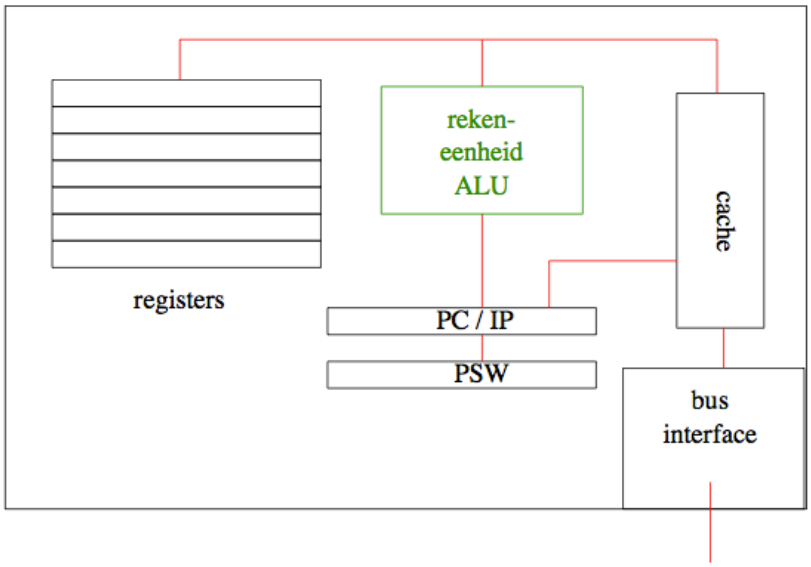
Computerarchitectuur en netwerken

Voorbeelduitwerking – Toets 1 – 4 oktober 2011
(Onder voorbehoud van typo's)

Opgave 1a



Opgave 1b

- ALU: dit is de Arithmetic Logic Unit, ofwel rekeneenheid. Dit is het onderdeel van de CPU waarmee de elementaire bewerkingen uitgevoerd worden, zoals optellen, aftrekken, vermenigvuldigingen en logische operaties.

- Registers: dit zijn opslagplaatsen voor gegevens in de chip zelf. De registers zijn veel sneller toegankelijk dan het geheugen. Ze worden gebruikt om de argumenten en resultaten van instructies tijdelijk in op te slaan.

- PC/IP: dit is de Program Counter of Instruction Pointer. Dit is een register dat aangeeft waar we in het programma zijn. De inhoud wijst naar een geheugenplaats waar de eerstvolgende instructie die uitgevoerd moet worden te vinden is.

- PSW: dit is de Program Status Word. Dit is een register met statusinformatie over het programma dat momenteel uitgevoerd wordt. In dit register is bijvoorbeeld te vinden of we momenteel in user mode of system mode draaien en wat de prioriteit is van het huidige interrupt (als er 1 actief is).

- Cache: dit is een relatief klein en snel 'tussengeheugen' dat zich bevindt tussen de registers/ALU en het interne geheugen. Dit geheugen heeft een bufferfunctie. Alle gegevens die recent naar het geheugen geschreven zijn of uit het geheugen opgehaald zijn, worden tijdelijk hier opgeslagen, zodat deze gegevens snel voorhanden zijn als ze opnieuw nodig zijn.

- Bus interface: dit is het onderdeel waarmee de CPU communiceert met de buitenwereld. De bus interface regelt de toegang tot de bus, ofwel het communicatiekanaal met de overige componenten van de computer, zoals het geheugen, de videokaart en de externe randapparatuur (disks, muis, keyboard).

Opgave 2a

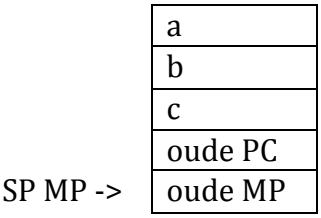
```
ldl -4
ldl -3
mul
ldl -4
ldl -3
mul
mul
stl -2    (immers: het resultaat moet op de plek van c komen, volgens de
           opgave)
```

Alternatief waar een slimme compiler mee zou kunnen komen:

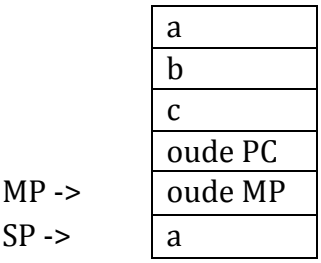
```
ldl -4
ldl -3
mul
lds 0     (of: ldl 1)
mul
stl -2
```

