Les B-03 Technologie: de werking van de processor

3.0. Doel

De gebruiker van een computer voert begrijpelijke informatie in (opdrachten, procedures, programma's, gegevens, bestanden) en krijgt via het beeldscherm, printer of modem ook begrijpelijke informatie uitgevoerd. In het inwendige van de computer daarentegen zorgt een grote hoeveelheid duizelingwekkend snelle elektrische stroompjes voor de echte afhandeling van opdrachten die voor de uitvoer zorgen zoals die aan de gebruiker verschijnt.

In de lesbrieven over binaire representatie van getallen, tekens en beelden is uitgelegd hoe informatie omgezet kan worden in reeksen van 0-en en 1-en. Via de processor van de computer worden deze reeksen van 0-en en 1-en met de verschillende componenten van de computer uitgewisseld. De reeksen van 0-en en 1-en hebben elke keer weer een andere betekenis. De ene keer is het een instructie voor de processor (sla op, tel op, etc.), de andere keer een adres (in het werkgeheugen of achtergrondgeheugen) en een andere keer weer een symbool (getal of symbool). In deze lesbrief wordt uitgelegd hoe de processor deze reeksen van 1-en en 0-en elektronisch verwerkt.

De lesbrief is in hoofdlijn een vertaling van informatie die te vinden is op de webpagina www.howstuffworks.com . Op deze webpagina zijn allerlei wetenswaardigheden te vinden omtrent de werking van zaken die je in het dagelijks leven tegenkomt en de computer in het bijzonder.

De begrippen die je aan het eind van deze lesbrief moet kennen en begrijpen zijn:

- ► microprocessor
- ► kloksnelheid (Hz) en MIPS
- ► assembleertaal
- ► onderdelen van de processor
- ► ALU
- ► instructieset
- ► Von Neumann cyclus

3.1 De geschiedenis van de microprocessor

Een **microprocessor** (ook wel CVE = Centraal Verwerkende Eenheid of CPU = Central Processing Unit genoemd) is een complete rekenmachine gemaakt op één enkele chip. De eerste microprocessor was de Intel 4004 en werd geïntroduceerd in 1971. De Intel 4004 was niet erg krachtig – de processor kon slechts optellen en aftrekken – en kon slechts 4 bits tegelijk verwerken. Het verbazingwekkende van de processor was op dat moment dat deze rekenmogelijkheden op één enkele chip aanwezig waren. Vóór de Intel 4004 werden computers namelijk gebouwd uit verzamelingen van chips en transistoren.

De eerste microprocessor die werd gebruikt in de PC was de Intel 8080, een 8-bits machine (1974). De Intel 8088 werd in 1979 geïntroduceerd en in 1982 toegepast in de IBM PC. De Intel microprocessor ontwikkelde vervolgens van de 8088 naar de 80286, 80386, 80486, Pentium I, Pentium II, Pentium III en de tegenwoordig (2003) gebruikte Pentium 4 processor. Al deze processoren zijn verbeteringen van het basisontwerp van de 8088. De Pentium 4 kan dan ook alle programmacode aan die de 8088 aankon, maar doet dat 5000 keer sneller!

De volgende tabel toont de verschillen tussen de verschillende processoren die Intel door de jaren heen heeft gemaakt:

Naam	Jaar	Transistoren	Micron	Klok snelheid	Data breedte	MIPS
8080	1974	6,000	6	2 MHz	8 bits	0.64
8088	1979	29,000	3	5 MHz	16 bits 8-bit bus	0.33
80286	1982	134,000	1.5	6 MHz	16 bits	1
80386	1985	275,000	1.5	16 MHz	32 bits	5
80486	1989	1,200,000	1	25 MHz	32 bits	20
Pentium	1993	3,100,000	0.8	60 MHz	32 bits 64-bit bus	100
Pentium II	1997	7,500,000	0.35	233 MHz	32 bits 64-bit bus	~300
Pentium III	1999	9,500,000	0.25	450 MHz	32 bits 64-bit bus	~510
Pentium 4	2000	42,000,000	0.18	1.5 GHz	32 bits 64-bit bus	~1,700

