­**Oblig 2 – DAT103**

**Oppg. 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| S | **0** | **1** | **1** | **1** | **0** | **1** | **1** | **0** |
| R | **1** | **0** | **1** | **0** | **1** | **1** | **0** | **0** |
| Q | **1** | **0** | **0** | **0** | **1** | **1** | **0** | **-** |

**Oppg. 2**

**1)**

**(a) Immediate (Umiddelbar)**

Operanden er innbakt i instruksjonen. Dersom operanden er kjent (en konstant) når programmet lages, kan verdien av denne legges inn i selve instruksjonen.

Ingen hukommelsetilganger bortsett fra å hente instruksjonen.

**(b) Direct (Direkte)**

Instruksjonen angir adressen til operand i RAM.

Trenger bare 1 hukommelsetilgang.

**(c) PC relative (PC relativ)**

Instruksjon har nummer på register som inneholder adresse til operand i RAM.

Lagrer addresse bits i instruksjonen hvis flest hukommelsestilganger er relativt nære til instruksjonen som skal bli utført. Så det er relativt.

**(d) Register**

Instruksjon har nummer på register som inneholder operand.

Trenger ingen hukommelsetilgang.

**2)­­**

|  |  |  |
| --- | --- | --- |
| Adresseringsmoduser | Eff. Adresse | Operand |
| Immediate 200 | 100 | 200 |
| Direct 200 | 200 | 400 |
| Base-register 200 | 200 | 99 |
| PC-relative 200 | 301 | 901 |
| Indirect 200 | 400 | 1000 |
| Register RR | - | 199 |
| Register indirect RR | 199 | 1000 |

**Oppg. 3**

**1)**

**Kildekode i hello.asm:**

; Program som skriver ut Hello World

; Konstanter

cr equ 13 ; Vognretur

lf equ 10 ; Linjeskift

section .data ; Datasegment

melding db 'Hello World!',cr,lf

lengde equ $ - melding

section .text ; Kodesegment

global \_start

\_start:

mov edx,lengde

mov ecx,melding

mov ebx,1

mov eax,4 ; sys\_write

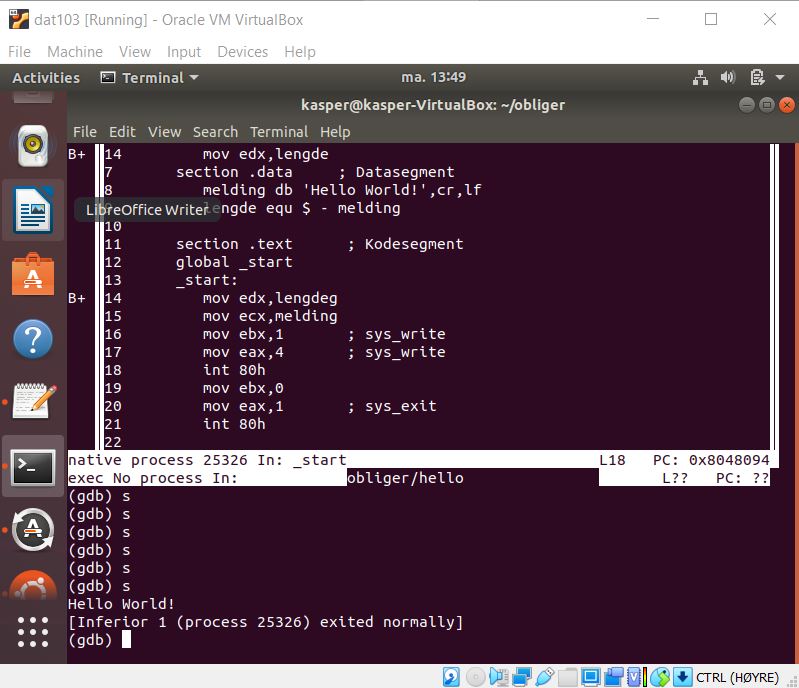
int 80h

mov ebx,0

mov eax,1 ; sys\_exit

int 80h

**Bilde av kjøring:**

****

**Forklaring av opsjoner:**

**nasm -f elf -F dwarf -g hello.asm:**

«-f elf» spesifiserer formatet til utdatafilen, som her er elf. «-F dwarf» bestemmer debuggingformatet som her er dwarf , og «-g» vil få nasm til å genere debug informasjon.

