Computersystemen

2007 oktober examen

In oktober 2007 kreeg men in het vak Computersystemen de volgende test voorgeschoteld:

```
0007 66A1[4F06]              mov eax, [a]
     66F72E[5306]            imul dword [b]
     662B06[5706]            sub eax, [c]
     7430                    jz twee
     66A1[5706]        een:  mov eax, [c]
     66F72E[5306]            imul dword [b]
     663906[5706]            cmp [c],eax
     74??                    je een
     66A1[4F06]              mov eax, [a]
met [a] = 5, [b] = 2, [c] = 10
```

1. Wat is de inhoud van de bevelenteller en het bevelregister TIJDENS de uitvoering van het eerste mov-bevel (mov eax, [a])?

2. Wat is de volgende inhoud van de bevelenteller en bevelregister?

3. Wat is de inhoud van bevelenteller en bevelregister net na uitvoering van het eerste sprongbevel (jz twee)?

4. Wat is de inhoud van bevelenteller en bevelregister NET NA de uitvoering van het tweede sprongbevel (je een)?

5. Wat is de volgende inhoud van de bevelenteller en bevelregister?

Ook in 2009 kreeg men een vergelijkbare test.

En ook in 2010 kwam een gelijkaardige vraag op het examen.

2009 januari examen

- wat is het verschil tussen een vertaler en een vertolker + leg uit
- wat is de 2-complement voorstelling van...
- dan moest je nog zo'n adressering kunnen zoals op de voorlaatste test
- een vraag over een autogarage iets, zoals er een gelijke in de boek staat.
- nog iets van de 3d kaarten het verschil tussen een vga en een xvga geloof ik.
- nog een vraag over iets van string naar integer zetten.
- omrekenen van CHS naar LBA-nummer, en omgekeerd
- verschil tussen hyperthreading processoren en multicore-processoren
- de 3 verschillende vormen van RAID
- en nog iets over de bevelenteller en checkbin

2010 januari examen

- Bepaal de binaire voorstellingen van a en b
- Tel de binaire voorstellingen a en b op
- imul dword [b], geef de voorstellingen van eax en edx
- idiv dword [c], geef de voorstellingen van eax en edx
- Waarde bevelenteller zoeken tov bevelregister. Met een positieve jne.
- Een vraag waarin je een stuk code met indexregisters enzo moet schrijven (pagina 75) + schrijf dit stukje code als subroutine + schrijf de oproepende commando's
- Iets over de stapel en waar de instruction pointer staat na een bepaalde handeling(hoofdstuk 4.3)
- Een vraag over de inhoud van het voorgeheugen (pagina 169, "toegepast als voorgeheugen zou dit....") Geef de inhoud als de CVE (bv.) 22DC2A04 vraagt.
- Rekenen met C,H,S (hoofdstuk 5.8.7) --> een schijf met C, H, S (2042, 8, 63), hoe veel sectoren heeft die + reken LBA 748 om naar (c,h,s)
- Schrijf een programma dat een bestand uitleest en voor elke regel bepaald hoe veel keer i voorkomt (hoofdletter I telt ook mee)
  Invoer:

```
O julissi na jalyni
O julissi na dytini
O bulo diti non slukati
Sestrone dina katsu.

O julissi na ti buku
...
(op het examen stond hier het volledige "lied", maar je snapt het idee wel)
```

Uitvoer: (12345678... zit NIET mee in uitvoerbestand)

```
12345678...
 3 O julissi na yalyni
 4 O julissi na dyntini
 3 O bulo diti non slukati
 1 Sestrone dina katsu.
 0
 3 O julissi na ti buku
enz...
```

- ...

2010 samenvatting commando's - Mattias

Een bestand met de commando's nodig voor de oefeningen met dank aan Mattias:

computersystemen comando's.docx.pdf

2011 januari examen

- Gegeven: [ijzel] en [sneeuw] en beiden worden opgeteld of vermenigvuldigd in ebx of eax. Er wordt gevraagd telkens de inhoud van eax en ebx te geven.
- Assembler: Een invoerrecord bevat
  - productnaam
  - hoeveelheid (in liter)
  - aandeel alcohol in ml/l
  - een karakter B (bier) of W (wijn.)
    - Er wordt gevraagd een uitvoerrecord te schrijven in assembler dat de hoeveelheid ALCOHOL in liter berekent van enkel de wijn.
- Bespreek DIMM
- Geef het schema van een hedendaagse chipset
- Een oefening met bevelenteller en bevelenregister (Hoofdstuk 2?)
- Welke parameters heeft ExitProcess nodig?
- Hoe weet de c.v.e dat het om een directe operand gaat, of om een adres na vertaling? Wat doet het vertaalprogramma? (anders geformuleerd)
- Verschil tussen PCI en PCI-E uitleggen.
- Een oefening waarbij een stuk asm code staat en 2 variabelen met hun inhoud(strings). op deze strings werd movsb en stosb gebruikt en dan moest je de inhoud van een variabele geven na de uitvoering van dat stuk code.
- Noem de verschillende wachttijden van een geheugen en geef ook het voordeel van bloktransfer(+tijdschema van deze wachttijden)
- Debiet berekenen van een harde schijf

2012 januari examen

- Gegeven: 3 constanten: 2 negatieven en 1 positief.

Gevraagd: inhouden van eax en edx geven na bevelen (mov eax, ... ; mov edx, ...; add eax, edx; imul dword ...)

Constanten	Definitie
vos	dd -70
kat	dd 57
wolf	dd -5

- Wat is de inhoud (volledig) van eax en edx na de uitvoering van "mov eax, [vos]; mov edx,[kat]"

- Zelfde vraag als daarna "add edx, [vos]" wordt uitgevoerd

- Zelfde vraag als daarna "imul dword [wolf] wordt uitgevoerd

- Gegeven: Stukje programma(Bevelregisters en de bevelen op dat adres) begin van bevelenteller.

Gevraagd: Bevelenteller geven voor een bevel, en na een bevel, en de 2 verschillend mogelijkheden na een jump-bevel

Adres	Opdracht
A1[9A010000]	mov eax, [a]
8B1D[9E010000]	mov ebx, [b]
81FB00000000	cmp ebx, 0
7410	JE gedaan
BA00000000	mov edx, 0
...	...
A3[A2010000]	gedaan: mov [ggd], eax

- Na het ophalen van mov eax, [A]

- Na het ophalen van cmp ebx, 0

- Na het uitvoeren van je gedaan (geef beide mogenlijkheden)

- Gegeven: rij met getallen, een getal n wordt ingevoerd.

Gevraagd: getalt op plaats n van de rij teruggeven, door gebruik te maken van subroutine en de stack.

- Gegeven: DIMM met 8 ics van 128 Megabyte, en 8 banks, maximum 4 DIMMS.

Gevraagd: Op welke manier kan FA545300 worden opgedeeld?

