Preben Bucher-Johannessen 06.10.17

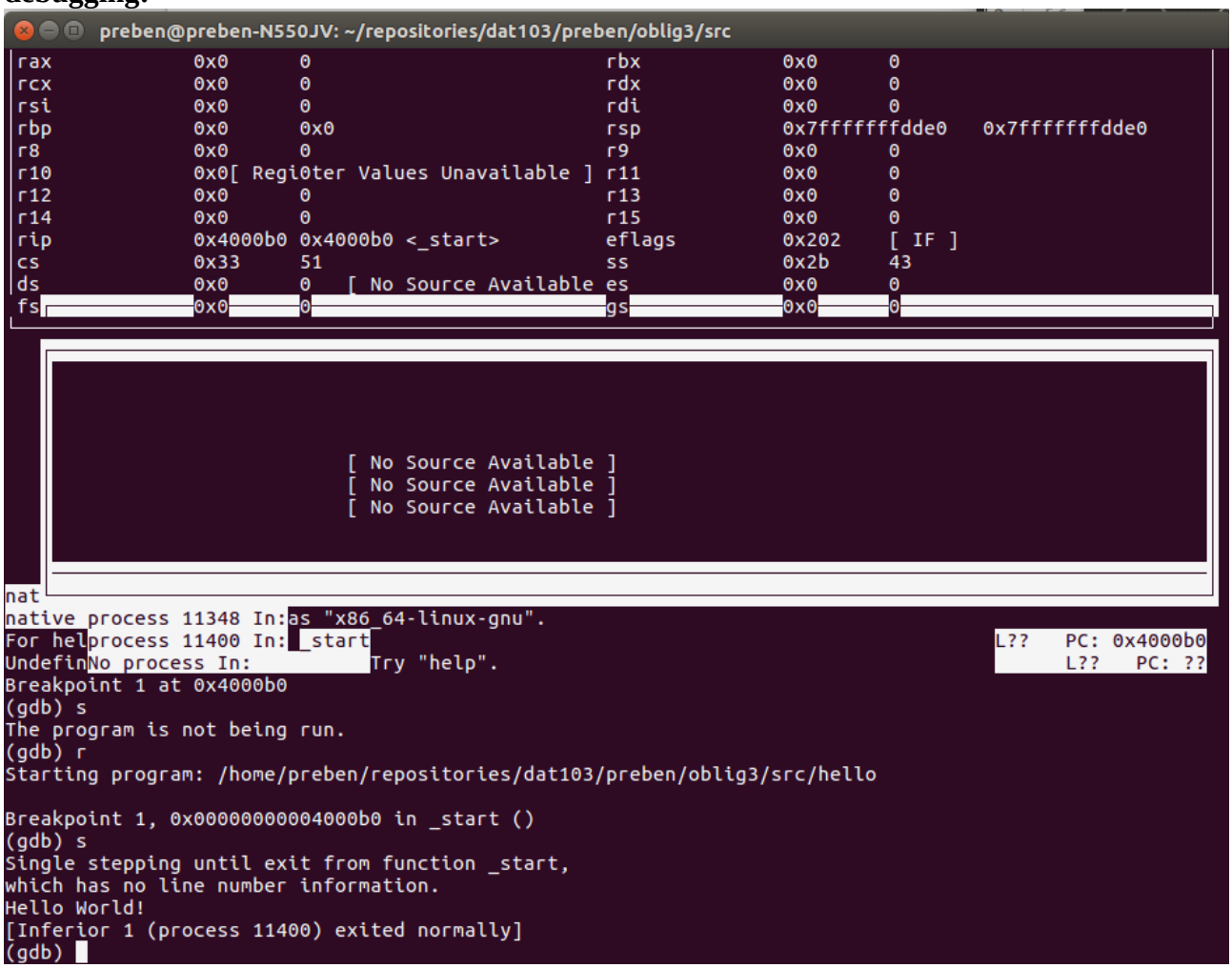
Obligatorisk oppgave nr 3 DAT103

Oppg 1

kildekode:

hello.asm
hello.o
hello

debugging:



opsjoner med forklaring:
-f - lar oss gi et bestemt output-format
elf – er kallet på format ELF32
-F – spesifiserer debug formatet
dwarf – debug er output formatet
-g – generer debug informasjon
hello.asm - er navnet på input filen

–m – Emulerer emulerings linkeren.
elf_i386 – Type emulering som skal kjøres.
–o hello – bruker hello som navn til programmet

hello.o – navnet på input filen.