Opgave 2b (uitgaande van de eerste oplossing)

Uitgangsituatie:



Na ldl -4:



Na ldl -3:

	a
	b
	c
	oude PC
MP ->	oude MP
	a
SP ->	b

Na mul:

	a
	b
	c
	oude PC
MP ->	oude MP
SP ->	a*b

Na ldl -4:

	a
	b
	c
	oude PC
MP ->	oude MP
	a*b
SP ->	a

Na ldl -3:

	a
	b
	c
	oude PC
MP ->	oude MP
	a*b
	a
SP ->	b

Na mul:

	a
	b
	c
	oude PC
MP ->	oude MP
	a*b
SP ->	a*b

Na mul:

	a
	b
	c
	oude PC
MP ->	oude MP
SP ->	(a*b)^2

Na stl -2 (tevens eindsituatie):

	a
	b
	(a*b)^2
	oude PC
SP MP ->	oude MP

Opgave 3a

De pagina's zijn 4kB (kilobyte) groot, ofwel 4096 bytes. De offset geeft aan welke byte je wilt benaderen binnen de pagina. De offset heeft dus een waarde tussen 0 en 4095. Er zijn 12 bits nodig om dit te coderen (2^12 = 4096).

Opgave 3b

Er worden 5 hexadecimale cijfers gebruikt om een paginanummer mee te coderen. Elk hexadecimaal cijfer heeft een waarde tussen 0 en 15, of, anders gezegd, kan worden gerepresenteerd met 4 bits. In totaal gaat het dus om 5 * 4 = 20 bits.

Opgave 3c

Een virtueel geheugenadres bestaat uit 20 bits voor het paginanummer en 12 bits voor de offset. Met de resulterende 32 bits kan 4294967296 bytes, ofwel

4GB (gigabyte) geheugen geadresseerd worden. Dat is dus tevens de maximale omvang van het virtuele geheugen.

Opgave 3d

De page table bevat voor elke pagina in het virtuele geheugen een verwijzing naar de lokatie van de pagina in het fysieke geheugen. Omdat de pagina's telkens op grenzen van 4kB beginnen, is het niet nodig een volledig fysiek geheugenadres op te nemen, maar kan worden volstaan met het registreren van het paginanummer (de vertaling naar een geheugenadres is dan: het paginanummer * 4096).

Merk op dat het virtuele paginanummer niet opgeslagen wordt; dat is immers de index in de page table. Er zijn maximaal 16^5 pagina's (immers: het paginanummer bestaat uit 5 hexadecimale cijfers), ofwel maximaal 1048576 page table entries.

Elk van deze page table entries neemt minimaal 20 bits in beslag: de entry bestaat zoals gezegd uit een paginanummer en een paginanummer bestaat uit 5 hexadecimale cijfers.

Minimaal is dus 20 bits * 1048576 ofwel 20 megabit geheugen nodig. In de regel zal een page table entry voor het gemak in volledige bytes gecodeerd worden. In dat geval zijn er 3 bytes (= 24 bits) per entry nodig. Er moet immers een paginanummer van 20 bits in passen. De resterende 4 bits per page table entry kunnen dan bijvoorbeeld nog voor permissie-data gebruikt worden (of desnoods leeg blijven). In dat geval is er 1048576 * 3 bytes = 3MB (megabyte) geheugen nodig voor de page table.

Opgave 4

Een draaiend proces kan worden afgewisseld door de volgende oorzaken:

- Het proces maakt z'n timeslice vol, waarna het OS de controle overneemt en een ander proces de CPU kan krijgen.

- Het proces heeft een resource nodig en voert daarom een system call uit, waarna het OS de controle overneemt.

- Er treedt een interrupt op, waardoor een interrupt routine geactiveerd wordt en het proces de controle over de CPU kwijtraakt.

- Het proces is helemaal klaar met z'n werk en termineert; de controle wordt vrijwillig aan het OS overgedragen.

Opgave 5a

Een kritieke sectie is een stuk programma waarin een shared resource bewerkt wordt. Een shared resource is een resource die toegankelijk is voor verschillende

processen of threads. Toegang tot een shared resource moet gecontroleerd gebeuren om te voorkomen dat processen/threads elkaars waarde overschrijven/de verkeerde waarde lezen (kortom: er is synchronisatie nodig). Kritieke secties worden gedefinieerd als er een risico is dat door gelijktijdige toegang inconsistenties ontstaan. Door een stuk programma als kritieke sectie aan te duiden kan bewerkstelligd worden dat slechts één proces of thread tegelijkertijd toegang krijgt tot dit stuk programma. Het gevolg daarvan is dat ook slechts één proces tegelijkertijd toegang heeft tot de shared resource.

