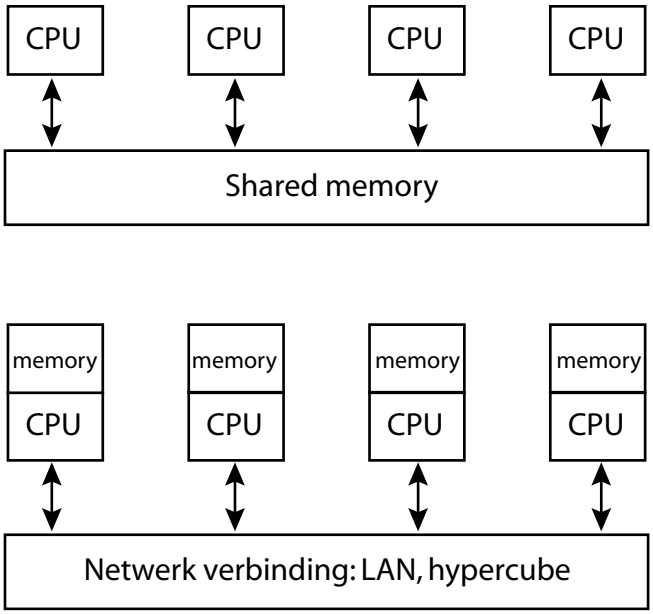
1 Aanvulling cosy deeltijd

1.1 Multiprocessor versus multicomputer

Het kenmerk van een multiprocessor is dat meer CPU hetzelfde geheugen delen. Voordeel van deze aanpak is het relatief eenvoudige programmeermodel. In principe kunnen alle CPU's overal bijkomen. Nadeel is de slechte schaalbaarheid. Willen we veel processorren toevoegen dan laat de bandbreedte van het geheugen een snelle toegang niet toe.



Figuur 1.1 Multiprocessor en multicomputer

Als de CPU's over cachegeheugen beschikken is het van belang de cache consistent (dat wil zeggen kloppend met het werkgeheugen) te houden. Een mogelijke oplossing is een snooping cache. Deze cache luistert het busverkeer af en kijkt over gecachte waarden overschreven worden. Bij het MESI cache coherentieprotocol (toegepast in Pentium) kent een cache-entry vier toestanden.

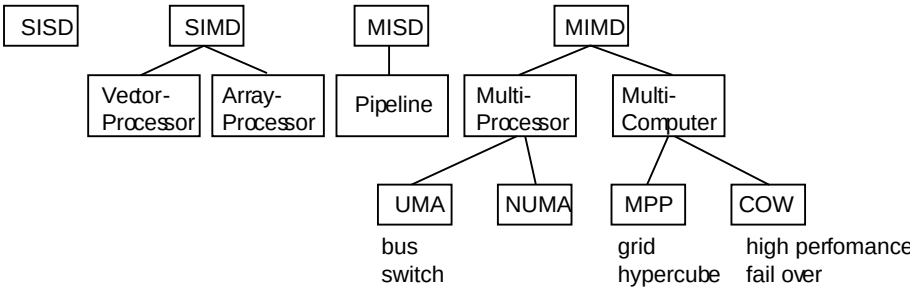
1. Modified (inhoud is veranderd, geheugen is nog niet aangepast)
2. Excusive (alleen deze cache heeft deze geheugenplek gecached en geheugen klopt met gecachte waarde)
3. Shared (gedeeld, meer caches bevatten de geheugeninhoud, deze klopt met de gecachte waarde)
4. Invalid (cache entry is niet geldig)

Tussen de vier toestanden zijn overgangen mogeijk die afhanegen van wat er precies gebeurd. Bij het ophalen van een waarde die gecached wordt is een entry exclusive.

Haalt een ander CPU dezelfde waarde op dan is het shared. Verandert een CPU een shared waarde dan wordt deze modified terwijl de de andere gecachte waarde invalid wordt. Wordt eeg waarde uit het geheugen opgevraagd die modified is dan wordt eerst de modified waarde naar het geheugen teruggeschreven voordat de opgevraagde waarde geleverd wordt.

