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

**Arhitektura računalnikov  
 ATMEL AVR STUDIO**

**SKOKI, PSEVDOUKAZI, PRIMERJANJE, NASLAVLJANJE - drugič**

# Naloge

## Naloga 1 - Sklad

## PUSH in POP

Sklad je prostor na koncu pomnilnika, ki se uporablja za začasno shranjevanje. Komentirajte spodnji program!

|  |  |
| --- | --- |
| Čemu služi register SP? |  |
| Kako je sestavljen? |  |
| Kakšne so vrednosti registrov po koncu programa? | r16:  r17:  r18: |

#include "m328pdef.inc"

.CSEG

reset: .ORG 0x0000

rjmp init

init: ldi r16, low(RAMEND)

out spl, R16

ldi r16, high(RAMEND)

out sph, R16 ; SP zdaj kaže na:

ldi r16, 0x20

ldi r17, 0x30

ldi r18, 0x40

push r18 ; SP zdaj kaže na:

push r17 ; SP zdaj kaže na:

pop r18 ; SP zdaj kaže na:

push r16 ; SP zdaj kaže na:

pop r17 ; SP zdaj kaže na:

pop r16 ; SP zdaj kaže na:

done: rjmp done

## Še vedno sklad 2

Za spodnji program upam, da še vedno *vemo*, zakaj ne dela.  
Popravite ga tako, da bomo lahko dobili vsoto števil 0x20 in 0x36. Zgledujte se po prejšnjem programu.  
Po operaciji seštevanja morajo biti vsi registri razen r0 nedotaknjeni.

|  |
| --- |
| .CSEG  .ORG 0x0000    mov r0, 0x20  mov r1, 0x36  add r0, r1 ;  ; rezultat mora na koncu biti v r0  forever: rjmp forever |

## Še vedno sklad 3: Umazani triki

Preizkusite naslednji program. Ta trik se pogosto uporablja, kadar se dela reverzni inženiring programov.

|  |  |
| --- | --- |
| Kaj se dogaja?  Zakaj je ta trik umazan? |  |

|  |
| --- |
| #include "m328pdef.inc"  .CSEG  reset: .ORG 0x0000  rjmp init    .org 0x0040  init: ldi r16, low(RAMEND)  out spl, R16  ldi r16, high(RAMEND)  out sph, R16  call nekaj1  nictakega: nop  rjmp nictakega  nekaj3: nop  nop  rjmp nekaj3  nekaj1: pop r16  pop r16  ldi r17, 0x48  push r17  ldi r17, 0x00  push r17  ret    done: rjmp done |

## Naloga 4

Vnesite spodnji program in ga prevedite. Poiščite .HEX datoteko in jo na Arduino naložite s programom XLoader.

Delovanje preizkusite s programom PLINK. Zaženete ga iz ukazne vrstice ali preko .BAT datoteke.

**plink.exe -serial COMn**

kjer je n številka COM porta, enaka 1, 2, ... itd.

; .include "m328pdef.inc" ; mcu je ATMega328p

.equ baudrate = 9600

.org 0x0000

start: rjmp program

.org 0x0080

hello: .db "Hello G2x", 0x0d, 0x0a, 0

.org 0x0120 ; oznaka program mora biti obvezno za org, èe hoèemo, da org "prime"

program: ser r16 ; portb v celoti izhod

out ddrb, r16

clr r16

out portb, r16 ; vse bite postavimo na 1

call setupUART ; pripravimo UART vrata

;hello world

again: ldi ZH, high(hello<<1)

ldi ZL, low(hello<<1)

ponovi: lpm r16, Z+

cpi r16, 0

breq zanka

call posljiznak

rjmp ponovi

zanka: call sprejmiznak ; poèakajmo na kak znak iz USART, da vemo, kdaj je druga stran pripravljena

in r16, portb

ldi r17, 0xFF

eor r16, r17

out portb, r16

rjmp again

cave\_johnson: rjmp cave\_johnson ; we are done here

byte2hex: ; podprogram v r17 in r16 vrne ascii vrednost vsebine r16

; npr. za vrednost 0x4B bo vrnil v r16 0x42, tj. 'B' in v r17 vrednost 0x34, tj. '4'

; r17 zgornji del, r16 spodnji del zloga

mov r17, r16

swap r17 ; najprej bomo obdelali zgornji del vrednosti

andi r17, 0x0F ; pobrišemo zgornje 4 bite

ori r17, 0x30 ; nalepimo èez 0x30, zdaj bo to ASCII cifra od '0' do '9', vendar samo, za vrednosti od 0 do 9

; za višje bo to znak, ki pa ni HEX številka ; ker mora biti to èrka od A do F, prištejemo vrednost 7,

; npr. da pridemo v primeru 3A (':') do vrednosti 0x41 ('A')

; ali ko bo vrednost 3F, vrednost 46 ('F').

cpi r17, 0x3A ; primerjamo, èe smo presegli '9'

brlo ni\_crka ; èe je manjše, potem ne

subi r17, -7 ; zloraba, ker nimamo addi

ni\_crka: andi r16, 0x0F ; z zgornjim delom smo opravili in je sedaj v r17, še spodnji del

ori r16, 0x30

cpi r16, 0x3A

brlo koncano

subi r16, -7

koncano: ret

setupUART: ; pripravi serijska vrata

; ne sekirajte se, èe še ne veste, èesa vsega ne poznate

; tu je zato, da primer dela tudi v resnici

; vrednosti not npr. TXEN0 in UBRR0H dobimo v datoteki m328pdef.inc

; in so lahko drugaène glede na model mikrokontrolerja

; 9600 baudov @ 16Mhz

; v IO register UBBR0 naložimo baudrate / 8

ldi r16, 0x67

ldi r17, 0x00

sts UBRR0L, r16

sts UBRR0H, r17

; RXEN0 = 4, TXEN0 = 3, znak << pomeni pomik v levo, >> desno

; 1<<TXEN0 = 0x40 (0b00000001 --> 0b00001000)

ldi r16, (1<<RXEN0)|(1<<TXEN0) ; vkljuèimo bita za sprejem in za oddajo

sts UCSR0B,r16

ldi r16, (3<<UCSZ00) ; 8 bitov, 1 stop bit

sts UCSR0C,r16

clr r16

sts UCSR0A, r16

ret

; procedura pošlje znak v r17 po UART

; poèakajmo, da bo oddajni vmesni pomnilnik na voljo

posljiznak: ; takrat bo bit UDRE0 enak 1

push r18

lds r18, UCSR0A

sbrs r18,UDRE0 ; preskoèi naslednjo instrukcijo, èe je ta bit 1

rjmp posljiznak ; bit še ni 1, skoèi nazaj

; daj podatek (r16) v vmesni pomnilnik,

sts UDR0,r16

pop r18

ret

; procedura sprejme znak po UART v r16

sprejmiznak: ; poèakajmo na znak

push r18

lds r18, UCSR0A

sbrs r18, RXC0

rjmp sprejmiznak

lds r16, UDR0

pop r18

vrnise: ret