Opgave 5b

Neem het voorbeeld van een bankoverschrijving:
- Proces A wil van rekening x naar rekening y overschrijven.
- Proces B wil van rekening z naar rekening x overschrijven.

Ga uit van deze volgorde van de relevante operaties:
- Proces A leest het saldo van x (de shared resource).
- Proces B leest het saldo van x.
- Proces A verlaagt het saldo van x.
- Proces B verhoogt het saldo van x.
- Proces A schrijft het nieuwe saldo van x terug (de shared resource).
- Proces B schrijft het nieuwe saldo van x terug.

De laatste operatie overschrijft het door proces A nieuw teruggeschreven saldo van x, waarmee de saldoverlaging door proces A verloren gaat.

Als de volledige toegang tot x (dus van lezen t/m schrijven) als kritieke sectie bestempeld zou zijn, dan zou slechts 1 proces bij x kunnen en dat proces zou de overschrijving eerst volledig moeten afronden voordat het andere proces x kan benaderen. Merk op dat het dan niet uitmaakt of eerst proces A of eerst proces B z'n werk doet.

Opgave 5c

Locking in de context van files. Het essentiële verschil is dat bij file locking de resource zelf op slot wordt gezet, terwijl bij een kritieke sectie feitelijk de code die toegang geeft tot de resource op slot wordt gezet (met als gevolg dat de resource zelf ook afgeschermd is).

Opgave 6a

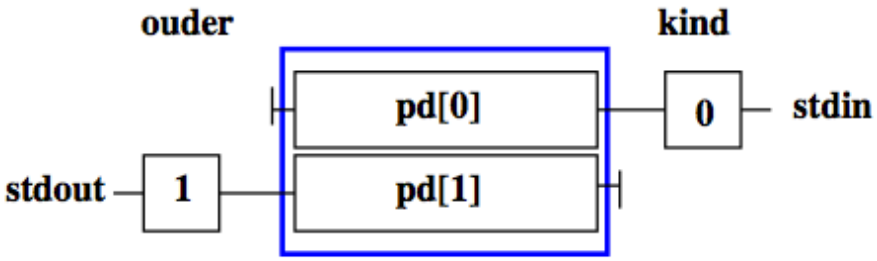
Er wordt een pipe aangemaakt binnen een proces, waarna dit proces een kind-proces aanmaakt; het kind sluit vervolgens het schrijfeinde van de pipe en de ouder sluit het leeseinde van de pipe, waarna de ouder via de pipe met het kind kan communiceren (richting: van ouder naar kind).

Opgave 6b

A: er wordt een kindproces afgesplitst dat een kopie is van het ouderproces.

B: er vindt een controle plaats of we in het kind of in de ouder zitten. Deze tak is voor het kind (pid == 0).
C: het kind sluit het schrijfeinde van de pipe.
D: het leeseinde van de pipe wordt naar de plek van de standaardinvoer gekopieerd in de open file table van het proces (fileid 0 is de standaardinvoer).
E: het kind sluit het leeseinde van de pipe, na het kopiëren.
F: het kind wisselt van code; er wordt nieuwe code geladen en uitgevoerd.
G: Deze tak is voor de ouder (pid != 0).

Opgave 6c



Opgave 7a

Het bericht dat er nieuwe data klaarstaat is veel korter dan de data zelf. Shared files of shared memory worden veelal gebruikt om grotere hoeveelheden data uit te wisselen (grofmazig). Voor het uitwisselen van korte berichten bestaan vaak efficiëntere IPC mechanismen binnen een OS.

Opgave 7b

Het bericht dat er nieuwe data klaarstaat kan bijvoorbeeld via message passing op efficiënte wijze worden verzonden.

