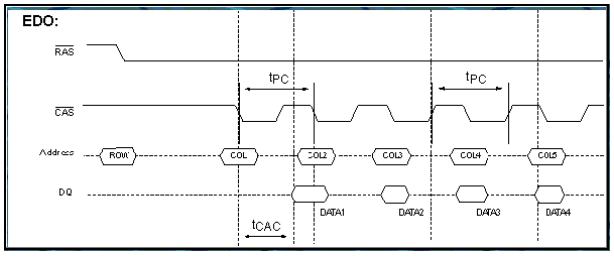

Figure 2.8. EDO-RAM

Extended Data Out-RAM is een aanpassing van het Fast Page-concept. Daarbij moest de memorycontroller wachten met het aanbieden van een nieuw kolomadres tot de vorige data gelezen waren. Bij EDO-RAM blijven de data op de uitgangen van het geheugen nog een tijd langer beschikbaar (zelfs tot na het aanbieden van het volgende kolomadres). Hierdoor wint men tijd: terwijl de data gelezen worden, kan men al het volgende kolomadres aanleggen. EDO-RAM kon gebruikt worden tot een busklok van 75 MHz met een timing van 5-2-2-2 klokcycli. Als we EDO-RAM dan nog combineren met een 64-bit bus (Pentium) geeft dit een maximaal debiet van 218 MB/s

:(8 "bytes"/"transfer" xx 4 "transfers"/"burst" xx 75 "Mcycli"/"sec")//(11 "cycli"/"burst")=218 "MB"/"sec"

2.8. Synchronous DRAM

Figure 2.9. afbeelding 20 Leesoperatie bij SD-RAM (bron: Micron)

Bij SDRAM gaat men nog een stap verder met het lokaliteitsprincipe. In plaats van uit te gaan van het lezen van naburige kolommen, wordt nu vertrokken van het idee dat opeenvolgende kolommen uitgelezen zullen worden. In het deel over cache geheugens zal duidelijk worden dat het RAM geheugen effectief op deze manier wordt aangesproken. De cyclus kan nu aangepast worden tot het aanleggen van een rij-adres, het selecteren van een kolom en vervolgens het inlezen of naar buiten brengen van een aantal opeenvolgende kolommen. Die kunnen naar buiten gebracht worden op het tempo van de klok. Vandaar spreekt men over synchroon DRAM. In de afbeelding 20 krijgen we een timing van 2-1-1-1. Daarnaast is SD-RAM geschikt voor busfrequenties tot 133 MHz (PC133), wat neerkomt op een maximaal debiet van 851 MBps.

(8 "bytes"/"transfer" xx 4 "transfers"/"burst" xx 133 "Mcycli"/"sec")//(5 "cycli"/"burst") = 851 "MB"/"sec"

2.8.1. DDR SDRAM - DDR2 - DDR3

Principieel werkt DDR op dezelfde manier als SDRAM. Er wordt nog steeds een rij-adres en een kolomadres aangelegd, waarna meerdere opeenvolgende cellen worden uitgelezen. Het verschil zit in het tempo waarop dit gebeurt. Bij Double Data Rate wordt data naar buiten gebracht op stijgende en dalende flank van de klok. Om dit te kunnen bereiken wordt gebruik gemaakt van een prefetch buffer.

In elke cyclus worden nu 2 bits getransfereerd naar het prefetch buffer, dat de data dan aan een dubbele snelheid naar buiten kan brengen.

afbeelding 21 write cyclus bij DDR-RAM (bron: Micron)Het voordeel van deze werkwijze is een hogere maximale bandbreedte (datasnelheid bij het effectief overbrengen van data). Het nadeel zit in een hogere latentietijd. Tussen het aanleggen van de adressen en het naar buiten brengen van de data verloopt iets meer tijd. De snelheid van de modules wordt uitgedrukt op een aantal verschillende manieren. Een eerste manier is in de naam, waar twee verschillende mogelijkheden bestaan. DDR400 en PC3200 duiden op hetzelfde soort geheugenchips. De 400 duidt de kloksnelheid aan (2x200MHz), de 3200 duidt de maximale transfersnelheid aan.

Op een 64-bit databus is die

8 "bytes"/"transfer" xx (2 xx 200 "MHz") = 3200 "MB"/"sec"

Er kan echter nog veel verschil zijn tussen twee PC3200 modules. De werkelijke snelheid hangt namelijk ook af van de totale latentietijd. Die kan op verschillende manieren worden aangegeven, maar een gangbare manier is het opgeven van vier getallen: TCL-Trcd-Trp-Tras.

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

In elke leescyclus is zeker Trcd en TCL nodig. Indien bursts uit verschillende rijen nodig zijn, dan is ook Trp belangrijk.

Figure 2.10. DDR-timing Tcl=2 (bron: Micron)

Een voorbeeld: PC3200 geheugen met parameters 2-2-2-6 heeft voor een burst met vier transfers van 8 bytes 2+2+4 x 0.5=6 klokcycli van 200 MHz nodig. Dit geeft een snelheid van 1067 MB/s.Voor twee dergelijke opeenvolgende transfers zijn

2(2+2+4 xx 0.5) + 2 = 14 klokcycli van 200 MHz nodig. Dit geeft 914 MBps.

Bij DDR kan ook het aantal cellen dat in een burst gelezen wordt variëren. Hetzelfde geheugen dat in een burst 8 transfers van 8 bytes uitvoert, haalt een snelheid van 1600MBps.Tras is in dit verhaal niet naar voor gekomen. Tras bepaalt de tijd waarin de volgende rij nog niet geladen mag worden. Deze moet groot genoeg zijn om de buffer niet te overschrijven voordat het volledig getransfereerd is over de databus. Deze parameter moet minimaal Trcd + TCL + 1 bedragen. Indien de parameter te klein is gaat uiteraard data verloren.

