Programmeren

COLLABORATORS			
	TITLE : Programmeren		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		June 4, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Programmeren

1.1 main

```
Nederlands eerste "Echte" magazine voor de beginnende en wat gevorderde
                        programmeurs onder ons

                    Deze magazine word uitgebracht door :

               ------------=*> E·R·R·O·R  <=*---------------


Programmeren
~~~~~~~~~~~~
 Voorwoord                Omrekenen van getallen


OPMERKING : Dit is een zogenaamde preview van het nieuwe zogenaamde
            coders magazine in het nederlands. Mocht je als beginner
            of gevorderde coder mee doen in de zin van sources geven
            enof methodes uitleggen bv over iff formaat hoe & wat.
            Dan laat me dit weten, mijn gegevens staan in de
            sectie 'voorwoord' als je me nodig hebt.

            Alvast bedankt voor je interesse.
```

1.2 voorwoord

```
Welkom,

Wij "ERROR" zijn een betrekkelijk nieuwe groep die ontstaan is door
diverse omstandigheden waardoor in dit groep nieuwe leden zitten maar
ook een aantal leden van het vroegere amigos.

Wegens dat we dus als groep opnieuwe moeten worden vertegenwoordigt
is het de bedoeling om bekend te raken, nou bij deze ben ik
(Lord Coma/eRRoR) begonnen met het idee van een coders magazine
en dan wel nederlands talig.
```

Verder, de grote reden achter de magazine is eigenlijk nog niet eens
het concept van reclame, nee. de bedoeling is om via deze magazine een
soort informatie bron te kweken voor beginners en gevorderden die beter
willen leren coden.

Ik moet bij deze wel vertellen dat ik persoonlijk wel iets van coden
afweet maar ook niet alles, en er kunnen wel eens waar fouten voor
komen in deze magazine, maar het gaat erom dat we zo er samen met
de lezers & schrijvers van deze magazine samen eruit komen om een
zo juist mogelijk informatie bron te creeeren en dat in het nederlands.

Na mijn mening is dit vrij uniek, althans zowiezo een magazine in
het nederlands en dan nog over coden.

Deze magazine is hopelijk een blijver, wat natuurlijk afhangt van
het publiek die dit gaat lezen of supporten op zijn of haar BBS.

Voor eventueele vragen, reacties , ideeen & suggesties dan ben ik
te bereiken op mijn eigen BBS :

LordsCave BBS Deventer 05700-10479 28k8

Of als je brieven stuurt, dan stuur deze naar :

Roelofs,D
Van Ostadestraat 18
7412 RS, Deventer

Alvast bedankt voor je interesse!.

1.3 getallen

Wat is nu een Decimaal getal ?
Wat is nu een Hexadecimaal getal ?
Wat is nu een Biniar getal ?

Van ...	Naar ...
Decimaal	Biniar
Biniar	Decimaal
Decimaal	Hexadecimaal
Hexadecimaal	Decimaal
Biniar	Hexadecimaal
Hexadecimaal	Biniar

1.4 decimaal

Nou, hier kunnen natuurlijk een heel simpel antwoord op geven.

Een decimaal getal is de gewone getallen reeks die we in het
normale leven gebruiken om te tellen, dus 1+1=2 enzovoorts.

Deze getallen zijn dus decimalen.

Het aanduiden van decimalen getallen in assembler heeft vaak
zijn eigen methode, dit gaat vrijwel vaak als :

```
MOVE.L #4,d6
```

Ga dit nu niet verwarren, dat er hier zou staan dat de execbase
in d6 komt te staan, ten tegendeel.. een execbase zal je altijd
via deze methode doen :

```
MOVE.L $4,d6
```

Al zal je mischien denken, althans voor die gene die weten waar
die $ (dollar teken voor staat) een hexadecimaal getal is, is
dit ook wel enigzins het geval, maar er word met een $ tevens
een "adress" mee bedoeld, als je hexadecimale getallen wilt
gebruiken in assembler als ik net met #4 beschreef zou je dat doen
als

```
MOVE.L #$4,d6
```

Maar aangezien een decimale 4, heximaal ook als 4 word aangeduid ga
ik hier niet een #$4 ervan maken, maar andere voorbeelden zijn dus :

```
MOVE.L #15,d6

MOVE.L #$F,d6
```

In beiden (d6) staat dan de waarden "15" , $F is de hexadecimale
waarde die overeenkomt met de decimale waarde #15 maar hier later
meer over.

1.5 hexadecimaal

Als eerst is een Hexadecimaal een korte vorm van noteren van waarden
waarvan de telwijze niet alleen numeriek is maar ook gebruik maakt
van een gedeelte van het alfabet (A t/m F)

Een Hexadecimaal tegen over een Decimaal reeks ziet er als volgt uit :

