Try-out 1 A - OPGAVE

Try-out 1: INT 21

Maak twee verschillende programma's in assembler.
Doel van het programma is je voornaam 10 maal achter elkaar printen gescheiden door spaties. Beide programma's moeten hetzelfde resultaat geven.
Met verschillende programma's wordt bedoeld dat er gebruik wordt gemaakt van verschillende DOS-subfuncties.
→ ***Voor TRY1a.COM maak je gebruik van subfunctie 02 van INT21 (printen van een teken).***
→ *Voor TRY1b.COM maak je gebruik van subfunctie 09 van INT21 (printen van een string).*

```
C:\>try1a
NAAM NAAM NAAM NAAM NAAM NAAM NAAM NAAM NAAM NAAM
C:\>
```

Try-out 1 A

Oplossing Try-out 1 A

```
-u 100 119
1481:0100 B90A00        MOV     CX,000A
1481:0103 B402          MOV     AH,02
1481:0105 B24E          MOV     DL,4E
1481:0107 CD21          INT     21
1481:0109 B241          MOV     DL,41
1481:010B CD21          INT     21
1481:010D CD21          INT     21
1481:010F B24D          MOV     DL,4D
1481:0111 CD21          INT     21
1481:0113 B220          MOV     DL,20
1481:0115 CD21          INT     21
1481:0117 E2EC          LOOP    0105
1481:0119 CD20          INT     20
-g
NAAM NAAM NAAM NAAM NAAM NAAM NAAM NAAM NAAM NAAM
Program terminated normally
```

Try-out 1 A – Stap voor stap

CODE	VERKLARING	PRAKTISCH
MOV CX, 000A	Vul het CX register met 000A (Verplaats de waarde 000A naar het CX register)	Teller waarde = 10 = A (HEX)
MOV AH, 02	Vul het AH register met 02	We willen subfunctie 02 gebruiken → Ah = 02. Sub functie 02 = print 1 karakter
MOV DL, 4E	Vul het DL register met 4E	Plaats de letter 'N' in het DL reg (→ zie ASCII tabel N = 4E)
INT 21	INTERRUPT 21	Interrupt 21 → sub functie 02 (Ah = 02) →Print de inhoud van DL = 4E → 4E = ASCII → N
MOV DL,41	Vul het DL register met 41	Plaats de ASCII waarde van 'A' in het DL reg
INT 21	INTERRUPT 21	… Print de inhoud van DL = A (AH = 02, DL = 41)
INT 21	INTERRUPT 21	Print de inhoud van DL (AH & DL blijven idem) dus Print 'A'
MOV DL,4D	Vul het DL register met 4D	Plaats de ASCII waarde voor 'M' in het DL reg
INT 21	INTERRUPT 21	Print de inhoud van DL = M
MOV DL,20	Vul het DL register met 20	Plaats de ASCII waarde voor ' ' in het DL reg
INT 21	INTERRUPT 21	Print de inhoud van DL = ' '
LOOP 0105	LOOP NAAR IP-Waarde 0105 Is (CX -1) = 0 ? → JA, verder naar volgende instructie → NEE, IP (terug) naar 105	Loop naar IP 105 (aantal loops = waarde in het CX register)
INT 20	INTERRUPT 20	Einde programma

Try-out 1 A - extra

Opslaan van het programma;

De waarde in het CX register bepaalt het aantal bytes dat wordt opgeslaan.

Uitvoeren van het program;

Q → QUIT (verlaat de DEBUG mode)
C:\> TRYOUT1A → Het programma TRYOUT1A wordt uitgevoerd

(her)Openen van het program in DEBUG;

DEBUG TRYOUT1A.COM
// DEBUG wordt geladen met het programma TRYOUT1A in het geheugen
U 100 → De programma-instructies worden getoond, startend vanaf IP 100

Try-out 1 A

VARIATIES 1A

- Schrijf je naam + achternaam: (gebruik een “*” als scheiding)
- Schrijf een andere tekst.
- Schrijf een visite kaartje.
  - Alles op 1 regel, gebruik een toepasselijk scheidingsteken.
- Maak gebruik van Line feed (ASCII 0A) en carriage return (ASCII 0D)

OPMERKINGEN:

- Maak de loop zo correct mogelijk. (Geen onnodige bewerkingen in de loop)
- Het opzoeken van de ASCII waarde van een karakter kan d.m.v het e-commando en het d-commando
  Voorbeeld:
  - e 200 “NAAM”
  - D 200

In DEBUG zie je rechts de karakters volgens ascii, links de HEX-waarde.

```
-e 200 "NAAM"
-d 200
1469:0200  4E 41 41 4D 4D F8 4D 03-4E 0E 4E 19 4E 24 4E 2F   NAAMM.M.N.N.N$N/
1469:0210  00 88 4F 30 4E 45 4E 50-4E 5B 4E 66 4E 71 4E 7C   ..N:NENPN[NfNqN|
1469:0220  4E 87 00 00 4E 92 4E 9D-4E A8 4E B3 4E BE 4E C9   N...N.N.N.N.N.N.
1469:0230  4E D4 4E DF 00 00 4E EA-4E F5 4E 00 4F 0B 4F 16   N.N...N.N.N.O.O.
1469:0240  4F 21 4F 2C 4F 37 7A 03-4F 00 13 01 FF 00 A7 05   O!O,O7z.O.......
1469:0250  80 00 CE 04 10 00 00 00-00 00 00 00 00 00 00 00   ................
1469:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
1469:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

Try-out 1 A

Wat kan er fout gaan?

- Oneindige loop;
  - → De loop verwijst naar de regel MOV CX, 000A
    - → in het voorbeeld → 0117: LOOP 0100
    - → Het CX register wordt in de loop terug overschreven (MOV CX, 000A in de loop) waardoor de teller nooit op nul kan komen!
- Zeer lange loop;
  - → Count-register is nul (default waarde is nul) !
    - → In het voorbeeld: MOV CX, 000A niet in het programma. Of mov CX, 0000.
    - → Loop wordt 65535 (FFFF) keer uitgevoerd!
    - → Na de eerste loop instructie wordt de inhoud van het CX register FFFF (0000-1). Aangezien dit niet gelijk is aan nul, wordt er verder herhaald. Dit tot de waarde van CX gelijk is aan nul!
    - → Test dit met de commando's r (= registers), t (= trace, uitvoeren van een instructie) en p (=proceed, een interrupt uitvoeren).
- 16x herhaling i.p.v 10x
    - → In het voorbeeld → 0100: MOV CX, 0010 (i.p.v 000A)
    - → De waarde in de registers zijn ALTIJD hexadecimaal. 0010 komt dus overeen met 16! 000A met 10!
- Programma loopt vast
  - → INT 20 niet op het einde van het programma.
    - → Na de laatste (geprogrammeerde) instructie wordt de IP verhoogd. De IP verwijst naar een volgende geheugen locatie. Deze betreffende geheugenlocatie is niet door jou geprogrammeerd, en bevat dus random data. Als deze data niet kan geïnterpreteerd worden door het instructieregister, loopt het programma vast!
  - → Loop naar een niet volledige instructie.
    - → De meeste instructies zijn 2 byte lang. (Vandaar dat de IP gaat van 103-105-107-…)
      In de loop kan je springen naar een onvolledige instructie. Hierdoor kan een programma vastlopen. (In het voorbeeld LOOP 106. Op IP 106 staat een halve instructie…)

Try-out 1: INT 21

Maak twee verschillende programma's in assembler.
Doel van het programma is je voornaam 10 maal achter elkaar printen gescheiden door spaties. Beide programma's moeten hetzelfde resultaat geven.
Met verschillende programma's wordt bedoeld dat er gebruik wordt gemaakt van verschillende DOS-subfuncties.
→ *Voor TRY1a.COM maak je gebruik van subfunctie 02 van INT21 (printen van een teken).*
→ ***Voor TRY1b.COM maak je gebruik van subfunctie 09 van INT21 (printen van een string).***

```
C:\>try1a
NAAM NAAM NAAM NAAM NAAM NAAM NAAM NAAM NAAM NAAM
C:\>

C:\>try1b
NAAM NAAM NAAM NAAM NAAM NAAM NAAM NAAM NAAM NAAM
```

Oplossing, Try-out 1 B