Opvolgers van DDR zijn DDR2 en DDR3. Behalve verbeteringen op het vlak van klokfrequenties en spanningen (DDR2 en DDR3 gebruiken telkens lagere spanningen) is het grootste verschil dat het prefetch buffer vergroot. Bij DDR2 worden vier bits gebufferd [3], bij DDR3 acht. DDR4, dat er binnenkort zit aan te komen, zal dit nogmaals verdubbelen.

Het gevolg is dat deze geheugens nog sneller data naar buiten kunnen brengen (hogere maximale transfer), maar dat dit weer een hogere latentietijd met zich meebrengt. Hou wel in gedachten dat slechts een klein stukje van het geheugen aan deze hoge snelheid werkt. Intern wordt nog steeds een relatief lage snelheid gebruikt om de cellen te beschrijven, maar door de buffers kan data toch aan een dubbele (DDR), viervoudige (DDR2) of achtvoudige (DDR3) snelheid naar buiten gebacht worden.

Table 2.1. DDR3 snelheden (bron: wikipedia)

Type geheugen	Alternatieve naam	kloksnelheid	Theoretische bandbreedte
PC3-10600 DDR3 SDRAM	DDR3-1333	167 MHz	10.667 GB/s
PC3-11000 DDR3 SDRAM	DDR3-1375	172 MHz	11 GB/s
PC3-12800 DDR3 SDRAM	DDR3-1600	200 MHz	12.8 GB/s
PC3-13000 DDR3 SDRAM	DDR3-1625	203 MHz	13 GB/s
PC3-14400 DDR3 SDRAM	DDR3-1800	225 MHz	14.4 GB/s
PC3-14900 DDR3 SDRAM	DDR3-1866	233 MHz	14.933 GB/s

Type geheugen	Alternatieve naam	kloksnelheid	Theoretische bandbreedte
PC3-15000 DDR3 SDRAM	DDR3-1866	233 MHz	14.933 GB/s
PC3-16000 DDR3 SDRAM	DDR3-2000	250 MHz	16 GB/s
PC3-17000 DDR3 SDRAM	DDR3-2133	266 MHz	17.066 GB/s
PC3-17600 DDR3 SDRAM	DDR3-2200	275 MHz	17.6 GB/s
PC3-19200 DDR3 SDRAM	DDR3-2400	300 MHz	19.2 GB/s
PC3-21300 DDR3 SDRAM	DDR3-2666	333 MHz	21.3 GB/s
PC3-24000 DDR3 SDRAM	DDR3-3000	375 MHz	24 GB/s

Een belangrijke opmerking, die reeds gedeeltelijk aangehaald werd, is dat DDR, DDR2 en DDR3 gebruik maken van een andere werkspanning. Deze wordt alsmaar lager om het gedissipeerd vermogen en de bijhorende warmteontwikkeling te verkleinen, wat nodig is om hogere kloksnelheden toe te laten. Bovendien hebben de modules ook een verschillend aantal aansluitpinnen, waardoor het duidelijk zal zijn dat ze niet compatibel zijn.

Om ongelukken te vermijden wordt daarom een andere behuizing gebruikt (inkeping in de module zit op een andere plaats).

2.9. Optimalisatietechnieken

De evolutie in DRAM technologie is er steeds op gericht om de maximale bandbreedte te verbeteren, terwijl bijzonder weinig aan de latentietijd werd gedaan. Deze verbeterde hooguit in absolute waarde, doordat de kloksnelheid verhoogde. Relatief gezien (dus uitgedrukt in klokcycli) is de latentietijd eerder gestegen. Uit het voorgaande zou al gebleken moeten zijn dat na elke rijtoegang een hersteltijd nodig is om de bufferrij(en) vrij te maken.

Figure 2.11. normale geheugentoegang

2.9.1. Interleaving

Een techniek die gebruikt kan worden om een deel van de dode tijd te vermijden, is interleaving. Meer bepaald gaat het dan om het vermijden van de tijd tussen twee rijen. Deze hersteltijd kan vermeden worden indien de volgende operatie doorgaat op een andere geheugenmodule. Om dit te bereiken worden bij interleaving naburige geheugenblokken verdeeld over verschillende geheugenbanken, die onafhankelijk van elkaar (en dus zonder hersteltijd) aangesproken kunnen worden. Belangrijke opmerking hierbij is dat er enkel snelheidswinst kan zijn als er gewisseld kan worden tussen banken. Het is dan ook belangrijk dat de memory controller weet of er al dan niet gewisseld kan worden tussen banken, zodat hij al dan niet rekening kan houden met de hersteltijd.

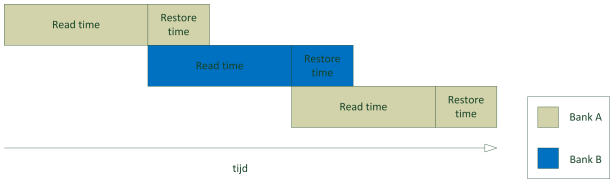

Figure 2.12. Memory interleaving

2.9.2. Dual channel

Een andere heel eenvoudige techniek om de snelheid te verhogen is het vergroten van de databus. Dit brengt wel een paar problemen met zich mee. Eerst en vooral moeten er extra aansluitingen voorzien worden op zowel de geheugenmodule als het moederbord en bovendien moeten de signaallijnen ook voorzien worden op het moederbord. Daarnaast zal er, zeker bij steeds toenemende kloksnelheden, een probleem ontstaan van tijdverschillen tussen de verschillende datalijnen.

Dual channel is een techniek die probeert de datasnelheid te verhogen door de databus naar de memory controller te verdubbelen, zonder de databus van de geheugenmodules te vergroten. Dit gebeurt weer door gebruik te maken van verschillende geheugenbanken. Elke geheugenbank heeft een databus van 64 bit, maar aangezien deze niet samenvallen is er een 128 bit databus naar de memory controller. De DIMM sockets op een moederbord zijn dus fysiek verbonden met één van de twee 64 bit kanalen. Uiteraard kan je enkel voordeel halen als geheugenmodules aangesloten zijn op de twee kanalen.