- Leg Zone Bit Recording uit.

- Gegeven cache(met blokken van 256 bytes)

dubbelwoord op adres FA057402 vinden.

- Wat zijn de doelstellingen van RAID.

Bespreek RAID-0, RAID-1 en RAID-5 en welke doelstellingen volbrengen ze?

- Gegeven: Een landbouwer heeft verschillende stukken grond.

Schrijf een programma dat als invoer meekrijgt:

* kol. 0-20 : De naam(plaats) waar het stuk grond ligt.

* kol. 22 : De functie van het stuk grond met 1 letter aangegeven(voor dieren of voedsel).

* kol. 24-31 : De oppervlakte

* kol. 33-40 : De prijs

En als uitvoer:

* kol. 6-25: De naam

* kol. 22- Prijs per meter

2013 januari examen

Veel vragen waarbij je het optellen van hexadecimale getallen nodig hebt. Zorg dat je dit vlot kunt!
Vragen i.v.m bevellenteller en bevelregister
Grote vraag om te programmeren.
Wat is lokaliteit? Ook moet je een adres van de GBE ontleden.

2015 januari examen

- Geef de inhoud van EAX en EDX na een MOV-bevel, een ADD-bevel en een IMUL-bevel volledig, dus zowel in binair (als in hexadecimaal) als in decimaal.
  - Opgelet: EDX wordt na het IMUL-bevel opgevuld met nullen of enen, naargelang het teken van de inhoud van EAX: is de inhoud van EAX na de vermenigvuldiging negatief, dan is EDX voor de rest opgevuld met enen. Is het resultaat positief, dan zijn dit nullen.
- Gegeven is een beschrijving van een SD-RAM en het hexadecimale adres van een byte. Geef DIMM-nummer, banknummer, rijnummer en kolomnummer van deze byte.
  - Zet hiervoor het hexadecimale adres om naar 32-bits binair adres. De eerste 2 bits vormen het DIMM-nummer, de volgende 3 bits het banknummer, de 14 bits daarna het rijnummer en de 10 daarop het kolomnummer. De laatste 3 bits vormen het bytenummer.
- Een vraag over wachttijden en bloktransfer, waarbij CAS "3-2-1" gegeven is.
- Een programmeeroefening in verband met rijen: geef één voor één een rij van 30 dubbelwoorden in en daarna een getal "n". Het n-de getal van de rij moet teruggegeven worden. Bijvoorbeeld bij groep = 1, 20, 300, 4000, 50000 en n = 4, moet als resultaat het vierde element van de rij, dus 4000, teruggegeven worden.

```
  %include "gt.asm"
  covar
  n:          resd 1
  uitvoer:    resd 1
  groep:      resd 30                  ; hier komen de 30 ingevoerde getallen
  een:        dd 1
  vier:       dd 4                     ; elk dubbelwoord is twee woorden, en dus 4 bytes, lang

  inleiding
            mov ecx, 30                ; er worden 30 getallen opgevraagd en na elk getal wordt ECX met 1 verminderd

  lus:        inv [groep+ebx]            ; gebruik hier om het even welk register, maar zorg dat ja de waarde ervan niet overschrijft!
            add ebx, [vier]            ; tel telkens 4 bij ebx op, zodat het ingevoerde getal na de 4 bytes van het vorig ingevoerde getal komt te staan
            loop lus                   ; deze loop stopt wanneer ecx op 0 staat, in dit geval dus na 30 keer

            inv [n]
```

```
mov eax, [n]
sub eax, [een]              ; trek 1 af van n, want n begint op 0 en niet op 1
imul dword [vier]           ; want 1 getal is 4 bytes
uit [groep+eax]   ; als n bijvoorbeeld 4 is, dan is groep+eax hier groep+12
```

- Bereken het aantal MT/s bij een busbreedte van 32 bits en een debiet 66.7 MB/s.
- Leg kort uit waarom het handig is wanneer een processor over een pijplijn beschikt.
- Wat betekent 01020201 in het bevel "MOV EAX, 02010102"? Waarvan is dit afhankelijk?
- Geef de waarde van ESP voor en na een aantal PUSH- en POP-bevelen.
  - Let hierbij op de lengte van het register of de waarde die/dat gePUSHt of gePOPt wordt.

- Schrijf voor onderstaande records een uitvoerbestand in de vorm van "75% van onze klanten behoort tot onze doelgroep", waarbij de doelgroep bestaat uit vrouwen die geboren zijn na 1985.
  - Hou dus minstens 2 variabelen bij: het aantal vrouwen geboren na 1985, en het totaal aantal klanten. Het percentage klanten dat tot de doelgroep behoort is dan (vrouwen x 100) / totaal.
  - Om de jaartallen met 1985 te vergelijken, moet je ze eerst omzetten naar binair. Hiervoor kun je tekstbin gebruiken.
  - Om het percentage uit te schrijven, moet dit eerst omgezet worden naar de ASCII-voorstelling. De subroutine die je daarvoor nodig hebt, kun je best van buiten leren.

Kolom 1 - naam	Kolom 25 - geslacht en geboortedatum	Kolom 50 - totale uitgaven in eurocent
An	V1986	5000
Betty	V1972	25000
Catherine	V1994	1500
Dave	M1989	8000

2017 januari examen

- Geef de inhoud van EAX en EDX tijdens een programma met MOV-, ADD-, IMUL, en IDIV bevelen en een voorwaardelijke sprong.

> Vergeet niet: Na IMUL DWORD komt het product in EDX:EAX Dit wil zeggen dat als het getal niet in EAX past, het verderloopt in EDX.
> Anders wordt EDX opgevuld met 0h of Fh, afhankelijk of het product positief of negatief is. Bij IDIV DWORD, komt uiteraard de rest in EDX

- Geef definities van herlocalisaties, radiale informatiedichtheid (met schets), RC-wachttijd.
- Analyseer een bevraging van het werkgeheugen.
- De stapel: In een subroutine die de absolute waarde van een getal geeft enkele push- en pop-bevelen aanvullen.

> Wat je op een stapel zet, met push, komt telkens bovenaan de stapel. Daarom moet je aan het eind van je subroutine de elementen in omgekeerde volgorde van het pushen terug afhalen met pop (telkens het bovenste)

- Vervolg op vorige: De waarde van de EIP (Instruction Pointer geven) na enkele push- en pop-bevelen

> Hier is het van belang de grootte van registers (eax, ax, al)... goed te kennen. Ook niet vergeten dat als je pusht, de EIP verlaagt, als je popt, de EIP verhoogt.

- Grote programmeeropdracht
  - Je krijgt een invoerbestand waarin op iedere rij een getal staat. Je moet voor ieder getal bepalen of het getal een deel is van de rij van Fibonacci. Je mag de eerste twee getallen (0 en 1) als constanten gebruiken.

