LABORATORIJSKE VAJE, 2. letnik - **03 -** Mikroprocesorji in mikrokontrolerji

**Arhitektura računalnikov  
 ATMEL AVR STUDIO**

**ARITMETIČNO LOGIČNE OPERACIJE,   
SKOKI, PSEVDOUKAZI**

# Naloge

Za izvedbo naloge boste potrebovali program Lightshot ali podoben program za zajemanje slike ekrana. Odprite program AVR Studio in ustvarite projekt v zbirniku. Odgovorite na vprašanja in odgovore označite krepko (bold).

## Naloga 1

Spomnite se, kako izgleda podatkovni naslovni prostor v procesorju ATMEGA 328p.   
Izvajajte spodnji program korak za korakom. Kaj opazite? \_\_\_\_\_\_stvar se začne pisati v regostra r8 in 9\_\_\_\_\_\_\_\_\_\_

## 

Kakšna je razlika med ukazoma **lds r0, 0x0010** in **mov r0, r16**? Odgovor se skriva v dokumentaciji!

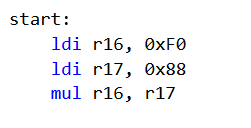
Razlika je v številu ciklov in v načinu kako prenese vrednosti.

Napiši ekvivalenta ukazov (ukaza, ki naredita isto):  
 sts 0x0008, r16 \_\_\_\_\_\_\_\_mov r8, r16\_\_\_\_\_\_\_\_\_  
 sts 0x0009, r17 \_\_\_\_\_\_\_\_mov r9, r17\_\_\_\_\_\_\_\_\_

## Naloga 2

V binarni obliki izračunajte na list papirja, koliko znaša 0x0F0 \* 0x88 ! Rezultat (v hex.): \_\_\_\_ 0x7F80\_\_\_\_\_\_\_

Poglej dokumentacijo in opiši, kako deluje ukaz MUL! \_\_\_\_\_\_\_\_\_množi dve števili



Izvedi program ukaz za ukazom! Primerjaj stanje registrov z rezultatom množenja zgoraj!   
Pojasni sosedu, kaj se je zgodilo!

Kolikšna je velikost faktorjev (v bitih)? \_\_\_\_\_\_\_\_\_\_ Kolikšna je velikost rezultata v bitih? \_\_\_\_\_\_\_\_\_  
Kje in kako vidite rezultat operacije množenja?

## Naloga 3

Preberi spodnje besedilo in na podlagi rezultata zgoraj opredeli, ali ATMEGA 328p uporablja big ali little endian način zapisa števil.

In big-endian format, whenever addressing memory or sending/storing words bytewise, the most significant byte — the byte containing the [most significant bit](https://en.wikipedia.org/wiki/Most_significant_bit) — is stored first (has the lowest address) or sent first, then the following bytes are stored or sent in decreasing significance order, with the least significant byte — the one containing the [least significant bit](https://en.wikipedia.org/wiki/Least_significant_bit) — stored last (having the highest address) or sent last.

Little-endian format reverses this order: the sequence addresses/sends/stores the least significant byte first (lowest address) and the most significant byte last (highest address). Most computer systems prefer a single format for all its data; using the system's native format is automatic. But when reading memory or receiving transmitted data from a different computer system, it is often required to process and translate data between the preferred native endianness format to the opposite format.

Uporablja \_\_\_\_\_little \_\_\_\_\_\_ način zapisa.

## Naloga 4

Vnesite naslednji program in ga izvajajte korak za korakom, vse dokler ne pridete na začetek.   
Ob vsakem koraku zapišite vsebino registra PC in koliko ciklov se je izvajal dotični ukaz.   
Odgovorite na vprašanja.

|  |  |  |
| --- | --- | --- |
| PC program counter | Trajanje v ciklih | Program |
| 0x00000000 |  |  |
| 0x00000050 | 1 |
|  |  |
|  |  |
|  |  |
| 0x00000051 | 1 |
| 0x00000052 | 1 |
| 0x00000053 | 2 |
|  |  |
|  |  |
| 0x00000054 | 2 |
| 0x00000056 | 2 |
| 0x00000058 | 2 |
| 0x00000060 | 2 |
|  |  |
|  |  |
|  |  |
| 0x00000040 | 1 |
| 0x00000041 | 1 |
| 0x00000042 | 2 |
|  |  |
|  |  |
| 0x00000043 | 2 |

### Vprašanja

|  |  |
| --- | --- |
| Koliko je 0x03 << 3? | 0x80 |
| Koliko je 0x01 << 3? | 0x18 |
| Katero operacijo pomeni oznaka << ? | Premakne bite za število mest |
| Se rezultat izraza izračuna med delovanjem programa ali pri prevajanju v strojno kodo? | Pretvori v prevajalniku |
| Čemu služita ukaza rjmp in jmp? | Rjmp je relative jump na neko oznako, jpm pa skače na določeno vrednost |
| .ORG je t.i. psevdo ukaz. Kaj to pomeni? | Da je namenjen prevajalniku |
| Čemu služi .ORG? | Da pove na kater naslov shrani ukaz |
| Če bi oznake nadomestili z vrednostmi PC, kakšne bi bile vrednosti za |  |
| start | 0x0050 |
| cont1 | 0x0054 |
| cont2 | 0x0040 |
| cont3 | 0x0044 |

## Naloga 5

Ukaza LDS (Load from Data Space) in STS (Store to Data Space) sta namenjena branju vrednosti iz podatkovnega naslovnega prostora oz. pisanju vanj. Ukaz BRNE ima enako vlogo, kot ukaz RJMP, le da se skok zgodi le, če je zastavica Z = 0.

Vnesite in preizkusite primer spodaj. Ugotovite, kaj program dela.

|  |  |
| --- | --- |
| .org 0x0000 | Nastavi število ukaza na 0 |
| rjmp start | Skoči na »start« |
|  |  |
| .org 0x0050 | Spremeni število ukaza na 0x50 |
| start: |  |
| ldi r16,0x05 | V register r16 naloži 0x05 |
| sts 0x0100, r16 | V prostor 0x0100 naloži vrednost r16 |
|  |  |
| loop: |  |
| lds r16, 0x0100 | V register r16 naloži vrednost prostora 0x0100 |
| dec r16 | Zmanjša r16 za 1 |
| sts 0x100, r16 | V 0x100 naloži vrednost registra r16 |
|  |  |
| brne loop | Ponavlja loop dokler ne bo rezultat 0 |
|  |  |
| forever: |  |
| rjmp forever | Skaće nazaj na forever do konca |

## Naloga 6 (ponovitev in dodatek):

Napišite program, ki bo seštel dve 16 bitni števili; 0x1234 in 0x9999 in rezultat spravil v RAM na naslov 0x0100 in 0x0101. Preverite rezultat s kalkulatorjem in v razhroščevalniku.

start:

ldi r16, 0x12

ldi r17, 0x23; prvo število

ldi r18, 0x99

ldi r19, 0x99

add r16, r18

adc r17, r18

sts 0x0100, r16

sts 0x0101, r17

rjmp forever

forever:

rjmp forever

## Naloga 7

Z zanko, podobno nalogi 5, naredite del programa, ki bo porabil natanko 10 ciklov (pri uri 1 Mhz je to natanko 10 μs).

start:

ldi r16, 3

pavza:

dec r16

brne pavza

rjmp forever

forever:

rjmp forever