```
-u 100 010C
1481:0100 B90A00        MOV     CX,000A
1481:0103 B409          MOV     AH,09
1481:0105 BA2001        MOV     DX,0120
1481:0108 CD21          INT     21
1481:010A E2FC          LOOP    0108
1481:010C CD20          INT     20
-e 120 "Naam $"
-g
Naam Naam Naam Naam Naam Naam Naam Naam Naam Naam
Program terminated normally
```

Opmerking → verschillende methode om de string in het geheugen te plaatsen:

- **E 120 “NAAM $”** // string tussen “ “
- **E 120 ‘N’ ‘A’ ‘A’ ‘M’ ‘ ’ ‘$’** // één karakter tussen ‘ ’ (Mag ook “N” “A” “A” “M” “ “ “$”)
- **E 120 4E 41 41 4D 20 24** // ASCII waarden (in HEX)
- **E 120 “NAAM” 20 24** // string in combinatie met ASCII waarde (20 = spatie + 24 = $)

Try-out 1 B – Stap voor stap

CODE	VERKLARING	PRAKTISCH
MOV CX, 000A	Vul het CX register met 000A (Verplaats de waarde 000A naar het CX register)	Teller waarde = 10 = A (HEX)
MOV AH, 09	Vul het AH register met 09	We willen sub functie 09 gebruiken → Ah = 09. Sub functie 02 = print een string Locatie van de string → adres waard in DX
MOV DX, 120	Vul het DX register met 120	120 is de adres waarde waar de string begint.
INT 21	INTERRUPT 21	Interrupt 21 → sub functie 09 (Ah = 09) →Print een string, adres in DL = 120 → Start met 120, tot de inhoud '$' (= einde)
LOOP 108	LOOP NAAR IP-Waarde 0108 Is (CX -1) = 0 ? → JA, verder naar volgende instructie → NEE, IP (terug) naar 108	Loop (totaal aantal loop = waarde CX)
INT 20	INTERRUPT 20	Einde programma

Try-out 1 B - extra

Opslaan van het programma;

```
R CX
26
N TRYOUT1B.COM
W
```

Opmerking: voor het programma zo klein mogelijk te maken, zou de string geplaatst kunnen worden, direct na “INT 20” (IP 010E). Zo wordt het totaal programma kleiner.

Uitvoeren van het program

Q → QUIT (verlaat de DEBUG mode)
TRYOUT1B → Het opgeslagen programma wordt uitgevoerd

(her)Openen van het program in DEBUG

DEBUG TRYOUT1B.COM
// DEBUG wordt geladen met het programma TRYOUT1B in het geheugen
U 100 → De programma-instructies worden getoond, startend vanaf IP 100

Try-out 1 B

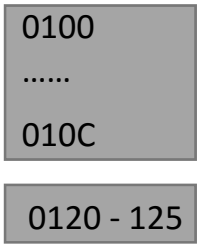
VARIATIES 1B

- Schrijf je naam + achternaam: (gebruik een “*” als scheiding)
- Schrijf een andere tekst.
- Schrijf een visite kaartje.
  - Alles op 1 regel, gebruik een toepasselijk scheidingsteken.
- Gebruik verschillende methode van het “e” (enter) commando.
- Maak gebruik van Line feed (ASCII 0A) en carriage return (ASCII 0D)

Try-out 1 B

Wat kan er fout gaan?

- Er komen allerlei vreemde tekens op het scherm.
  - De uit te printen karakters staan niet op het adres (adres in DX register), of de string is niet beëindigd met een ‘ $ ‘.
    → INT21 & subfunctie 09 zal een reekst van karakters op het scherm tonen, startend vanaf de adres locatie in DX. (DX=pointer). Na het eerste karakter wordt het karakter van de volgende geheugenlocatie getoond, dit tot een $-teken (= ASCII 24) de string afsluit. Als er geen $-teken staat, blijft de uitwerking van interrupt 21 de karkaters printen. Dit tot er ergens (toevallig) een $-teken staat.
    → Als je zelf geen karakters ingeeft op de geheugenlocatie (‘e’ commando) wordt er ‘random’ data afgeprint. (Meestal een reeks van ‘vreemde’ tekens op het scherm.)
- De string wordt niet correct getoond, na het opslaan van het programma.
  → Bij het opslaan geef je via het CX register aan hoeveel byte er wordt gesaved.
  → Stel nu dat de instructies (=startend van adres 0100) worden gesaved, maar de data van de te printen string (=adres in DX) niet …. (in het voorbeeld; CX = 0C i.p.v. 26 voor het saven.)
  → Bij het uitvoeren van het (gesaved) programma zullen de instructies worden uitgevoerd. Maar op de locatie waar het DX register naar verwijst, staat nu ‘random’ data. Dit resulteert in het printen van “gekke tekens”.
  → Oplossing, de te printen data moet mee worden gesaved bij het programma.



Instructies

Geheugenlocatie string

Door enkel de code van de instructies op te slaan. Gaat de inhoud van de string verloren!
In dit voorbeeld is de minimale waarde van CX tijdens het opslaan 26.

Try-out 2: Regel vullen

Schrijf een gestructureerd programma in assembler (TRY2.COM), waarbij een volledige regel (80 karakters) wordt opgevuld met het karakter met ASCII-waarde 2 (het teken ☻).

Pas dit programma aan zodat een volledig scherm opgevuld wordt (25 lijnen)