INVOER:

> Kolom 7 |
>
> 34
>
> 89
>
> 7
>
> 13

UITVOER:

```
34 IS HET       10e GETAL VAN FIBONACCI
89 IS HET       14e GETAL VAN FIBONACCI
 7 IS GEEN GETAL VAN FIBONACCI
13 IS HET        8e GETAL VAN FIBONACCI
```

2018 januari examen

Geef definities van:

- native command queuing
- multithreading
- wet van moore
- principe van lokaliteit

- Geef de inhoud van EAX en EDX tijdens een programma met MOV-, ADD-, IMUL, en IDIV bevelen en een voorwaardelijke sprong.

> Vergeet niet: Na IMUL DWORD komt het product in EDX:EAX Dit wil zeggen dat als het getal niet in EAX past, het verderloopt in EDX.
>
> Anders wordt EDX opgevuld met 0h of Fh, afhankelijk of het product positief of negatief is. Bij IDIV DWORD, komt uiteraard de rest in EDX

- Vraag over de stapel

- In een programma de vlaggen

- Raid-0 Raid-1 Raid-5: Bij elk zeggen wat de invloed is op betrouwbaarheid en snelheid van lezen of schrijven. Ook uitleggen.

- Grote programmeer oefening

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 ?? ?? 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
```

2018 oefeningen hoofdstuk 4 de stapel oplossingen

DE STAPEL

Korte vraag

Gegeven volgend programma om de rij van Fibonacci te berekenen:

```
fibGetallen:   dd 0
               dd 1
               resd 98
...
               mov ecx, 98
               mov eax, fibGetallen + 8
lus:            push eax
               call berekenVolgendFibGetal
               add eax, 4
               loop lus
```

De code definieert een rij van 100 getallen, waarvan de eerste twee getallen (0 en 1) als constanten gedefiniëerd worden. De volgende 98 getallen worden berekend door middel van een lus waar telkens de 'berekenVolgendFibGetal'-methode opgeroepen wordt. Deze methode berekent één getal. Het adres waar het te berekenen getal moet worden opgeslagen wordt via de stapel doorgegeven. De methode kan dan aan de hand van dat adres (dat dus wijst naar een getal in de 'fibGetallen'-rij) de twee voorgaande getallen ophalen, optellen en het resultaat wegschrijven.

Teken de inhoud van de stack op de moment dat de subroutine 'berekenVolgendFibGetal' begint. Vul daarna de subruoutine aan:

```
berekenVolgendFibGetal:
               push ebp
               mov ebp, esp
               push eax
```

```
push esi
; kopieer het adres van het ide
; getal (dat op de stack staat)
; naar ESI
mov esi, [ebp + 8]
; kopieer de waarde van het getal
; ervoor naar eax
mov eax, [esi - 4]
; tel de waarde van het getal dat
; twee plaatsen ervoor staat erbij op
add eax, [esi - 8]
; kopieer de berekende waarde naar de array
mov [esi], eax
pop esi
pop eax
pop ebp
ret 4
```

De stapelwijzer (1)

Bepaal voor onderstaand stuk code de uiteindelijke waarde van de stapelwijzer (register ESP), als je weet dat de initiële waarde van ESP 0000029Ah is.

PROGRAMMA

De volgende instructies worden uitgevoerd.

```
pop ecx
push dword [520h]
sub eax, [404h]
idiv dword [404h]
pop ah
pop ch
mov eax, [520h]
```

STAPELWIJZER (na uitvoering, in hexadecimaal)

De waarde van ESP is:
0000029C

De stapelwijzer (2)

Bepaal voor onderstaand stuk code de uiteindelijke waarde van de stapelwijzer (register ESP), als je weet dat de initiële waarde van ESP 0000011Ah is.
PROGRAMMA

De volgende instructies worden uitgevoerd.

pop dword [700h]
push dword [248h]
pop dl
push dword [700h]
pop cx
add eax, [248h]

STAPELWIJZER (na uitvoering, in hexadecimaal)

De waarde van ESP is:
00000119

De stapelwijzer (3)

Bepaal voor onderstaand stuk code de uiteindelijke waarde van de stapelwijzer (register ESP), als je weet dat de initiële waarde van ESP 0000016Ch is.
PROGRAMMA

De volgende instructies worden uitgevoerd.

pop dword [62Ch]
pop al
add eax, [62Ch]
push dword [62Ch]
push dword [438h]
push dword [408h]
pop dword [62Ch]

STAPELWIJZER (na uitvoering, in hexadecimaal)

De waarde van ESP is:
00000169

De stapelwijzer (4)

Bepaal voor onderstaand stuk code de uiteindelijke waarde van de stapelwijzer (register ESP), als je weet dat de initiële waarde van ESP 0000006Eh is.
PROGRAMMA

De volgende instructies worden uitgevoerd.

pop dword [324h]
idiv dword [324h]
add eax, 5
mov [324h], eax
push bl
sub eax, 39
add eax, [324h]
idiv dword [61Ch]

STAPELWIJZER (na uitvoering, in hexadecimaal)

De waarde van ESP is:
00000071

De stapelwijzer (5)

Bepaal voor onderstaand stuk code de uiteindelijke waarde van de stapelwijzer (register ESP), als je weet dat de initiële waarde van ESP 0000025Ah is.
PROGRAMMA

De volgende instructies worden uitgevoerd.

push dword [208h]
push dword [208h]
mov eax, 53
push dword [924h]
imul dword [908h]

STAPELWIJZER (na uitvoering, in hexadecimaal)

De waarde van ESP is:
0000024E

De stapelwijzer (6)

Bepaal voor onderstaand stuk code de uiteindelijke waarde van de stapelwijzer (register ESP), als je weet dat de initiële waarde van ESP 000002A4h is.
PROGRAMMA

De volgende instructies worden uitgevoerd.

add eax, 27
pop ah
idiv dword [73Ch]

```
push ch
imul dword [638h]
pop dword [638h]
sub eax, [73Ch]
sub [638h], eax
```

STAPELWIJZER (na uitvoering, in hexadecimaal)

De waarde van ESP is:
000002A8

SUBROUTINES

ASCII-omzetting

Het stukje programma om een integer om te rekenen naas ASCII komt in vele programma's (soms meermaals) voor. Schrijf een programma dat een getal aan de gebruiker vraagt, dit getal omzet naar ASCII, en wegschrijft naar een uitvoerbestand. Zorg er voor dat het stukje code dat gebruikt wordt om een getal om te zetten naar ASCII als een subroutine geschreven is (d.w.z.: met call en ret instructies). Het getal dat de gebruiker invoert is niet langer dan 10 cijfers.

