

Computer systems 2017 -2018

Assembler

“De eerste stappen”

HOGESCHOOL PXL

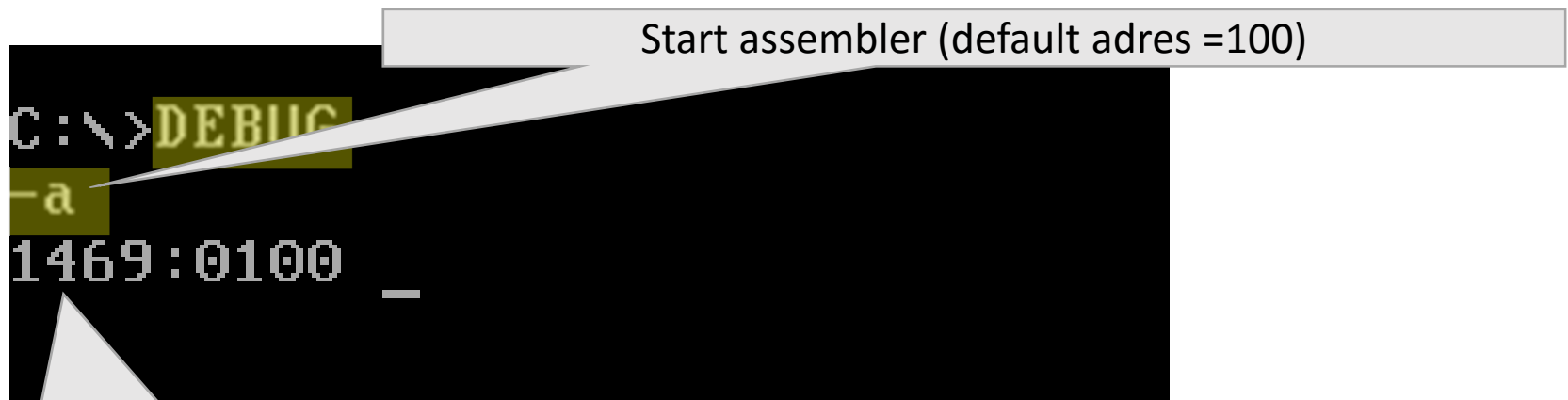


Assembler, de eerste stappen ...

Instructie 'a' → = start van assembler,
Default op adres 0100

Starten vanaf een ander adres:

→ bijvoorbeeld a 01FF → start assembler vanaf adres 01FF



```
C:\>DEBUG
-a
1469:0100
```

Start assembler (default adres =100)

Opmerking: De segmentwaarde (1469) kan verschillen!

Assembler, de eerste stappen ...

Basis bewerkingen

MOV AX, BX

MOV AL, BL

MOV AH, 02

Opgelet MOV instructies!
MOV [DESTINATION] [SOURCE]

De waarde van het BX register wordt
verplaatst naar het AX register

**Het vullen van registers in een assembler
programma wordt gedaan met de MOV
instructie.**

ADD AX, BX

ADD AX, 20

ADD AH, 20

SUB AX, BX

ADD = optelling van verschillende registers
SUB = een aftelling ...

INC AX

DEC AX

INC = increment (één bijtellen)
DEC = decrement (één aftellen)

Assembler, de eerste stappen ...

Basis bewerkingen

MOV instructie:

MOV AH, 02

“Plaats de waarde 02 in het AH register”.

MOV DL, AL

“Plaats de waarde van het AL register in het DL register”

Opmerking voor MOV DL, AL:

- DL staat voor het laag gedeelte (8bit) van het DX register (16bit)
- De actuele waarde in het DL register wordt overschreven.
- De actuele waarde in het AL register blijft hetzelfde.

MOV DX, AX

“Plaats de waarde van het AX register (=16-bit) in het DX register (=16bit)”

Assembler, de eerste stappen ...

Interrupts

= signaal naar de processor voor een “speciale gebeurtenis”

Verschillende interrupt routines + subroutines:

- *INT 20 = Afsluiten van een programma!!*
- *INT 21 = interactie met het scherm*
 - SUBF. 02 = AH=02 → één teken op het scherm tonen (uit DL).
 - SUBF. 09 = AH=09 → String naar het scherm (pointer in DX).
 - SUBF. 01 = AH=01 → Tekens inlezen van het scherm (in AL).

