Å løse eksamensoppgaver

- Programmeringsoppgaver
- Flervalgsoppgaver
- Prøveeksamen
- Eksamen

Programmeringsoppgaver

Vil typisk være å skrive en funksjon eller oversette en funksjon fra C til assemblerkode.

Hvorfor gi slike oppgaver?

- Se om studenten har forstått assemblerprogrammering.
- La studenten velge hvordan han/hun vil løse oppgaven.

Hint

Du kan programmere ganske fritt så lenge koden gjør det den skal.

Eksempel: `for (i=1; i<=n; ++i) ...`
kan godt oversettes til

```
      movl  ⟨n⟩,%ECX
Lab1: ...
      loop  Lab1
```

Det vesentlig er å overbevise sensor om at du kan dette.

Eksamensoppgaven

Funksjonen **`strcpy`** kopierer én tekst til et angitt sted. Du skal skrive funksjonen

```
char *multistrcpy (char *res, ...)
```

som kan kopiere vilkårlig mange tekster til `res`; etter siste tekst kommer parameteren 0. Funksjonen skal skrives i x86-assemblerspråk.

Standard start og slutt

```
        .globl  multistrcpy
# Navn:         multistrcpy.
# Synopsis:     En strcpy for mange tekster.
# C-signatur:   char *multistrcpy (char *res, ...).
# Registre:     %AL  - tegnet som skal kopieres
#               %EDX - peker på neste parameter
#               %ESI - hvor det skal kopieres fra
#               %EDI - res, der resultatet havner

multistrcpy:
        pushl   %ebp            # Standard
        movl    %esp,%ebp       # funksjonsstart
        pushl   %edi            # Gjem unna %EDI
        pushl   %esi            # og %ESI.

        movl    8(%ebp),%edi    # Hent parameter res.
        leal    12(%ebp),%edx   # Sett %EDX til å peke på parameter nr 2.
        cld                     # La adressene øke under kopering.
          :
          :

msc_x:  movb    $0,(%edi)       # Avslutt res med 0-byte.
        movl    8(%ebp),%eax    # Angi svarverdien.
        popl    %esi            # Hent tilbake %ESI
        popl    %edi            # og %EDI.
        popl    %ebp            # Standard
        ret                     # retur.
```

Ytre løkke

```
        # Ytre løkke: løper gjennom alle paramtrene.
msc_l:  cmpl    $0,(%edx)       # Hvis neste parameter er 0,
        je      msc_x           # er vi ferdig.

        movl    (%edx),%esi     # Hent neste parameter, og
        addl    $4,%edx         # legg den i %ESI.
          ⋮
```

Indre løkke

```
        # Indre løkke: kopierer alle tegnene.
msc_c:  cmpb    $0,(%esi)       # Hvis neste tegn er 0,
        je      msc_l           # er kopieringen ferdig,
                                # og vi kan gjenta ytre løkke.
        movsb                   # Flytt ett tegn (og oppdater %ESI og %EDI).
        jmp     msc_c           # Gjenta indre løkke.
```

Flervalgsoppgaver

Her vil du få oppgitt små kodesnutter i C og assembler med spørsmål om hva som blir resultatet.

Hvorfor gi slike oppgaver?

- Studenten skal vise forståelse for både C-kode og assemblerinstruksjoner.
- Studenten skal få slippe å skrive seg i hjel.

Hint

Vær ekstremt nøye med å lese koden!

Hensikten med oppgavene er ikke å «lure» studenten, men å sjekke detaljforståelse.

Hint

Bruk hjelpemidlene!

Slå opp instruksjonene om du er i den minste tvil.

Hint

Ikke vær redd for å oppgi flere svaralternativer eller for å gjette!

Du kan ikke få verre enn blankt svar.

Flervalgsoppgavene

```
extern unsigned int f1 (unsigned int a, unsigned int b, unsigned int c);

int main (void)
{
  unsigned int v = f1(0x12345678, 0x000000ff, 0x0000ff00);

  printf("Test 1: 0x%08x\n", v);
}
```

```
        .globl  f1
f1:     pushl   %ebp
        movl    %esp,%ebp

        movl    8(%ebp),%eax
        andl    12(%ebp),%eax
        orl     16(%ebp),%eax

        popl    %ebp
        ret
```

1. Test 1: 0x00000000
2. Test 1: 0x0000ff00
3. Test 1: 0x0000ff78
4. Test 1: 0x0000ffff
5. Test 1: 0x1234ff78
6. Det vil oppstå en feil («exception»)
7. Noe annet enn alternativene over

Flervalgsoppgavene

```
extern int f2 (int a[], int n);

int data[6] = { 33, -2, 49, -88, 0, 13 };

int main (void)
{
  int v = f2(data, 6);

  printf("Test 2: %d\n", v);
}
```

```
        .globl  f2
f2:     pushl   %ebp
        movl    %esp,%ebp

        movl    8(%ebp),%edx
        movl    12(%ebp),%ecx

        movl    (%edx),%eax
f2_l:   decl    %ecx
        jz      f2_x
        addl    $4,%edx
        cmpl    (%edx),%eax
        jge     f2_l
        movl    (%edx),%eax
        jmp     f2_l

f2_x:   popl    %ebp
        ret
```

