2018 oefeningen hoofdstuk 3 rijen oplossingen

```
RIJEN



Hoger of lager of gemiddeld?
----------------------------

Schrijf een programma dat 6 getallen inleest en de volgende getallen berekent en toont:

    Het gemiddelde
    Hoeveel getallen groter zijn dan het gemiddelde
    Hoeveel getallen kleiner zijn dan het gemiddelde



%include "gt.asm"
covar
gemiddelde: resd 1
groterdangemiddelde: resd 1
kleinerdangemiddelde: resd 1
getal: resd 6
een: dd 1
zes: dd 6
inleiding
mov eax, 0
mov ecx, [zes]
mov edi, 0

hoger:
cmp ecx, 0
jle verder

inv [getal + edi]
add eax, [getal + edi]
add edi, 4
sub ecx, 1
```

```
jmp hoger

verder:
imul dword [een]
idiv dword [zes]
mov [gemiddelde], eax

mov edi, 0
mov ebx, 0
mov edx, 0
mov ecx, [zes]

lus:
cmp ecx, 0
jle einde

mov eax, [getal + edi]
add edi, 4
sub ecx, 1

cmp eax, [gemiddelde]
jl kleinerdan
jg groterdan
je lus

kleinerdan:
add ebx, 1
jmp lus

groterdan:
add edx, 1
jmp lus

einde:
mov [kleinerdangemiddelde], ebx
mov [groterdangemiddelde], edx
uit [gemiddelde]
uit [groterdangemiddelde]
uit [kleinerdangemiddelde]

slot
```

Sorteren

Schrijf een programma dat 7 getallen inleest en deze gesorteerd van klein naar groot afdrukt.
Om te sorteren kan men als volgt te werk gaan:

```
doe 6 maal:
  { i = 0;
    doe 6 maal
      { vergelijk element(i) met element(i+1);
        if (element(i) > element(i+1) {
          verwissel;
        }
        i = i + 1;
      }
  }
```

```
%include "gt.asm"
covar
getal: resd 7
inleiding
mov ecx, 7
mov edi, 0

invoer:
    jle ordenen
    inv [getal + edi]
    add edi, 4
    loop invoer

ordenen:
    mov eax, 6

grotelus:
    cmp eax, 0
```

```
    jle uitvoer

lus:
    mov ecx, 6
    mov edi, 0

grootstnaarachteren:
    cmp ecx, 0
    jle volgendelus

    mov edx, [getal + edi]
    add edi, 4

    cmp edx, [getal + edi]
    jle grootstnaarachteren

mov ebx, [getal + edi]
mov [getal + edi], edx
sub edi, 4
mov [getal + edi], ebx
add edi, 4
sub ecx, 1
jmp grootstnaarachteren

volgendelus:
sub eax, 1
jmp grotelus



uitvoer:
    mov edi, 0
    mov ecx, 7

lustwee:
    cmp ecx, 0
    jle einde

    uit [getal + edi]
    add edi, 4
    sub ecx, 1
    jmp lustwee
```

```
einde:
slot
```

MOVSB/STOSB

1

Wat staat er in het geheugen als achtereenvolgens onderstaande instructies worden uitgevoerd?
GEHEUGENINHOUD (initieel, hexadecimaal)

Initieel is de inhoud van het geheugen:

```
a: 19 1B 1E ?? ?? 1C ?? 26 ??
```

PROGRAMMA

De volgende instructies worden uitgevoerd.

```
std
mov edi, a+6
mov eax, Dh
mov ecx, 2
rep stosb
```

GEHEUGENINHOUD (na uitvoering, hexadecimaal)

Vul aan:

```
a: 19 1B 1E ?? ?? 0D 0D 26 ??
```

2

Wat staat er in het geheugen als achtereenvolgens onderstaande instructies worden uitgevoerd?
GEHEUGENINHOUD (initieel, hexadecimaal)

Initieel is de inhoud van het geheugen:

a: ?? 23 ?? 20 ?? 1D 27 ?? ?? ?? 22
PROGRAMMA

De volgende instructies worden uitgevoerd.

```
std
mov edi, a+7
mov eax, 18h
mov ecx, 7
rep stosb
```

GEHEUGENINHOUD (na uitvoering, hexadecimaal)

Vul aan:

a: ?? 18 18 18 18 18 18 18 ?? ?? 22

3

Wat staat er in het geheugen als achtereenvolgens onderstaande instructies worden uitgevoerd?
GEHEUGENINHOUD (initieel, hexadecimaal)

Initieel is de inhoud van het geheugen:

a: 1C 21 ?? 1A 15 25 ?? 1D 1F 1B 1F
PROGRAMMA

De volgende instructies worden uitgevoerd.

```
std
```

```
mov edi, a+9
mov eax, 10h
mov ecx, 10
rep stosb
```

