Univerzitet u Sarajevu
Elektrotehnički fakultet
Ugradbeni sistemi 2023/24.

Izvještaj za laboratorijsku vježbu 9

GNU ARM Assembler

Ime i prezime: **Kanita Kadušić**
Broj index-a: **19327**

Sarajevo, maj 2024.

Sadržaj

1 Pseudokod 1
1.1 Zadatak 1 1
1.2 Zadatak 2 1
1.3 Zadatak 3 1
1.4 Zadatak 4 1
2 Analiza programskog rješenja 2
3 Korišteni hardverski resursi 3
3.1 Raspberry Pi 2 Model B 3
4 Zaključak 4
5 Prilog 5
5.1 Zadatak 1: Izvorni kôd 5
5.2 Zadatak 2: Izvorni kôd 6
5.3 Zadatak 3: Izvorni kôd 8
5.4 Zadatak 4: Izvorni kôd 9

1 Pseudokod

1.1 Zadatak 1

učitaj redni broj željenog Fibonaccijevog broja
učitaj početne vrijednosti
generiši naredni Fibonaccijev broj
provjeri da li su generisani svi brojevi do željenog

1.2 Zadatak 2

ispiši poruku za unos elemenata
unesi elemente niza
pozovi rutinu za sortiranje niza
ispiši poruku prije ispisa sortiranog niza
ispiši sortirani niz
završi program

1.3 Zadatak 3

očitaj unos sa tastature
postepeno pretvaraj karaktere u cifre
spremi rezultat u memoriju
završi program

1.4 Zadatak 4

ispiši poruku za unos broja elemenata
unesi broj elemenata
pretvori unos u broj
ispiši poruku za unos elemenata niza
učitaj elemente niza
sortiraj niz
izračunaj i ispiši rezultate
završi program

2 Analiza programskog rješenja

S obzirom na prirodu zadataka Laboratorijske vježbe 9, te činjenicu da je svaka linija kôda objašnjena u izvornom kôdu, daljnja analiza neće biti priložena.

3 Korišteni hardverski resursi

3.1 Raspberry Pi 2 Model B

	Komponenta	*Opis*	*Količina*
1	ARM Cortex-A7	jezgra	1
2	mrežni kabal	spajanje sa *switch*-erom	1
3	USB Micro punjač	napajanje i komunikacija	1

4 Zaključak

Zadaci u okviru Laboratorijske vježbe 9 su klasični, jednostavni zadaci, ali čija implementacija u asembleru čini stvari mnogo napornijim.

Uz mnogo strpljenja, koncentracije i ponekih „trikova“, realizacija je protekla uredno.

Za kraj, nova znanja stečena u okviru Laboratorijske vježbe 9 podrazumijevaju upoznavanje s ARM instrukcijskim setom, te korištenje GNU asemblera za asembliranje kôda za ARM arhitekturu.

5 Prilog

5.1 Zadatak 1: Izvorni kôd

```
01: .section .data
02:
03: N: .word 48                 @ N-ti Fibonaccijev broj
04: fibonacci: .space 200       @ prostor za do 50 Fibonaccijevih brojeva
05:
06: .section .text
07:
08: .global _start
09:
10: _start:
11:     ldr r1, =N              @ r1 <- &N
12:     ldr r2, [r1]            @ r2 <- N
13:     ldr r3, =fibonacci      @ r3 <- &fibonacci
14:
15:     mov r4, #1              @ F1 = 1
16:     str r4, [r3], #4        @ fibonacci[0] = 1 | r3 <- r3 + 4
17:
18:     mov r4, #1              @ F2 = 1
19:     str r4, [r3], #4        @ fibonacci[1] = 1 | r3 <- r3 + 4
20:
21:     sub r2, r2, #2          @ N -= 2
22:     cmp r2, #0
23:     beq done                @ if N = 0: done
24:
25: generate:
26:     ldr r4, [r3, #-4]       @ r4 <- F(N - 1)
27:     ldr r5, [r3, #-8]       @ r4 <- F(N - 2)
28:     add r4, r4, r5          @ F(N) = F(N - 1) + F(N - 2)
29:     str r4, [r3], #4        @ fibonacci[...] = sadrzaj(r4) | r3 <- r3 + 4
30:
31:     sub r2, r2, #1          @ N--
32:     cmp r2, #0
33:     bne generate            @ if N != 0: generate
34:     beq done                @ if N = 0: done
35:
36: done:
37:     mov r0, #0              @ status kod 0
38:     mov r7, #1              @ syscall za exit
39:     swi 0                   @ prekid za syscall
40:     b done
```