Figure 2.13. geheugentoegang met Dual Channel

Behalve het aanschakelen van meerdere modules is het dan ook belangrijk om ze in de juiste sockets te steken, zodat je beide kanalen gebruikt (en niet twee modules op hetzelfde kanaal). Moederborden met meerdere sockets op de kanalen, hebben overeenkomstige plaatsen op die kanalen die eenzelfde kleur hebben. Op deze manier zijn er dus matched sockets. Op deze matched sockets moet dus geheugen geïnstalleerd worden.

Figure 2.14. dual channel sockets

Dit geheugen moet identiek zijn in capaciteit, anders zou een deel van het geheugen niet bereikbaar zijn in dual channel mode. Verder moeten de modules in principe niet gelijk zijn. Zelfs modules met verschillende snelheden zijn mogelijk, al zal dan tegen de traagste snelheid gewerkt worden. In principe betekent echter dat er in de praktijk wel problemen kunnen ontstaan, zodat moederbord fabrikanten het gebruik van identieke modules aanraden. Deze worden door verschillende geheugenfabrikanten ook aangeboden.

Hoewel dual channel een veelbelovende techniek is, blijkt uit benchmarks dat de winst (ondertussen) toch marginaal is. Een deel van de verklaring hiervoor zou liggen in het cache geheugen, dat steeds groter en efficiënter wordt. Later zullen we zien dat de cache geheugens het snelheidsverschil tussen geheugen en processor moeten opvangen. Je kan trouwens zelf aan de berekeningen bij DDR al zien dat dual channel weinig invloed heeft, tenzij echt grote opeenvolgende stukken gelezen worden. Vergelijk de tijd die nodig is voor een burst van 4 transfers met die voor een burst van 8 transfers. Op een dual channel systeem zullen de bursts immers maar half zo groot zijn als bij single channel.

2.9.3. Buffered/registered RAM en foutdetectie

Een term die regelmatig terug te vinden is bij RAM geheugen is (un)buffered of registered. De termen buffered en registered hebben dezelfde betekenis. Op basis van de eerdere uitleg zou je kunnen vermoeden dat elk DRAM geheugen gebufferd is met de statische bufferrij. De term buffered slaat in dit geval niet op die statische rij, maar wel op de aanwezigheid van een extra bufferchip. Deze bufferchip doet dienst als elektrisch buffer/versterker tussen de geheugenIC's en

de rest van het systeem. Op die manier kan het bijvoorbeeld problemen met onvoldoende stroom helpen oplossen. De bufferchip is dikwijls te herkennen aan zijn dwarse plaatsing op de module.

Twee andere termen die je kan terugvinden zijn (non)ECC en parity. Beiden houden verband met het vermogen van het geheugen om fouten te laten detecteren. In het geval van pariteit wordt per byte een pariteitsbit berekend. Dit bit wordt mee verzonden. De ontvanger herberekent de pariteitsbit en vergelijkt het resultaat van zijn berekening met het ontvangen pariteitsbit. Indien ze verschillen is er een fout geweest en moet de transfer opnieuw gebeuren.

ECC werkt op gelijkaardige manier, maar maakt gebruik van een hash functie. Voordeel van deze manier van werken is dat meervoudige bitfouten gedetecteerd kunnen worden of dat enkelvoudige bitfouten gecorrigeerd kunnen worden. Pariteit kan enkel enkelvoudige bitfouten detecteren. Uiteraard kost deze foutcontrole ook rekenkracht en dus tijd. Door de betrouwbaarheid van moderne geheugens hebben deze technieken enkel zin in kritische toepassingen (bijvoorbeeld servers, procescontrole, ...)

2.10. Cache geheugen

Zelfs met alle optimalisatietechnieken blijft er een snelheidsprobleem. Instructies die de processor moet uitvoeren, zitten in het geheugen en als de snelheid waarmee het geheugen instructie kan leveren, vergeleken wordt met de snelheid waarmee de processor ze kan uitvoeren, blijkt die laatste een stuk sneller. Een belangrijk verschil, want een processor kan nu eenmaal niet sneller instructies afwerken dan dat ze door het geheugen aangeboden kunnen worden. Sneller geheugen maken, ligt voor de hand. Zoals eerder al aangehaald is statisch geheugen sneller dan dynamisch geheugen.

Belangrijk nadeel van statisch geheugen is dat het per byte veel duurder is dan dynamisch geheugen. De hoeveelheid statisch geheugen in een computersysteem zal dus eerder klein zijn en het komt erop aan deze kleine hoeveelheid zo efficiënt mogelijk te gebruiken. Een tweede probleem is dat niet enkel de snelheid van het geheugen problemen geeft, maar ook de snelheid van de interface naar het geheugen. Om dit probleem aan te pakken was er ooit een snelle back-side bus die de verbinding met het cache geheugen verzorgde. Ondertussen zijn er al verschillende niveaus van cachegeheugens in de processor geïntegreerd, zodat de verbinding met dit geheugen ook in de processor zelf zit en daardoor veel sneller kan zijn.

2.10.1. Werking

Figure 2.15. cache als buffer tussen CPU en geheugen

Het cache geheugen vormt een buffer dat het snelheidsverschil tussen processor en CPU moet opvangen. Aangezien het cache geheugen een stuk kleiner zal zijn dan het dynamisch geheugen, komt het erop aan om de nodige gegevens klaar te hebben zitten in het cache geheugen.

Om dit te realiseren wordt weer uitgegaan van het lokaliteitsprincipe. Zowel het cache geheugen als het hoofdgeheugen worden onderverdeeld in blokken van dezelfde grootte. De blokken in het cache geheugen (cache lines) kunnen een kopie bevatten van een blok uit het hoofdgeheugen.