```
%include "gt.asm"
covar
outarea: resb 70
        db 0Dh, 0Ah
getal: resd 1
inleiding
openuit
call leeg

inv [getal]
push dword [getal]
push dword 9
```

```
call omzetascii
schrijf
slot

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



omzetascii:
push ebp
mov ebp, esp
push eax
push ebx
push edx
push edi
mov eax, [ebp + 12]

std
mov edi, outarea
add edi, [ebp + 8]
mov ebx, 10

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

pop edi
pop edx
pop ebx
```

```
pop eax
pop ebp
ret 8
```

Grootste gemene deler

Schrijf een programma dat aan de gebruiker twee getallen vraagt, de grootste gemene deler ervan berekent, en het resultaat aan de gebruiker toont. Zorg er voor dat het stukje code dat gebruikt wordt om de grootste gemene deler te berekenen als een subroutine geschreven is. Je kan hiervoor je code van de oefeningen van hoofdstuk 2 hergebruiken. Zorg er voor dat de twee getallen via registers EAX en EBX aan de subroutine doorgegeven worden. Het resultaat van de subroutine komt in register EDX. Na uitvoering van de subroutine mag enkel de waarde van register EDX aangepast zijn.

```
%include "gt.asm"
covar
getaleen: resd 1
getaltwee: resd 1
ggd: resd 1
inleiding
inv [getaleen]
inv [getaltwee]

mov eax, [getaleen]
mov ebx, [getaltwee]

push dword edx
push dword eax
push dword ebx
call grootstegemenedeler
pop dword edx

mov [ggd], edx

uit [ggd]
```

slot

```
grootstegemenedeler:
push ebp
mov ebp, esp
push eax
push ebx
push edx

mov eax, [ebp + 12]
mov ebx, [ebp + 8]
mov [ebp + 16], ebx

terug:
cmp ebx, 0
je verder

mov edx, 0
mov [ebp + 16], ebx
idiv dword [ebp + 16]
mov eax, ebx
mov ebx, edx
jmp terug

verder:
pop edx
pop ebx
pop eax
pop ebp
ret 8
```

Kleinste gemene veelvoud

Schrijf een programma dat aan de gebruiker twee getallen vraagt, het kleinste gemene veelvoud ervan berekent, en het resultaat aan de gebruiker toont. Zorg er voor dat het stukje code dat gebruikt wordt om het kleinste gemene veelvoud te berekenen als een subroutine geschreven is. Je kan hiervoor je code van de oefeningen van hoofdstuk 2 en van de vorige oefening hergebruiken. Zorg er voor dat de twee getallen via registers EAX en EBX aan de subroutine doorgegeven worden. Het resultaat van de subroutine komt in register ECX. Na uitvoering van de subroutine mag enkel de waarde van register ECX aangepast zijn.

```
%include "gt.asm"
covar
getaleen: resd 1
getaltwee: resd 1
ggd: resd 1
kgv: resd 1
inleiding
inv [getaleen]
inv [getaltwee]

mov eax, [getaleen]
mov ebx, [getaltwee]

push dword ecx
push dword edx
push dword eax
push dword ebx
call kleinstegemeneveelvoud
pop dword ecx

mov [kgv], ecx

uit [kgv]
slot



kleinstegemeneveelvoud:
push ebp
mov ebp, esp
push eax
push ebx
```

```
push edx

mov eax, [ebp + 12]
mov ebx, [ebp + 8]
mov [ebp + 16], ebx

terug:
cmp ebx, 0
je verder

mov edx, 0
mov [ebp + 16], ebx
idiv dword [ebp + 16]
mov eax, ebx
mov ebx, edx
jmp terug

verder:
mov ebx, 1
mov eax, [ebp + 12]
imul dword [ebp + 8]
imul dword ebx
idiv dword [ebp + 16]

mov [ebp + 20], eax

pop edx
pop ebx
pop eax
pop ebp
ret 12
```

Faculteit

Schrijf een programma dat aan de gebruiker een waarde, n, vraagt, n! berekent, en het resultaat toont.

n! = n * (n-1) * (n-2) * ... * 3 * 2 * 1

Zorg er voor dat het stukje code dat gebruikt wordt om n! te berekenen als een subroutine geschreven is.

```
%include "gt.asm"
covar
n: resd 1
nuitroepteken: resd 1
inleiding
inv [n]

push dword [nuitroepteken]
push dword [n]
call nfaculteit
pop dword [nuitroepteken]

uit [nuitroepteken]
slot


nfaculteit:
push ebp
mov ebp, esp
push eax
push ebx
mov ebx, [ebp + 8]
mov eax, 1

lus:
cmp ebx, 1
jg rekenen
jmp einde

rekenen:
imul dword ebx
sub ebx, 1
jmp lus
```

```
einde:
mov [ebp + 12], eax
pop ebx
pop eax
pop ebp
ret 4
```

2019 oplossingen oefeningen

Met dank aan de Github van Martijn

Bestanden:

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
uitvoer: resd 1

inleiding



openin
lees
cld

sub eax, eax
sub esi, esi
sub ebx, ebx
mov al, 'a'
mov ecx, 70

terug:
cmp al, [inarea + esi]
jne verder
add ebx, 1
```

```
verder:
add esi, 1
sub ecx, 1
cmp ecx, 0
je einde
jmp terug

einde:
mov [uitvoer], ebx
uit [uitvoer]

slot
```

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

```
%include 'gt.asm'
covar

inarea: resb 70
uitvoer: resd 1

inleiding


openin
lees
cld

sub eax, eax
sub esi, esi
sub ebx, ebx
mov al, 'a'
mov ecx, 70

terug:
mov al, 'a'
cmp al, [inarea + esi]
je verder
mov al, 'e'
cmp al, [inarea + esi]
```

```
je verder
mov al, 'i'
cmp al, [inarea + esi]
je verder
mov al, 'o'
cmp al, [inarea + esi]
je verder
mov al, 'u'
cmp al, [inarea + esi]
je verder
jmp skip
verder:
add ebx, 1
skip:
add esi, 1
sub ecx, 1
cmp ecx, 0
je einde
jmp terug

einde:
mov [uitvoer], ebx
uit [uitvoer]

slot
```

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
uitvoer: resd 1

inleiding
```

```
openin
lees
cld

sub eax, eax
sub esi, esi
sub ebx, ebx
sub edx, edx
mov al, 'a'
mov ecx, 10

terug:
mov al, [inarea + edi]
cmp al, [inarea + esi]
jne verder
add ebx, 1
verder:
add esi, 1
sub ecx, 1
cmp ecx, 0
je next
jmp terug
next:
add edi, 70
sub edx, 1
cmp edx, 0
je einde


einde:
mov [uitvoer], ebx
uit [uitvoer]

slot
```

Rijen:

Schrijf een programma dat 9 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
getal: resd 9
ding: resd 1
meer: resd 1
minder: resd 1
negen: dd 9
een: dd 1

inleiding
sub ecx, ecx
sub eax, eax
sub edi, edi
mov ecx, 9



hoger:
    cmp ecx, 0
    jle verder
    inv[getal + edi]
    add eax, [getal + edi]
    add edi, 4
    sub ecx, 1
    jmp hoger

verder:
mov [ding], eax
imul dword [een]
idiv dword [negen]
mov [ding], eax
mov ecx, 9
sub edi, edi
```

```
terug:
    mov eax, [ding]
    cmp eax, [getal + edi]
    jl nogverder
    je natel
    sub eax, eax
    add eax, 1
    add [meer], eax
    add edi, 4
    jmp na

nogverder:
    mov eax, [ding]
    cmp eax, [getal + edi]
    sub eax, eax
    add eax, 1
    add [minder], eax

natel:
    add edi, 4
na:
    loop terug


uit [ding]
uit [minder]
uit [meer]
slot
```