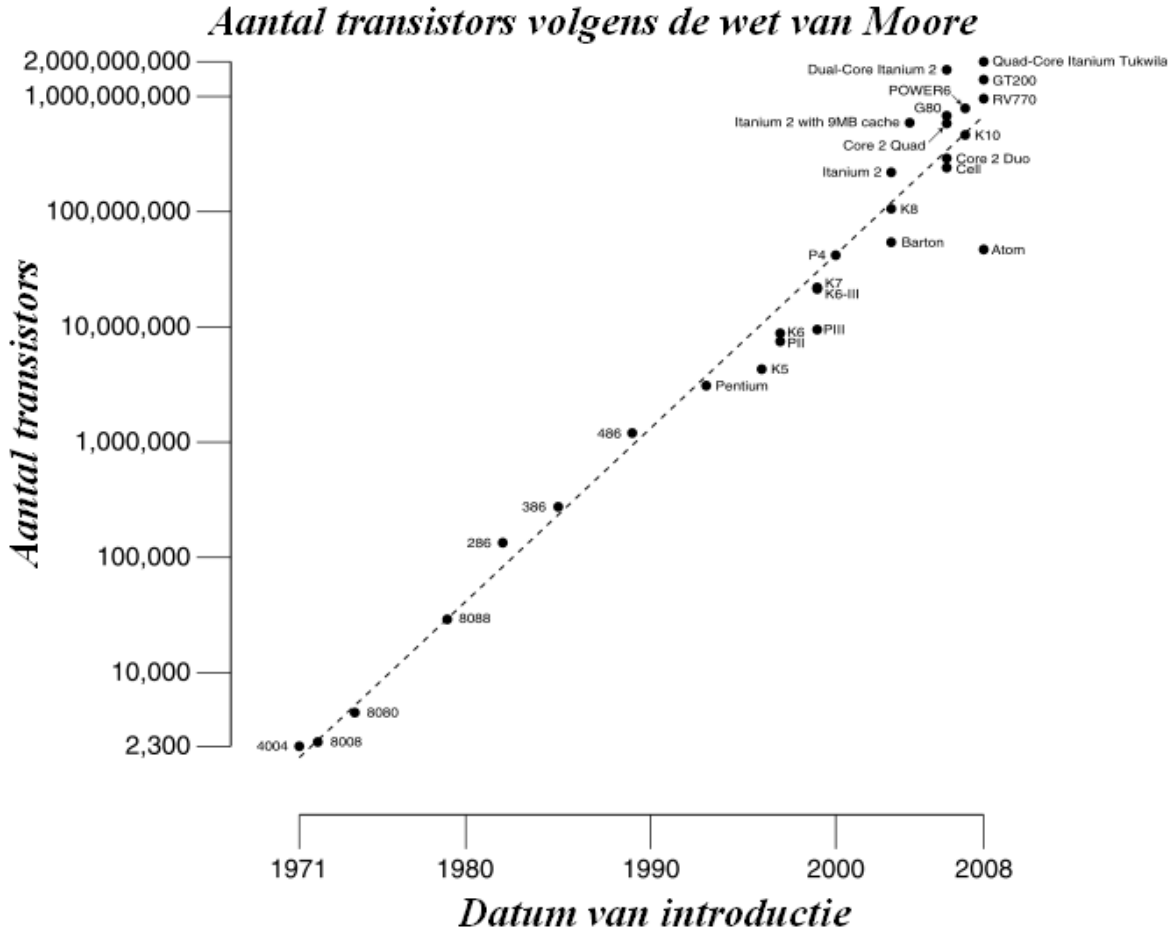
tabel 3.1. overzicht van Intel microprocessoren tot het jaar 2000

Jaar	het jaar van introductie
Transistoren	het aantal transistoren op de chip
Micron	dikte van de smalste draad op de chip N.B. Een menselijke haar is 100 micron dik.
Kloksnelheid	de snelheid waarmee de computerklok “tikt”
Databreedte	de breedte van de ALU (Arithmetic Logical Unit). Een 8-bits ALU kan twee 8-bits getallen optellen / aftrekken / vermenigvuldigen / enz.
MIPS	een grove schatting van het aantal opdrachten dat de processor per seconde kan verwerken (Miljoenen Instructies Per Seconde).