Een leescyclus verloopt als volgt: * de processor vraagt een adres * Indien dat adres in een blok ligt dat in de cache te vinden is, wordt er gesproken van een cache-hit en heeft er een snelle leescyclus plaats vanuit de cache. * In het andere geval (een cache-miss) wordt via een trage leescyclus het gevraagde woord opgehaald uit het centrale geheugen terwijl ook onmiddellijk het blok waarin dit woord zich bevindt naar de cache gekopieerd wordt.

Aangezien opeenvolgende geheugentoegangen meestal op naburige adressen doorgaan, is de kans groot dat een volgende toegang een cache-hit geeft en dus snel verwerkt kan worden. Een belangrijke parameter voor de snelheid van het cache geheugen is dus de hitrate. Dit is het percentage geheugentoegangen dat rechtstreeks via het snelle cache geheugen kan verlopen. De hitrate ligt typisch tussen 80 en 99%.

Een schrijfcyclus kan gelijkaardig verlopen, maar er stelt zich wel een nieuw probleem. In het geval van een schrijfcyclus worden er immers gegevens aangepast. Indien er een cache-hit is, lijkt het logisch om de inhoud van het cache geheugen aan te passen en pas later (bij het verwijderen van de cacheline) wordt de inhoud van het hoofdgeheugen aangepast. Deze manier van werken heet write-back cache. Belangrijk nadeel hiervan is dat op een bepaald ogenblik de inhoud van het hoofdgeheugen niet consistent is met het cache geheugen. Dit kan bijvoorbeeld bij DMA-toegangen problemen opleveren. Tweede nadeel is dat het wegschrijven naar het geheugen gebeurt op het ogenblik dat de pagina uit het cache geheugen verwijderd wordt. Dit is het ogenblik waarop in het cache geheugen plaats gemaakt wordt voor een nieuwe pagina. Dit vertraagt dus het inladen van de nieuwe pagina.

Een alternatief is gebruik maken van write through cache. In dit geval worden steeds zowel het hoofdgeheugen als het cache geheugen aangepast. Nadeel is duidelijk dat steeds gebruik gemaakt wordt van het tragere hoofdgeheugen.

Dit kan gedeeltelijk opgevangen worden doordat de processor niet moet wachten tot de volledige cyclus is afgewerkt, maar bij opeenvolgende schrijfopdrachten zal het toch leiden tot vertragingen. Bij schrijfcycli zijn er nog verschillen mogelijk: write allocate en write no-allocate. Bij write allocate zal bij een cache miss het geheugenblok in het cache geheugen geladen worden, bij write no-allocate niet. Het voordeel is dat een schrijfoperatie naar het geheugen typisch sneller is dan het ophalen van een volledig blok. Bijvoorbeeld bij write-through geheugen is makkelijk in te zien dat het interessant kan zijn om het blok niet volledig in te laden.

2.10.2. Soorten caches

Er zijn een aantal onderscheiden te maken tussen cache geheugens. Een eerste belangrijk verschil is dat tussen level 1 en level 2 caches. In principe zijn zelfs nog meer niveau's van cache geheugens mogelijk. L1 cache is het cache geheugen dat het dichtst bij de processor staat. Het moet dan ook het snelste geheugen zijn, zodat het de processor kan volgen. Naarmate de snelheid van de processor toeneemt, werd het verschil in snelheid zo groot dat het interessant werd om een tweede niveau buffer in te zetten. Dit geheugen is minder kritisch op het vlak van snelheid (het moet de processor niet rechtstreeks voorzien van gegevens) en kan dus op andere vlakken geoptimaliseerd worden. Zo zal L2 cache typisch minder snel, maar wel een stuk groter zijn.

Een ander verschil tussen cache geheugens is dat L1 cache dikwijls opgedeeld worden in data cache en instructie cache. Hiervoor zijn verschillende redenen te bedenken. Een belangrijke reden is dat met pipelining, een deel van de processor (instruction fetch unit) instructies ophaalt en tegelijkertijd een ander deel (operand fetch) gegevens ophaalt om de bewerkingen uit te voeren.

Als beide geheugens gescheiden zijn, kunnen de twee units onafhankelijk van elkaar hun operaties uitvoeren. Anderzijds kunnen de cache geheugens ook geoptimaliseerd worden. Zo zal een processor nooit wijzigingen aanbrengen in de instructies die hij uitvoert. Voor de instructiecache moeten geen voorzieningen getroffen worden voor schrijfoperaties.

2.10.3. Overschrijfstrategieën

Bij een cache miss zal in de meeste gevallen een volledig geheugenblok ingeladen worden. Dit betekent dat in het cache geheugen plaats zal moeten gemaakt worden. Er zal met andere woorden een lijn geselecteerd moet worden, die uit het cache geheugen verwijderd zal worden. Het selecteren van die lijn moet uiteraard doordacht gebeuren om de hit rate zo hoog mogelijk te houden. In principe is het best om de cacheline te verwijderen die het langst niet zal gebruikt worden. Het probleem met deze keuze is dat er kennis over de toekomst voor nodig is. In plaats daarvan zijn er een aantal strategieën die op een andere manier proberen de meest geschikte lijn te selecteren:

First In First Out (FIFO)
: selecteert de lijn die al het langst in het cachegeheugen zit. Het is een erg eenvoudig te implementeren techniek, aangezien de lijnen eigenlijk gewoon cyclisch overschreven worden. Deze techniek is daarentegen weinig efficiënt op het vlak van het optimaliseren van de hitrate. Een lang geleden, maar veel gebruikte lijn zal bijvoorbeeld eerder verwijderd worden dan een lijn, waar maar een keer data uit gelezen wordt, maar die wel later is ingeladen.

Least Recently Used (LRU)
: selecteert de lijn die al het langst niet meer gebruikt wordt. De kans dat deze lijn dan plots weer gebruikt zal worden, is een stuk kleiner dan bij FIFO. Hierdoor zal deze techiek leiden tot een hogere hitrate. Nadeel is dan weer dat er een pak meer bij komt kijken om bij te houden welke lijn geselecteerd wordt. Deze berekening kost zowel geld (om te implementeren) als tijd (om de lijn te selecteren). In het bijzonder als uit een groot aantal lijnen gekozen moet worden, is het gebruik van deze techniek niet aangewezen.

