Računalniška arhitektura

2. Domača naloga – poročilo

Luka Šveigl

1. NALOGA:

Napišite program v zbirnem jeziku za procesor ARM, ki prešteje število vejic, pik, klicajev in vprašajev v nizu NIZ ter zapiše rezultat v 8-bitno spremenljivko REZULTAT.

NIZ: .asciz “Koliko je v tem nizu vejic, pik, klicajev in vprašajev? Vesele praznike in srečno novo leto. Ostanite zdravi! Se slišimo…”
REZULTAT: .byte 0

Program:

```
        .text
REZULTAT: .byte 0
NIZ:    .asciz  "Koliko je v tem nizu vejic, pik, klicajev in vprašajev? Vesele praznike in srecno novo leto. Ostanite zdravi! Se slišimo…"
        .align
        .global __start
__start:
        @ Program, ki presteje vejice, pike, klicaje in vprasaje (locila)
        mov r2, #',' @ V registre shranimo vrednosti za primerjavo
        mov r3, #'.'
        mov r4, #'!'
        mov r5, #'?'

        mov r6, #0 @ Stevec locil

        adr r0, REZULTAT
        adr r1, NIZ

LOOP:
        ldrb r7, [r1]

        cmp r7, #0 @ Preverimo, ce je niza konec
        beq END_OF_LOOP  @ Ce je niza konec, koncamo program

        cmp r7, r2 @ Pogledamo, ce je znak vejica
        addeq r6, r6, #1

        cmp r7, r3 @ Pogledamo, ce je znak pika
        addeq r6, r6, #1

        cmp r7, r4 @ Pogledamo, ce je znak klicaj
        addeq r6, r6, #1

        cmp r7, r5 @ Pogledamo, ce je znak vprasaj
        addeq r6, r6, #1

        add r1, r1, #1 @ Premik na naslednji znak

        b LOOP
END_OF_LOOP:
        strb r6, [r0]
__end:  b __end
```

Uporabljeni registri:

R0 – bazni register za naslavljanje naslova REZULTAT
R1 – bazni register za naslavljanje naslova NIZ
R2 – register, ki vsebuje znak ‘,’
R3 – register, ki vsebuje znak ‘.’
R4 – register, ki vsebuje znak ‘!’
R5 – register, ki vsebuje znak ‘?’
R6 – register, ki je števec ločil
R7 – register, v katerega naložimo posamezen znak niza

Testni podatki:

Testni podatek je niz, ki je bil podan v navodilu naloge: "Koliko je v tem nizu vejic, pik, klicajev in vprašajev? Vesele praznike in srecno novo leto. Ostanite zdravi! Se slišimo…"

Rezultat programa:

Rezultat programa je 5, saj ta stavek vsebuje 5 ločil, ki so vejica, pika, klicaj ali vprašaj. Ločila … na koncu ne štejemo, saj to niso 3 pike, temveč en znak, ki je v ASCII naslovljen z kodo 0x85

Delovanje programa:

Program deluje tako, da gre v zanki čez vse znake podanega niza, in jih primerja z iskanimi vrednostmi. Če je trenutni znak enak kateri izmed iskanih vrednosti, se števcu ločil prišteje 1. V primeru, da je najdeni znak 0x00, pa program ve, da je prišel do konca niza in zato zanko zaključi, nato pa še shrani vrednost števca v spremenljivko REZULTAT.

2. NALOGA:

V zbirniku za procesor ARM napišite program, ki v tabeli predznačenih števil TABELA poišče največje in najmanjše število in ju zapiše v 16-bitni spremenljivki MIN in MAX. Iskanje največjega in najmanjšega števila izvedite v zanki. Kot komentar na koncu programa dopišite še najmanjšo in največjo vrednost (šestnajstiško) in na katerem indeksu se nahajata v tabeli. Prvi element ima indeks 0.

```
TABELA: .hword 1, 5, -1, 8, -130, 1024, 10, 64, -80, 256
MAX:      .space 2
MIN:      .space 2
VELIKOST_TABELE: .byte 10
```

Program:

```
          .text
TABELA:   .hword 1, 5, -1, 8, -130, 1024, 10, 64, -80, 256
MAX:      .space 2
MIN:      .space 2
VELIKOST_TABELE: .byte 10
        .align
        .global __start
__start:
         adr r0, TABELA
         adr r1, VELIKOST_TABELE
         ldr r2, [r1]

         ldrsh r5, [r0]

         mov r3, r5 @MAX
         mov r4, r5 @MIN
LOOP:
        ldrsh r5, [r0]

        cmp r5, r3
        movge r3, r5
        cmp r5, r4
        movle r4, r5

        add r0, r0, #2
        subs r2, r2, #1

        bne LOOP
        adr r6, MAX
        strh r3, [r6]

        adr r6, MIN
        strh r4, [r6]
@ Najmanjsa vrednost v tabeli je -130 oz 0xFF7E, z indeksom 4
@ Najvecja vrednost v tabeli je 1024 oz 0x0400, z indeksom 5
__end:    b __end
```