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 02

“Een eerste programma zal simpelweg de letter ‘a’ op het scherm tonen.”

Een aantal opmerkingen voor het eerste programma:

1. Een karakter printen kan met interrupt 21 en subfunctie 02
2. Wanneer interrupt 21 subfunctie 02 wordt opgeroepen bepaalt de waarde in het DL register het karakter dat wordt afgeprint.
→ Het karakter dat overeenkomt met de ASCII waarde in het DL register wordt op het scherm getoond bij het uitvoeren van INT21 subfunctie 02.
3. Alvorens interrupt 21 wordt opgeroepen moet dus:
 1. Het register AH gevuld zijn met de waarde 02. Dit is om aan te duiden dat we subfunctie 2 willen gebruiken.
 2. Het register DL gevuld zijn met de ASCII waarde van het te printen karakter.
(bijvoorbeeld de waarde 41 voor het karakter A)

Opmerking: Dit moet je niet vanbuiten kennen. Wel kunnen toepassen!

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 02

Assembler code:

verklaring code:

| | |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| MOV AH, 02 | → 02 in het AH register. (Dit is om subfunctie 02 te 'activeren') |
| MOV DL, 41 | → 41 = ASCII 'A' INT21 subfunctie 02 zal dit karakter afprinten. |
| INT 21 | → De interrupt drukt één karakter af Aangezien AH=02 → subfunctie. Het karakter is 'A' Aangezien het DL register 41 als waarde heeft). |
| INT 20 | → INT 20 sluit het programma af! Wanneer deze interrupt wordt opgeroepen beëindigd het programma. |

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 02

A 100: start assembler mode vanaf adres 100

```
-a 100
1469:0100 MOV AH, 02
1469:0102 MOV DL, 41
1469:0104 INT 21
1469:0106 INT 20
1469:0108
```

Hier kan rechtstreeks assembler code worden ingegeven.
De adressen worden automatisch aangepast. (Dit voorbeeld is een programma van adres 100 tot adres 108)

... druk op 'enter' om uit assembler mode te gaan.

Opmerking:

Het DX register is 16 bit groot en kan worden opgesplitst in een DL (8bit) en DH (8bit) register.
In dit voorbeeld wordt enkel het DL register gebruikt.

De instructie MOV DX, 0041 zou hetzelfde resultaat geven...

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 02

Test het programma m.b.v. 'r' & 't' & 'p' = "stap voor stap"

```
-r
AX=014C  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1469  ES=1469  SS=1469  CS=1469  IP=0100  NU UP EI PL NZ NA PO NC
1469:0100 B402          MOV     AH,02
-t
AX=024C  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1469  ES=1469  SS=1469  CS=1469  IP=0102  NU UP EI PL NZ NA PO NC
1469:0102 B241          MOV     DL,41
-t
AX=024C  BX=0000  CX=0000  DX=0041  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1469  ES=1469  SS=1469  CS=1469  IP=0104  NU UP EI PL NZ NA PO NC
1469:0104 CD21          INT     21
-
```

r = Opvragen van Register waarde

t = Trace (instructie wordt uitgevoerd en (nieuwe) register waardes worden getoond.

p = Proceed (Interrupt wordt uitgevoerd)

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 02

INT21 & AH=02 → Karakter in het DL register wordt afgedrukt (41 in het voorbeeld).

```
AX=0241 BX=0000 CX=0000 DX=0041 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1469 ES=1469 SS=1469 CS=1469 IP=0104  NV UP EI PL NZ NA PO NC
1469:0104 CD21
INT 21
-p
A
AX=0241 BX=0000 CX=0000 DX=0041 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1469 ES=1469 SS=1469 CS=1469 IP=0106  NV UP EI PL NZ NA PO NC
1469:0106 CD20
INT 20
-p
Program terminated normally
```

De (ASCII) waarde in register DL wordt afgedrukt op het scherm.