**ld -m elf\_i386 -o hello hello.o:**

«-m elf» vil emulere linken elf\_i386. «-o hello» vil bestemme navnet på outputfilen

**«gdb -tui hello»**:

vil laste opp informasjonen i hello i tui, som er et user interface for debugging

**2)**

**Ferdig kildekode (.asm fil ligger også vedlagt):**

; Konstanter

cr equ 13 ; Vognretur

lf equ 10 ; Linjeskift

SYS\_EXIT equ 1

SYS\_READ equ 3

SYS\_WRITE equ 4

STDIN equ 0

STDOUT equ 1

STDERR equ 2

; Datasegment

section .bss

siffer resb 4

; Datasegment

section .data

meld db "Skriv to ensifrede tall skilt med mellomrom.",cr,lf

db "Summen av tallene maa vaere mindre enn 10.",cr,lf

meldlen equ $ - meld

feilmeld db cr,lf, "Skriv kun sifre!",cr,lf

feillen equ $ - feilmeld

crlf db cr,lf

crlflen equ $ - crlf

; Kodesegment med program

section .text

global \_start

\_start:

mov edx,meldlen

mov ecx,meld

mov ebx,STDOUT

mov eax,SYS\_WRITE

int 80h

; Les tall, innlest tall returneres i ecx

; Vellykket retur dersom edx=0

call lessiffer

cmp edx,0 ; Test om vellykket innlesning

jne Slutt ; Hopp tilavslutning ved feil i innlesing

mov eax,ecx ; Første tall/siffer lagres i reg eax

call lessiffer

; Les andre tall/siffer

; vellykket: edx=0, tall i ecx

cmp edx,0 ;Test om vellykket innlesning

jne Slutt

mov ebx,ecx ; andre tall/siffer lagres i reg ebx

call nylinje

add eax,ebx

mov ecx,eax

call skrivsiffer ; Skriv ut verdi i ecx som ensifret tall

Slutt:

mov eax,SYS\_EXIT

mov ebx,0

int 80h

; ---------------------------------------------------------

skrivsiffer:

; Skriver ut sifferet lagret i ecx.

mov al, cl

aam

add ax, 3030h

mov [siffer], ah

mov [siffer + 1], al

mov ecx, siffer

mov edx, 2

mov ebx,STDOUT

mov eax,SYS\_WRITE

int 80h

mov eax,SYS\_EXIT

mov bl, al

int 80h

; ---------------------------------------------------------

lessiffer:

; Leter forbi alle blanke til neste ikke-blank

; Neste ikke-blank returneres i ecx

push eax

push ebx

Lokke:

; Leser et tegn fra tastaturet

mov eax,SYS\_READ

mov ebx,STDIN

mov ecx,siffer

mov edx,1

int 80h

mov ecx,[siffer]

cmp ecx,' '

je Lokke

cmp ecx,'0'; Sjekk at tast er i område 0-9

jb Feil

cmp ecx,'9'

ja Feil

sub ecx,'0'; Konverter ascii til tall.

mov edx,0 ; signaliser vellykket innlesning

pop ebx

pop eax

ret ; Vellykket retur

Feil:

mov edx,feillen

mov ecx,feilmeld

mov ebx,STDERR

mov eax,SYS\_WRITE

int 80h

mov edx,1 ; Signaliser mislykket innlesning av tall

pop ebx

pop eax

ret ; Mislykket retur

; ---------------------------------------------------------

; Flytt cursor helt til venstre på neste linje

nylinje:

push eax

push ebx

push ecx

push edx

mov edx,crlflen

mov ecx,crlf

mov ebx,STDOUT

mov eax,SYS\_WRITE

int 80h

pop edx

pop ecx

pop ebx

pop eax

ret

; End \_start

**3)**

; konstanter

SYS\_EXIT equ 1

SYS\_READ equ 3

SYS\_WRITE equ 4

STDIN equ 0

STDOUT equ 1

STDERR equ 2

section .bss

num resb 10

section .text

global \_start

\_start:

mov ecx, 0

loop1:

inc ecx

cmp ecx, 10

jl loop1

je loop2

loop2:

dec ecx

cmp ecx, 0

jg loop2

je Finished

skrivsiffer:

; Skriver ut sifferet lagret i ecx. Ingen sjekk på verdiområde.

push eax

push ebx

push ecx

push edx

add ecx,'0'; converter tall til ascii.

mov [num],ecx

mov ecx,num

mov edx,1

mov ebx,STDOUT

mov eax,SYS\_WRITE

int 80h

pop edx

pop ecx

pop ebx

pop eax

ret

Finished:

call skrivsiffer

mov eax,1

mov ebx,0

int 80h