```
Decimaal     : 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16
Hexadecimaal : 0  1  2  3  4  5  6  7  8  9   A   B   C   D   E   F  10
```

Als dit je eerste aanraking is met hexadecimale getallen dan kan ik me
indenken dat dit verwarrend is, maar op zich is deze vorm van noteren
ook net zo "makkelijk" te gebruiken als je het eenmaal door hebt in
plaats van de decimale reken stelsel

een paar voorbeelden :

```
Decimaal 10    =  Hexadecimaal #$0A
Decimaal 255   =  Hexadecimaal #$FF
Decimaal 93    =  Hexadecimaal #$5D
```

Als je wilt weten hoe je dat zelf kunt omrekenen moet je deze button even drukken.

1.6 biniar

Nou , ook dit is weer een rekenwijze die je na wat oefening ook snel onder de vingers hebt, en biniar is de eigenlijke moedertaal van al je IC's, al is de operating system zelf in een soort gelijke C taal geschreven.

Maar goed, Biniar.. bits & bytes..

De reken wijze van biniare getallen is vrij simpel, als ik nu eens een voorbeeld geef waar ik alleen een 1 byte (8 bits) nodig heb.:

Decimaal #78 = Biniar %01001110

Nou, als je het door zult hebben zo is dat echt niet moelijk te begrijpen. Het gaat pas echt "moelijk" worden zodra je met 4 bytes (32 bits) gaat werken, dan heb je wel een calculator en een stuk papier nodig.

Maar voor 1 byte ziet de tabel er zo uit

	(1 Byte = 8 bits)							
	Bit#7	Bit#6	Bit#5	Bit#4	Bit#3	Bit#2	Bit#1	Bit#0
Decimale waarde :	128	64	32	16	8	4	2	1

Zoals je ziet, elke byte van RECHTS naar links heeft altijd de vorige bit's waarde vermeningvuldig * 2, dus als je met een 2 bytes waarde zou werken, zou een Bit#8 dus een Decimale waarde van 128*2=256 hebben.

Maar goed, nagelang mijn voorbeeld die een decimale waarde van #78 heeft ga ik nu eens voordoen hoe het nu echt zit.

Decimale waarde = #78 , Biniare situatie %00000000

Ik ga kijken welke Decimale waarde van een van de 8 bits ik maximaal kan pakken om deze af te trekken van mijn waarde #78, in dit geval kan ik bit#6 (waarde 64) er van af trekken.

78-64 = 14 , Dit betekent dat bit#6 dus 1 word %01000000

Verder kijkend, zie ik dat Bit#3 de gene is die ik weer van 14 kan afhalen, endus :

14-8 = 6 , Dit is dan bit#3 die dus 1 word, %01001000

en zo verder..

6-4 = 2 , Dit is dan bit#2 die word gezet %01001100

2-2 = 0 , En nu word als laatste bit#1 gezet %01001110

Kort samen gevat in een soort "Denk wijze" :

```
            Ja / Nee :
78 - 128       - 0   : Hou over, 78
78 - 64     1  -     : Hou over, 14
14 - 32        - 0   : Hou over, 14
14 - 16        - 0   : Hou over, 14
14 - 8      1  -     : Hou over,  6
 6 - 4      1  -     : Hou over,  2
 2 - 2      1  -     : Hou over,  0
 0 - 0           0   : Hou over,  0
```

Opmerking, 0 - 0 is natuurlijk onzinning, dus die word dan ook als NEE beshouwt en blijft die specifieke bit#0 op 0 dus.

Nou als je alle bits naast elkaar zet van boven naar beneden krijg je dus : %01001110

Nou, Ik moet natuurlijk zeggen, 8 bits.. komt natuurlijk voor, maar je kan maximaal met 8 bits maar een waarde bereiken van 0 t/m 255 Dus als je met Decimale getallen van bijvoorbeeld 946 zit kom je dus in aanraking met de 2 bytes (16 bits) systeem, die natuurlijk net zo werkt als de 1 byte (8 bits) alleen werk je met wat hogere waarden.

1.7 dectobin

Het omrekenen van decimale getallen naar biniar heb ik al grotendeels beschreven in een andere sectie, druk deze button om daar heen te gaan.

1.8 bintodec

Het omrekenen van een biniar getal is vrijwel heel simpel, het is een kwestie van kennis of een rekenmachine bij meerdere byte's om daarvan de waarde te weten, zoals ik al eerder heb vermeld heeft een 1 byte ofwel 8 bits een max waarde van 0 t/m 255 dus

```
                  BIT#7  BIT#6  BIT#5 BIT#4  BIT#3  BIT#2  BIT#1  BIT#0
Decimale waarde :   128     64     32    16      8      4      2      1
```

Dus als ik een gegeven biniare reeks heb van bijvoorbeeld :

%10101010

dan kun je de waarde gewoon uitrekenen als op deze manier :

