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
```

Revision #1
Created 17 June 2021 12:25:54 by Jasper G.
Updated 3 December 2021 22:13:09 by Jasper G.