5.2 Zadatak 2: Izvorni kôd

```
01: .equ ARRAY_SIZE, 10                    @ broj elemenata niza
02:
03: .section .data
04:
05: prompt_input:  .asciz "Unesite elemente niza kao ASCII karaktere:\n"
06: prompt_output: .asciz "Sortirani niz:\n"
07:
08: array: .space ARRAY_SIZE               @ zauzimanje prostora za niz
09:
10: .section .text
11:
12: .global _start
13:
14: _start:
15:     @ Ispis poruke za unos elemenata
16:     mov r7, #4                         @ syscall za write
17:     mov r0, #1                         @ file descriptor 1 (stdout)
18:     ldr r1, =prompt_input              @ adresa prompt_input
19:     mov r2, #44                        @ duzina prompt_input
20:     swi #0                             @ prekid za syscall
21:
22:     @ Unos elemenata niza
23:     mov r7, #3                         @ syscall za read
24:     mov r0, #0                         @ file descriptor 0 (stdin)
25:     ldr r1, =array                     @ adresa array
26:     mov r2, #ARRAY_SIZE                @ duzina array
27:     swi #0                             @ prekid za syscall
28:
29:     @ Poziv rutine za sortiranje niza
30:     bl selection_sort
31:
32:     @ Ispis poruke prije ispisa niza
33:     mov r7, #4                         @ syscall za write
34:     mov r0, #1                         @ file descriptor 1 (stdout)
35:     ldr r1, =prompt_output             @ adresa prompt_output
36:     mov r2, #16                        @ duzina prompt_output
37:     swi #0                             @ prekid za syscall
38:
39:     @ Ispis sortiranog niza
40:     mov r7, #4                         @ syscall za write
41:     mov r0, #1                         @ file descriptor 1 (stdout)
42:     ldr r1, =array                     @ adresa array
43:     mov r2, #ARRAY_SIZE                @ duzina array
44:     swi #0                             @ prekid za syscall
45:
46:     @ Izlaz iz programa
47:     mov r7, #1                         @ syscall za exit
48:     swi #0                             @ prekid za syscall
49:
```

```
50: selection_sort:
51:     ldr r0, =array                  @ r0 <- &array
52:     mov r1, #-1                     @ i = -1
53:
54: outer_loop:
55:     add r1, #1                      @ i++
56:     cmp r1, #ARRAY_SIZE
57:     bge sort_done                   @ if i >= ARRAY_SIZE: sort_done
58:
59:     ldrb r2, [r0, r1]               @ r2 <- array[i]
60:     mov r3, r1                      @ j = i
61:
62: inner_loop:
63:     add r3, #1                      @ j++
64:     cmp r3, #ARRAY_SIZE
65:     bge outer_loop                  @ if j >= ARRAY_SIZE: outer_loop
66:
67:     ldrb r4, [r0, r3]               @ r4 <- array[j]
68:     cmp r2, r4
69:     ble inner_loop                  @ if sadrzaj(r2) <= sadrzaj(r4): inner_loop
70:
71:     strb r4, [r0, r1]               @ u suprotnom
72:     strb r2, [r0, r3]               @ razmijeni sadrzaj u memoriji
73:     mov r2, r4                      @ r2 <- r4
74:
75:     b inner_loop                    @ ponovi inner_loop
76:
77: sort_done:
78:     bx lr                           @ povratak iz rutine
```