Schrijf een programma dat 6 getallen inleest en deze gesorteerd van klein naar groot afdrukt. Om te sorteren kan men als volgt te werk gaan:

```
doe 5 maal:
  { i = 0;
    doe 5 maal
      { vergelijk element(i) met element(i+1);
        if (element(i) > element(i+1) {
          verwissel;
        }
```

```
      i = i + 1;
    }
  }

%include "gt.asm"
covar
getal: resd 6
nul: dd 0
hulp1: resd 1
hulp2: resd 2
teller: resd 1

inleiding



sub esi, esi
sub ecx, ecx



inv [getal]
inv [getal + 4]
inv [getal + 8]
inv [getal + 12]
inv [getal + 16]
inv [getal + 20]

begin:
sub ecx, ecx
sub esi, esi

vergelijk:
cmp ecx, 5
je verder
mov eax, [getal + esi]
mov edx, [getal + esi + 4]
cmp eax, edx
jle skip
mov [getal + esi + 4], eax
mov [getal + esi], edx
skip:
```

```
add ecx, 1
add esi, 4
jmp vergelijk

verder:
sub ecx, ecx
mov ecx, [teller]
add ecx, 1
mov [teller], ecx
cmp ecx, 5
jle begin




uit [getal]
uit [getal + 4]
uit [getal + 8]
uit [getal + 12]
uit [getal + 16]
uit [getal + 20]




slot
```

Strings in bestanden:

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

hulp: resd 1
getal1: resd 1
getal2: resd 2
inarea: resb 70
outarea: resb 70
        db 13,10
vb1: dd "HET EERSTE GETAL IS:"
vb2: dd "HET TWEEDE GETAL IS:"
vb3: dd "DE SOM IS:          "
spatie: dd " "
lijn: dd "============================"
som: dd "DE SOM IS:"

inleiding
openuit

inv[getal1]
inv[getal2]

# EERSTE GETAL
cld
mov ecx, 70
mov al, " "
mov edi, outarea
rep stosb

mov ecx, 20
mov esi, vb1
mov edi, outarea
rep movsb

mov eax, [getal1]
```

```
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
schrijf

# TWEEDE GETAL
cld
mov ecx, 70
mov al, " "
mov edi, outarea
rep stosb

mov ecx, 20
mov esi, vb2
mov edi, outarea
rep movsb

mov eax, [getal2]

mov edi, outarea + 27
std
mov ebx, 10
lus1: mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne lus1
schrijf
```

```
# STREEP
cld
mov ecx, 70
mov al, " "
mov edi, outarea
rep stosb

cld
mov ecx, 27
mov al, "="
mov edi, outarea
rep stosb
schrijf

# SOM
cld
mov ecx, 70
mov al, " "
mov edi, outarea
rep stosb

mov ecx, 20
mov esi, vb3
mov edi, outarea
rep movsb

mov eax, [getal1]
add eax, [getal2]

mov edi, outarea + 27
std
mov ebx, 10
lus2: mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne lus2
```

```
schrijf




slot

;BELASTING BEREKENEN

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

hulp: resd 1
inarea: resb 70
outarea: resb 70
        db 13,10
spatie: dd " "
tien: dd 10
vijf: dd 5
twee: dd 2
een: dd 1

inleiding
sub esi, esi
openuit
openin
```

```
lus:
cld
mov ecx, 70
mov al, " "
mov edi, outarea
rep stosb

lees
cmp eax, 0
je einde

cld
mov ecx, 70
mov esi, inarea
mov edi, outarea
rep movsb


mov esi, inarea + 30
mov ecx, 10
tekstbin
mov [hulp], eax
uit [hulp]

imul dword [een]
cmp eax, 2500
jg next
sub eax, eax
jmp past


next:
cmp eax, 5000
jg next1
sub eax, 2500
idiv dword [tien]
jmp past

next1:
```

```
cmp eax, 10000
jg next2
sub eax, 5000
idiv dword [vijf]
add eax, 250
jmp past

next2
sub eax, 10000
imul eax, 2
idiv dword [vijf]
add eax, 1250
past:


mov edi, outarea + 49
std
mov ebx, 10
luss: mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne luss
schrijf


jmp lus




einde:
```

slot

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

hulp: resd 1
hulp1: resd 1
inarea: resb 70
outarea: resb 70
        db 13,10
spatie: dd " "
tien: dd 10
vijf: dd 5
twee: dd 2
een: dd 1
totaalbruto: resd 1
```

```
totaalafhouding: resd 1

inleiding
sub esi, esi
openuit
openin

lus:
cld
mov ecx, 70
mov al, " "
mov edi, outarea
rep stosb

lees
cmp eax, 0
je einde

cld
mov ecx, 36
mov esi, inarea
mov edi, outarea
rep movsb

cld
mov esi, inarea + 21
mov ecx, 4
tekstbin
mov [hulp], eax
uit [hulp]

cld
mov esi, inarea + 31
mov ecx, 4
tekstbin
mov [hulp1], eax
uit [hulp1]



imul dword [hulp]
```

```
mov [hulp], eax
add [totaalbruto], eax

mov edi, outarea + 48
std
mov ebx, 10
luss: mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne luss

mov eax, [hulp]
imul dword [een]
imul dword [twee]
idiv dword [vijf]
add [totaalafhouding], eax

mov edi, outarea + 58
std
mov ebx, 10
lus2: mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne lus2

schrijf
jmp lus
einde:

cld
mov ecx, 70
mov al, " "
```

```
mov edi, outarea
rep stosb
schrijf

mov eax, [totaalbruto]
mov edi, outarea + 48
std
mov ebx, 10
lusbruto: mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne lusbruto

mov eax, [totaalafhouding]
mov edi, outarea + 58
std
mov ebx, 10
lusafhouding: mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne lusafhouding

schrijf



slot
```

Strings:

Schrijf een programma dat afdrukt:
WHO IS MY TAILOR?
MY TAILOR IS CHRISTIAN DIOR
MY TAILOR IS RICH
IS MY TAILOR RICH?

```
%include "gt.asm"
covar
outarea: resb 70
db 0Dh, 0Ah
vb1: db 'WHO IS MY TAILOR?'
vb2: db 'MY TAILOR IS CHRISTIAN DIOR'
vb3: db 'MY TAILOR IS RICH'
vb4: db 'IS MY TAILOR RICH?'
spatie: db ' '



inleiding
openuit

mov ecx, 70
mov esi, spatie
mov edi, outarea
rep movsb

mov ecx, 17
mov esi, vb1
mov edi, outarea
rep movsb
schrijf

mov ecx, 70
mov esi, spatie
mov edi, outarea
rep movsb



sub esi, esi
```

```
mov ecx, 27
mov esi, vb2
mov edi, outarea
rep movsb
schrijf

mov ecx, 70
mov esi, spatie
mov edi, outarea
rep movsb

sub esi, esi

mov ecx, 17
mov esi, vb3
mov edi, outarea
rep movsb
schrijf

mov ecx, 70
mov esi, spatie
mov edi, outarea
rep movsb

sub esi, esi

mov ecx, 18
mov esi, vb4
mov edi, outarea
rep movsb
schrijf


slot
```

2019 samenvatting - Axel Hamelryck

Met dank aan de Github van Martijn en natuurlijk Axel Hamelryck

Computersystemen_axelele_2019_compressed.pdf

2020 augustus examen

de oplossingen van verschillende studenten voor het augustus examen. Met dank aan ISW en natuurlijk de betrokken studenten:

Elias Beddegenoodts:

Name
Size
Modified
EliasBeddegenoodtsDeel2
.txt
Actions
2 KB6 months ago
Oefening_Auto_Lukas_De_Ruysscher_Examen_CS_Aug_2020
.txt
Actions
2 KB6 months ago
r0785293_Timo_Taverniers
.txt
Actions
2 KB6 months ago
theorie_oplossingen
.txt
Actions
1 KB6 months ago
4 files7 KB
Nextcloud - ISW Leuven
Privacy policy

Get your own free account

```
%include 'gt.asm'
covar
	inarea: resb 70
	outarea: times 70 db (' ')
		 db 0Ah
	aantalliter: resd 1
	aantalkm: resd 1
	duizend: dd 1000
	result: resd 1
	een: dd 1
	aantalwagens: resd 1
inleiding
		sub eax, eax
		mov ebx, 0

		openin
		openuit

	;lezen lijn per lijn
	volgendelijn:
		lees
		cmp eax, 0
		je einde

		mov ecx, 70
		mov al, ' '
		mov edi, outarea
		rep stosb

		;bestaande inhoud inarea naar outarea plaatsen
		mov ecx, 34
		mov esi, inarea
		mov edi, outarea
		rep movsb

		;liter to Int
		mov ecx, 4
		mov esi, inarea + 30
		tekstbin
```

```
        mov [aantalliter], eax

        ; km to Int
        mov ecx, 5
        mov esi, inarea + 20
        tekstbin
        mov [aantalkm], eax

        ;berekening
        mov eax, [aantalliter]
        imul dword [duizend]
        idiv dword [aantalkm]
        cmp eax, 60
        jg jump
        mov [result], eax

        ;schrijven in uitvoer
        mov ecx, 2
        mov eax, [result]
        mov edi, outarea + 40
        call omzetascii

        schrijf
jump:
        jmp volgendelijn

omzetascii:
        mov ebx, 10
        std
        lus:
                mov edx, 0
                idiv ebx
                or dl, 30h
                xchg al, dl
                stosb
                xchg al, dl
                cmp eax, 0
                jne lus
        cld
        ret
```

```
        einde:
slot
```

Lukas De Ruysscher:

```
 
Name
Size
Modified
 
EliasBeddegenoodtsDeel2
.txt
Actions
2 KB 6 months ago
 
Oefening_Auto_Lukas_De_Ruysscher_Examen_CS_Aug_2020
.txt
Actions
2 KB 6 months ago
 
r0785293_Timo_Taverniers
.txt
Actions
2 KB 6 months ago
 
theorie_oplossingen
.txt
Actions
1 KB 6 months ago
4 files 7 KB 
Nextcloud - ISW Leuven
Privacy policy

Get your own free account

%include "gt.asm"

covar
inarea: resb 70
```

```
outarea: resb 70
        db 0Ah
txtPercent: db '% van de wagens heeft een verbruik onder de 60.'
aantalOnder: dd 0
aantalRecords: dd 0
honderd: dd 100
duizend: dd 1000
zestig: dd 60
aantalKM: resd 1
aantalLiter: resd 1
verbruik: resd 1
hulp: resd 1
percentage: resd 1
inleiding

sub eax, eax
sub ebx, ebx
sub ecx, ecx
sub edx, edx

openin
openuit

lezen:
        lees
        cmp eax, 0
        je einde

;outarea leegmaken
        mov ecx, 70
        mov al, ' '
        mov edi, outarea
        rep stosb



;aantal liters lezen
        mov ecx, 5
        mov esi, inarea + 30
        tekstbin
        mov [aantalLiter], eax
```

```
;aantal km lezen
        mov ecx, 5
        mov esi, inarea + 20
        tekstbin
        mov [aantalKM], eax



;verbruik berekenen
        mov eax, [aantalLiter]
        imul dword [duizend]
        idiv dword [aantalKM]
        mov [verbruik], eax
        mov ebx, [aantalRecords]
        add ebx, 1
        mov [aantalRecords], ebx
        cmp eax, [zestig]
        jl minder
        jmp lezen

;aantal bijhouden en percentage te berekenen
minder:
        mov ebx, [aantalOnder]
        add ebx, 1
        mov [aantalOnder], ebx

        mov ecx, 35
        mov esi, inarea
        mov edi, outarea
        rep movsb

        mov ecx, 3
        mov eax, [verbruik]
        call omzetascii
        mov edi, outarea + 40
        jmp lezen

einde:
        mov eax, [aantalOnder]
        imul dword [honderd]
```

```
        idiv dword [aantalRecords]
        mov [percentage], eax

        mov edi, outarea
        call omzetascii

 mov ecx, 47
        mov esi, txtPercent
        mov edi, outarea + 3
        rep movsb
        schrijf
slot

omzetascii:
    mov ebx,10
    std
    lus:
        mov edx,0
        idiv ebx
        or dl,30h
        xchg al,dl
        stosb
        xchg al,dl
        cmp eax,0
        jne lus
    ret
```

Timo Taverniers:

```
Name
Size
Modified
EliasBeddegenoodtsDeel2
.txt
Actions
2 KB6 months ago
```