GEHEUGENINHOUD (na uitvoering, hexadecimaal)

Vul aan:

a: 10 10 10 10 10 10 10 10 10 10 1F

4

Wat staat er in het geheugen als achtereenvolgens onderstaande instructies worden uitgevoerd?
GEHEUGENINHOUD (initieel, hexadecimaal)

Initieel is de inhoud van het geheugen:

a: ?? 1D 27 ?? 1F ?? ?? ??
PROGRAMMA

De volgende instructies worden uitgevoerd.

```
cld
mov edi, a+4
mov eax, 1Dh
mov ecx, 4
rep stosb
```

GEHEUGENINHOUD (na uitvoering, hexadecimaal)

Vul aan:

a: ?? 1D 27 ?? 1D 1D 1D 1D

5

Wat staat er in het geheugen als achtereenvolgens onderstaande instructies worden uitgevoerd?
GEHEUGENINHOUD (initieel, hexadecimaal)

Initieel is de inhoud van het geheugen:

a: 25 ?? ?? 26 16 1F 26 26 ?? ?? ??
PROGRAMMA

De volgende instructies worden uitgevoerd.

```
std
mov edi, a+9
mov eax, Fh
mov ecx, 3
rep stosb
```

GEHEUGENINHOUD (na uitvoering, hexadecimaal)

Vul aan:

a: 25 ?? ?? 26 16 1F 26 0F 0F 0F ??

6

Wat staat er in het geheugen als achtereenvolgens onderstaande instructies worden uitgevoerd?
GEHEUGENINHOUD (initieel, hexadecimaal)

Initieel is de inhoud van het geheugen:

a: 26 20 22 21 23 25 20 ?? ?? 16 14

b: 18 1D 23 14 1A 23 1B 1B

PROGRAMMA

De volgende instructies worden uitgevoerd.

```
cld
mov edi, a+5
mov esi, b+2
mov ecx, 5
rep movsb
```

GEHEUGENINHOUD (na uitvoering, hexadecimaal)

Vul aan:

a: 26 20 22 21 23 23 14 1A 23 1B 14

b: 18 1D 23 14 1A 23 1B 1B

STRINGS

Karakterstrings

Schrijf een programma dat afdrukt:

```
WHO IS MY TAILOR?
MY TAILOR IS CHRISTIAN DIOR
MY TAILOR IS RICH
IS MY TAILOR RICH?
```

Definieer zo weinig mogelijk karakterstrings, bvb.:

```
'MY TAILOR?'
'IS RICH'
'WHO'
'CHRISTIAN DIOR'
```

```
%include "gt.asm"
covar
outarea: times 70 db (' ')
        db 0Dh, 0Ah
mytailor: db 'MY TAILOR?'
isrich: db 'IS RICH'
who: db 'WHO'
christiandior: db 'CHRISTIAN DIOR'
lchristiandior: EQU $-christiandior
inleiding
openuit
cld
mov ecx, 70
mov al, ' '
mov edi, outarea
rep stosb


mov ecx, 3
mov esi, who
mov edi, outarea
rep movsb

mov ecx, 2
mov esi, isrich
mov edi, outarea+4
rep movsb

mov ecx, 10
mov esi, mytailor
mov edi, outarea+7
rep movsb
schrijf


mov ecx, 17
mov al, ' '
mov edi, outarea
rep stosb
```

```
mov ecx, 9
mov esi, mytailor
mov edi, outarea
rep movsb

mov ecx, 2
mov esi, isrich
mov edi, outarea+10
rep movsb

mov ecx, lchristiandior
mov esi, christiandior
mov edi, outarea+13
rep movsb
schrijf


mov ecx, 17
mov al, ' '
mov edi, outarea+10
rep stosb

mov ecx, 7
mov esi, isrich
mov edi, outarea+10
rep movsb
schrijf


mov ecx, 24
mov al, ' '
mov edi, outarea
rep stosb

mov ecx, 2
mov esi, isrich
mov edi, outarea
rep movsb

mov ecx, 9
mov esi, mytailor
```

```
mov edi, outarea+3
rep movsb

mov ecx, 4
mov esi, isrich+3
mov edi, outarea+13
rep movsb

mov ecx, 1
mov esi, mytailor+9
mov edi, outarea+17
rep movsb
schrijf
slot
```

BESTANDEN

Letters zoeken...

Invoer: een bestand met 1 lijn tekst van 70 karakters, bvb.:

Het spaanse graan heeft de orkaan doorstaan...