Uit de tabel kan je opmaken dat er een verband is tussen de **kloksnelheid** en het **aantal MIPS**. De 8088 is geklokt op 5 MHz (5 miljoen "tikken" per seconde) en voert 0,33 MIPS (0,33 miljoen opdrachten per seconde) uit. Dat betekent dat er één opdracht in 15 kloktikken wordt uitgevoerd. Moderne processors voeren opdrachten vaak in 2 kloktikken uit.

Voor de meest recente lijst met Intel processors:
http://nl.wikipedia.org/wiki/Lijst_van_Intel-processors

De explosieve ontwikkeling van de microtechnologie die geleid heeft tot steeds snellere processors werd in 1965 al voorspeld door **Gordon Moore**, één van de oprichters van Intel.



figuur 3.1.Wet van Moore – ontwikkeling aantal transistors per chip

Moore's (in 1970 aangescherpte) wet voorspelt dat iedere twee jaar het aantal transistors in chips verdubbelt.

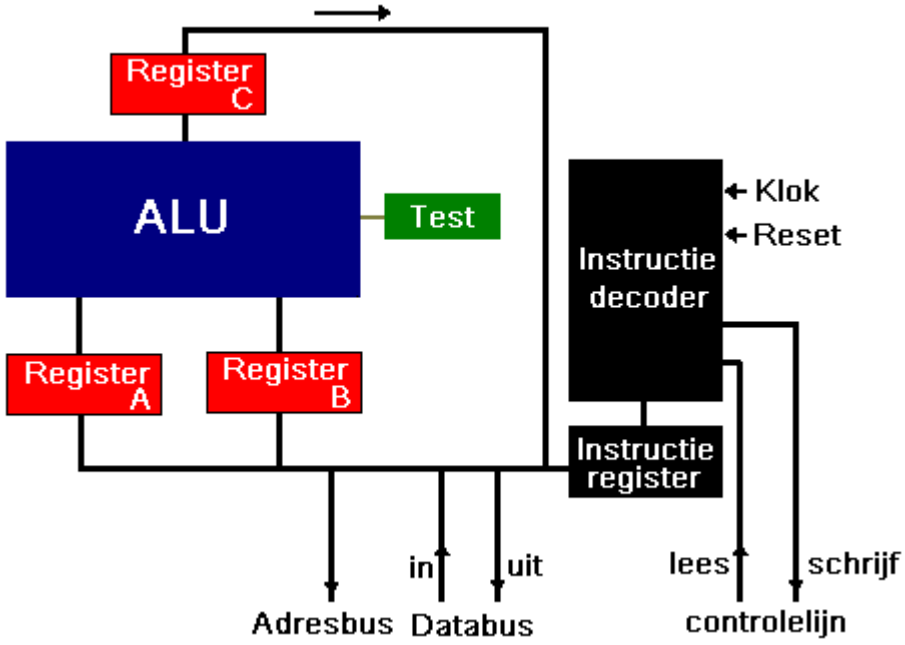
Anno 2006 lijkt de groei van de kloksnelheid van de chips te stagneren en komt niet meer boven de 3,8 GHz. Als oplossing daarvoor plaatsen de chipfabrikanten meerdere processoren (ook wel cores genoemd) op een chip. **Dual cores** worden in grote mate in consumentencomputers gebruikt, en er zijn zelfs chips met 80 floating-point cores. Deze parallelisatie kan echter alleen goed benut worden als de software hierop wordt aangepast (zoals met multithreading of SOA). Zo kan de Wet van Moore nog wel een tijdje geldig blijven.

Eind 2006 kondigde de toen gepensioneerde Gordon Moore aan dat zijn wet niet eeuwig geldend zou zijn. Ook de microtechnologie heeft tenslotte zijn grenzen.

3.2 De opbouw van een (eenvoudige) processor

De processor voert een aantal machine instructies uit. De taal waarin deze instructies zijn geschreven heet ook wel de **assembleertaal** (assembler code). Uitgaand van deze machine instructies doet een processor eigenlijk maar drie dingen:

1) Gebruik makend van de **ALU** (Arithmetic Logical Unit) verricht de processor wiskundige handelingen als optellen, aftrekken, delen en vermenigvuldigen.
2) De processor verplaatst gegevens van de ene (geheugen)plaats naar de andere.
3) De processor kan de beslissing nemen met een nieuwe instructieset aan de slag te gaan.



figuur 3.2. globale indeling van de processor

Een eenvoudige processor bestaat uit de componenten:

1. de ALU en zijn registers
De ALU voert een rekenkundige bewerking uit op de inhoud van twee registers (A en B) en plaats het resultaat in een register (C) of in een testregister

2. adresbus (8, 16 of 32-bits):
zendt een adres naar het geheugen

3. databus (8, 16 of 32-bits):
zendt gegevens naar of ontvangt gegevens van het geheugen

4. schrijf (WR) en lees (RD) lijn:
dienen om aan te controleren of er gegevens moeten worden geschreven naar of gelezen uit het geheugen.

