Blekinge Tekniska Högskola

DV1464/DV1493 – Datorteknik

Laboration I - Rekursiv fakultetsberäknig i ARM assembler

Författare: Carina Nilsson, Erik Bergenholtz

1 Inledning

Syftet med den här laborationen är att stifta en första bekantskap med assemblerprogrammering, samt hur subrutiner (funktioner) kan skrivas och anropas i assembler. Här ges också tillfälle att grundligt lära sig hur en stack fungerar genom att implementera en rekursiv subrutin. Labbmiljön vi använder är en Raspberry Pi. Det finns även en ARM-simulator som kan användas om man vill sitta hemma och programmera, men inte har någon Pi. Det finns en länk till nedladdningsstället för den på It's. Laborationen består av tre delar, där bara den tredje delen behöver redovisas. Uppgiften görs lämpligen i grupper om två. Annan gruppstorlek ska beviljas av handledaren. Grupper med fler än tre deltagare godtas inte. Redovisningen sker genom att du visar upp programkörning och kod för handledaren. De två första delarna är till för uppvärmning. Tänk på att laborationerna i den här kursen kräver betydligt mer arbete än den tid som finns vid det handledda laborationstillfället.

2 Laborationsuppgifter

2.1 Del 1 - Kör ditt första assemblerprogram

Här finns ett litet kodexempel som kan skrivas in och köras som en första bekantskap med assemblerprogrammering. Stega igenom programmet i debugger eller simulator.

```
.data
numbers:
    .word       2, 3, 8, 3, 9, 12, 0
sum:
    .word       0

.text
.global main
main:
    LDR r1, =numbers
    MOV r0, #0
again:
    LDR r2, [r1]
    CMP r2, #0
    BEQ finish
    ADD r0, r0, r2
    ADD r1, r1, #4
    BAL again
finish:
    LDR r1, =sum
    STR r0, [r1]
halt:
    BAL halt
.end
```

2.2 Del 2 - Skriv en funktion i assembler

Det här programexemplet innehåller en subrutin också.
Huvudprogrammet gör ett ”anrop” (hopp) till subrutinen i koden. Det ger ett bra tillfälle att träna på att använda stacken lite grann. Subrutinen `findMax` gör egentligen ingenting ännu. Komplettera funktionens kod så att den returnerar det största värdet i den array av heltal, som avslutas med talet noll, vars adress skickas in som parameter till funktionen. I programmet som anropar funktionen används arrayen `test` som argument till funktionsanropet.

```
.data

test:
        .word 1
        .word 3
        .word 5
        .word 7
        .word 9
        .word 8
        .word 6
        .word 4
        .word 2
        .word 0

textA: .asciz "Lab1, Assignment 2\n"
textB: .asciz "The max is %d\n"
textC: .asciz "Done\n"

.text
.global main
.extern printf

/*****************************************************************
  Function finding maximum value in a zero terminated integer array
*****************************************************************/
findMax:
    PUSH    {lr}

/* Add code to find maximum value element here! */
/* Any registers altered by the function beside r0-r3 must be preserved */

    POP     {pc}
```

```
/**********************
 main function
**********************/
main:
    PUSH    {lr}
    LDR     r0, =textA
    BL      printf
    LDR     r0, =test
    BL      findMax
    MOV     r1, r0
    LDR     r0, =textB
    BL      printf
    LDR     r0, =textC
    BL      printf
    POP     {pc}

.end
```

2.3 Del 3 - Rekursiv assemblerfunktion för fakultetsberäkning

Till den här uppgiften ges ingen hjälpkod.
Skriv en rekursiv subrutin i ARM-assembler som räknar ut fakultet på ett tal:

$$n! = n \cdot (n-1) \cdot \cdots \cdot 2 \cdot 1 = n \cdot (n-1)!$$

En sådan subrutin kan i ett högnivåspråk se ut så här:

```
1  int factorial(int number)
2  {
3      if (number > 1)
4          return(number * factorial(number - 1));
5      else
6          return(1);
7  }
```

Tänk på hur parametrar och stack ska hanteras så att rekursion blir möjlig.
Beräkningsfunktionen ska inte göra några utskrifter!

Skriv också ett huvudprogram med en loop som anropar den här funktionen och presenterar resultaten av fakultetsberäkning för talen n = 1,2,3,4,5,6,7,8,9,10.