Oppg. 2

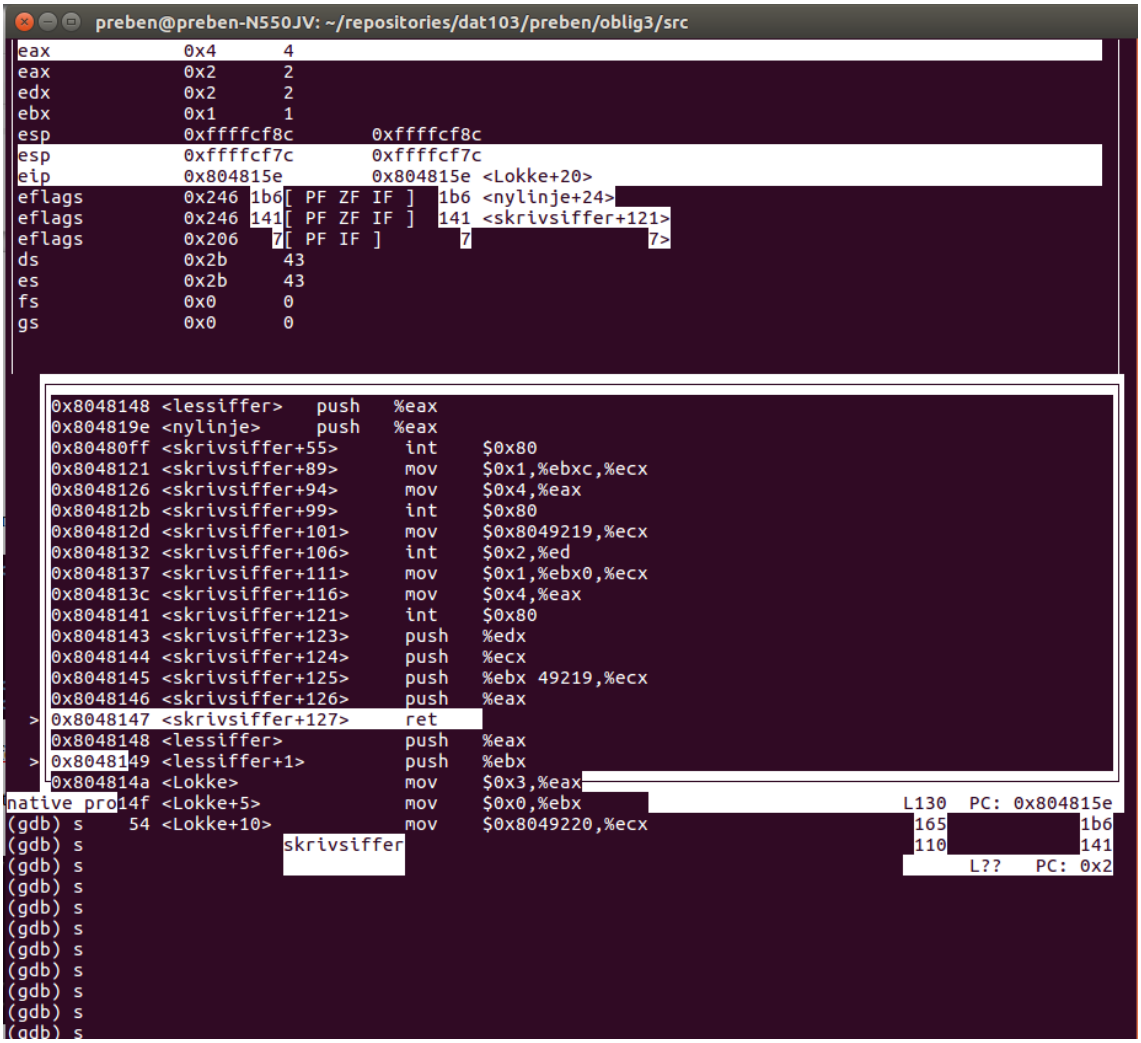
Kopierte koden fra pdf filen til filen summer2.asm og kompilerte og kjørte denne:

```
preben@preben-N550JV:~/repositories/dat103/preben/oblig3/src$ ./summer2
Skriv to ensifrede tall skilt med mellomrom.
Summen av tallene maa vaere mindre enn 10.
3 4

7Segmentation fault (core dumped)
preben@preben-N550JV:~/repositories/dat103/preben/oblig3/src$
preben@preben-N550JV:~/repositories/dat103/preben/oblig3/src$
```

Endret deretter koden i filen summer.asm og brukte div for å ta vare på tiersiffer og enersiffer til summen av det brukeren putter inn i programmet. Ener og tiersiffer blir lagret i eax og edx. Jeg skrev disse ut i riktig rekkefølge etter hverandre samt en tekst som forklarer litt bedre. Resultatet av kjøringen ble som følger:

```
preben@preben-N550JV:~/repositories/dat103/preben/oblig3/src$ ./summer
Skriv to ensifrede tall skilt med mellomrom.
7 5

Summen av tallene ble: 12
Segmentation fault (core dumped)
preben@preben-N550JV:~/repositories/dat103/preben/oblig3/src$
preben@preben-N550JV:~/repositories/dat103/preben/oblig3/src$
```

debugging:


preben@preben-N550JV: ~/repositories/dat103/preben/oblig3/src
eax 0x4 4
eax 0x2 2
edx 0x2 2
ebx 0x1 1
esp 0xffffcf8c 0xffffcf8c
esp 0xffffcf7c 0xffffcf7c
eip 0x804815e 0x804815e <Lokke+20>
eflags 0x246 1b6[PF ZF IF] 1b6 <nylinje+24>
eflags 0x246 141[PF ZF IF] 141 <skrivsiffer+121>
eflags 0x206 7[PF IF] 7 7>
ds 0x2b 43
es 0x2b 43
fs 0x0 0
gs 0x0 0
0x8048148 <lessiffer> push %eax
0x804819e <nylinje> push %eax
0x80480ff <skrivsiffer+55> int $0x80
0x8048121 <skrivsiffer+89> mov $0x1,%ebxc,%ecx
0x8048126 <skrivsiffer+94> mov $0x4,%eax
0x804812b <skrivsiffer+99> int $0x80
0x804812d <skrivsiffer+101> mov $0x8049219,%ecx
0x8048132 <skrivsiffer+106> int $0x2,%ed
0x8048137 <skrivsiffer+111> mov $0x1,%ebx0,%ecx
0x804813c <skrivsiffer+116> mov $0x4,%eax
0x8048141 <skrivsiffer+121> int $0x80
0x8048143 <skrivsiffer+123> push %edx
0x8048144 <skrivsiffer+124> push %ecx
0x8048145 <skrivsiffer+125> push %ebx 49219,%ecx
0x8048146 <skrivsiffer+126> push %eax
> 0x8048147 <skrivsiffer+127> ret
0x8048148 <lessiffer> push %eax
> 0x8048149 <lessiffer+1> push %ebx
0x804814a <Lokke> mov $0x3,%eax
native pro14f <Lokke+5> mov $0x0,%ebx L130 PC: 0x804815e
(gdb) s 54 <Lokke+10> mov $0x8049220,%ecx 165 1b6
(gdb) s skrivsiffer 110 141
(gdb) s L?? PC: 0x2
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s

Oppg. 3

Lagde en fil som heter java.asm og kompilerte og kjørte den som over:

```
preben@preben-N550JV:~/repositories/dat103/preben/oblig3/src$ nasm -f elf -F dwarf -g java2.asm
preben@preben-N550JV:~/repositories/dat103/preben/oblig3/src$ ld -m elf_i386 -o java2 java2.o
preben@preben-N550JV:~/repositories/dat103/preben/oblig3/src$ ./java2
Kjoerer loop
Loop ferdig, resultat av a:
0
preben@preben-N550JV:~/repositories/dat103/preben/oblig3/src$
```

debugging etter første runde av loop:

```
Register group: general
eax            0x1        1
ecx            0x13       19
edx            0xe        14
ebx            0x1        1
esp            0xffffcfa0       0xffffcfa0
ebp            0x0        0x0
esi            0x0        0
edi            0x0        0
eip            0x80480a8        0x80480a8 <looper+3>
eflags         0x293      [ CF AF SF IF ]
cs             0x23       35
ss             0x2b       43
ds             0x2b       43
es             0x2b       43
fs             0x0        0
gs             0x0        0

   0x8048080 <_start>       mov    $0xe,%edx
   0x8048085 <_start+5>     mov    $0x8049110,%ecx
   0x804808a <_start+10>    mov    $0x1,%ebx
   0x804808f <_start+15>    mov    $0x4,%eax
   0x8048094 <_start+20>    int    $0x80
   0x8048096 <_start+22>    mov    $0x14,%ecx
   0x804809b <_start+27>    mov    $0x0,%ebx
   0x80480a0 <_start+32>    mov    $0x0,%eax
B+ 0x80480a5 <looper>       cmp    $0xa,%ebx
 > 0x80480a8 <looper+3>     jl     0x80480b1 <increment>
   0x80480aa <looper+5>     jge    0x80480b4 <decrement>
   0x80480ac <loopcont>     inc    %ebx
   0x80480ad <loopcont+1>   loop   0x80480a5 <looper>
   0x80480af <loopcont+3>   jmp    0x80480b7 <printres>
   0x80480b1 <increment>    inc    %eax
   0x80480b2 <increment+1>  jmp    0x80480ac <loopcont>
   0x80480b4 <decrement>    dec    %eax
   0x80480b5 <decrement+1>  jmp    0x80480ac <loopcont>

native process 27543 In: looper                                   L39   PC: 0x80480a8
(gdb) b looper
Breakpoint 1 at 0x80480a5: file java2.asm, line 38.
(gdb) r
Starting program: /home/preben/repositories/dat103/preben/oblig3/src/java2
Kjoerer loop
Breakpoint 1, looper () at java2.asm:38
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s

Breakpoint 1, looper () at java2.asm:38
(gdb) s
(gdb)
```

Preben Bucher-Johannessen 06.10.17

Forklaring av fil: Lagde en loop som går igjennom fra ecx 20 til 0 og sammenligner for hver gang om ebx er mindre eller større eller lik 10. hvis mindre hopper programmet til inkrementerings delen og hopper så tilbake. Hvis større hopper den til dekrementeringsdelen.

Etter at loopen er ferdig hopper programmet til utskriftdelen som skriver ut resultatet av loopen som ASCII.