```
C:\>TRY2
☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻
C:\>
```

Try-out 2 - Oplossing

```
1481:0100 B95000          MOV     CX,0050
1481:0103 B402            MOV     AH,02
1481:0105 B202            MOV     DL,02
1481:0107 CD21            INT     21
1481:0109 E2FC            LOOP    0107
1481:010B CD20            INT     20
```

```
A
MOV     CX,0050     → zet de teller op 80 (= 50hex)
MOV     AH,02       → zet de subfunctie op 02
MOV     DL,02       → plaats ascii teken 02 in DL
INT     21          → print het teken
LOOP    0107        →  Loop naar adres 107  (Aantal keer loop = waarde in CX)
INT     20          → einde
```

Try-out 2 - Oplossing

Try-out 2 – variant, oplossing met int21, subfunctie 09

```
a                                   ; Try-out 2 – één lijn ☺, gebruik van subfunctie 09
MOV AH, 09                          ; Subfunctie 09
MOV DX, 110                         ; adres string is 110, vul register met 110
INT 21                              ; print de string
INT 20

F 110 L50 2                         ; Fill vanaf adres 110 met een lengte van L50(h), met de waarde 02
E 160 '$'                           ; Enter adres 160 met het teken '$'

r cx
61
n try02.com
w
```

Try-out 2 - Oplossing

Try-out 2 – volledig scherm, gebruik makend van een dubbele loop en hulpregister BX

```
a
MOV AH, 02      → Subfunctie 02 = één teken printen
MOV BX, 0019    → (Hulp)register BX vullen met 25
MOV CX, 0050    → Countregister CX vullen met 80
MOV DL, 02      → ☺ teken in DL register
INT 21          → printen van 1x ☺
LOOP 010A       → LOOP  ( printen van 1x ☺ )
MOV DL, 0A
INT 21          → printen van 1x line feed (0A)
MOV DL, 0D
INT 21          → printen van 1x Carrige return (0D)
DEC BX          → Hulpregister BX met 1 verminderen (DEC = decrement)
MOV CX, BX      → Verplaatsen van BX naar CX (CX wordt gebruikt voor de loop)
LOOP 105        → Loop naar “MOV CX, 0050”
INT 20          → Einde programma
```

Try-out 2

VARIATIES:

- Maak gebruik van een ander symbool.
- Maak de oefening, eenmaal met subfunctie 02 en eenmaal met subfunctie 09.
- Maak een lijn met afwisselend ASCII waarde 01 en ASCII waarde 02.
- Maak een ander patroon (minder of meer lijnen/kolommen)

Opmerkingen:

- 80 Regels = 50 hexadecimaal !!
- Als je gebruik maakt van subfunctie 09, kan je best het ‘F’ commando gebruiken voor de string in het geheugen te plaatsen. Vergeet ‘$’ niet!
- Save het programma, test het in c:\>

Try-out 3: Omzetting van “case”

Het programma TRY3.COM laat toe om een letter in te geven (subfunctie 01 van INT 21). Er mag verondersteld worden dat er een hoofdletter wordt ingegeven (controleren is hier dus nog niet nodig).

Deze hoofdletter wordt vervolgens omgezet naar een kleine letter en op de volgende positie op het scherm geplaatst.