5. kloklijn
zendt de klokpuls naar de processor

6. resetlijn
stelt de programmateller op 0 en herstart het uitvoeren van een programma.

3.3 De instructieset van de processor

De verzameling van mogelijke machine-instructies (**instructieset**) bij de processor van de vorige pagina is:

- **LOADA mem** Laad register A vanuit geheugenadres ….
- **LOADB mem** Laad register B vanuit geheugenadres ….
- **CONB con** Laad een constante waarde in register B
- **SAVEB mem** Bewaar de inhoud van register B in geheugenadres ….
- **SAVEC mem** Bewaar de inhoud van register C in geheugenadres ….
- **ADD** Tel A en B op en bewaar het resultaat in C
- **SUB** Trek A en B af en bewaar het resultaat in C
- **MUL** Vermenigvuldig A en B en bewaar het resultaat in C
- **DIV** Deel A en B en bewaar het resultaat in C
- **COM** Vergelijk A en B en bewaar het resultaat in C
- **JUMP addr** Spring naar adres ….
- **JEQ addr** Spring, indien gelijk, naar adres ….
- **JNEQ addr** Spring, indien ongelijk, naar adres ….
- **JG addr** Spring, indien groter dan, naar adres ….
- **JGE addr** Spring, indien groter of gelijk, naar adres ….
- **JL addr** Spring, indien kleiner dan, naar adres ….
- **JLE addr** Spring, indien kleiner of gelijk, naar adres ….
- **STOP** Stop programma executie

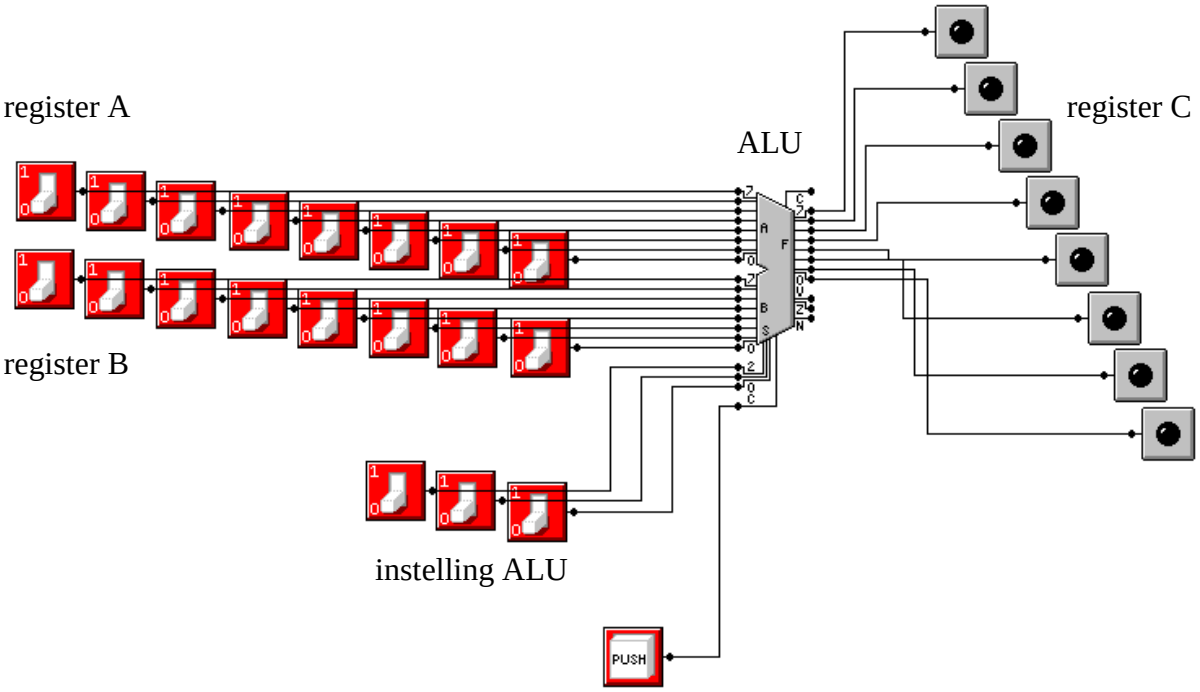
Met deze basisinstructies kan de processor allerlei handelingen verrichten. We beginnen met een eenvoudige optelling.