Least Frequently Used (LFU)
: deze techniek zal de lijn selecteren die het minst frequent gebruikt wordt. De techniek haalt gelijkaardige resultaten als LRU, maar heeft ook dezelfde nadelen.

Adaptive Replacement Cache (ARC)
: combineert zowel LRU als LFU. Hierdoor worden nog betere resultaten gehaald op het vlak van hitrate, maar de bewerkingen en de implementatie ervan worden anderzijds ook complexer.

Random
: selecteert willekeurig een lijn. Dit is een eenvoudige te implementeren techniek, die toch aanvaarbare resultaten haalt, in het bijzonder als er een

groot aantal lijnen zijn waaruit geselecteerd moet worden. Deze techniek wordt soms gecombineerd met LRU, door een bit te koppelen aan een lijn. Dit bit geeft aan of de lijn al gebruikt is. Als alle lijnen gebruikt zijn, worden alle bits weer gereset. Als dan een lijn gekozen moet worden, zal random geselecteerd worden uit alle lijnen die gemarkeerd zijn als “niet gebruikt”.

2.11. Associativiteit

De associativiteit van het cache geheugen bepaalt op welke cachelines een welbepaald geheugenblok terecht kan komen. De associativiteit bepaalt dus ook het aantal lijnen waaruit geselecteerd kan worden en bepaald dus zowel rechtstreeks (beperking van lijnen) als onrechtstreeks (welke overschrijfstrategie) mee de hitrate.

2.11.1. Fully Associative cache

Figure 2.16. principe fully associative cache

Bij fully associative cache kan een blok uit het geheugen terecht komen in gelijk welke cacheline. Om te kunnen bepalen welk geheugenblok in een bepaalde cacheline zit, wordt een elke cacheline een tag gekoppeld. Deze is nodig om te kunnen bepalen of het blok aanwezig is in het cache en eventueel de juiste cacheline te kunnen selecteren.

Figure 2.17. voorbeeld fully associative geheugen

Een voorbeeld zie je in bovenstaande afbeelding. De processor beschikt over een 32-bit adresbus en een cache geheugen van 16kB met cachelines die 64-byte breed zijn. Om binnen elke cacheline het juiste byte te selecteren zijn er 6 bits nodig. Hiervoor worden uiteraard de minst significante gebruikt. De overige 26 bits worden gebruikt om de inhoud van de cacheline te identificeren. Je zou dit kunnen zien als het nummer van het geheugenblok. Merk immers op dat voor elk byte van een geheugenblok, deze 26 bits van het adres steeds overeenkomen. Het zijn dan ook deze bits die opgeslagen worden in de tag. Naast de cachelines en de tags die eraan gekoppeld zijn, zijn er een aantal comperatoren voorzien. Bij het begin van een cyclus worden deze comperatoren gebruikt om de bovenste 26 bits van het adres te vergelijken met de inhoud van de tags. Indien de uitkomst

van een van de comperatoren een gelijkheid aangeeft, is er een cache-hit en is meteen ook de juiste cacheline geselecteerd.

In het voorbeeld zijn er 256 tags van 26 bits elk (6656 bits) die vereist zijn naast de data-cache. Dit komt neer op bijna 5% van de cache-capaciteit. Vermits elke pagina uit het geheugen in om het even welke cache-line geplaatst kan worden, kan de cache optimaal benut worden. Bovendien kan gebruikt gemaakt worden van de optimale overschrijfstrategie. Het nadeel van associative cache is dat het een vrij dure implementatie is. Er is immers behoefte aan veel supersnel geheugen om de tags te implementeren. Daarnaast moeten ook de snelle comparatoren gerealiseerd worden. Binnen de tijd van een cyclus moet immers geweten zijn of er een cache hit is.

Fully associative cache geeft goede resultaten. Indien deze techniek dan ook nog gecombineerd wordt met de meest complexe overschrijfstrategieën, wordt het geheel extreem complex en duur. Zoals al aangehaald bij de overschrijfstrategieën is hier het aantal lijnen meestal te groot om gebruik te maken van LRU, LFU of ARC, zonder daarvoor een andere prijs te betalen.

2.11.2. Direct mapped cache

Figure 2.18. principe direct mapped cache

Bij direct mapped cache kan elk geheugenblok slechts in één cacheline terecht. Dit maakt dat overschrijfstrategieën overbodig zijn, er is namelijk maar een plaats waar het geheugenblok terecht kan en de inhoud van die cacheline zal verwijderd worden. Evident nadeel is dat als er geen keuze mogelijk is, de optimale keuze niet gemaakt kan worden. Deze manier van werken heeft dus een negatief effect op de hitrate. Om dit te staven met een extreem voorbeeld: bij direct mapped cache is het mogelijk dat een cacheline plaats moet maken voor een andere, terwijl een deel van het cache nog niet in gebruik is. Doordat een geheugenblok maar op een lijn terecht kan, worden de tags kleiner en is er ook slechts een comparator nodig.

Figure 2.19. voorbeeld direct mapped

Een voorbeeld zal dit verduidelijken... De processor beschikt overeen 32-bit adres bus en een cache geheugen van 16kB met cachelines die 64-byte breed zijn. Naast de zes bits die nodig zijn om een byte binnen de cacheline te selecteren, zijn

er nu ook acht bits nodig om een van de 256 (=16kB/64B) lijnen te selecteren. Op die manier blijven er maar 18 bits over voor de tag. Dit soort cache-organisatie is dus veel goedkoper en eenvoudiger dan de vorige: er is slechts 1 comparator (van 18 bits) nodig, en het tag-gedeelte is ook kleiner: 256 x 18 bits (4608 bits, 3,4%). Dit soort cache wordt toegepast op plaatsen waar de snelheid minder kritisch is, met andere woorden in de caches die verder van de processor staan.