```
C:\>TRY3
Gg
```

Oplossing Try-out 3

```
A
MOV     AH,01     → subfunctie 01 om in te lezen
INT     21        → ingelezen karakter zit nu in AL
MOV     AH,02     → subfunctie 02 om karakter op scherm te tonen
ADD     AL,20     → 32 (HEX 20) erbij tellen om van hoofdletter naar kleine letter te gaan. (zie ASCII tabel …)
MOV     DL,AL     → karakter van AL naar DL kopiëren
INT     21        → Karakter in DL afprinten (= kleine letter op scherm tonen)
INT     20        → Einde
```

Try-out 3:

VARIATIES:

- Verander een kleine letter, naar een hoofdletter.
- Start met een getal (0-8), tel één erbij en toon op het scherm.
- Schoon het programma op door de regel te printen “geef een hoofdletter… “ & “De kleine letter is…”

```
C:\>
Geef een hoofdletter >> F
De kleine letter is >> f
```

Try-out 4A: Letter ingeven en afprinten

*Maak een assemblerprogramma in debug waardoor het mogelijk wordt een letter in te geven. Deze letter dient dan tien keer op het scherm te worden gedrukt, zoals hieronder wordt aangegeven (**NIET** gescheiden door een spatie).*

```
C:\>TRY4a
ssssssssss
C:\>
```

```
C:\>TRY4b
s
 s
  s
   s
    s
     s
      s
       s
        s
         s
          s
C:\>
```

Try-out 4A: Oplossing

```
a
MOV     CX,0009     → zet de teller op 9
MOV     AH,01       → subfunctie 01
INT     21          → int 21 oproepen om teken in te lezen
MOV     AH,02       → subfunctie 02
MOV     DL,AL       → plaats het ingelezen teken in DL
INT     21          → int 21 oproepen om teken te printen
LOOP    10B         → spring terug naar adres 10B (printen van teken)
INT 20              → einde
```

Try-out 4B: Letter ingeven en afprinten

Pas dit programma aan door de tekens te scheiden door middel van een linefeed of een spatie. Je zal merken dat dit programma niet de verwachte output geeft. Probeer de fout te vinden door te debuggen (stap voor stap de werking van elke instructie in je programma doorlopen en controleren op zijn juiste werking d.m.v. trace en proceed). Als je de fout gevonden heb, tracht ze dan op te lossen.

```
C:\>TRY4b
S
 S
  S
   S
    S
     S
      S
       S
        S
         S
          S
           S
C:\>
```

Try-out 4B: Oplossing

```
a
MOV     CX,0009     → zet de teller op 9
MOV     AH,01
INT     21
MOV     AH,02
MOV     BL,AL       → plaats het ingelezen teken veilig in BL
MOV     DL,0A       → plaats de linefeed in DL
INT     21          → int 21 oproepen om teken te printen
MOV     DL,BL       → plaats het ingelezen teken vanuit BL in DL
INT     21          → int 21 oproepen om teken te printen
LOOP    10B         → spring terug naar adres 10B (linefeed in DL)
INT     20          → einde
```

Opmerking:
Wanneer INT21 een karakter afprint, wordt dit karakter in het AL register geplaatst alvorens het af te printen. In deze oefening is het belangrijk om het ingelezen karakter (in AL) veilig weg te plaatsen in een ander register. Anders wordt dit karakter overschreven door het printen van de linefeed. (0A in AL…)

Try-out 4 A & B

Variaties:

- Maak dezelfde oefening, maar toon eerste een “>” (ASCII 10) alvorens de andere tekens te tonen. (Let op, plaats de waarde in het AL-REG veilig in het BL register alvorens INT 21 uit te voeren)
- Maak dezelfde oefening, maar scheidt de verschillende karakters met een ‘-‘.
- Maak dezelfde oefening, maar druk al de letters af, op een volgende regel.
- Maak dezelfde oefening, maar druk de letter af, op een volgende regel, in groepjes van 3.
- Maak dezelfde oefening, start met een hoofdletter, geef de volgende 10 letters in een kleine letter
- Geef een hoofdletter, toon op de volgende regel, de volgende letter 10x. Toon op de daarop volgende regel volgende letter 10x. Toon hierbij de letters als volgt; 5xhoofdletters ‘-’ 5x kleine letters.
- Schoon de oefening op door gebruik te maken van tekst. Bijvoorbeeld:

```
Geef een hoofdletter: S
->>>>    s * s * s * s * s * s * s * s * s * s *
```

Try-out 5: Alfabet in hoofd- en kleine letters

Maak een assemblerprogramma in debug dat het volledig alfabet toont, afwisselend met hoofdletters en kleine letters.