3.4 De werking van de ALU: een eenvoudige optelling

Als de processor een eenvoudige optelling wil verrichten moet het volgende gebeuren:

- de data moeten via de data-in lijn in de registers A en B worden gezet
- de ALU moet worden ingesteld op “optellen”
- de output moet in register C worden weggeschreven

We kunnen de werking van de ALU in deze situatie met MMLogic nabouwen:



figuur 3.3. nabootsen processor m.b.v. MMLOGIC

Bouw de schakeling na in MMLogic. We kunnen met deze schakeling een aantal interessante eigenschappen van deze 8-bits ALU onderzoeken.

Daarvoor is het eerst nodig om te weten hoe de ALU ingesteld dient te worden:

000 = optellen
001 = aftrekken
010 = vermenigvuldigen
011 = delen

OPDRACHTEN

Opdracht 3.1
Zet in register A het getal 6 (00000110) en in register B het getal 3 (00000011). Voer met de ALU de volgende berekeningen uit en controleer het (binaire) antwoord in register C:

a) 6 + 3
b) 6 – 3
c) 6 * 3
d) 6 / 3

Opdracht 3.2
Hoe gaat de ALU om met te grote getallen?
Maak een extra LED-je aan de C-uitgang van de ALU vast. Bereken achtereenvolgens:

a) 128 + 32
b) 128 + 64
c) 128 + 128
d) Welke functie heeft de C-uitgang van de ALU bij optellen?

Opdracht 3.3
Hoe gaat de ALU om met negatieve getallen?
Maak een extra LED-je aan de C-uitgang van de ALU vast. Bereken achtereenvolgens:

a) 6 – 5
b) 6 – 6
c) 6 – 7
d) Welke functie heeft de C-uitgang van de ALU bij aftrekken?

Opdracht 3.4
Hoe gaat de ALU om met breuken? Bereken achtereenvolgens:

a) 48 / 6
b) 48 / 8
c) 48 / 10
d) Welke beperking heeft de ALU bij delen?

Tot zover onze kennismaking met de ALU. Wil je meer weten hoe computers met negatieve getallen en kommagetallen werken? Ga naar het collegedictaat van dhr. A. Brouwer van de Technische Universiteit Eindhoven: http://www.win.tue.nl/~aeb/ca/computers.html

3.5. Van C-programma naar machine-instructies

In het volgende voorbeeld wordt beschreven hoe een programma geschreven in de programmeertaal C kan worden vertaald in machine instructies die door de processor kunnen worden uitgevoerd.

We bekijken het volgende programma geschreven in de taal C.
Dit programma rekent het getal 5! = 5 · 4 · 3 · 2 · 1 uit:

```
a = 1;
f = 1;
while (a <= 5)
{
        f  = f * a ;
        a = a + 1;
}
```

Dit programma van 7 regels code blijft een getal f met beginwaarde 1 vermenigvuldigen met een getal a, dat steeds met 1 wordt opgehoogd zolang de waarde 5 nog niet overschreden is:

```
a =1
f =1
is a <= 5?      f = 1 * 1
                a =1 + 1
                is a <=5?       f = 1 * 2
                                a = 2 + 1
                                is a <=5?       f = 2 * 3
                                                a = 3 + 1
                                                is a <= 5?      f = 6 * 4
                                                                a = 4 + 1
                                                                is a <= 5?      f  =  24 * 5
                                                                                a = 5 + 1     STOP
```