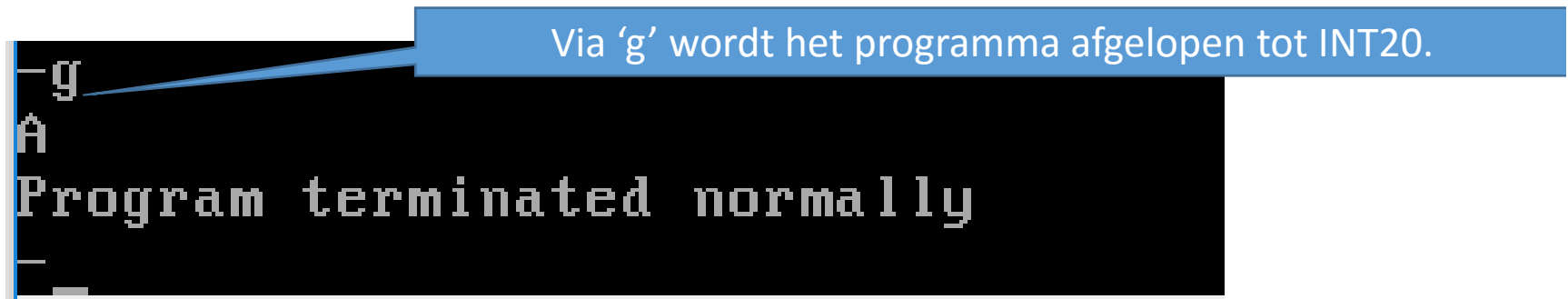
Einde programma ...

Een interrupt wordt uitgevoerd door proceed ('p').

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 02

Test van het programma m.b.v. 'g' = GO



A screenshot of a DOS command prompt window. The prompt is a hyphen followed by a space. The user has entered the command 'g' and pressed enter. The output shows 'A' on the next line, followed by 'Program terminated normally' on the line after. A blue callout box points from the 'g' command to the text 'Via 'g' wordt het programma afgelopen tot INT20.'

```
-g  
A  
Program terminated normally  
-
```

Via 'g' wordt het programma afgelopen tot INT20.

Opgelet!

Zorg dat de instructiepointer verwijst naar de locatie waar de programma code start!! (Meestal 100)

Controleren/veranderen van de instructiepointer door het commando 'r ip'

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 09

In een tweede programma wordt niet één karakter maar een reeks van karakters afgeprint. Dit kan met behulp van INT 21 subfunctie 09.

INT21–SUB09 zal een reeks van karakters afprinten.

De (te printen) string wordt bepaalt door de waarde in het DX register.

In tegenstelling tot INT21 SUB02 bevat het DX register niet de ASCII waarde van het te printen karakter maar het adres waar de te printen karkaters zich bevinden. (=pointer)

Wanneer INT21-SUB09 wordt opgeroepen zullen achtereenvolgens karkaters geprint worden, startend vanaf de adreslocatie weergegeven in het dx register. Wanneer INT21-sub09 de ASCII waarde van een dollar teken (=ASCII 24) tegenkomt wordt de interrupt (en het printen van karakters) beëindigd.

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 09

Invoeren van karakters op een bepaalde geheugenlocatie.

Dit kan m.b.v. het 'e' (=enter) commando.

Bijvoorbeeld:

```
E 150 41 42 43 44 45 46 47 24
```

- Op adres locatie 150 staat de waarde 41, op adres locatie 151 de waarde 42, op adreslocatie 153 de waarde 43, ...
- De waardes 41 42 43 44 45 46 47 komen overeen met de ASCII karakters A B C D E F G.
- De string wordt afgesloten met 24 (=ASCII teken '\$') (Zie definitie interrupt 21 subfunctie 09)

Ander mogelijkheid voor (dezelfde) ingaven:

```
E 150 "ABCDEFGG$" → via " kan je een reeks van ASCII tekens invoeren.
```

```
E 150 "ABCDEFGG" 24 → een combinatie is mogelijk
```

```
E 150 'A' 'B' 'C' 'D' 'E' 'F' 'G' '$' → via " kan je één ASCII teken invoeren.
```

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 09

MOV AH, 09 → *09 in het AH register zorgt voor subfunctie 09*
(INT 21 sub09= printen van een reeks karakters)

MOV DX, 0150 → *De string start op adres 0150.*
(Einde van de string wordt aangegeven door '\$' = ASCII 24)

INT 21 → *INT 21*
 → *sufubunctie 09 (omdat AH = 09)*
 - De karakters vanaf adreslocatie 150 worden
 getoond. (Omdat regsiter DL = 150)
 - Laatste karakter is G (ASCII 47) omdat op de
 volgende geheugenlocatie (adres 157) een \$-
 teken staat.