2.11.3. Set-associative cache

Figure 2.20. principe set-associative cache

Set-associative cache is de tussenvorm. Hierbij worden de cachelines gegroepeerd in sets. Elk geheugenblok kan terecht in slechts een set, maar binnen die set kan het in elke cacheline terecht.

Figure 2.21. voorbeeld set associative

Een voorbeeld: een processor met een 32-bit brede adresbus en 16kB 4-way set associative cache geheugen met cachelines van 64 byte. 4-way betekent dat een set bestaat uit vier cachelines. De 256 lijnen worden nu verdeeld in 256/4=64 sets.

Zes bits zijn nog steeds nodig om een byte in de cache line te selecteren. Daarnaast zijn er nu een aantal bits nodig om een set te kiezen. Aangezien er 64 sets zijn, zijn er hiervoor zes bits nodig. De resterende bits (20) worden gebruikt voor de tag. Als nu een adres aangeboden wordt, wordt een set van vier lijnen geselecteerd en worden de vier tags vergeleken met de meest signicante bits van het adres. Als één van de tags gelijk is aan deze bits is er een cache hit.

Rekenvoorbeeld (meer voorbeelden vind je in [4]) Opgave: Een 4-way set associative cache heeft een grootte van 64KB. De CPU werkt met 32-bit addressering, elk geheugenwoord bevat 4 bytes. Elke cache-line bevat 16 bytes. De cache gebruikt een write-through policy. Bereken de totale benodigde hoeveelheid geheugen (geheugen+tags) die nodig is om dit te implementeren... Antwoord:

het aantal bits is een adres is 32 (a) Een cacheline omvat 16 bytes, wat betekent dat de vier (b) LSB's niet moeten geadresseerd worden. Het aantal sets is dan 64KB/(4*16) = 1024 (c) Het aantal bits dat nodig is om deze te kunnen adresseren: log2 1024= 10 bits (d) Het aantal tag bits wordt dan: 32 – 4 (b) – 10 (d) = 18 bits

Om de totale grootte dan te berekenen kan je redeneren: - elke cachelijn heeft 16 bits data, en een tag van 18 bits. Dit komt op een totaal van 146 bits. - dit moet je met vier vermenigvuldigen (‘4’-way) en met het aantal sets (d), wat resulteert in 598016 bits. Het percentage overhead is dus (totaal benodigde bits)/(*nuttige*^ bits) of 14%

REMARK: bereken aan de hand van vorig voorbeeld hoe de overhead zich gedraagt bij 8-way en 16-way cache geheugen.

De eigenschappen van set associative liggen dan ook tussen de twee vormen in. Het laat minder mogelijkheden voor het kiezen waar een cacheline terecht kan dan fully associative, maar meer dan direct mapped. Anderzijds heeft het minder bits nodig voor de tags en ook minder comparatoren dan bij fully associative cache. Vergeleken met direct mapped is het dan weer complexer. Meer algemeen geldt dat naarmate de associativiteit toeneemt er meer keuze is in de cache lijnen, maar dat de kost die hieraan verbonden is ook hoger wordt. Omwille van de beperking van het aantal cachelines in een set is set-associative cache gemakkelijker te combineren met betere overschrijfstrategieën. Mede daardoor is fully associative cache een eerder zeldzame vorm van cache.

Opmerking: in literatuur worden de termen ‘cache line’, ‘cache block’, ‘cache entry’ en ‘cache set’ vaak door elkaar gebruikt. [4]

2.11.4. Snelheid van de cache

In het voorgaande hebben we reeds een aantal eigenschappen aangehaald die mee de efficiëntie van het cache geheugen bepalen. Dit is uiteraard een heel belangrijke eigenschap, aangezien bij cache hits, de processor aan zijn volle snelheid kan werken.

cache misses in functie van grootte en associativiteit (bron: wikipedia)In bovenstaande afbeelding toont een grafiek het aandeel missers in functie van de grootte van het cache geheugen en dit voor verschillende associativiteit. Een deel van de missers zijn onvermijdelijk. Het zijn de zogenaamde coldstart misses, die afkomstig zijn van de eerste keer dat toegang tot een blok gezocht wordt. Een tweede deel hangt samen met de capaciteit. Dit neemt duidelijk af naarmate het cache geheugen groter wordt. Een derde deel hangt samen met de associativiteit. In deze grafiek komt duidelijk naar voor dat met toenemende associativiteit het aantal missers afneemt. Naast de hitrate zijn er nog een aantal andere eigenschappen, die samen de snelheid van het volledige geheugen bepalen.

Deze eigenschappen zijn: Hit time:: de tijd nodig om bij een hit de gegevens op te halen. Miss time:: de tijd nodig om bij een miss het nieuwe geheugenblok te laden en de gevraagde gegevens beschikbaar te maken Miss penalty:: de extra tijd die nodig is bij een cache miss (eigenlijk misstime-hit time) Hit rate:: percentage toegangen die rechtstreeks langs de cache gaan

De formule voor de totale toegangstijd van de gecombineerde caches is dan:

"t"_"totaal"="HR"_"L1" xx "t"_"L1" + (1 - "HR"_"L1") xx "HR"_"L2" xx t_"L2" + (1 - "HR"_"L1") xx (1 - "HR"_"L2") x "t"_"RAM"

Voorbeeld: Een computer heeft een L1-cache, toegangstijd 1 ns en een hitrate van 96%. De L2 cache heeft een hitrate van 88 % bij een toegangstijd van 2 ns. Het externe geheugen heeft een toegangstijd van 8 ns. De gemiddelde toegangstijd wordt dan:

0.96 xx 1 "ns" + 0.04 xx 0.88 xx 2 "ns" + 0.04 xx 0.12 * 8 "ns" = 1.0688 "ns"

Probeer zelf eens uit wat de invloed is van de toegangstijd van het RAM-geheugen in dit systeem. Probeer eens wat er gebeurt zonder L2 cache. De verschillende factoren worden niet op dezelfde manier beïnvloed door de eigenschappen van het cache. Bijvoorbeeld de asscociativiteit van de cache biedt meer mogelijkheden op het vlak van cachelines selecteren, waardoor de hitrate hoger kan zijn. Daar staat tegenover dat een complexere berekening nodig is om een lijn te selecteren, waardoor bij een cache miss er extra tijd verloren gaat (hogere miss time).