We vertalen het programma in machine instructies. Het programma in machine instructies geeft precies weer hoe de onderdelen van de processor moeten worden gebruikt:

```
0       CONB 1          // plaats het getal a=1 in register B)
1       SAVE 128        // bewaar het getal a=1 op geheugenplaats 128
2       CONB 1          // plaats het getal f=1 in register B
3       SAVE 129        // bewaar het getal f=1 op geheugenplaats 129
4       LOADA 128       // plaats het getal a vanuit het geheugen in register A
5       CONB 5          // plaats het getal 5 in register B
6       COM             // vergelijk de registers A (a) en B (5)
7       JG 17           // spring, indien groter (a > 5) naar adres 17 ( = STOP )
8       LOADA 129       // plaats het getal f vanuit het geheugen in register A
9       LOADB 128       // plaats het getal a vanuit het geheugen in register B
10      MUL             // vermenigvuldig f met a
11      SAVEC 129       // bewaar het getal f = f * a op geheugenplaats 129
12      LOADA 128       // plaats het getal a vanuit het geheugen in register A
13      CONB 1          // plaats het getal 1 in register B
14      ADD             // tel de getallen a en 1 bij elkaar op
15      SAVEC 128       // bewaar het getal a = a + 1 op geheugenplaats 128
16      JUMP 4          // spring naar adres 4 ( = HERHAAL )
17      STOP
```

De maker van de processor heeft aan alle mogelijke instructies een uniek nummer gegeven:

1. LOADA	**10. COM**
2. LOADB	**11. JUMP**
3. CONB	**12. JEQ**
4. SAVEB	**13. JNEQ**
5. SAVEC	**14. JG**
6. ADD	**15. JGE**
7. SUB	**16. JL**
8. MUL	**17. JLE**
9. DIV	**18. STOP**

Het programma in assembleertaal kan nu volledig met getallen worden beschreven:

```
0     3       // CONB 1
1     1
2     4       // SAVE 128
3     128
4     1       // CONB 1
5     1
6     4       // SAVEB 129
7     129
8     1       // LOADA 128
9     128
10    3       // CONB 5
11    5
12    10      // COM
13    14      // JG 17 – NU: REGEL 31
14    31
15    1       // LOADA 129
16    129
17    2       // LOADB 128
18    128
19    8       // MUL
20    5       // SAVEC 129
21    129
22    1       // LOADA 128
23    128
24    3       // CONB 1
25    1
26    6       // ADD
27    5       // SAVEC 128
28    128
29    11      // JUMP 4 – NU: REGEL 11
30    8
31    18      // STOP
```

Het programma van 7 regels C code en 17 regels machine instructies is uiteindelijk in 32 getallen vertaald.

Dat wil zeggen dat het programma nu volledig vertaald kan worden in bytes die door de processor gelezen en begrepen kunnen worden:

```
0     3     00000011    // CONB 1
1     1     00000001
2     4     00000100    // SAVE 128
3     128   10000000
4     1     00000001    // CONB 1
5     1     00000001
6     4     00000100    // SAVEB 129
7     129   10000001
8     1     00000001    // LOADA 128
9     128   10000000
10    3     00000011    // CONB 5
11    5     00000101
12    10    00001010    // COM
13    14    00001110    // JG 17 – NU: REGEL 31
14    31    00011111
15    1     00000001    // LOADA 129
16    129   10000001
17    2     00000010    // LOADB 128
18    128   10000000
19    8     00001000    // MUL
20    5     00000101    // SAVEC 129
21    129   10000001
22    1     00000001    // LOADA 128
23    128   10000000
24    3     00000011    // CONB 1
25    1     00000001
26    6     00000110    // ADD
27    5     00000101    // SAVEC 128
28    128   10000000
29    11    00001011    // JUMP 4 – NU: REGEL 4
30    8     00001000
31    18    00010010    // STOP
```