INT 20 → *programma afsluiten*

Vullen van de geheugenlocatie:

E 150 41 42 43 44 45 46 47 24

Ander mogelijkheden:

E 150 "ABCDEFGG\$"

E 150 "ABCDEFGG" 24

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 09

```
-a 100
1469:0100 MOV AH, 09
1469:0102 MOV DX, 0150
1469:0105 INT 21
1469:0107 INT 20
1469:0109
-e 0150 41 42 43 44 45 46 47 24
```

Controle van de geheugen locaties.

```
-d 150
1469:0150 41 42 43 44 45 46 47 24-74 B1 BE 32 01 8D 8B 1E ABCDEFG$t..2....
1469:0160 8E FC 12 A8 33 D2 29 E3-13 8B C2 03 C3 69 02 00 ....3.).....i..
1469:0170 0E F8 83 FF FF 74 11 26-01 1D E2 F3 81 00 94 FA ..t.&.....
1469:0180 00 F0 74 16 81 C2 00 10-EB DC 5B 40 42 26 1E 83 ..[EB&..
1469:0190 EF 10 19 90 EB 45 A1 E2-97 8B 3E FB 8B 36 0A 80 .E....>..6..
1469:01A0 9A F F0 01 06 02 00 2D-B1 0A 00 BB BB F5 FA 8E .-.....
1469:01B0 D6 F E7 FB 8B C5 2E FF-28 50 2F B4 40 BB 1A B9 ....(P/.@...
1469:01C0 16 F 8C CA 00 05 A8 BA-1C 01 CD 21 B8 FF 4C 05 .....!..L.
```

Waarden in HEX

Waarden volgens ASCII

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 09

Test van het programma m.b.v. 'r' & 't' & 'p' = "stap voor stap"

```
-r
AX=0241 BX=0000 CX=0000 DX=0041 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1469 ES=1469 SS=1469 CS=1469 IP=0100  NV UP EI PL NZ NA PO NC
1469:0100 B409          MOV     AH,09
-t
AX=0941 BX=0000 CX=0000 DX=0041 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1469 ES=1469 SS=1469 CS=1469 IP=0102  NV UP EI PL NZ NA PO NC
1469:0102 BA5001       MOV     DX,0150
-t
AX=0941 BX=0000 CX=0000 DX=0150 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1469 ES=1469 SS=1469 CS=1469 IP=0105  NV UP EI PL NZ NA PO NC
1469:0105 CD21        INT     21
```

r = opvragen van Register waarde

t = Trace (instructie wordt uitgevoerd en (nieuwe) register waardes worden getoond.

p = Proceed (Interrupt wordt uitgevoerd)

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 09

INT 21 + AH = 09 → Karakters op geheugen locatie [DX] worden afgeprint.

DX geeft de startlocatie weer en '\$-teken' het einde van de te printen karakters

```
AX=0941 BX=0000 CX=0000 DX=0150 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1469 ES=1469 SS=1469 CS=1469 IP=0105 NU UP EI PL NZ NA PO NC
1469:0105 CD21 INT 21
-p
ABCDEFGH
AX=0924 BX=0000 CX=0000 DX=0150 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1469 ES=1469 SS=1469 CS=1469 IP=0107 NU UP EI PL NZ NA PO NC
1469:0107 CD20 INT 20
-p
Program terminated normally
```

Karakters op geheugen locatie 150 worden geprint. (einde = '\$')

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 09

Test van het programma m.b.v. 'g' = GO

```
-g  
ABCDEFGG  
Program terminated normally  
-
```

Opgelet!

Zorg dat de instructiepointer verwijst naar de locatie waar de programma code start!!

Assembler, de eerste stappen ...

DEBUG, geheugenbenadering

ENTER COMMANDO "e"

- E 100* → aanspreking van geheugenlocatie 100.
- E 100 FF* → geheugenlocatie 100 wordt overschreven met de waarde FF
- E 100 "string"* → geheugenlocatie 100 = ASCII waarde 's', 101 = 't' , ...
- E 100 'A'* → geheugenlocatie 100 = ASCII waarde van A (=41)

FILL COMMANDO "f"

- F 100 500 00* → geheugenlocatie 100 tot 500 = 00
- F 100 L20 AA* → vanaf 100, met een lengte van '20' = AA

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 01

INT 21 SUB 01:

INT21 subfunctie 01 zal een karakter inlezen.