Schrijf een programma dat telt hoeveel keer de letter 'a' voorkomt. Toon de uitvoer aan de gebruiker.

```
%include "gt.asm"
covar
inarea: resb 70
hulpd: resd 1
inleiding
openin
lees
cld
mov eax, 0
mov ebx, 0
```

```
mov ecx, 70
mov edi, 0
mov al, 'a'

lus:
cmp al, [inarea + edi]
jne verder
add ebx, 1

verder:
add edi, 1
loop lus

mov [hulpd], ebx
uit [hulpd]
slot
```

Klinkers zoeken

Herschrijf de vorige oefening om alle klinkers (a,e,i,o,u) te tellen.

```
%include "gt.asm"
covar
inarea: resb 70
hulpd: resd 1
inleiding
openin
lees
cld
mov eax, 0
mov ebx, 0
mov ecx, 70
mov edi, 0
```

```
lus:
mov al, 'a'
cmp al, [inarea + edi]
je gelijk

mov al, 'e'
cmp al, [inarea + edi]
je gelijk

mov al, 'i'
cmp al, [inarea + edi]
je gelijk

mov al, 'o'
cmp al, [inarea + edi]
je gelijk

mov al, 'u'
cmp al, [inarea + edi]
je gelijk
jmp verder

gelijk:
add ebx, 1

verder:
add edi, 1
loop lus

mov [hulpd], ebx
uit [hulpd]
slot
```

Woorden zoeken

Schrijf een programma dat een woord uitleest uit het invoerbestand. Het woord staat op de

eerste lijn (die verder uit allemaal spaties bestaat). Lees dan het bestand verder uit, en tel hoe vaak het woord nog voorkomt in de rest van de tekst. Toon het resultaat aan de gebruiker. (Elke lijn bevat één woord gevolgd door spaties.)

```
%include "gt.asm"
covar
inarea: resb 70
woord: resb 70
aantal: dd 1
een: dd 1
zeventig: dd 70
inleiding
openin
mov ebx, 0

lees
cld
mov ecx, 70
mov esi, inarea
mov edi, woord
rep movsb

lezen:
lees
cmp eax, 0
je einde

mov ecx, 70
mov esi, 0

lus:
    mov al, [woord + esi]
    cmp al, [inarea + esi]
    jne lezen
    add esi, 1
    loop lus
add ebx, 1
jmp lezen

einde:
```

```
mov [aantal], ebx
uit [aantal]
slot
```

STRINGS IN BESTANDEN

Tel op

Schrijf een programma dat 2 getallen leest via inv en de som afdrukt in het uitvoerbestand. De getallen bestaan uit niet meer dan 5 cijfers. Als uitvoer willen we:

```
HET EERSTE GETAL IS:     39
HET TWEEDE GETAL IS:     40
==========================
DE SOM IS:               79
```

```
%include "gt.asm"
covar
outarea: resb 70
        db 0Dh, 0Ah
getal1: resd 1
getal2: resd 1
som: resd 1
tekstgetal1: db 'HET EERSTE GETAL IS:'
tekstgetal2: db 'HET TWEEDE GETAL IS:'
tekstsom: db 'DE SOM IS:'
inleiding
openuit
inv [getal1]
inv [getal2]

; som
mov eax, [getal1]
add eax, [getal2]
```

```
mov [som], eax

; eerste getal
cld
call legelijn

mov ecx, 20
mov esi, tekstgetal1
mov edi, outarea
rep movsb

mov eax, [getal1]
call omzetascii
schrijf

; tweede getal
call legelijn

mov ecx, 20
mov esi, tekstgetal2
mov edi, outarea
rep movsb

mov eax, [getal2]
call omzetascii
schrijf

; lijn
call legelijn

mov ecx, 27
mov al, '='
mov edi, outarea
rep stosb
schrijf

; som twee getallen
call legelijn

mov ecx, 10
mov esi, tekstsom
```

```
mov edi, outarea
rep movsb

mov eax, [som]
call omzetascii
schrijf
slot

legelijn:
mov ecx, 70
mov al, ' '
mov edi, outarea
rep stosb
ret

omzetascii:
mov edi, outarea + 27
std
mov ebx, 10

lus:
mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne lus

cld
ret
```

