Lær assembly fra eksperten

Hvordan undersøke assemblykode lagd av GCC

Asbjørn Djupdal
ARM Norway, IDI NTNU

29. januar 2013

1 Introduksjon

Dette dokumentet beskriver hvordan man kan undersøke assemblykode generert av C-kompilatoren GCC. Dette kan være nyttig i startfasen når man lærer seg assemblyprogrammering, da GCC ofte vil produsere bra assemblykode.

2 Skaffe seg GCC

Har du en PC med Ubuntu (eller en annen GNU/Linux-variant) er det enkelt. For å installere GCC for Intel-prosessorer skriver du følgende kommando i en terminal:

```
apt-get install gcc
```

Denne vil kun kompilere til x86 assemblykode. Ønsker du i stedet å jobbe med assemblykode for en annen prosessor trenger du en *krysskompilator*. Krysskompilator for ARM kan du installere slik:

```
apt-get install gcc-arm-linux-gnueabi
```

Mange andre GCC krysskompilatorer finnes for ubuntu.

Det finnes GCC krysskompilatorer gratis for både Microsoft Windows og Apple OSX. Google f.eks "linaro toolchain binaries" eller "arm gcc toolchain".

3 Lage ARM assemblykode av C-filer

For å kompilere en C kildefil til ARM assembly må du benytte krysskompilatoren for ARM. Kompiler med følgende kommando:

```
arm-linux-gnueabi-gcc -O0 -S kildefil.c
```

Forklaring:

- `-S`: Du ønsker assemblyfil som resultat (den vil hete kildefil.s)
- `-O0`: Du ønsker optimaliseringsnivå 0 (ingen optimalisering). Her kan du velge `-O0`, `-O1`, `-O2` eller `-O3` hvor `-O3` er raskest, eller `-Os` som betyr mest størrelsesoptimal.

4 Eksempel

Lagre følgende fil som `eksempel.c`:

```
int a = 1;
int b = 2;

int x;

void test(void) {
  x = a + b;
}
```

Kompiler med følgende kommando:

```
arm-linux-gnueabi-gcc -O0 -S eksempel.c
```

Du får da resultatfilen `eksempel.s` (se vedlegg 1). Denne er full av rare assembler pseudoops. De fleste av disse (alle utenom `.word`) kan dere se bort fra. Skreller dere vekk alt "unødig" sitter dere igjen med følgende:

```
a:
  .word  1
b:
  .word  2

test:
  str  fp, [sp, #-4]!
  add  fp, sp, #0
  ldr  r3, .L2
  ldr  r2, [r3, #0]
  ldr  r3, .L2+4
  ldr  r3, [r3, #0]
  add  r2, r2, r3
```

```
  ldr  r3, .L2+8
  str  r2, [r3, #0]
  add  sp, fp, #0
  ldmfd  sp!, {fp}
  bx  lr

.L2:
  .word  a
  .word  b
  .word  x
```

5 Forslag

Lek dere litt med forskjellige typer småprogrammer og prøv å forstå hva som lages av assemblykode. Prøv både løkker og funksjoner. Undersøk hva som er effekten av å deklarere variabler inne i eller utenfor funksjoner, og effekten av nøkkelord som *static* og *volatile*.

Bruk google eller databladet til prosessoren for å slå opp hva instruksjonene gjør.

Prøv også forskjellige optimaliseringsnivåer, det kan være veldig lærerikt.

6 Binærfiler

Ønsker dere å undersøke selve binærkoden, altså resultatet etter at både C-kompilator og assembler har gjort jobben, kan dere gjøre det slik:

```
arm-linux-gnueabi-gcc -O0 -c eksempel.c
```

Denne lager en objektfil som heter `eksempel.o`. Undersøk objektfila på denne måten:

```
arm-linux-gnueabi-objdump -S eksempel.o
```

Hver linje vil her vise instruksjoner, både som ferdig assemblerte instruksjonsord (2. kolonne) og på assembly-format.

Vedlegg 1

```
  .arch armv5t
  .fpu softvfp
  .eabi_attribute 20, 1
  .eabi_attribute 21, 1
  .eabi_attribute 23, 3
  .eabi_attribute 24, 1
  .eabi_attribute 25, 1
  .eabi_attribute 26, 2
```

```
  .eabi_attribute 30, 6
  .eabi_attribute 34, 0
  .eabi_attribute 18, 4
  .file  "add.c"
  .global  a
  .data
  .align  2
  .type  a, %object
  .size  a, 4
a:
  .word  1
  .global  b
  .align  2
  .type  b, %object
  .size  b, 4
b:
  .word  2
  .comm  x,4,4
  .text
  .align  2
  .global  test
  .type  test, %function
test:
  @ args = 0, pretend = 0, frame = 0
  @ frame_needed = 1, uses_anonymous_args = 0
  @ link register save eliminated.
  str  fp, [sp, #-4]!
  add  fp, sp, #0
  ldr  r3, .L2
  ldr  r2, [r3, #0]
  ldr  r3, .L2+4
  ldr  r3, [r3, #0]
  add  r2, r2, r3
  ldr  r3, .L2+8
  str  r2, [r3, #0]
  add  sp, fp, #0
  ldmfd  sp!, {fp}
  bx  lr
.L3:
  .align  2
.L2:
  .word  a
  .word  b
  .word  x
  .size  test, .-test
  .ident  "GCC: (Ubuntu/Linaro 4.7.2-1ubuntu1) 4.7.2"
  .section  .note.GNU-stack,"",%progbits
```