# Vježba iz Ugradbenih računalnih sustava

1. U memoriji inicijalizirati četiri riječi te ih učitati u registre. Također, inicijalizirati četiri okteta koje je potrebno pročitati i staviti na stog.

.equ adresa, 0x2000 /\* adresa bloka u memoriji \*/

.equ stack\_top, 0x1000 /\* vrh stoga na adresi 0x1000\*/

.global \_start

.text

\_start: /\* Početak programa \*/

MOV sp, #stack\_top /\* postavlja se stack pointer na vrh stoga \*/

adr r0, pod /\* stavlja se u ro adresa podataka \*/

LDR r1, [r0],#4 /\* u r1 prvi podatak iz memorije 0x000F \*/

LDR r2, [r0],#4 /\* u r2 drugi podatak iz memorije 0x00FF \*/

LDR r3, [r0],#4 /\* u r3 treci podatak iz memorije 0x0FFF \*/

LDR r4, [r0],#4 /\* u r4 cetvrti podatak iz memorije 0xFFFF \*/

LDRB r5, [r0],#1 /\* u r6 prvi byte podataka 0x01 \*/

STR r5, [sp],#-4 /\* na stog se stavlja r5 \*/

LDRB r5, [r0],#1

STR r5, [sp],#-4

LDRB r5, [r0],#1

STR r5, [sp],#-4

LDRB r5, [r0],#1

STR r5, [sp],#-4

stop:

B stop /\* program end and get into endless loop \*/

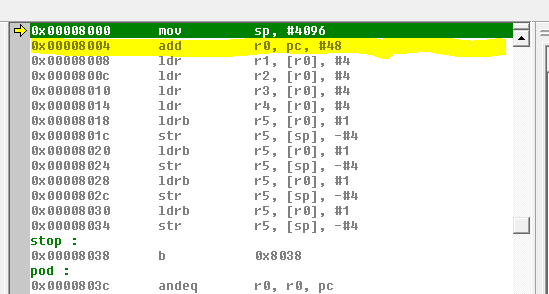
pod:

.word 0x000f, 0x00ff, 0x0fff, 0xffff

.byte 0x01, 0x02, 0x03,0x04

.end

Kod kada se upiše u memoriju izgleda kao na slici:



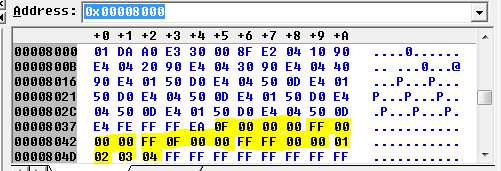
Za primijetiti je naredbu:

**Adr r0, pod**

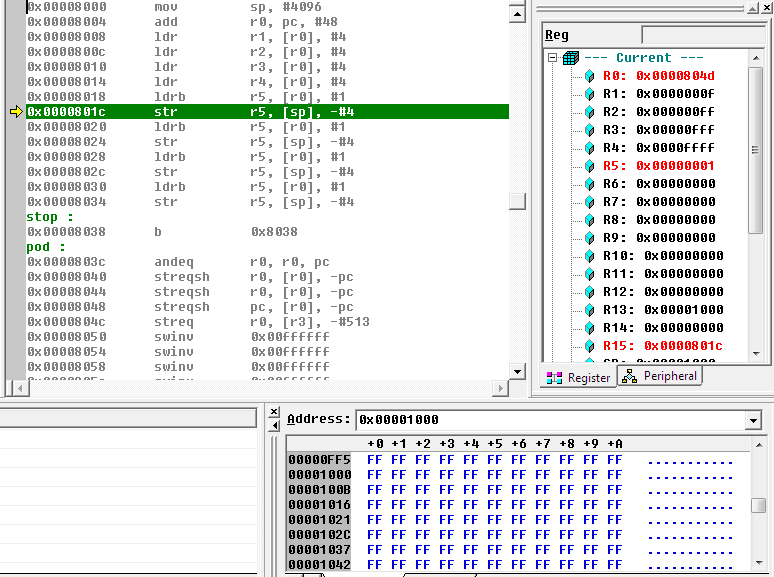
Asembler prevodi u:

**add** r0, pc, #48

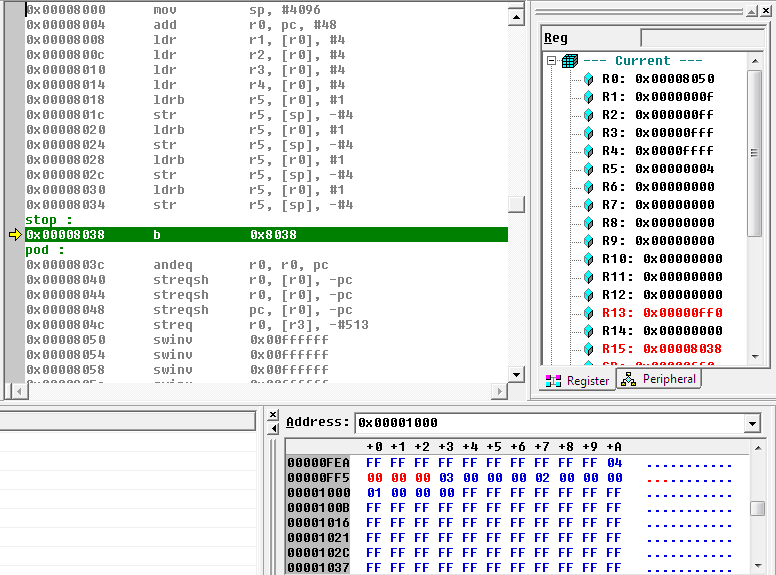
Radi se o relativnom adresiranju u kojem asembler izračuna relativnu razliku između trenutne vrijednosti PCa i adrese na kojoj se nalazi labela **pod:** te zbroj pc + razlika upisuje u r0. Sada je u r0 adresa podataka.



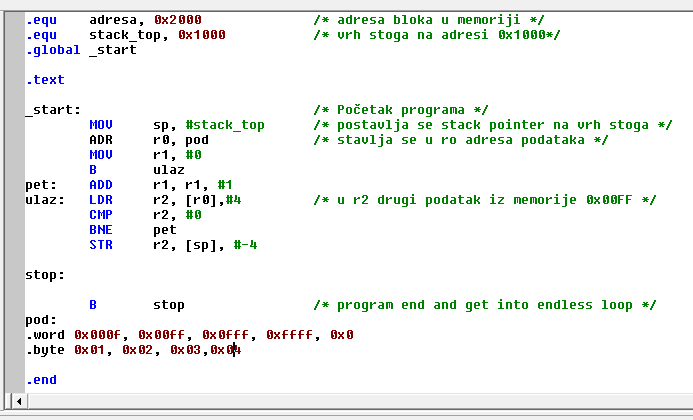
Nakon izvođenja naredbi za upis u spremnik stanje je:



Odnosno nakon spremanja na stog



1. U bloku memorije koji slijedi u području iza koda prebrojiti koliko ima podatka do prvog podatka 0.



1. Napišite program za izračun drugog krojena iz cijelog broja prema sljedećem algoritmu:

(x + 1)2 = x2 + 2x + 1  
neka je x = p = 0, tada je (x + 1)2 = q = 1  
tada se korijen može računati temeljem jednadžbe:  
 q = q + 2\*p + 1;  
 p = p + 1;  
 dok je q manje ili jednako zadanom broju

.text

\_start:

adr r3, pod /\* adresa podatka

ldr r4, [r3] /\*u r4 broj iz kojeg se vadi korijen

mov r0, #0 /\*u r0 p

mov r1, #1 /\*u r1 q

petlja:

add r0, r0, #1 /\*inkrementira se p

mov r2, r0, LSL #1 /\*u r2 (p+1)\*2

add r2, r2, r1 /\*u r2 q + (p+1)=\*2

add r1, r2, #1 /\*u r1 q + (p+1)=\*2 +1

cmp r1, r4 /\*usporedi s brojem

ble petlja /\*ako je manji od broja ponovo

kraj:

b kraj

pod:

.word 6154

1. Napišite program za dijeljenje dva cijela broja!  
   Dijeljenje raditi po principu oduzimanja  
   rez = x/y

rez = 0;   
x = x - y  
dok je x > 0  
 rez = rez + 1  
 x = x -y

.text

\_start:

adr r0, pod

ldr r1, [r0]

ldr r2, [r0,#4]

mov r3, #0

b l1

l0: add r3, r3, #1

l1: subs r2, r2, r1

bgt l0

kraj:

b kraj

pod:

.word 53, 436

1. U memoriji od adrese 0x1000 nalazi se blok sa 32-bitnim podacima u formatu dvojnog komplementa. Blok je nepoznate duljine, ali se zna da je zaključen podatkom F000 0000. Napišite program za koji treba u svim podacima u bloku zamijeniti dvije najniže skupine od 4 bita (vidi sliku). Negativni podaci se ne mijenjaju.

/\* PROMJENITI POSTAVKE PROJEKTA \*/

/\* u Project -> Settings -> Debbug -> Download \*/

/\* postaviti Download address 0x0000 \*/

/\* i opcija Program entry point \*/

/\* postaviti adresu podataka na 0x00001000 \*/

.org 0x00001000

.data

.word 0x5a22115a, 0x11223345, -5, 0xF0000000

/\* postaviti adresu programa na 0x00008000 \*/

/\* .org znači povećati prethodni .org za iznos \*/

/\* 0x00001000 + 0x00007000 = 0x00008000 \*/

.org 0x00007000

\_strat:

/\* u r0 maska 0xFFFFFF00 \*/

mov r0, #0xFF

mov r0, r0, LSL #8

orr r0, r0, #0xFF

mov r0, r0, LSL #8

orr r0, r0, #0xFF

mov r0, r0, LSL #8

/\* u r1 uvjet zaustavljanje 0xF0000000 \*/

mov r1, #0xF0

mov r1, r1, LSL #24

/\* baza podataka 0x1000 u r2 \*/

mov r2, #0x10

mov r2, r2, LSL #8

ponovo:

/\* pročitati podatak u r3 \*/

ldr r3,[r2],#4

/\* usporedi s uvjetom zaustavljanje u r1 \*/

cmp r3, r1

beq kraj

/\* odaberi gornja 4 bita zadnjeg bayta \*/

and r4, r3, #0xF0

/\* pomakni ih u desno za 4 mjesta \*/

mov r4, r4, LSR #4

/\* odaberi donja 4 bita zadnjeg bayta \*/

and r5, r3, #0x0F

/\* pomakni ih u lijevo za 4 mjesta \*/

mov r5, r5, LSL #4

/\* stvori rezultat \*/

and r3, r3, r0

orr r3, r3, r4

orr r3, r3, r5

b ponovo

kraj:

b kraj

četvrti bajt

treći bajt

drugi bajt

RJEŠENJE:

1. Napisati potprogram koji dijeli dva cijela broja koja se prenose preko stoga. Rezultat također vratiti preko stoga.

RJEŠENJE:

.equ stog, 0x1000

\_start:

adr r0, podaci

mov sp, #stog

/\* pročitati podatke koji sudjeluju u dijeljenju \*/

/\* i staviti ih na stog \*/

ldr r1, [r0], #4

str r1, [sp, #-4]!

ldr r1, [r0], #4

str r1, [sp, #-4]!

/\* pozovi proceduru za dijeljenje \*/

bl dijeli

/\* pročitaj rezultat dijeljenja sa stoga \*/

ldr r1, [sp], #4

kraj:

b kraj

dijeli:

ldr r1, [sp], #4

ldr r2, [sp], #4

mov r3, #0

b l1

l0: add r3, r3, #1

l1: subs r2, r2, r1

bgt l0

/\* rezultat na stog \*/

str r3, [sp, #-4]!