Een ander voorbeeld zijn de grootte van de cachelines. Grotere cachelines geven in eerste instantie een hogere hit rate, omdat meer naburige gegevens gekopieerd worden. Anderzijds zijn grotere cachelines nefast voor de miss time. Bij heel grote cachelines zal bovendien zelfs de hitrate dalen, omdat lijnen te snel uit het cachegeheugen verdwijnen (omdat bij dezelfde grootte er nu eenmaal minder lijnen zijn).

Een groter cachegeheugen geeft dan weer meer cachelines. Mits een goede overschrijfstrategie toegepast wordt en de associativiteit hoog genoeg is, kan de hit rate toenemen. Nadeel is dat de selectie van een lijn dan weer complexer wordt waardoor de miss time weer toeneemt.

2.12. Virtueel geheugen

Virtueel geheugen is een oplossing om het tekort aan fysiek geheugen op te lossen. Het tekort aan geheugen komt van steeds groter wordende toepassingen en bestanden die bewerkt worden. Bovendien worden ook verschillende programma's naast elkaar uitgevoerd. Toch zijn er vandaag ook een aantal situaties waarin een computer ruim voldoende heeft met het fysieke geheugen. In dat geval biedt virtueel geheugen weinig meerwaarde.

2.12.1. Werking

Virtueel geheugen zal gebruik maken van de beschikbare ruimte op een of ander medium voor massaopslag om het geheugen groter te laten lijken.

Meestal zal het gebruikte medium een harde schijf zijn. We zullen hier in het volgende van uitgaan. Op de harde schijf zal ruimte voorzien worden die gebruikt wordt als swapruimte. Het kan een swap-file zijn of een swap-partitie. Indien de processor toegang wil tot een bepaald adres, zal aan de hand van het adres nagegaan worden of de gevraagde gegevens aanwezig zijn in het fysieke geheugen. Indien de gegevens in het fysieke geheugen zitten, worden ze opgehaald en wordt de instructie gewoon verder afgewerkt. Als de gegevens niet in het fysieke geheugen zitten, treedt er een page fault op. Dit is een speciale exceptie. De processor wordt met andere woorden onderbroken en zal de exceptieroutine uitvoeren. In dit geval gaat het om een roll-back exception. De processor zal eerst terugkeren naar de toestand vlak voor de uitvoering van de instructie, die het probleem veroorzaakte en vervolgens de exception handler uitvoeren. De exception handler zal indien nodig plaats maken en vervolgens de nodige gegevens verplaatsen naar het fysieke geheugen. Uiteraard zal het hier niet gaan over een byte, maar wel over een groter stuk geheugen. We zullen later terugkomen op de grootte van dit geheugenblok.

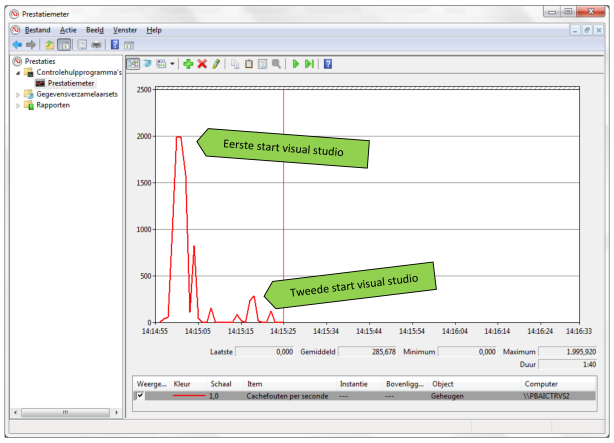

Figure 2.22. Page-faults bij opstarten software

Na de uitvoering van de handler keert de processor terug naar de toestand bij de oproep van de handler. Dit is uiteraard de instructie die in eerste instantie het probleem veroorzaakte. Deze keer zal bij de uitvoering van die instructie blijken dat de gegevens beschikbaar zijn in RAM.

Om virtueel geheugen te ondersteunen is, naast het opslagmedium, ook een processor nodig die de roll-back exception ondersteund. Daarnaast moet ook het besturingssysteem de nodige routines omvatten en tenslotte is er ook behoefte om bij te kunnen houden waar de gegevens zich bevinden (fysiek geheugen of swap). Belangrijk is ook dat bepaalde delen nooit mogen verdwijnen uit het fysieke geheugen. Een voorbeeld is de handler: als die op de swap staat, is er geen routine meer beschikbaar om hem naar het fysiek geheugen te verplaatsen.

2.12.2. Paging - segmenting

De meest gebruikte manier om met virtueel geheugen te werken, is zowel de swap als het fysieke geheugen te verdelen in stukken van dezelfde grootte. Zo'n blok wordt dan een pagina genoemd. Voor elke pagina zou dan in een tabel opgeslagen kunnen worden waar de pagina zich bevindt. In deze tabel zou dan een bit aangeven of het fysiek dan wel virtueel aanwezig is en de andere bits zouden de locatie kunnen aanduiden. Uiteraard moet deze tabel ook ten allen tijde in het fysiek geheugen aanwezig blijven. Soms is het noodzakelijk om de grootte van deze tabel te beperken.