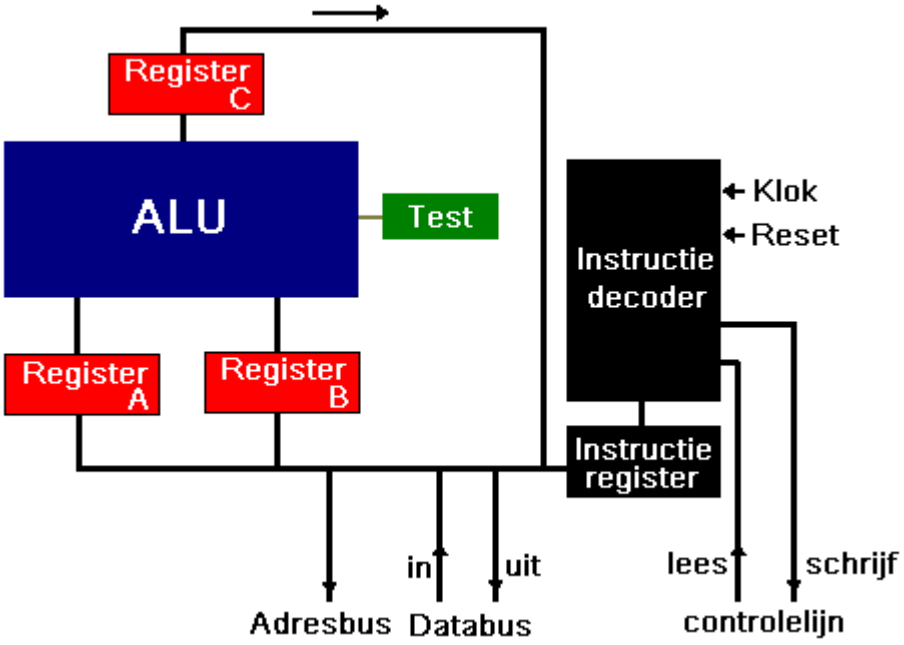
Natuurlijk wordt niet van je verwacht dat je de programmeertaal C en de machine-instructies nu volledig begrijpt. Wel moet je nu een indruk hebben hoe het mogelijk is dat een processor programma-informatie kan verwerken.

Elke instructie van een programma is te vertalen in een aantal getallen, een bitstroom die door de instructiedecoder gelezen en door de onderdelen van de processor verwerkt kan worden.

3.6 De verwerking van machine-instructies

We weten nu hoe een programma kan worden vertaald in machine-instructies en dat deze machine-instructies door de processor kunnen worden verwerkt omdat ze precies aangeven welk onderdeel van de processor wat moet doen. Maar hoe gaat dat dan in zijn werk?

Bij het uitvoeren van het C-programma wordt het programma in assembleertaal van de harde schijf via een externe databus naar het RAM-geheugen gekopieerd. Op een aantal geheugenplaatsen in het RAM-geheugen staan de machine-instructies achter elkaar geplaatst.



figuur 3.4. verwerking van instructies door de processor

De instructies worden één voor één gelezen door de instructiedecoder. Bij elke machine-instructie komt als eerste een byte binnen die aangeeft naar welke locatie bepaalde informatie moet gaan. In het programmavoorbeeld:

```
0     3       00000011   // CONB 1
1     1       00000001
2     4       00000100   // SAVE 128
3     128     10000000
```

staat de eerste byte voor “plaats in register B”. Op de buslijnen van de processor bevinden zich buffers die drie toestanden kunnen hebben (tri-state buffers). Deze kunnen worden geactiveerd (toestand 1) zodat ze informatie doorlaten. De eerste byte zal ervoor zorgen dat de bus naar register B wordt geactiveerd. De tweede byte bevat informatie “het getal 1”. Deze byte wordt op de bus geplaatst en zal belanden in het geactiveerde register B. De derde byte “bewaar op geheugenplaats” zorgt voor het activeren van de adresbus en data-uit bus. Het daarop volgende adres “128” en de data uit register B “het getal 1” kunnen nu via de juiste buslijnen worden verstuurd.

De hierboven beschreven afhandeling van een machine-instructie wordt ook wel de **Von Neumann cyclus** genoemd:

- haal de opdracht op
- plaats de opdracht in de juiste registers
- verwerk de opdracht

De cyclus is vernoemd naar de wetenschapper Von Neumann die als eerste dit verwerkingsprincipe bedacht.