/\* vrati se u program \*/

mov pc, r14

podaci:

.word 2456, 52

**Zadatak:**

Napisati potprogram za zbrajanje dva broja. Brojeve u potprogram prenijeti preko registara.

Rješenje:

.global \_start

.text

\_start:

adr r0, podaci

LDR r1, [r0], #4

LDR r2, [r0], #4

BL broji

ADD r3, r3, #0

zbroji:

ADD r3, r1, r2

MOV r15, r14

stop:

B stop

podaci:

.word 30, 40

.end

**Zadatak:**

Napisati potprogram za zbrajanje dva broja. Parametre u potprogram prenijeti preko stoga. U potprogramu sačuvati vrijednosti regisrata koji se koriste. Rezultat vratiti preko stoga i očistiti stog.

Rješenje:

stavljaju se parametri na stog

skida se rezultat sa stoga

stavlja se rezultat na stog

obnavljaju se registri sa stoga

skidaju se parametri sa stoga

stavljaju se registri na stog

.global \_start

.equ stack\_top, 0x1000

.text

\_start:

MOV sp, #stack\_top

adr r0, podaci

LDR r1, [r0], #4

LDR r2, [r0], #4

STR r1, [sp], #-4

STR r2, [sp], #-4

BL zbroji

LDR r3, [sp]

B stop

zbroji:

STR r1, [sp], #-4

STR r2, [sp], #-4

LDR r2, [sp,#16]

LDR r1, [sp,#12]

ADD r3, r1, r2

LDR r2, [sp,#4]!

LDR r1, [sp,#4]!

STR r3, [sp,#8]!

MOV r15, r14

stop:

B stop

podaci:

.word 0x30, 0x40

.end

## ZADACI ZA VJEŽBU

1. U memoriji od adrese 0x1000 nalazi se blok sa 32-bitnim podacima u formatu dvojnog komplementa. Blok je nepoznate duljine, ali se zna da je zaključen podatkom F000 0000. Napišite program za koji treba u svim pozitivnim podacima u bloku zamijeniti dvije najniže skupine od 4 bita (vidi sliku). Negativni podaci se ne mijenjaju.

četvrti bajt

treći bajt

drugi bajt

1. U memoriji sustava nalazi se blok 32-bitnih podataka u formatu dvojnog komplementa. Blok podataka počinje s podatkom koji se nalazi na adresi iza koda, a završava sa podatkom koji se nalazi na adresi 0x0100 većoj. Napisati program za koji će zbrojiti sve podatke iz bloka. Zbroj mora biti u 64-bitnom formatu dvojnog komplementa.
2. U bloku memorije koji počinje od adrese iza koda, nalaze se 16-bitni brojevi u dvostrukom komplementu. Nije poznato koliko blok sadrži podataka, ali se zna da je zaključen 0xFFFF. Izračunati sumu svih podataka.
3. Napisati potprogram koji množi dva broja. Parametre predati potprogramu preko stoga, te vratiti rezultat isto preko stoga. Nakon završetka potprograma „očistiti“ stog.
4. Napisati program koji će u bloku 32 bitovnih podataka nepoznate duljine sve negativne brojeve postaviti na početak bloka, a sve pozitivne na kraj. Blok je zaključen podatkom 00000000. Početna adresa bloka je 0x1000.
5. Napisati program koji će bloku 32 bitovnih podataka nepoznate duljine izračunati srednju vrijednost. Blok je zaključen podatkom 00000000. Početna adresa bloka je iza koda.
6. Napisati program koji čita niz znakova – *string* (*zero terminated string*) proizvoljne dužine zapisan na adresi iza koda i koji pretvara sva velika slova u mala.
7. U bloku memorije koji slijedi u području iza koda prebrojiti koliko ima neparnih podatka. Brojanje završava prvim podatkom 0.
8. Program u C-u prebaci u asembler:

**int** x = 20, y = 35, rez = 0;  
**if** (x < y)  
 rez = y –x;  
**else**  
 rez = x – y;