Univerzitet u Sarajevu

Elektrotehnički fakultet
Odsjek za automatiku i elektroniku

Laboratorijska vjezba 5

GNU ARM asembler

Bazdar Faris
Predmet: Ugradbeni sistemi
Akademska godina: 2019/2020

Sarajevo, maj 2020.

1 Zadatak 1

```
.data
        n: .word 47 @duzina do koje idemo sa fibonacijem
        prvi: .word 0 @ pomocne varijable za racunanje
        drugi: .word 1
        sljedeci: .word 0
        .align 4
        a: .skip 188 @alokacija kontinualna memorije
.text
.global _start
_start:

        ldr r6, =a
        ldr r0, =prvi @ucitavanje adrese u registar
        ldr r1, =drugi
        ldr r0, [r0] @ucitavanje vrijednosti iz adrese u
           registar
        ldr r1, [r1]

        mov r2, #1
        ldr r3, =n
        ldr r3, [r3]
        ldr r5, =sljedeci @ vrijednost koja se upisuje u niz
        ldr r5, [r5]

        loop:
                add r5, r0, r1 @dodajemo r0+r1 u r5
                mov r0, r1 @mijenjamo r0 sa r1
                mov r1, r5 @mijenjamo r1 sa r5

                str r5, [r6] @skladistimo vrijednost u niz
                add r6, r6, #4 @niz je tipa int koji je velicine
                    4 bajta pa prolazimo kroz niz dodavajuci 4

                adds r2, r2, #1 @povecavamo brojac
                cmp r2, r3 @poredimo brojac
                blt loop @ukoliko je vrijednost brojaca manja od
                    @maximuma do kojeg idemo onda se vracamo na
                   @pocetak loop labele
```

Opis koda: Ukoliko je vrijednost do koje treba ici 48 dodje do overflowa zbog velicine registra, te se pokaze nepravilna vrijednost.

2 Zadatak 2

```
        .data
string:
        .asciz "        " @moguce je proizvoljno postaviti duzinu
            teksta
        .text
.global _start
_start:

        mov r7, #3
        mov r0, #0
        ldr r1, =string @ovaj blok koda sluzi za unos stringa
        mov r2, #7 @broj karaktera koji se unose ukljucujuci
           prostor za nul terminirajuci znak(ili enter)
        swi #0

        mov r7, #0
        mov r0, #4 @konstanta koja odredjuje dokle ide vanjska
           petlja
        mov r1, #3 @konstanta koja odredjuje dokle ide
           unutrasnja petlja

        @sortiranje je izvedeno koristeci bubble sort algoritam
           (barem sam ja tako zamislio)

loopOUT:
        mov r8, #0
        ldr r2, =string @predstavlja adresu nekog bajta u
           stringu
        add r3, r2, #1 @predstavlja adresu odma iza prethodno
           navedene
loopIN:
        ldrb r4, [r2] @uzimaju se vrijednosti koje se nalaza na
           adresama koje su pohranjene u registru
        ldrb r5, [r3] @- || -
        cmp r4, r5 @usporedba potrebna za sortiranje po rastucem
            redoslijedu, porede se ascii vrijednosti
        ble true @ukoliko je prethodni uslov ispunjen prelazi se
            na labelu true
        mov r6, r5 @ova i sljedece 2 komande predstavljaju
           zamjenu sadrzaja u registru
        mov r5, r4
        mov r4, r6
        strb r4, [r2] @ ova i sljedeca komanda vrse pohranu
           zamijenjenog sadrzaja
        strb r5, [r3]
true:
```

```
add r2, r2, #1 @pomicemo se na sljedeci karakter u
   stringu
add r3, r3, #1
cmp r8, r1 @kontrolni uslov za unutarasnju petlju
ble loopIN
add r7, r7, #1
cmp r7, r0 @kontrolni uslov za vanjsku petlju
ble loopOUT

mov r7, #4
mov r0, #1
ldr r, =string @ispis stringa
mov r2, #6
swi #0
```

3 Zadatak 3

```
        .data
broj:
        .asciz "-12\0" @maksimalno moze biti 7 cifri +
           terminirajuci znak
        .text
.global _start
_start:

        mov r3, #10 @konstanta za mnozenje
        ldr r2, =broj @ucitavamo adresu stringa
        ldr r0, =broj @brojac
        mov r1, #0 @ovdje zapisujemo cifru (memorijska lokacija
           specificirana u zadatku)
        ldrb r7, [r2]
        cmp r7, #45 @provjera da li je prvi znak minus
        beq NEG @ako jeste prelazimo odma na loopNEG
loopPOZ:
        ldrb r4, [r2]
        subs r4, r4, #48 @normiramo na decimalni sistem,
           oduzimamo vrijednost 0 iz ASCII sistema
        mla r1, r1, r3, r4 @komanda za mnozenje i sabiranje te
           pohranu u r1, tj. mnozi se r1 sa r3 i doda se rr3 u
           r1
        add r2, r2, #1 @pomjeramo broj
        add r0, r0, #1 @pomjeramo brojac
        ldrb r5, [r0] @poredi se nenormirani string, zbog toga
           postoje 2 pohrane na pocetku r2 i r0
        cmp r5, #0 @ukoliko smo dosli do terminirajuceg znaka tj
           . NULL onda zavrsavamo petlju
        bne loop

        cmp r7, #45
        bne KRAJ
NEG:

add r2, r2, #1
add r0, r0, #1 @preskacemo samo minus, dole se kod isti ponavlja
    za upisivanje cifre, samo je sada registar u koji se upisuje
    r8
mov r8, #0
loopNEG:

ldrb r4, [r2]
        subs r4, r4, #48 @normiramo na decimalni sistem,
           oduzimamo vrijednost 0 iz ASCII sistema
        mla r1, r1, r3, r4 @komanda za mnozenje i sabiranje te
```

```
            pohranu u r1, tj. mnozi se r1 sa r3 i doda se rr3 u
               r1
        add r2, r2, #1 @pomjeramo broj
        add r0, r0, #1 @pomjeramo brojac
        ldrb r8, [r0] @poredi se nenormirani string, zbog toga
           postoje 2 pohrane na pocetku r2 i r0
        cmp r8, #0 @ukoliko smo dosli do terminirajuceg znaka tj
           . NULL onda zavrsavamo petlju
        bne loop

@potrebno je jos izvrsiti upis u nas zeljeni registar r7
   negativan broj, tj. komplement broja iz registra 8
@za to je koristena komanda mvn
mvn r7, r8 @sada je upisan negativan broj

KRAJ: @cisto da se moze preskociti loopNEG u slucaju pozitivnog
   broja
```

Napomena: U zadacima, ukoliko je koristen izraz "string" u komentaru koda, ustvari se misli na niz karaktera smjesten u memoriju, jer sam tip string ne postoji na ovom nivou programiranja