Uporabljeni registri:

R0 – bazni register za naslavljanje naslova TABELA
R1 – bazni register za naslavljanje naslova VELIKOST_TABELE
R2 – register, v katerem je shranjena vrednost VELIKOST_TABELE, uporabljen je kot števec.
R3 – register, v katerem je shranjena največja vrednost tabele
R4 – register, v katerem je shranjena najmanjša vrednost tabele
R5 – register, v katerega se shrani posamezno št. v tabeli
R6 – bazni register za naslavljanje naslovov MAX in MIN

Testni podatki:

Testni podatek za program je tabela, podana v navodilih naloge, torej : TABELA: .hword 1, 5, -1, 8, -130, 1024, 10, 64, -80, 256.

Rezultat programa:

Najmanjša vrednost v tabeli je -130, oziroma 0xFF7E, ki se nahaja na indeksu 4, največja pa 1024 oziroma 0x0400, ki se nahaja na indeksu 5.

Delovanje programa:

Program deluje tako, da se na začetku kot največja in najmanjša vrednost privzame kar prva vrednost v tabeli. Nato gremo z uporabo zanke čez vsak element tabele, in ga primerjamo z do sedaj največjo oz. najmanjšo najdeno vrednostjo. Če je trenutni element večji oz. manjši kot do sedaj najdene vrednosti, le te nadomestimo z trenutnim elementom. Zanka se ponavlja, dokler števec ni enak 0.

3. NALOGA:

Napišite program v zbirnem jeziku za procesor ARM, ki izračuna rezultat pri celoštevilskem množenju dveh 8-bitnih nepredznačenih spremenljivk STEV1 in STEV2 brez ukaza za množenje. Rezultat zapišite v 8-bitno spremenljivko REZULTAT. Predpostavite lahko, da sta števili različni od 0. Program napišite tako, da v zanki prištevate k produktu STEV1 tolikokrat, kot je vrednost STEV2.

Pravilnost programa preverite na različnih testnih podatkih. Kot komentar na koncu programa zapišite še na katerih testnih podatkih ste preverili delovanje in kakšen je bil rezultat v desetiškem in šestnajstiškem zapisu.

STEV1: .byte 24
STEV2: .byte 7
REZULTAT: .space 1

Program:

```
        .text

STEV1:    .byte 24
STEV2:    .byte 7
REZULTAT: .space 1

        .align
        .global __start
__start:
        adr r0, STEV1
        adr r1, STEV2
        ldrb r2, [r0] @ STEV1
        ldrb r3, [r1] @ Stevec (STEV2)

        mov r4, #0

LOOP:
        add r4, r4, r2

        subs r3, r3, #1
        bne LOOP

        adr r5, REZULTAT
        strb r4, [r5]

@ Testni podatki: STEV1 = 24, STEV2 = 7, REZULTAT = 168 oz 0xA8
@ Testni podatki: STEV1 = 15, STEV2 = 15, REZULTAT = 225 oz 0xE1
@ Testni podatki: STEV1 = 2, STEV2 = 1, REZULTAT = 2 oz 0x02
@ Testni podatki: STEV1 = 3, STEV2 = 5, REZULTAT = 15 oz 0x0F
__end:    b __end
```

Uporabljeni registri:

R0 – bazni register za naslavljanje naslova STEV1
R1 – bazni register za naslavljanje naslova STEV2
R2 – register, v katerega naložimo vrednost STEV1
R3 – register, v katerega naložimo vrednost STEV2, uporabljen kot števec
R4 – register, v katerega shranjujemo vrednost ob seštevanju, hrani rezultat
R5 – bazni register za naslavljanje naslova REZULTAT

Testni podatki in rezultati:

Testni podatki: STEV1 = 24, STEV2 = 7, REZULTAT = 168 oz. 0xA8
Testni podatki: STEV1 = 15, STEV2 = 15, REZULTAT = 225 oz. 0xE1
Testni podatki: STEV1 = 2, STEV2 = 1, REZULTAT = 2 oz. 0x02
Testni podatki: STEV1 = 3, STEV2 = 5, REZULTAT = 15 oz. 0x0E

Delovanje programa:

Program deluje tako, da na začetku v register, ki hrani rezultat shrani vrednost, 0, kot števec zanke pa se uporabi vrednost STEV2. V zanki se rezultatu prišteje vrednost STEV1, tolikokrat, kolikor je vrednost STEV2. Po izvedbi zanke se rezultat shrani v spremenljivko REZULTAT.