```
C:\>Try5
AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
```

Try-out 5: Oplossing

```
A
MOV     CX,001A     → plaats 26 in het countregister (=1A HEX)
MOV     AH,02       → zet subfunctie op 02
MOV     DL,41       → plaats hoofdletter A in DL
INT     21          → print de hoofdletter
ADD     DL,20       → tel 20 Hex bij om naar kleine letter te gaan
INT     21          → print de kleine letter
SUB     DL,1F       → trek terug 1F (Hex) af van DL om de volgende hoofdletter te krijgen
LOOP    0107        → spring naar adres 107
INT     20          → einde
```

Opmerking:

$20_{(HEX)} - 1 = 1F_{(HEX)}$ → $32_{(10)} - 1 = 31_{(10)}$

Try-out 5: Oplossing - variant

```
a
MOV CX, 001A      → 26 in teller
MOV AH, 02        → subfunctie 02, enkel karakter printen
MOV DL, 41        → Start met 'A'
INT 21            → Print hoofdletter
ADD DL, 20        → +20 = kleine letter
INT 21            → printen kleine letter
SUB DL, 20        → Tel 20 af = terug hoofdletter
INC DL            → 1 erbij = volgende hoofdletter
LOOP 107          → herhaal (26 keer)
INT 20
```

Try-out 5: Oplossing – variant, met hulpregister

```
A
MOV     CX,001A     → plaats 26 in het countregister
MOV     AH,02       → zet subfunctie op 02
MOV     BL,41       → plaats hoofdletter A in BL
MOV     DL,BL       → kopieer BL naar DL
INT     21          → print de hoofdletter
ADD     DL,20       → tel 20H bij om naar kleine letter te gaan
INT     21          → print de kleine letter
ADD     BL,01       → Verhoog het hulpregister met 1 om de volgende hoofdletter te krijgen
LOOP    0107        → spring naar adres 107
INT     20          → einde
```

Try-out 5

VARIANTEN:

- zZyYxXwWvVuUtTsSrRqQpPoOnNmMlLkKjJiIhHgGfFeEdDcCbBaA
- 0123456789
- 0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9
- AB*CD*EF*GH*IJ*JK*MN*OP*QR*ST*UV*WX*YZ
- …

Opmerkingen:

Het verschil tussen een hoofdletter en een kleine letter is 20(HEX). één minder is dan 20 HEX is 1F, niet 19!

Try-out 6: Piece of cake

Schrijf een programma waarmee vanuit het geheugen de string "Piece of cake" tien keer op het scherm wordt geplaatst.

```
C:\> Try6
Piece of cake
Piece of cake
Piece of cake
Piece of cake
Piece of cake
Piece of cake
Piece of cake
Piece of cake
Piece of cake
Piece of cake
```

Try-out 6: Oplossing

```
A
MOV     CX,000A     → zet de teller op 10
MOV     AH,09       → zet de subfunctie op 09
MOV     DX,010F     → plaats de stringpointer op adres 10F
INT     21          → print de string
LOOP    0108        → spring terug naar adres 108
INT     20          → einde


e 10f "Piece of cake" 0d 0a 24
```

Try-out 6

Alternatieven

Maak de oefening, maar houdt telkens een lege regel tussen de verschillende strings.

Maak de oefening, maar houdt tussen de regels 1 maal een regel met “-”, de volgende maal een regel met “*”.

```
Piece of cake
-----------------
Piece of cake
****************
... ...
```

Vraag eerst via INT21(SUB01) een getal tussen 1 en 5. Hierna toon je string “piece of cake’, net zoveel als de waarde in AL (=ingegeven waarde). Opmerking ASCII waarde van getallen is 30 tot 39! (De waarde in het cx register wordt dus AL-30!)