BTW-berekening

```
Invoer (meerdere lijnen):

   kol  1-20: naam
   kol 31-40: inkomen

Uitvoer (voor ieder invoerrecord):

   kol  1-20: naam (idem als invoer)
   kol 31-40: inkomen (idem als invoer)
   kol 42-50: belasting

Deze belasting wordt als volgt berekent:

    Als inkomen <= 2500, dan is belasting = 0
    Als inkomen > 2500 en <= 5000, dan is belasting = (inkomen - 2500) * 10%
    Als inkomen > 5000 en <= 10000, dan is belasting = (inkomen - 5000) * 20% + 250
    Als inkomen > 10000, dan is belasting = (inkomen - 10000) * 40% + 1250



%include "gt.asm"
covar
inarea: resb 70
outarea: resb 70
        db 0Dh, 0Ah
een: dd 1
tien: dd 10
twintig: dd 20
veertig: dd 40
honderd: dd 100
inleiding
openin
openuit

invoerbestand:
lees
cmp eax, 0
je einde

; uitvoer leeg maken
cld
mov ecx, 70
```

```
mov al, ' '
mov edi, outarea
rep stosb

; naam en inkomen kopiëren
mov ecx, 40
mov esi, inarea
mov edi, outarea
rep movsb

; inkomen (string naar integer)
mov ecx, 10
mov esi, inarea + 30
tekstbin

; belasting berekenen
cmp eax, 2500
jle eerste
cmp eax, 5000
jle tweede
cmp eax, 10000
jle derde
jmp vierde

eerste:
mov eax, 0
jmp verder

tweede:
sub eax, 2500
imul dword [tien]
idiv dword [honderd]
jmp verder

derde:
sub eax, 5000
imul dword [twintig]
idiv dword [honderd]
add eax, 250
jmp verder
```

```
vierde:
sub eax, 10000
imul dword [veertig]
idiv dword [honderd]
add eax, 1250

; belasting (integer naar string)
verder:
mov edi, outarea + 49

lus:
std
mov edx, 0
idiv dword [tien]
add dl, 30h
xchg al, dl
stosb
xchg al, dl

cmp eax, 0
jne lus

schrijf
jmp invoerbestand

einde:
slot
```

Loonberekening

Maak een invoerbestand met meerdere lijnen als volgt:

```
   kol  1-20: naam
   kol 21-25: aantal dagen
   kol 31-35: dagloon
```

1) Schrijf een programma dat de eerste record van dit invoerbestand leest en afdrukt. De uitvoer wordt dus:

   kol 1-35: idem als op invoerrecord
   kol 36-70: blanco

2) Wijzig uw programma zodat nu het invoerbestand record per record afgedrukt wordt.

3) Voeg volgende functie toe aan uw programma: voor ieder invoerrecord wordt nu ook het brutoloon (= aantal-dagen * dagloon) berekend en afgedrukt in kol 42-50.

4) Voeg volgende functie toe aan uw programma: voor ieder invoerrecord wordt nu ook de afhouding (= brutoloon * 40%) berekend en afgedrukt in kol 52-60.

5) Voeg volgende functie toe aan uw programma: nadat de gegevens van het laatste invoerrecord verwerkt (en afgedrukt) zijn, wordt er: (a) een blanco lijn gedrukt, en (b) het totaal van de kolommen brutoloon en afhouding afgedrukt.

Het is verboden de bevelen voor een volgend punt in te typen zonder dat men er zich van vergewist heeft dat de voorgaande punten perfect opgelost zijn!

```
%include "gt.asm"
covar
inarea: resb 70
outarea: resb 70
        db 0Dh, 0Ah
hulp: resd 1
brutoloon: resd 1
afhouding: resd 1
tien: dd 10
veertig: dd 40
honderd: dd 100
inleiding
openin
openuit

; nog invoer?
invoerbestand:
lees
```

```
cmp eax, 0
je einde

; uitvoer leeg
cld
call leeg

; copy invoer to uitvoer
mov ecx, 35
mov esi, inarea
mov edi, outarea
rep movsb

; aantal dagen to integer
mov ecx, 5
mov esi, inarea + 20
tekstbin
mov [hulp], eax

; dagloon to integer
mov ecx, 5
mov esi, inarea + 30
tekstbin

; brutoloon berekenen
imul dword [hulp]
mov [hulp], eax
mov ebx, [brutoloon]
add ebx, eax
mov [brutoloon], ebx

; brutoloon afdrukken
mov edi, outarea + 49
call omzetascii

; afhouding berekenen
mov eax, [hulp]
imul dword [veertig]
idiv dword [honderd]
mov ebx, [afhouding]
add ebx, eax
```

```
mov [afhouding], ebx

; afhouding afdrukken
mov edi, outarea + 59
call omzetascii

schrijf
jmp invoerbestand

einde:
; blanco lijn
call leeg
schrijf

; totaal brutoloon afdrukken
mov eax, [brutoloon]
mov edi, outarea + 49
call omzetascii

; totaal afhouding afdrukken
mov eax, [afhouding]
mov edi, outarea + 59
call omzetascii

schrijf
slot

leeg:
mov ecx, 70
mov al, ' '
mov edi, outarea
rep stosb
ret


omzetascii:
std

lus:
mov edx, 0
idiv dword [tien]
```

```
add dl, 30h
xchg al, dl
stosb
xchg al, dl

cmp eax, 0
jne lus
cld
ret
```

Revision #1
Created 2021-06-17 12:25:19 CEST by Jasper G.
Updated 2021-12-03 22:13:09 CET by Jasper G.