Als INT21 subfunctie 01 wordt opgeroepen zal het programma wachten op een invoer van het toetsenbord.

Eenmaal het karakter ingelezen zal de ASCII waarde van het ingelezen karakter in het DL register staan.

Opmerking:

Dit is enkel een voorbeeld programma. Het programma Zal enkel karakter inlezen. Verder niets!

In de uitwerking zal je niets merken, maar via trace en proceed kan je nagaan wat INT21 subfunctie 01 doet.

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 01

MOV AH, 01

→ Register waarde van AH = 01. Dit duidt op het gebruik van subfunctie 01 = inlezen karakter

INT 21

→ INT 21 wordt opgeroepen, de actuele waarde in het AH register is 01 dus INT21 met subfunctie 01. Het programma wacht op een invoer van het toetsenbord. De invoer wordt opgeslaan in het DL register.

INT 20

```
C:\>DEBUG
-a
1469:0100 MOV AH, 01
1469:0102 INT 21
1469:0104 INT 20
1469:0106
_
```

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 01

Test van het programma m.b.v. 'r' & 't' & 'p' = "stap voor stap"

```
-r
AX=0100  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1469  ES=1469  SS=1469  CS=1469  IP=0100  NU UP EI PL NZ NA PO NC
1469:0100 B401          MOV     AH,01

-t
AX=0100  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1469  ES=1469  SS=1469  CS=1469  IP=0102  NU UP EI PL NZ NA PO NC
1469:0102 CD21          INT     21
```

r = Opvragen van Register waarde

t = Trace (instructie wordt uitgevoerd en (nieuwe) register waardes worden getoond.

p = Proceed (Interrupt wordt uitgevoerd)

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 01

INT 21 + AH = 02 → Karakter in het AL register wordt afgedrukt.

```
AX=0100 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1469 ES=1469 SS=1469 CS=1469 IP=0102  NV UP EI PL NZ NA PO NC
1469:0102 CD21      INT     21
-p
L
AX=014C BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1469 ES=1469 SS=1469 CS=1469 IP=0104  NV UP EI PL NZ NA PO NC
1469:0104 CD20      INT     20
-p
Program terminated normally
```

Ingaven van het karakter L = ASCII waarde 4C

De ingegeven waarde wordt opgevangen in het AL register!

r = opvragen van Register waarde

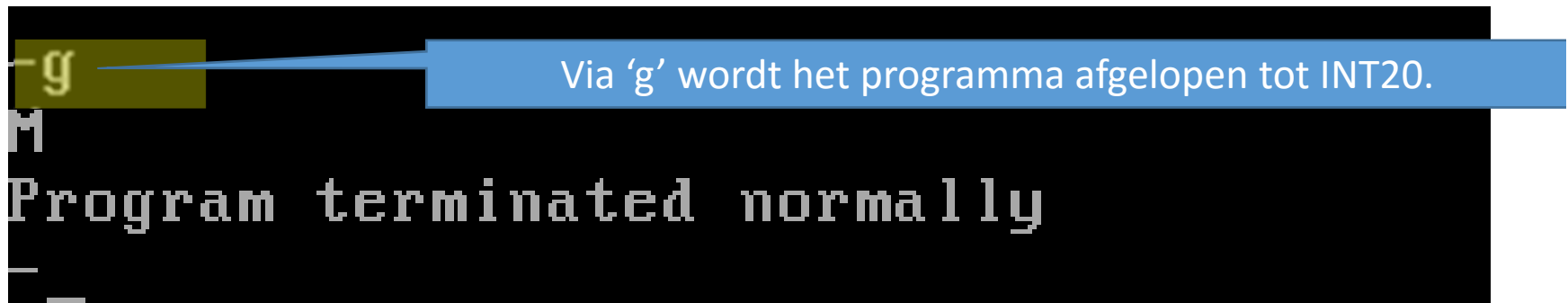
t = Trace (instructie wordt uitgevoerd en (nieuwe) register waardes worden getoond.

p = Proceed (Interrupt wordt uitgevoerd)

Assembler, de eerste stappen ...