Oefening_Auto_Lukas_De_Ruysscher_Examen_CS_Aug_2020

.txt

Actions

2 KB 6 months ago

r0785293_Timo_Taverniers

.txt

Actions

2 KB 6 months ago

theorie_oplossingen

.txt

Actions

1 KB 6 months ago

4 files 7 KB

Nextcloud - ISW Leuven

Privacy policy

Get your own free account

```
%include "gt.asm"
covar

inarea: resb 70
outarea: times 70 db (' ')
db 0Ah

uitvoerlijn: db '% van de wagens heeft een verbruik onder de 60.'
km: resd 1
L: resd 1
resultaat: resd 1
aantal: resd 1
eco: resd 1

inleiding

sub eax, eax
sub edx, edx
sub ecx, ecx
sub ebx, ebx
```

```
sub esi, esi
sub edi, edi

openin
openuit
cld

next:
 lees
 cmp eax, 0
 je einde
 mov esi, inarea
 lodsb
 
 mov edi, outarea
 mov al, ' '
 mov ecx, 70
 rep stosb
 
 mov edi, outarea
 mov esi, inarea
 mov ecx, 34
 rep movsb

 mov eax, 0
 mov esi, inarea+20
 mov ecx, 5
 tekstbin
 mov [km], eax
 ;uit [km]
 
 mov eax, 0
 mov esi, inarea+31
 mov ecx, 4
 tekstbin
 mov [L], eax
 ;uit [L]
 
 mov eax, [L]
 mov edx, 0
```

mov ebx, 1000
imul ebx
mov edx, 0
mov ebx, [km]
idiv ebx
mov [resultaat],eax
;uit [resultaat]

std
mov eax, [resultaat]
mov ebx, 10
mov edi, outarea+41
lus: mov edx, 0
idiv ebx
or dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne lus
cld

schrijf

mov eax, [aantal]
add eax, 1
mov [aantal], eax
mov eax, [resultaat]
cmp eax, 60
jg next
mov eax, [eco]
add eax, 1
mov [eco], eax

jmp next

einde:
;uit [aantal]
;uit [eco]

```
mov edx, 0
mov eax, [eco]
mov ebx, 100
imul ebx
mov ebx, [aantal]
idiv ebx
mov [resultaat], eax
;uit [resultaat]

mov edi, outarea
mov ecx, 70
mov al, ' '
rep stosb

mov eax, [resultaat]
cmp eax, 100
je eindec

std
mov eax, [resultaat]
mov ebx, 10
mov edi, outarea+1
eilu: mov edx, 0
idiv ebx
or dl, 30h
xchg al, dl
stosb
xchg al,dl
cmp eax, 0
jne eilu
cld

mov esi, uitvoerlijn
mov edi, outarea+2
mov ecx, 47
rep movsb

schrijf
jmp eindefinaal
```

```
eindec:
        std
        mov eax, [resultaat]
        mov ebx, 10
        mov edi, outarea+2
eiluc:  mov edx, 0
        idiv ebx
        or dl, 30h
        xchg al, dl
        stosb
        xchg al,dl
        cmp eax, 0
        jne eiluc
        cld

        mov esi, uitvoerlijn
        mov edi, outarea+3
        mov ecx, 47
        rep movsb

        schrijf




eindefinaal:
slot
```

Onbekend theorie oplossing:

```
Name
Size
Modified
EliasBeddegenoodtsDeel2
.txt
Actions
```

2 KB 6 months ago

Oefening_Auto_Lukas_De_Ruysscher_Examen_CS_Aug_2020

.txt

Actions

2 KB 6 months ago

r0785293_Timo_Taverniers

.txt

Actions

2 KB 6 months ago

theorie_oplossingen

.txt

Actions

1 KB 6 months ago

4 files 7 KB

Nextcloud - ISW Leuven

Privacy policy

Get your own free account

173

128

45

32

13

08

05

04

1

1

0

0000 0000 0000 0000 0000 0000 1010 1101

1111 1111 1111 1111 1111 1111 0101 0010
0000 0000 0000 0000 0000 0000 0000 0001

1111 1111 1111 1111 1111 1111 0101 0011

FFFFFF53

77
64

13
8
4
1

0000 0000 0000 0000 0000 0000 0100 1101
0000004D

94
64

30
16

14
8

6
4
--
2
2
--
0

0000 0000 0000 0000 0000 0000 0101 1110

1111 1111 1111 1111 1111 1111 1010 0001

0000 0000 0000 0000 0000 0000 0000 0001

1111 1111 1111 1111 1111 1111 1010 0010

FFFFFFA2

173

 6

1038

1024

0014

0000 0000 0000 0000 0000 0100 0000 1110

0000040E

13

Bevel 1 na 7

2 na 8

3 na 9

4 10

5 11

6 12

7 13

8 14

9 15

10 16

11 17

12 18

13 19

14 20

15 21

16 22

17 23

18 24

19 25

20 26

21 27

22 28

23 29

24 30

25 31

26 32

27 33

28 34

29 35

30 36

31 37

32 38

33 39

34 40

35 41

000000B4

2020 januari examen

Examen bestond uit 5 vragen:

- Geef de inhoud van EAX en EDX tijdens een programma met MOV-, ADD-, IMUL, en IDIV bevelen (32 bits!!!!!!)

> Vergeet niet: Na IMUL DWORD komt het product in EDX:EAX Dit wil zeggen dat als het getal niet in EAX past, het verderloopt in EDX.
>
> Anders wordt EDX opgevuld met 0h of Fh, afhankelijk of het product positief of negatief is. Bij IDIV DWORD, komt uiteraard de rest in EDX

- Geef definities van: Symbolentabel, Zone bit recording (ZBR), Wet van Moore, RC-wachttijd bij SD RAM en Lineare informatiedichtheid (met kleine schets)
- Geef benamingen van 3 dingen die staan aangeduid op de foto (Werkgeheugen <-> CPU)
- Geef het bevel dat wordt uitgevoerd, bevelenteller(EIP), bevelregister, ESP en zowel de Nul-vlag(ZF) als het Teken-vlag(SF) van een gegeven programma met oa push en pop bevelen
- Grote programmeer oefening (getypt op PC in een .txt MAAR je kan deze niet uitvoeren!)
  - Je krijgt een invoerbestand waarin 4 kolommen staan. De eerste kolom bevat de letter A of V (met A wordt het aankopen van een materiaal bedoelt, met V de verkoop ervan), de tweede kolom een materiaal, de derde kolom de aantal eenheden en in de vierde kolom staat de prijs per eenheid. Op basis van de producten moet je berekenen hoeveel winst of verlies een bedrijf maakt.
  - Op de examenToledo staat oa de lus om te vertalen naar ASCII
  - Een voorbeeld van het invoerbestand dat gegeven is en het uitvoerbestand dat je moet maken:

INVOER:

```
A    ijzer             6     15
V    roestrvrij staal    5     48
A    inox              158   5
```

UITVOER:

```
Winst   1286 €
```

2021 januari examen

Oplossingen voor de oefeningen van het januari examen met dank aan ISW en Jens Gervais:

Oefening 1:

Name
Size
Modified
1_ComputersystemenAssemblyOefening1_JensGervais
.txt
Actions
1 KB 5 months ago
1_ComputersystemenAssemblyOefening1Invoer_JensGervais
.txt
Actions
< 1 KB 5 months ago
1_ComputersystemenAssemblyOefening2_JensGervais
.txt
Actions
< 1 KB 5 months ago
3 files 2 KB
Nextcloud - ISW Leuven
Privacy policy

Get your own free account

```
%include "gt.asm"
covar
 outarea: resb 70
  DB 0Dh, 0Ah
 inarea: resb 70
```

aantalkm: resd 1

liters: resd 1

duizend: dd 1000

verbruik: resd 1

verbruikLagerDan: resd 1

zestig: dd 60

totaal: resd 1

honderd: dd 100

tekst: DB '% van de wagens heeft een verbruik onder de 60.'

inleiding

openin ; open het invoerbestand

openuit ; open het uitvoerbestand

cld

lijnlezen:

lees

cmp eax, 0

je einde

mov edx, [totaal]

add edx, 1

mov [totaal], edx

call lijnvullen

mov ecx, 34

mov esi, inarea

mov edi, outarea

rep movsb

mov ecx, 5

mov esi, inarea + 21

tekstbin

mov [aantalkm], eax

mov ecx, 4

mov esi, inarea + 31

tekstbin

mov [liters], eax

mov eax, [liters]

imul dword [duizend]

idiv dword [aantalkm]

mov [verbruik], eax

```
mov eax, [zestig]

cmp [verbruik], eax
jge lijnlezen
mov edx, [verbruikLagerDan]
add edx, 1
mov [verbruikLagerDan], edx

mov ecx, 2
mov edi, outarea + 42
mov eax, [verbruik]
call asciiBerekening

schrijf
jmp lijnlezen
einde:
call lijnvullen
schrijf

mov eax, [verbruikLagerDan]
imul dword [honderd]
idiv dword [totaal]
mov edi, outarea + 3
call asciiBerekening

mov esi, tekst
mov edi, outarea + 4
mov ecx, 47
rep movsb
schrijf
slot

lijnvullen:
mov ecx, 70
mov al, ' '
mov edi, outarea
rep stosb
sub edi, edi
ret
```

```
asciiBerekening:
	mov ebx, 10
	std
	lus:
		mov edx, 0
		idiv ebx
		or dl, 30h
		xchg al, dl
		stosb
		xchg al, dl
		cmp eax, 0
		jne lus
	cld
	ret
```

Oefening 1 invoer:

```
 
Name
Size
Modified
 
1_ComputersystemenAssemblyOefening1_JensGervais
.txt
Actions
1 KB 5 months ago
 
1_ComputersystemenAssemblyOefening1Invoer_JensGervais
.txt
Actions
< 1 KB 5 months ago
 
1_ComputersystemenAssemblyOefening2_JensGervais
.txt
Actions
< 1 KB 5 months ago
3 files 2 KB
```

Nextcloud - ISW Leuven

Privacy policy

Get your own free account

OPEL MERIVA	855	45
OPEL ASTRA	935	47
FORD Focus	1230	62
BMW 735i	1100	85
VW GOLF GTD	895	44
FIAT CROMA 1.9	725	39
Tesla Model S	5	0

Oefening 2:

Name

Size

Modified

1_ComputersystemenAssemblyOefening1_JensGervais

.txt

Actions

1 KB 5 months ago

1_ComputersystemenAssemblyOefening1Invoer_JensGervais

.txt

Actions

< 1 KB 5 months ago

1_ComputersystemenAssemblyOefening2_JensGervais

.txt

Actions

< 1 KB 5 months ago

3 files 2 KB

Nextcloud - ISW Leuven

Privacy policy

Get your own free account

Bereken fibonacci (https://en.wikipedia.org/wiki/Fibonacci_number)

```
%include "gt.asm"
covar
    x: resd 1
    eersteFibo: dd 0
    tweedeFibo: dd 1
    nieuweFibo: resd 1
inleiding
    inv [x]
    uit [eersteFibo]
    uit [tweedeFibo]
    mov ecx, [x]
    sub ecx, 2
vlgFibo:
    mov eax, [eersteFibo]
    add eax, [tweedeFibo]
    mov [nieuweFibo], eax
    uit [nieuweFibo]
    mov eax, [tweedeFibo]
    mov [eersteFibo], eax
    mov eax, [nieuweFibo]
    mov [tweedeFibo], eax
    loop vlgFibo
slot
```

Algemeen

In het begin zijn de lessen zeer makkelijk, maar een keer je met het werkelijke programmeren begint wordt het gauw moeilijk. Probeer zo vaak mogelijk aanwezig te zijn in de lessen en aarzel niet om hulp te vragen tijdens het programmeren aan de leerkrachten. Zorg ook dat je zoveel mogelijk oefenigen afmaakt tijdens de praktijklessen, uit ondervinding weten we dat het heel verleidelijk is spelletjes te spelen. Als je de oefeningen goed kan zal het examen ook wel lukken: de moeilijkste oefeningen zijn immers veel moeilijker dan de programmeeropdrachten op het examen. Er worden ook 2 testen gehouden tijdens het jaar. De eerste test gaat alleen over hoofdstuk 1 de andere gaat over programmeren. Normaal gezien wordt duidelijk op voorhand gezegd wanneer de test plaatsvindt en wat je goed moet kennen. Zorg dat je hier goed op scoort, dit levert je een groot voordeel op bij het examen.

2016: De testen zijn een manier om makkelijk aan je punten te geraken, studeer hier zeker voor! Dit zijn zo goed als gratis examenpunten.

2019: De eerste test gaat over hoofstuk 1, de tweede over hoofdstuk 2. De oefeningen zijn vrij gelijkaardig aan die op EduBlend.

Assembly testomgeving

De oorspronkelijke maker is meneer Swennen, alle credits naar hem.

Notitie: als je zelf Linux draait kan je ook gewoon de repository clonen en NASM installeren.
https://gitlab.iswleuven.be/isw/general/computersystemen-assembler of
https://github.com/swenr/CompSys_Assembler

Stap 1

OVA importeren in VMWare of VirtualBox.

Stap 2

Inloggen met SSH
Gebruikersnaam: compsys
Wachtwoord: t

Stap 3

Stap 3.1

Ga naar `*/home/compsys/assembler*`

```
cd ~/assembler
```

Stap 3.2

```
git pull && ./vertaal gtine
```

Stap 3.2

Zet je programma in oef.asm, zet je input in oef.in

Stap 3.3

```
./vertaal oef
```

Stap 3.4

```
./voeruit oef
```

Stap 3.5

De uitvoer komt op het scherm en in bestand oef.uit

Step 3.6

Bugs melden aan Swennen ;)