5.3 Zadatak 3: Izvorni kôd

```
01: .section .data
02:
03:     size: .int 12                   @ velicina bafera za unos
04:     buffer: .space 12               @ prostor za unos broja kao ASCII znakova
05:
06: .section .bss
07:
08:     number: .word 0                 @ memorijska lokacija za broj
09:
10: .section .text
11:
12: .global _start
13:
14: _start:
15:     @ Syscall za citanje sa stdin
16:     mov r7, #3                      @ syscall za read
17:     mov r0, #0                      @ file descriptor 0 (stdin)
18:     ldr r1, =buffer                 @ ucitaj adresu bafera
19:     mov r2, #12                     @ maksimalni broj bajtova za citanje
20:     swi #0                          @ prekid za syscall
21:
22:     ldr r0, =buffer                 @ ucitaj adresu bafera sa unesenim podacima
23:     mov r1, #0                      @ inicijaliziraj rezultat na 0
24:     mov r2, #0                      @ inicijaliziraj privremeni registar za rezultat
25:
26: convert_loop:
27:     ldrb r3, [r0], #1               @ ucitaj naredni karakter iz bafera
28:     cmp r3, #10                     @ provjeri je li kraj linije ('\n')
29:     beq store_number                @ ako je kraj linije, spremi broj
30:     sub r3, r3, #48                 @ pretvori ASCII karakter u cifru
31:
32:     @ r1 = r1 * 10
33:     mov r4, r1, lsl #3              @ r4 = r1 * 8
34:     add r1, r4, r1, lsl #1          @ r1 = r4 + r1 * 2 = r1 * 10
35:
36:     add r1, r1, r3                  @ dodaj cifru u rezultat
37:     b convert_loop                  @ nastavi konverziju
38:
39: store_number:
40:     ldr r0, =number                 @ ucitaj adresu memorijske lokacije
41:     str r1, [r0]                    @ spremi broj u memorijsku lokaciju
42:
43:     @ Izlaz iz programa
44:     mov r0, #0                      @ status kod 0
45:     mov r7, #1                      @ syscall za exit
46:     swi 0                           @ prekid za syscall
```

