Obligatorisk oppgave nr 3 i INF2270 våren 2015

Frist

Fristen er satt til: **mandag 11. mai 2015 kl 12.00**. Oppgaven skal leveres via Devilry.

Funksjonen *sprintf*

Standardfunksjonen **sprintf** i C fungerer som **printf** men resultatet havner i en tekstvariabel i stedet for å bli skrevet ut. For eksempel vil

```
char str[200];
int  x;

x = 4;
sprintf(str, "Her er %d %s.", x, "siffer");
```

resultere i at `str` inneholder «**Her er 4 siffer.**» (samt en 0-byte som avslutning). Les mer i *man sprintf* om hvorledes **sprintf** fungerer; ikke glem at **sprintf** også returnerer en verdi.

Oppgaven er å skrive funksjonen **sprinter** som er en forenklet utgave av **sprintf**. Nærmere bestemt kan vi anta følgende forenklinger:

- De eneste %-spesifikasjonene som kan forekomme, er **%c**, **%d**, **%f**, **%s**, **%x** og **%%**.
- Det er ingen breddeangivelse (som **%12d**) og heller ingen modifikasjoner av %-spesifikasjonene (som **+**, **-**, **0** eller **l** (for «long»)).
- Signaturen er

  ```
  int sprinter(unsigned char *res, unsigned char *format, ...);
  ```

  Funksjonen kan altså ha vilkårlig mange parametre, men alltid minst to.
- Hvis formatet inneholder ulovlige spesifikasjoner som **%a** eller **%22d**, kan du selv velge hva resultatet skal være, men funksjonen skal ikke gi kjørefeil.

Oppgaven

Oppgaven er å skrive funksjonen ***sprinter*** i x86-assemblerspråk. Den skal løses individuelt, så vi forventer at alle innleverte

løsninger er forskjellige. Det er heller ikke lov å bruke en kompilator til å generere koden. Ytterligere regler finnes i /studier/admin/obligatoriske-aktiviteter/mn-ifi-oblig.html som forutsettes lest av alle som skal levere obligatoriske oppgaver i INF2270.

Koden skal skrives slik at den kan assembleres med kommandoen **gcc -m32** på Ifis Linux-maskiner.

Legg vekt på oversiktlig programmering og gode kommentarer! Gruppelæreren kan nekte å rette besvarelsen hvis det er for vanskelig å forstå hvorledes funksjonen virker.

Det er lov å la **sprinter** kalle andre funksjoner, men da skal de også skrives i x86-assemblerspråk og legges ved. De eneste unntaketene er at de er lov å bruke funksjonene **my_ftoa** (se ~inf2270/programmer/Oblig-3/my_ftoa.c) for å konvertere et flyt-tall til en tekststreng og **add_int** i samme fil.

Det finnes et testprogram ~inf2270/programmer/Oblig-3/test-sprinter.c som gruppelærerne vil bruke under rettingen; det kan være lurt å prøve det selv.

Er du i tvil om noe i oppgaven, så spør gruppelæreren eller meg.

Hint

- Skriv aller først noen C-programmer som bruker den originale **sprintf** slik at du blir sikker på hvorledes **sprintf** virker.
- Oppgaven skal fungere for tegn som er kodet i **ISO 8859-1** (også kjent som **Latin-1**). Vær sikker på at alle filene bruker dette tegnsettet. Du kan sjekke tegnsettet med

  ```
  $ file -i oblig3.html
  oblig3.html: text/html; charset=iso-8859-1
  ```

  Hvis tegnsettet er noe annet enn «ascii» eller «iso-8859-1», må man justere oppsettet i redigeringsprogrammet sitt eller fikse filen, for eksempel slik:

  ```
  $ iconv -f UTF-8 -t LATIN1 min-utf-8-fil >min-latin1-fil
  ```

- Før du begynner å skrive assemblerkode, bør du løse problemet i et høynivåspråk du kjenner godt (for eksempel C) slik at du vet nøyaktig hva som skal gjøres.
- De færreste klarer å skrive en slik funksjon i ett uten testing. Start heller med en implementasjon av **strcpy** og utvid den gradvis til å kunne håndtere **%%**, så **%c**, **%s**, **%f** og til sist **%x** og **%d** (som er vanskeligst).
- **%f** skriver ut et flyt-tall, nærmere bestemt en 64-bits **double**. De tar altså dobbelt så stor plass som **int**-verdier.
- Siden det eneste du skal gjøre med flyt-tallsverdiene er å overføre dem som parameter til **my_ftoa**, trenger du ikke bruke noen flyt-tallsoperasjoner (men du kan gjerne benytte deg av dem om du foretrekker det).
- Let etter ideer i ukeoppgavene som har vært gitt og fasit til disse.

Lykke til!

4. april 2015

Dag Langmyhr