Een eerste programma! INT21 sub 01

Test van het programma m.b.v. 'g' = GO



```
M  
Program terminated normally  
_
```

Opgelet!

Zorg dat de instructiepointer verwijst naar de locatie waar de programma code start!! (meestal 0100)

Assembler, de eerste stappen ...

De Loop instructie

De LOOP instructie

- *Afhankelijk van de waarde in CX, 'springt' de IP naar een opgegeven waarde.*
- *Het springen gebeurt door de IP van waarde te veranderen.*

Voorbeeld LOOP 110

- *De loop instructie verminderd de CX waarde met 1.*
- *De loop springt (IP wordt 110) naar het opgegeven adres zolang $cx > 0$ (=sprong)*
- *Indien $CX = null \rightarrow IP$ naar volgende geheugenlocatie (=geen sprong)*

Opmerking: CX staat voor het count register!

Assembler, de eerste stappen ...

De Loop instructie, voorbeeld

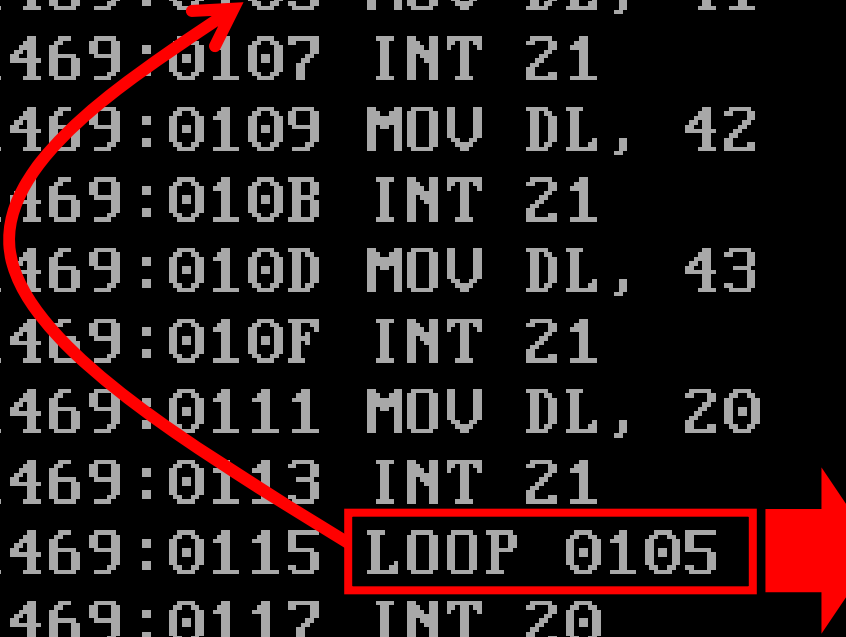
```
-a 100
1469:0100 MOV CX, 0005
1469:0103 MOV AH, 02
1469:0105 MOV DL, 41
1469:0107 INT 21
1469:0109 MOV DL, 42
1469:010B INT 21
1469:010D MOV DL, 43
1469:010F INT 21
1469:0111 MOV DL, 20
1469:0113 INT 21
1469:0115 LOOP 0105
1469:0117 INT 20
1469:0119
```

- Teller waarde = 5
- Subfunctie 02
- DL = 41 (is ASCII karakter A)
- INT 21 (SUB02) = printen van één karakter. Het karakter staat in DL-register, in dit geval de ASCII waarde van 41 = A.
- Herhaling van INT21 met telkens een andere waarde in het DL register...
- ...

Assembler, de eerste stappen ...

De Loop instructie

```
-a 100
1469:0100 MOV CX, 0005
1469:0103 MOV AH, 02
1469:0105 MOV DL, 41
1469:0107 INT 21
1469:0109 MOV DL, 42
1469:010B INT 21
1469:010D MOV DL, 43
1469:010F INT 21
1469:0111 MOV DL, 20
1469:0113 INT 21
1469:0115 LOOP 0105
1469:0117 INT 20
1469:0119
```



CX = 0005 = “teller waarde” = 5

LOOP 0105:

Verminder teller waarde (-1)

→ ALS CX > 0 dan LOOP
(= IP terug naar **0105**)

→ ALS CX == 0 dan einde LOOP
(=IP naar volgende waarde)

```
-g
ABC ABC ABC ABC ABC
Program terminated normally
```