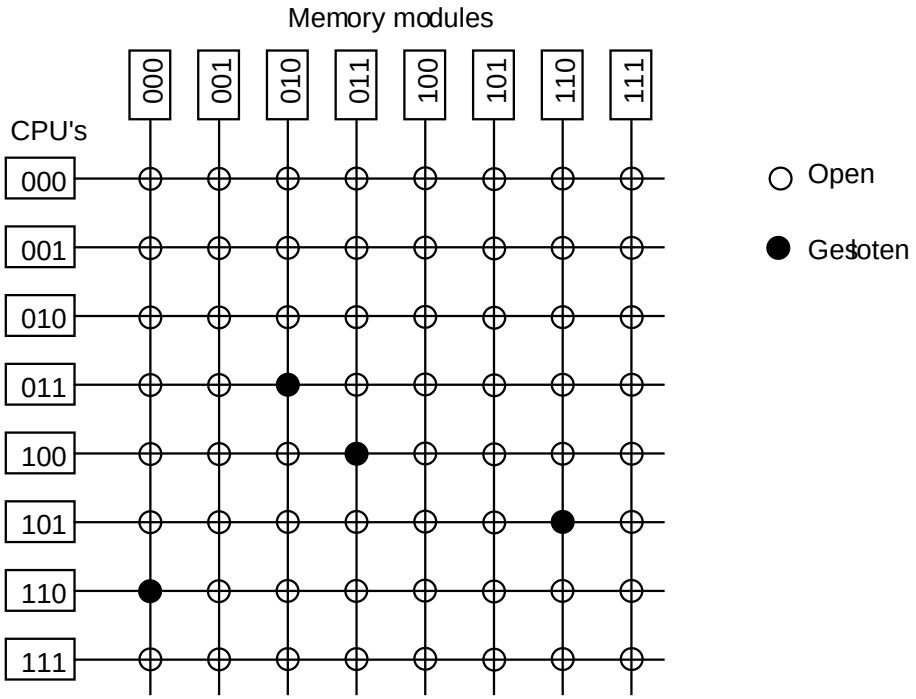
1.2 Multicomputers

Een multicomputer is een systeem dat uit meer computers bestaat. De computers zijn via een netwerk verbonden. Dit kan een LAN zijn, maar ook meer geavanceerde systemen zoals een grid en een hypercube zijn mogeijk. Klassificate van Flynn. De volgende figuur geeft de klassificatie van flynn met met klasse mogelijke uitwerkingen.