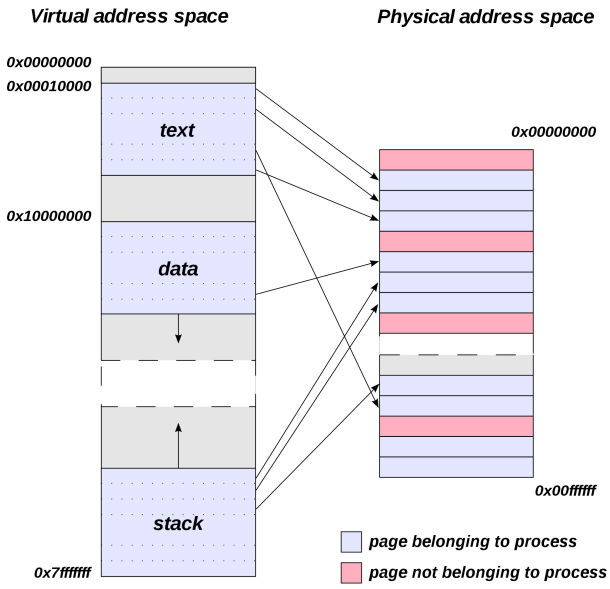

Figure 2.23. Paging table (Wikimedia Commons, BSD)

Een voorbeeld: op een 32-bit processor met pagina's van 4kB is een tabel nodig van 4MB (op voorwaarde dat de elementen van de tabel 32 bit groot zijn). Om de tabellen klein te houden kan gewerkt worden met meerdere niveaus van tabellen. In het reeds aangehaalde voorbeeld zouden er 1024 tabellen van 4kB nodig zijn. Deze tabellen bevinden zich op het tweede niveau. Op het eerste niveau is er een tabel van 4kB, die toelaat een van de 1024 secundaire tabellen te selecteren.

Het alternatief voor paging is werken met segmenten. In grote lijnen is het verhaal hetzelfde, er is bijvoorbeeld ook een tabel die bijhoudt waar de segmenten zich bevinden. Het grote verschil is dat de gegevens nu niet opgedeeld worden in pagina's van gelijke grootte. Het grootste gevolg hiervan situeert zich op het vlak van geheugenfragmentatie. Bij het vervangen van geheugenpagina's of segmenten is er nu extra tijd (traagheid van harde schijf laat complexe berekeningen toe). Er kan dus een optimale keuze gemaakt worden. Bij segmentering kan het gebeuren dat een groot segment vervangen wordt door een veel kleiner segment, waardoor een stuk van het geheugen niet in gebruik zou zijn. Dit soort fragmentatie heet externe fragmentatie.

Bij paging gebeurt dit uiteraard niet. Daar worden immers pagina's van gelijke grootte vervangen. Bij paging kan wel interne fragmentatie optreden. In feite worden de gegevens van een segment verdeeld in pagina's van 4kB. Voor een bestand van 13kB geeft dit drie volledig gevulde pagina's en een pagina met 3kB ongebruikte ruimte.

2.12.3. Snelheid en virtueel geheugen

Virtueel geheugen heeft de reputatie om de computer te vertragen. Hoewel dit strikt genomen wel waar kan zijn, klopt dit niet helemaal. De reputatie komt namelijk van het swappen van gegevens naar de harde schijf. Aangezien de harde schijf een stuk trager is dan het RAM, gaat dit uiteraard een stuk trager dan gewoon lezen van gegevens uit het RAM.

Dit is echter geen eerlijke vergelijking. Indien de swap ruimte niet gebruikt zou kunnen worden, zou er gewoon onvoldoende geheugen beschikbaar zijn om in deze situatie te komen. De gebruiker zou zelf bestanden of programma's moeten afsluiten om te kunnen doen wat het swappen veroorzaakte. Het gebruik van virtueel geheugen biedt de gebruiker dus in eerste instantie extra mogelijkheden. Als die extra mogelijkheden gebruikt worden, gaat dat inderdaad trager.

Los daarvan kan het gebruik van virtueel geheugen wel vertraging geven. Een eerste evidente reden is dat in plaats van een adres gewoon op te zoeken in het geheugen, nu het adres eerst opgezocht moet worden in twee tabellen, alvorens de eigenlijke operatie kan doorgaan. Aangezien de tabel in het geheugen zit, zijn er dus drie geheugentoegangen nodig om een gegeven te manipuleren. Dit probleem wordt grotendeels opgelost door een Translation Lookaside Buffer. Dit is een snel soort cachegeheugen in de processor waar de meest recent gebruikte fysieke adressen opgeslagen worden. De hitrate van dit buffer ligt weer bijzonder dicht bij 100%, zodat deze vertraging vrijwel geen probleem meer is.

Een tweede vertraging kan ontstaan als het besturingssysteem onnodig voorbereidingen treft om een toekomstige swap-operatie te versnellen. Zoals daarnet vermeld is swappen een trage operatie. Bovendien moet eerst een pagina geswapt worden naar de harde schijf, waarna de gevraagde pagina in het fysieke geheugen geladen kan worden. Dit betekent tweemaal 4kB verplaatsen naar en van de harde schijf. Om op het ogenblik dat er gegevens uit de swap-ruimte nodig zijn, de transactie te versnellen, kan het besturingssysteem proberen om op de achtergrondpagina's te zoeken die waarschijnlijk niet onmiddellijk gebruikt zullen worden en deze dan al naar de swap te verplaatsen.

Als dan gegevens uit de swap nodig zijn, kan dat met een 4kB verplaatsing van de harde schijf. Het kan natuurlijk gebeuren dat de verplaatste pagina's toch eerder nodig zijn dan dat er een swapping nodig is. In dat geval zullen de verplaatste pagina's, wanneer ze terug nodig zijn, uit de swap moeten gehaald worden.

Hierdoor reageert de computer trager dan in het geval er geen virtueel geheugen zou zijn.

2.13. Bronvermelding bij dit hoofdstuk

Chapter 3. Literatuurlijst

Onderstaande werken bevatten een stuk diepgaandere informatie over deze materie. Ze kunnen dan ook een ideaal vertrekpunt vormen voor een verdere studie van de computerarchitectuur.

[STALLINGS] Andy Hunt & Dave Thomas. *Computer Organization and architecture, Ninth edition*. Pearson education. 2012.

[RAMA] Umakishura Ramachandran. *Computer Systems: An Integrated Approach to Architecture and Operating Systems*. Manning Publications. 2010.