Lab 4

Thursday, 20 May 2021 13:11

Vad har jag lärt mig under denna laboration?

Denna gång var indirect call det nya! När man trycker på en knapp får man ut 1-5, och utifrån det skulle man köra en subrutin. Detta går ju inte riktigt att loopa på ett bra sätt tänkte jag. CPSE var väldigt nära det jag ville göra, compare, skip if equal. Fast jag ville göra compare, skip if not equal. Då hade man kunnat skriva ihop en snabb switch sats. Man skulle behöva många labels och repeterande kod. Jag kollade vidare och hittade "icall". Att kunna kalla på den delen där Z pekaren pekar. Efter lite exprimenerande lyckades jag hitta en bra lösning.

Först lagrar jag de 5 labels jag vill kalla på I flash minnet. På detta vis.

```
ACTION:
    .db     BACKLIGHT_TOGGLE
    .db     MOVE_CURSOR_LEFT
    .db     PREVIOUS_LETTER
    .db     NEXT_LETTER
    .db     MOVE_CURSOR_RIGHT
```

Det är väldigt viktigt att alla börjar på ny .db, så att de verkligen tar upp 2 hela bytes. Då en label är 2 bytes. Om inte så kommer de ibland bara vara en byte stor, vilket gör det inkonsekvent och svårt hitta rätt I minnet.

Konceptet för vad jag ville göra var att ha en lista med subrutiner och en pekare som pekar på dem, sedan ta denna peckare +x steg framåt. Denna funktion laddar Z pekaren på ACTION I flash och stegar fram med två steg för varje r16. Alltså om man laddar r16 med 2 kommer den att kalla på "MOVE_CURSOR_LEFT".

```
KEY_ACTION:
    setp    Z, (ACTION*2)
    dec         r16
    adc         ZL, r16  // Varje label är 2 bytes, så för varje r16 ska man lägga till 2, 2*r16, alltså r16+r16
    adc         ZL, r16
    lpm         r16, Z+
    lpm         r17, Z+
    mov         ZH, r17
    mov         ZL, r16
    icall
    ret
```

Subrutinen sätter pekaren I flash minnet där jag har sparat ett par labels. Sedan laddar man in värdet från flash minnet I r16 och r17, dessa två bildar ett 16 bitars tall som motsvarar en label. Sedan laddar man in pekaren med detta 16 bitars tall. Z pekaren pekar nu på den labeln I flash som den ska kalla. Sedan kör man icall som kallar på där Z pekaren står. På detta sätt slippar man compare massa i onödan. Man kör bara direkt den man vill istället, mycket smidigt.

Det bästa med denna subrutin är att den är så generell, man kan väldigt enkelt byta vilka subrutiner man vill kalla på. Den går även att expandera. Just nu om du laddar r16 med 6 blir det helt knas, men om du lägger till en till label i ACTION listan så löser det sig. Man skulle kunna ha 255 olika ACTIONS på detta viss. Det går även väldigt enkelt att modifiera koden till att istället läsa labelsen från SRAM. På detta sätt hade man kunnat byta ut funktioner av knappar "on the fly".

Sammanfattningsviss skulle jag säga att jag skapat en kompakt och effektiv switch-sats i assembler.

Vad var svårast/krångligast/mest tidskrävande?

Att få min switch sats att fungerar var ju lite små krånligt, mycket läsa I flash minnet hur labels beter sig när man lagrar dom. Annars gick labben väldigt effektivt frammåt.