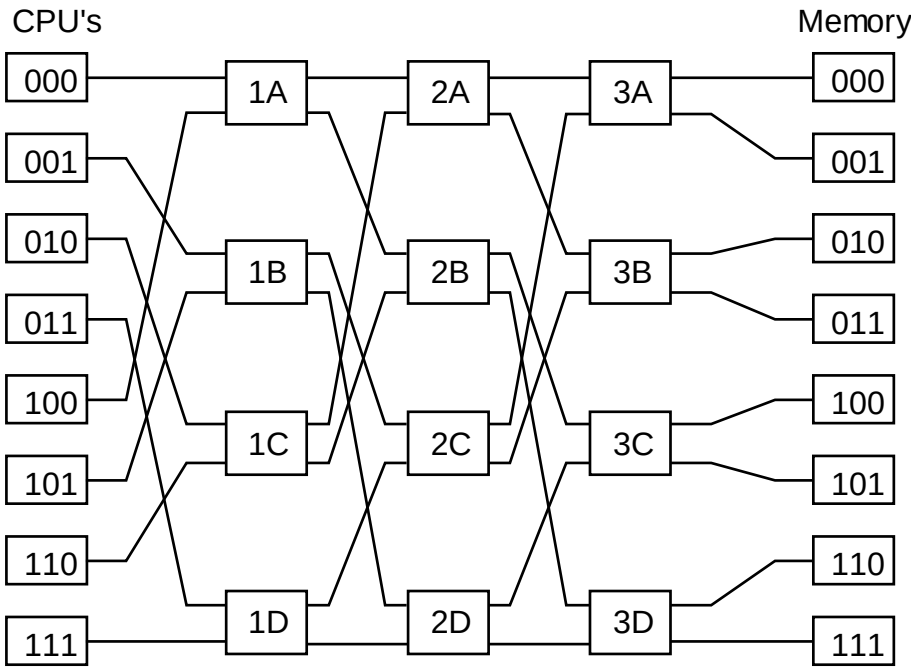
Figuur 1.2 Klassificatie van Flynn

Voor multiprocessoren kennen we de UMA en de NUMA implementatie. UMA staat voor Uniform memory access, dat wil zeggen dat elke processor op dezelfde wijze bij het shared memory kan komen. Passen we hiervoor een bus toe dat blijkt dit geen goede oplossing vanwege de slechte schaalbaarheid. Het geheuegen is eigenlijk al te langzaam voor een CPU, laat staan als het om meer CPU's gaat. Daarom worden hier speciale architecturen toegepast zoals een schakelnetwerk (kruisschakelaar) of een omegaswitch. Bij een schakelnetwerk zijn de CPU's en de memory modules via een kruisnetwerk verbonden. Elke CPU kan op zeker moment bij een memory-module komen. We noemen dit een non-blocking network. Let op: het is wel mogeijk dat een CPU niet bij het geheugen kan omdat een andere CPU al met dei module bezig is.



Figuur 1.3 Netwerk met kruisschakelaars

Bij een omega swicth gebruiken we schakelelementen met twee ingangen en twee uitgangen. Afhankelijk van de aansturing gaat de verbinding rechtdoor of is de verbinding gekruist. De aansturing wordt bepaald door de adresseringbits. Telkens zal een van de bits de schakelaar in de juiste stand zetten om bij de module te komen.

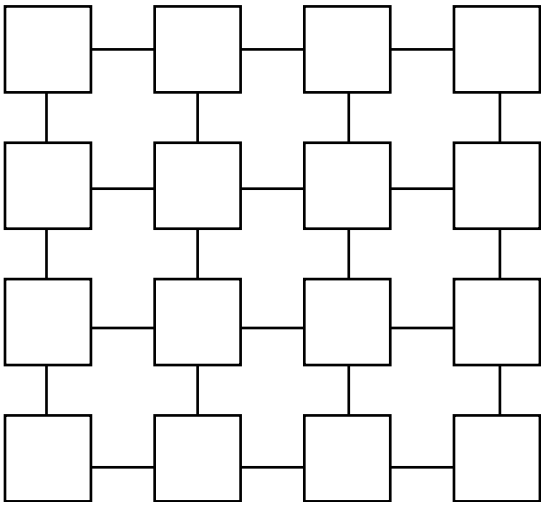


Figuur 1.4 Omega netwerk

Bij dit netwerk is het denkbaar dat twee CPU's een verschillende module willen bereiken maar waarbij een CPU de switches in zo'n stand heeft gezet dat de tweede zijn module niet kan bereiken. We spreken van een blocking network. Let erop dat we hier van elektronische switches spreken die dus zeer snel schakelen. Dergelijke switches zijn ook in ATM netwerken terug te vinden.

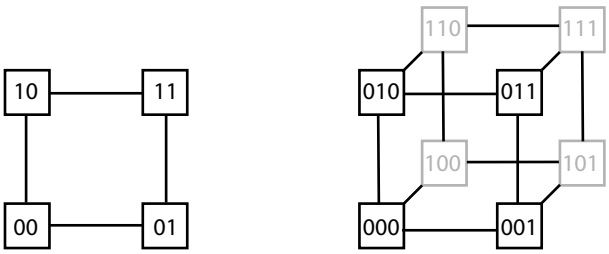
1.3 Multicomputers

Voor multicomputers onderscheiden we speciale architecturen zoals de grid en de hypercube. Daarnaast is een multicomputer ook op basis van standaard netwerktechnologie te bouwen. We krijgen dan clusters van werkstations (COW) of failover systemen als het om betrouwbaarheid gaat. Voor de speciale architecturen hebben we de grid waarbij de computers (behalve de systemen aan de rand en op de hoeken) allemaal vier netwerkverbindingen hebben.



Figuur 1.5 Grid verbindingsnetwerk

In het geval van een $N \times N$ grid is de maximaal af te leggen weg tussen twee nodes $2(N-1)$ hops. Ingavel van een 4×4 grid is dat 6 hops. Bij een hypercube is de topologie wat anders georganiseerd. We zien alke node als een hoekpunt van een vierkant, een kubus of een kubus mer meer dan drie dimensies. De eenvoudigste benadering van een dergelijk systeem is om elke dimensie een eigen bit te geven in een nodenummer.