```
BIT#7  = 1, dus waarde ervan is,      128
BIT#6  = 0, dus heeft geen waarde,      0
BIT#5  = 1, dus waarde ervan is        32
BIT#4  = 0, dus heeft geen waarde,      0
BIT#3  = 1, dus waarde ervan is         8
BIT#2  = 0, dus heeft geen waarde,      0
```

```
BIT#1  = 1, dus waarde ervan is         2
BIT#0  = 0, dus heeft geen waarde,      0
------------                          -----
 %10101010              =              170
```

1.9 dectohex

Het omrekenen van decimale getallen naar hexadecimaal heb ik al grotendeels beschreven in een andere sectie, druk deze button om daar heen te gaan.

1.10 hextodec

Het omrekenen van een hexadecimaal naar een decimaal gaat vrijwel makkelijk maar wel is het handig om een rekenmachine te gebruiken bij grotere waarden, want deze doet het op de meeste momenten sneller voor je, maar goed, we gaan voor de kleine getallen wel ff ons best doen om dat zonder rekenmachine uit te rekenen.

Een hexadecimaal waarde loopt van 0 t/m F dus een klein overzicht :

```
Hexadecimaal : 0 1 2 3 4 5 6 7 8 9  A  B  C  D  E  F
Decimaal     : 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

Zoals je ziet zijn er 16 waarden in zon reeks, want na F komt $10.
De hexadecimale waarde van $10 is dus Decimaal 16. Hoe je dit nu uitrekent is vrij simpel.

```
Hexadecimaal =              1 0
                           /   \
                       *16^1   *16^0
                       /         \
                    1*16          0*1
                   -----         -----
                     16     +      0    = Decimaal 16
```

Dus bij een hexadecimale getal van bijvoorbeeld $9A3F is natuurlijk lekker een hoog getal, maar het uitrekenen ervan is vrijwel simpel want je gaat altijd dit systeem toe passen :

```
                           $9A3F
```

Je zit dus 4 waarden, dus de 16^.. reeks loopt als volgt :

```
                  16^3   16^2   16^1  16^0
```

Als we dit nu even toepassen als zonet met de vorige voorbeeld krijg je dus dit :

```
                        9        A         3        D
```

```
        /         |         |         \
     16^3       16^2      16^1       16^0
      /           |         |           \
 9 * 4096     10 * 256   3 * 16        13 * 1
 --------     --------   ------        ------
  36864    +    2560   +   48     +      13


                  = 39485
```

1.11 bintohex

Om een biniar getal naar een hexadecimaal getal te gaan rekenen doe
je dat als volgt :

Biniar getal : %01011001

Zoals je ziet is dit een decimale waarde van 89 maar om hier nu een
Hexadecimaal getal van te krijgen ga je iets "grapigs" uithalen en
dat is te zeggen van dat we deze 8 bits in 2 gedeeltes maken, dus.

```
                        %01011001

                          word

              %0101          en          %1001
```

Nu moet je er van uit gaan dat we nu met 4 bits te maken hebben van
een zogenaamde nieuwe biniare getal, dat betekent dus dat

```
BIT#3   BIT#2   BIT#1   BIT#0
    8       4       2       1
```

de "nieuwe" waardes zijn, als je nu 4 bits allemaal 1 waren, dan zou
dat een totaal zijn van 15, ofwel $F
Dus wat doe je nu, je telt van beide gedeeltes het totaal bij
elkaar, Dus zo :

%0101 = Decimaal 5   en   %1001 = Decimaal 9

nu dan zet je 5 en 9 naast elkaar met een $ teken ervoor en je hebt je
Hexadecimale waarde.. dus : $59

Want 5*(16^1) + 9*(16^0) = Decimaal 89

1.12 hextobin

Nou dit is net zo eenvoudig als we dit andersom deden wat ik besproken
had in een andere sectie van het omrekenen van Decimalen naar Hexadecimalen.
maar nu doe je dit gewoon zo.

Laten we zeggen, ik heb een hexadecimale waarde van $F9
Ik splits dit getal in 2 cijfers..

$F en een $9

Decimaal is de $F een waarde van 15, en een $9 van gewoon 9
Zet ik het getal 15 dus om in een biniare vorm krijg ik dus.

%1111 = Decimaal 15 en Hexadecimaal een $f

En het getal 9 is in een biniare vorm van.

%1001 = Decimaal 9 en Hexadecimaal een $9

Voeg ik deze samen, waarbij natuurlijk ik eerst de $F moet plaatsen
en daar achteraan de $9 word dit dus :

%1111 en %1001 = %11111001

En daar hebben we onze biniare vorm van de hexadecimale waarde van $F9