3.7 Moderne processoren

De hierboven beschreven processor is een vereenvoudigde weergave van de Intel 8088 processor. Deze werkte met 8-bits datalijnen, adreslijnen, registers enz. De (relatieve) eenvoud van de processor maakt het mogelijk om de werking van processors te begrijpen. Dat is een stuk lastiger als je een moderne Intel Pentium 4 processor zou bekijken. De principes van de originele Intel 8088 processor zijn in deze moderne processor nog steeds terug te vinden. De processor is alleen nog veel verder ontwikkeld en complexer geworden:

- Er wordt gewerkt met 32 bits datalijnen, adreslijnen, registers enz. in plaats van met 8-bits datalijnen, adreslijnen, registers enz.
- Er worden minder energieverbruikende halfgeleiders gebruikt, waardoor de chip minder warmte ontwikkelt. Ook zijn de technieken om de halfgeleiders op de chip aan te brengen, te etsen, verfijnd. Door deze ontwikkelingen kunnen er meer onderdelen (transistors, registers en cache geheugen) op eenzelfde oppervlakte worden geëtst.
- Doordat de moderne processor uit meer onderdelen bestaat is ook de instructieset groter geworden. Moeilijke berekeningen die op oudere processors uit tientallen machine-instructies bestonden (neem het voorbeeld van het berekenen van 5!) kunnen nu in enkele instructies worden uitgevoerd.
- Doordat de onderdelen op de chip dichter op elkaar zijn komen te liggen konden de machine-instructies sneller achter elkaar worden geschakeld. De klokfrequentie van de processor (aantal tikken per seconde) is toegenomen van 5 MHz (5 miljoen kloktikken per seconde in 1979) tot 3 GHz (3 miljard kloktikken per seconde in 2003).
- Moderne processors hebben meer dan één instructiedecoder en kunnen daardoor naast de instructie die verwerkt wordt ook volgende instructies voorbereiden of instructies die los daarvan staan parallel verwerken. Moderne processors werken dus met meerdere kanalen (pipelines). Deze techniek wordt dan ook wel pipelining genoemd

Bovenstaande ontwikkelingen hebben de processor aanzienlijk sneller gemaakt.

Daarnaast moet worden vermeld dat in deze lesbrief de principes van processors die Intel maakt zijn behandeld. Processors gemaakt door AMD maken gebruik van een andere en snellere techniek om opdrachten af te handelen. Om de prestaties van Intel processors en AMD processors te vergelijken is het dus niet voldoende om naar de kloksnelheid te kijken: AMD processors met een lagere kloksnelheid kunnen dezelfde prestaties leveren als Intel processors met een hogere kloksnelheid.

Bronnen:

http://howstuffworks.com
http://docent.hogent.be/~ldc392/MICROPRO.HTM

ANTWOORDEN

Opdracht 3.1

opgave	register A	register B	instelling	register C	antwoord
6 + 3	00000110	00000011	000	00001001	9
6 – 3	00000110	00000011	001	00000011	3
6 * 3	00000110	00000011	010	00010010	18
6 / 3	00000110	00000011	011	00000010	2

Opdracht 3.2

opgave	register A	register B	instelling	register C	antwoord
128 + 32	10000000	00100000	000	10100000	160
128 + 64	10000000	01000000	000	11000000	192
128 + 128	10000000	10000000	000	00000000	256

Het LED-je brandt en geeft aan dat er **overflow** opgetreden is bij het optellen. De optelling heeft een resultaat dat niet meer in 8 bits is weer te geven.

Samen met de extra (overflow) bit ontstaat het optelresultaat: 100000000

Opdracht 3.3

opgave	register A	register B	instelling	register C	antwoord
6 – 5	00000110	00000101	001	00000001	1
6 – 6	00000110	00000110	001	00000000	0
6 - 7	00000110	00000111	001	11111111	-1

Het LED-je brandt en geeft aan dat er een negatief getal is ontstaan bij het aftrekken.

Samen met de extra bit ontstaat het aftrekresultaat: 1 11111111

Dit getal wordt vertaald in een negatief getal door het tegengestelde te nemen van 11111111, dat is dus 00000000 en daar nog één van af te trekken (zie voor de reden hiervan het collegedictaat http://www.win.tue.nl/~aeb/ca/computers.html).

Zo is het aftrekresultaat van de opgave 0 – 29 het getal 1 11100011.
De bijbehorende waarde is het tegengestelde, dus 00011100 = -28 min één geeft -29

Opdracht 3.4

opgave	register A	register B	instelling	register C	antwoord
48 / 6	00110000	00000110	011	00001000	8
48 / 8	00110000	00001000	011	00000110	6
48 / 10	00110000	00001010	011	00000100	4

De ALU kan alleen in gehelen rekenen en rondt dus af. Naast de ALU zijn processors daarom ook uitgerust met een Floating Point Unit (FPU), die met kommagetallen kan rekenen.