1. Test 2: -88
2. Test 2: 0
3. Test 2: 13
4. Test 2: 33
5. Test 2: 49
6. Det vil oppstå en feil («exception»)
7. Noe annet enn alternativene over

Flervalgsoppgavene

```
extern int f3 (char *a);

char *txt = "En bitte liten test!";

int main (void)
{
  int v = f3(txt+9);

  printf("Test 3: %d\n", v);
}
```

```
        .globl  f3
f3:     pushl   %ebp
        movl    %esp,%ebp
        pushl   %edi

        movl    8(%ebp),%edi
        movl    $0xffffffff,%ecx
        cld
        movb    $' ',%al

        repnz scasb

        movl    $-2,%eax
        subl    %ecx,%eax

        popl    %edi
        popl    %ebp
        ret
```

1. Test 3: -1
2. Test 3: 0
3. Test 3: 2
4. Test 3: 5
5. Test 3: 20
6. Det vil oppstå en feil («exception»)
7. Noe annet enn alternativene over

Flervalgsoppgavene

```
extern int f4 (int a, int b);

int main (void)
{
  int v = f4(4,1);
  printf("Test 4: %d\n", v);
}
```

```
        .globl  f4
f4:     pushl   %ebp
        movl    %esp,%ebp

        movl    8(%ebp),%edx
        pushl   %edx
        movl    12(%ebp),%edx
        pushl   %edx
        call    sub
        popl    %edx
        popl    %edx

        popl    %ebp
        ret

sub:    pushl   %ebp
        movl    %esp,%ebp

        movl    8(%ebp),%eax
        subl    12(%ebp),%eax

        popl    %ebp
        ret
```

1. Test 4: -3
2. Test 4: 0
3. Test 4: 3
4. Test 4: 4
5. Test 4: 5
6. Det vil oppstå en feil («exception»)
7. Noe annet enn alternativene over

Flervalgsoppgavene

```
extern float f5 (float a, float b, float c);

int main (void)
{
  float v = f5(1.0, 2.0, -1.0);

  printf("Test 5: %f\n", v);
}
```

```
        .globl  f5
f5:     pushl   %ebp
        movl    %esp,%ebp

        flds    8(%ebp)
        flds    12(%ebp)
        flds    16(%ebp)
        faddp
        fsqrt
        faddp

        popl    %ebp
        ret
```

1. Test 5: 0.000000
2. Test 5: 0.732051
3. Test 5: 1.000000
4. Test 5: 1.414214
5. Test 5: 2.000000
6. Det vil oppstå en feil («exception»)
7. Noe annet enn alternativene over

Flervalgsoppgavene

```
extern int f6 (int a[], int n);

int data[6] = { 2, 3, 5, 7, 11, 13 };

int main (void)
{
  int v = f6(data,6);

  printf("Test 6: %d\n", v);
}
```

```
        .globl  f6
f6:     pushl   %ebp
        movl    %esp,%ebp

        movl    8(%ebp),%edx
        movl    12(%ebp),%ecx
        movl    $0,%eax

f6_l:   subl    $1,%ecx
        js      f6_x
        addl    (%edx),%eax
        incl    %edx
        jmp     f6_l

f6_x:   popl    %ebp
        ret
```

1. Test 6: 0
2. Test 6: 6
3. Test 6: 13
4. Test 6: 28
5. Test 6: 41
6. Det vil oppstå en feil («exception»)
7. Noe annet enn alternativene over

Frivillig prøveeksamen

- Tid: Tirsdag 20.5 kl 10.15–12.
- Sted: Oppmøte i Store auditorium
- Formål:
  1. Teste programvaren
  2. Teste egne kunnskaper

Eksamen

Eksamen er **11. juni 2014** klokken 14.30–18.30.

Sted

Eksamen vil finne sted i **Ole-Johan Dahls hus**. Følge med på StudentWeb for å få vite hvilke rom.

Hjelpemidler

- *Alle* trykte og skrevne hjelpemidler er tillatt.
- Batteridrevet lommekalkulator er lov — og kan være nyttig.
- Selv om eksamen skjer på datamaskin, har dere ikke tilgang til annen programvare enn den sikre nettleseren.
- I tillegg til å svare på datamaskinen kan dere levere svar (med diagrammer etc) på papir.

Sjekk godkjenningslister

- Besøk **//devilry.ifi.uio.no/** og sjekk godkjentlisten for kurset for å se din status.
- Ta kontakt med gruppelærer (eller meg) hvis du mener det er gjort feil.
- I siste instans: Møt opp på eksamensdagen og krev å få avlegge eksamen.

Problemer under eksamen?

Faglærerne vil være til stede eksamenslokalene hele tiden. Da er det anledning til å stille spørsmål om *alt*.

Bruk tiden riktig

Det er mye bedre å svare litt på alle spørsmålene enn perfekt på noen og intet på andre.

Forslag: Bruk så mye av tiden som oppgaven teller (dvs bruk 25% av tiden på en oppgave som teller 25%).