Undersøk den produserte Assembler koden og sammenlign med din egen kildekode fra Del 2, er det noen store forskjeller mellom det du lagde og det GCC produserer?

* En av forskjellene mellom den selvgenererte assembler-koden og vår kildekode fra oppgave 2, er at det ikke er benyttet en loop som går beregner fibonacci-verdiene på samme måte som jeg har gjort i min.
* En vesentlig forskjell er størrelsen på koden, da kildekoden jeg skrev endte på 31 linjer istedenfor 66 hos den genererte koden.
* Den genererte kildekoden er også mye vanskeligere å forstå, da det er en rekke ulike funksjoner som ikke er hverken oversiktlige eller tydelige hva gjør. I min kildekode er det brukt enklere og færre «kommandoer» som gjør det enklere å skjønne hva som foregår.

Endre -Os til -O2, hvordan påvirker dette den produserte koden? Prøv å endre til -O3, ser du noen forandring nå?

* Endringen gjør koden mer optimalisert. Dersom vi endrer det til -O3 blir koden enda mer optimalisert, og vil bruke mindre tid på å returnere tallet fra fib.

Hvilke argumenter er det for og imot å bruke en kompilator sammenlignet med å skrive Assembler? Tenk spesielt på hvor mye jobb det ville være å oversette programsnuttene i Del 1, 2 og 3 til en annen prosessorarkitektur.

* En stor fordel ved å bruke en kompilator er at det tar vesentlig kortere tid å produsere c-kode enn å skrive Assembler. Spesielt med tanke på så mange ulike måter det er å skrive Assembler til ulike prosessorarkitekturer.
* En ulempe ved å bruke en kompilator er det at koden i mange tilfeller ofte blir lenger og mindre forståelig. Dette kan vi se ved å sammenlikne koden fra oppgave 2 med den kompilerte c-koden fra oppgave 3. Da ser vi at den kompilerte koden er både lenger og mindre oversiktlig, og ikke minst vanskeligere å forstå.