Figuur 1.6 Hypercube verbindingsnetwerk

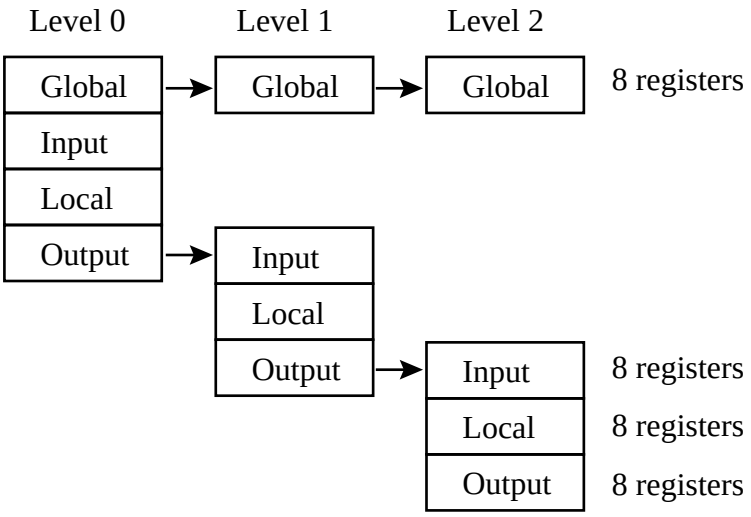
Zo kent een driedimensionale kubus drie bits (voor x, y en z-richting.) Op dezelfde wijze kunnen we ook een vierdimensionale kubus maken (met vier bits per node nummer). Een dergelijke kubus heeft dus 16 hoekpunten. De maximale af te leggen weg is in zo'n geval 4 hops (namelijk als alle bits in de nodenummers tussen bron en bestemming verschillen).

Bij NUMA hebben systemen lokaal geheugen, shared geheuegen en ook caches zijn

mogelijk. Het lokaal geheugen kan alleen voor een specifieke CPU toegankelijk zijn, maar hey kan ook onderdeel van het shared geheugen zijn. In dat laatste geval heeft de CPU voor wie dit lokaal geheugen is een snellere toegang tot dit geheugen. Het consistent houden van caches is een speciaal probleem

1.4 Aanvulling RISC

Gebruik van grote registerset in RISC architecturen. Twee technieken die men toepast bij CPU met zeer veel registers zullen we hier besreken. Bij de SPARC wordt een techniek toegepast die sliding register window heet. De achterliggende gedachte is dat we het geheugenverkeer willen minimaliseren. Kijken we naar een eenvoudig C-programma dan blijkt dat we hier te maken hebben met een samenstel van functies die worden aangeroepen. Bij een functieaanroep worden parameters meegegeven. Een functie kent zelf lokale variablen en kan zelf ook weer andere functies aanroepen. De Sparc werkt per functie dan met 32 registers die in vier groepen verdeeld zijn. Allereerst hebben we acht registers voor globale variabelen die voor alle functies toegankelijk zijn. Dan hebben we acht registers met parameters die aan de functie meegegeven worden. We noemen dit de input. Voor lokale variabelen zijn acht registers beschikbaar. Roepen we een andere functie aan, dan zetten we de parameters voor deze functie in de acht output-registers. Als we dan de functie echt aanroepen, worden deze output-registers de input voor de aangeroepen functie. De volgende acht registers uit de grote registerbank op de CPU zijn voor lokale variabelen enzovoorts. In de figuur is de situatie tot drie niveu's aangegeven.



Figuur 1.7 Register window van de SPARC

Een tweede techniek om goed gebruik te maken van registers voor opslag van variabelen is het gebruik van levende en dode variabelen. Als een variabele gedeclareerd wordt wordt hij vaak nog niet gebruikt. Bij het compileren is goed te volgen, wanneer een variabele gebruikt gaat worden (levend wordt) en wanneer hij niet meer gebruikt wordt (dood is). Voor elke variabele is op deze wijze een levenslijn uit te zetten. Variabelen met levenslijnen die niet overlappen kunnen hetzelfde register gebruiken. Op deze wijze hoeven we niet zo snel het externe geheugen aan te spreken en boeken we weer tijdswinst.