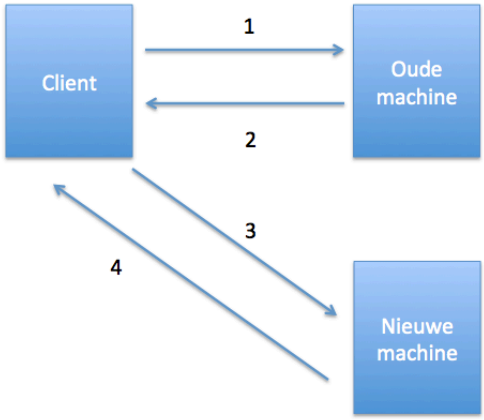
Opgave 7c

Het proces zou de shared resource regelmatig kunnen benaderen om te kijken of er al nieuwe data klaarstaat. Dat is feitelijk een vorm van polling. Het nadeel is dat het benaderen van een shared resource een relatief dure operatie is (system calls!) en vaak het resultaat van de operatie uitsluitend de constatering is dat er nog geen nieuwe gegevens klaarstaan.

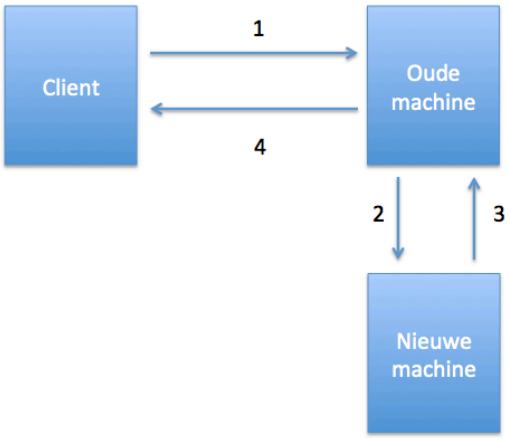
Opgave 8a

De oorspronkelijke machine kan een foutmelding geven. De client moet nu zelf op zoek gaan naar de nieuwe lokatie van de server. Dit kan via een name server (figuur weggelaten).

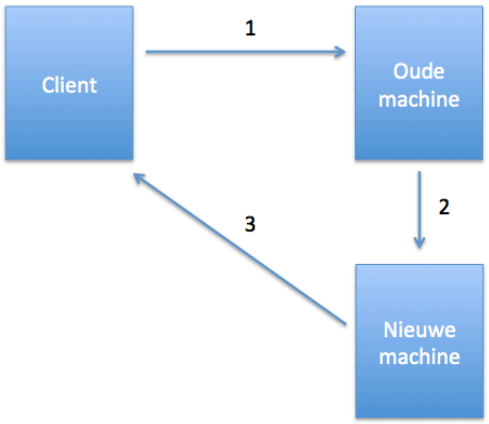
De oorspronkelijke machine kan de client laten weten waar naartoe de server verhuisd is. Daarna neemt de client zelf contact op met de nieuwe machine.



De oorspronkelijke machine kan het request doorsturen naar de machine waar de server zich nu bevindt. De nieuwe machine kan het antwoord terugsturen naar de oude machine, die op zijn beurt weer het antwoord naar de client stuurt.



De oorspronkelijke machine kan het request doorsturen naar de machine waar de server zich nu bevindt. De nieuwe machine kan het antwoord rechtstreeks terugsturen naar de client, die daardoor weet dat de server verhuisd is.



Opgave 8b

De name server is een single point of failure: als dit proces onderuit gaat, gaat feitelijk het hele netwerk down, omdat de processen niet meer kunnen achterhalen waar de services te vinden zijn.

De name server is een bottle neck: deze server moet veel requests afhandelen. Er zal al gauw een wachtrij ontstaan van requests, waardoor de processen die een request hebben uitstaan noodgedwongen in de wacht worden gezet.

Opgave 8c

Elke server doet een broadcast zodra deze start om z'n diensten aan te kondigen aan alle clients op het netwerk (en bij voorkeur ook periodiek daarna, zodat ook nieuwe clients op de hoogte zijn). De clients zien de broadcasts en slaan de gegevens van de servers in hun lokale database op.

Een client die een server wil benaderen maar niet weet waar deze zich bevindt, doet een verzoek via een broadcast. Een server die de dienst kan verlenen reageert daarop, eveneens via een broadcast. Alle andere clients zien de broadcasts en slaan de gegevens van de servers in hun lokale database op.

Antwoorden multiple choice vragen

1. **D**
2. **B** (deze vraag is wat verwarrend geformuleerd: 'voor welke van de onderstaande operaties moet een programma system mode rechten hebben' zou beter zijn geweest)
3. **C**
4. **C**
5. **D** (in deze oefentoets is dit antwoord van de pagina gevallen. De bijbehorende tekst was: 'Alle pagina's moeten even groot zijn'.)
6. **E**
7. **D**
8. **B**
9. **C**
10. **A**