5.4 Zadatak 4: Izvorni kôd

```
001: .section .data
002:
003: prompt_broj: .asciz "Unesite broj elemenata niza:\n"
004: prompt_elementi: .asciz "Unesite elemente niza:\n"
005: prompt_min: .asciz "Najmanji broj: %d\n"
006: prompt_max: .asciz "Najveci broj: %d\n"
007: prompt_opseg: .asciz "Opseg: %d\n"
008: prompt_medijan: .asciz "Medijan: %d\n"
009:
010: buffer: .asciz "                    "
011: array: .space 40                @ prostor za do 10 cijelih brojeva
012:
013: max_elements: .word 10
014: num_elements: .word 0
015:
016: .section .text
017:
018: .global _start
019: .extern printf
020:
021: _start:
022:     @ Ispis poruke za unos broja elemenata
023:     mov r7, #4                  @ syscall za write
024:     mov r0, #1                  @ file descriptor 1 (stdout)
025:     ldr r1, =prompt_broj        @ adresa prompt_broj
026:     mov r2, #31                 @ duzina prompt_broj
027:     swi #0                      @ prekid za syscall
028:
029:     @ Unos broja elemenata
030:     mov r7, #3                  @ syscall za read
031:     mov r0, #0                  @ file descriptor 0 (stdin)
032:     ldr r1, =buffer             @ adresa buffer
033:     mov r2, #20                 @ duzina buffer
034:     swi #0                      @ prekid za syscall
035:
036:     @ Pretvori unos u broj
037:     ldr r1, =buffer             @ ucitaj adresu buffer-a
038:     mov r2, #0                  @ postavi rezultat na 0
039:     mov r3, #0                  @ broj je na pocetku pozitivan
040:
041: parse_num:
042:     ldrb r4, [r1], #1           @ ucitaj bajt i inkrementiraj r1
043:     cmp r4, #' '                @ provjeri da li je jednak razmaku
044:     beq parse_num               @ preskoci ako jeste
045:     cmp r4, #'-'                @ provjeri da li je jednak minusu
046:     bne check_digit_num         @ ako nije provjeri je li cifra
047:     mov r3, #1                  @ ako jeste znaci da je broj negativan
048:     b parse_num                 @ nastavi parsirati
049:
```

```
050: check_digit_num:
051:     sub r4, r4, #'0'            @ pretvori ASCII u cifru
052:     cmp r4, #9                  @ provjeri da li je cifra
053:     bhi end_parse_num           @ ako nije zavrsi parsiranje
054:     mov r5, r2, lsl #3          @ r5 = r2 * 8
055:     add r2, r5, r2, lsl #1      @ r2 = r5 + r2 * 2 = r2 * 10
056:     add r2, r2, r4              @ dodaj cifru na rezultat
057:     b parse_num                 @ nastavi parsirati
058:
059: end_parse_num:
060:     cmp r3, #0                  @ provjeri da li je broj pozitivan
061:     beq store_num               @ ako jeste upisi broj elemenata
062:     rsb r2, r2, #0              @ negiraj rezultat
063:
064: store_num:
065:     ldr r0, =max_elements       @ ucitaj adresu max_elements
066:     ldr r1, [r0]                @ ucitaj sadrzaj max_elements
067:     cmp r2, r1                  @ uporedi num_elements sa max_elements
068:     bhi exit_error              @ if num_elements > max_elements: end
069:     ldr r0, =num_elements       @ ucitaj adresu num_elements
070:     str r2, [r0]                @ upisi broj elemenata
071:
072:     @ Ispis poruke za unos elemenata niza
073:     mov r7, #4                  @ syscall za write
074:     mov r0, #1                  @ file descriptor 1 (stdout)
075:     ldr r1, =prompt_elementi    @ adresa prompt_elementi
076:     mov r2, #24                 @ duzina prompt_elementi
077:     swi #0                      @ prekid za syscall
078:
079:     @ Ucitavanje elemenata
080:     ldr r0, =num_elements       @ ucitaj adresu num_elements
081:     ldr r3, [r0]                @ ucitaj sadrzaj num_elements
082:     mov r4, #0                  @ inicijalizacija index-a na 0
083:
084: read_element_loop:
085:     cmp r4, r3                  @ uporedi index sa num_elements
086:     bge selection_sort          @ if index >= num_elements: break
087:
088:     @ Unos elementa
089:     mov r7, #3                  @ syscall za read
090:     mov r0, #0                  @ file descriptor 0 (stdin)
091:     ldr r1, =buffer             @ adresa buffer
092:     mov r2, #20                 @ duzina buffer
093:     swi #0                      @ prekid za syscall
094:
095:     @ Pretvori unos u broj
096:     ldr r1, =buffer             @ ucitaj adresu buffer-a
097:     mov r2, #0                  @ postavi rezultat na 0
098:     mov r5, #0                  @ broj je na pocetku pozitivan
099:
```

```
100: parse:
101:     ldrb r6, [r1], #1           @ ucitaj bajt i inkrementiraj r1
102:     cmp r6, #' '                @ provjeri da li je jednak razmaku
103:     beq parse                   @ preskoci ako jeste
104:     cmp r6, #'-'                @ provjeri da li je jednak minusu
105:     bne check_digit             @ ako nije provjeri je li cifra
106:     mov r5, #1                  @ ako jeste znaci da je broj negativan
107:     b parse                     @ nastavi parsirati
108:
109: check_digit:
110:     sub r6, r6, #'0'            @ pretvori ASCII u cifru
111:     cmp r6, #9                  @ provjeri da li je cifra
112:     bhi end_parse               @ ako nije zavrsi parsiranje
113:     mov r7, r2, lsl #3          @ r7 = r2 * 8
114:     add r2, r7, r2, lsl #1      @ r2 = r7 + r2 * 2 = r2 * 10
115:     add r2, r2, r6              @ dodaj cifru na rezultat
116:     b parse                     @ nastavi parsirati
117:
118: end_parse:
119:     cmp r5, #0                  @ provjeri da li je broj pozitivan
120:     beq store                   @ ako jeste upisi broj elemenata
121:     rsb r2, r2, #0              @ negiraj rezultat
122:
123: store:
124:     ldr r0, =array              @ ucitaj adresu array
125:     add r0, r0, r4, lsl #2      @ daj &array[r4]
126:     str r2, [r0]                @ upisi element
127:     add r4, r4, #1              @ inkrementiraj index
128:     b read_element_loop         @ ponovi za sljedeci element
129:
130: selection_sort:
131:     mov r4, #0                  @ i = 0
132:     ldr r0, =num_elements       @ ucitaj adresu num_elements
133:     ldr r3, [r0]                @ ucitaj sadrzaj num_elements
134:
135: outer_loop:
136:     cmp r4, r3                  @ uporedi 'i' i num_elements
137:     bge print_results           @ if i >= num_elements: print
138:     mov r6, r4                  @ min_index = i
139:     add r5, r4, #1              @ j = i + 1
140:
141: inner_loop:
142:     cmp r5, r3                  @ uporedi 'j' i num_elements
143:     bge update_min              @ if j >= num_elements: break
144:     ldr r1, =array              @ ucitaj adresu array
145:     ldr r8, [r1, r5, lsl #2]    @ r8 = tekst[j]
146:     ldr r7, [r1, r6, lsl #2]    @ r7 = tekst[min_index]
147:     cmp r8, r7                  @ if tekst[j] < tekst[min_index]:
148:     bge skip_update             @ skip if r8 >= r7
149:     mov r6, r5                  @ min_index = j
150:
```

```
151: skip_update:
152:     add r5, r5, #1              @ j = j + 1
153:     b inner_loop                @ nastavi inner_loop
154:
155: update_min:
156:     cmp r4, r6                  @ uporedi 'i' i min_index
157:     beq no_swap                 @ if i == min_index: no_swap
158:     ldr r1, =array              @ ucitaj adresu array
159:     ldr r8, [r1, r4, lsl #2]    @ r8 = tekst[i]
160:     ldr r7, [r1, r6, lsl #2]    @ r7 = tekst[min_index]
161:     str r7, [r1, r4, lsl #2]    @ tekst[i] = tekst[min_index]
162:     str r8, [r1, r6, lsl #2]    @ tekst[min_index] = tekst[i]
163:
164: no_swap:
165:     add r4, r4, #1              @ i = i + 1
166:     b outer_loop                @ nastavi outer_loop
167:
168: print_results:
169:     @ Izracunaj opseg
170:     ldr r3, =num_elements       @ ucitaj adresu num_elements
171:     ldr r3, [r3]                @ ucitaj sadrzaj num_elements
172:     ldr r0, =array              @ ucitaj adresu prvog elementa
173:     ldr r1, [r0]                @ ucitaj prvi element
174:     add r2, r0, r3, lsl #2      @ ucitaj &array[num_elements]
175:     sub r2, r2, #4              @ ucitaj adresu posljednjeg elementa
176:     ldr r2, [r2]                @ ucitaj posljednji element
177:     sub r3, r2, r1              @ izracunaj opseg (max - min)
178:
179:     @ Izracunaj medijan
180:     ldr r4, =num_elements       @ ucitaj adresu num_elements
181:     ldr r5, [r4]                @ ucitaj sadrzaj num_elements
182:     mov r6, r5, lsr #1          @ izracunaj index medijana (r5 / 2)
183:     tst r5, #1                  @ provjeri da li je num_elements neparan
184:     beq even_median             @ paran broj elemenata
185:
186:     @ Neparan broj elemenata
187:     ldr r0, =array              @ ucitaj adresu array
188:     add r7, r0, r6, lsl #2      @ daj &array[r6]
189:     ldr r7, [r7]                @ ucitaj medijan
190:     b median_calculated         @ medijan izracunat
191:
192:     even_median:
193:     ldr r0, =array              @ ucitaj adresu array
194:     add r7, r0, r6, lsl #2      @ daj &array[r6]
195:     ldr r7, [r7]                @ ucitaj array[r6]
196:     sub r6, r6, #1              @ index prethodnog sredisnjeg elemenata
197:     add r0, r0, r6, lsl #2      @ daj &array[r6 - 1]
198:     ldr r6, [r0]                @ ucitaj array[r6 - 1]
199:     add r7, r7, r6              @ suma ucitanih elemenata
200:     mov r7, r7, lsr #1          @ podijeli s 2
201:
202: median_calculated:
203:     nop
204:
```

```
205: @ Ispis rezultata koristeci 'printf'
206: exit_program:
207:     @ Premjestanje rezultata
208:     mov r10, r1
209:     mov r9, r2
210:     mov r8, r3
211:     mov r6, r7
212:
213:     @ Ispisi minimalni element
214:     push {fp, lr}
215:     add fp, sp, #4
216:     ldr r0, =prompt_min
217:     mov r1, r10
218:     bl printf
219:     nop
220:     sub sp, fp, #4
221:     pop {fp, lr}
222:
223:     @ Ispisi maksimalni element
224:     push {fp, lr}
225:     add fp, sp, #4
226:     ldr r0, =prompt_max
227:     mov r1, r9
228:     bl printf
229:     nop
230:     sub sp, fp, #4
231:     pop {fp, lr}
232:
233:     @ Ispisi opseg
234:     push {fp, lr}
235:     add fp, sp, #4
236:     ldr r0, =prompt_opseg
237:     mov r1, r8
238:     bl printf
239:     nop
240:     sub sp, fp, #4
241:     pop {fp, lr}
242:
243:     @ Ispisi medijan
244:     push {fp, lr}
245:     add fp, sp, #4
246:     ldr r0, =prompt_medijan
247:     mov r1, r6
248:     bl printf
249:     nop
250:     sub sp, fp, #4
251:     pop {fp, lr}
252:
```

```
253: exit_success:
254:     mov r0, #0                 @ status kod 0
255:     mov r7, #1                 @ syscall za exit
256:     swi #0                     @ prekid za syscall
257:
258: exit_error:
259:     mov r0, #12                @ status kod 0
260:     mov r7, #1                 @ syscall za exit
261:     swi #0                     @ prekid za syscall
```