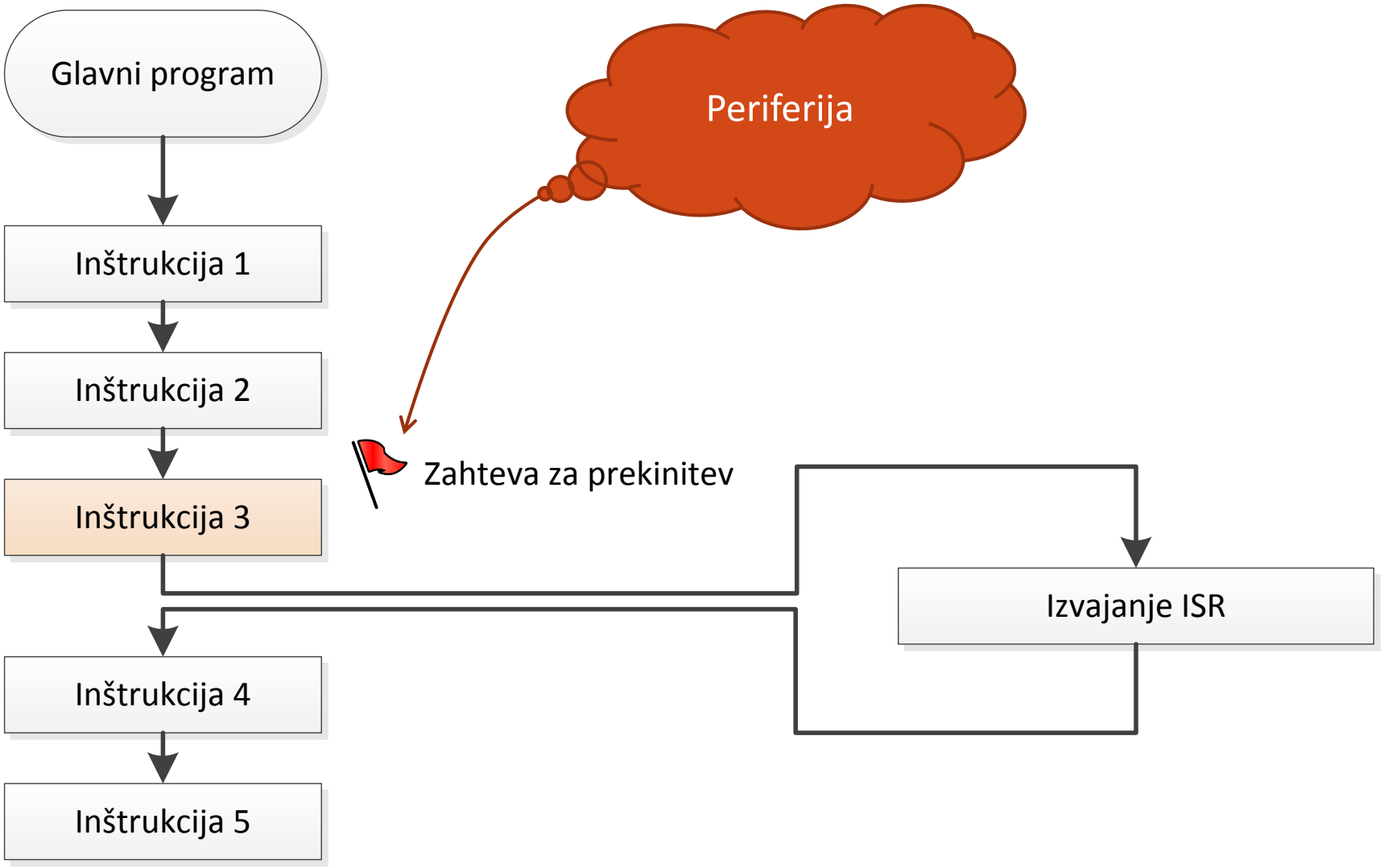
# Osnove mikroprocesorske elektronike

doc. dr. Marko Jankovec

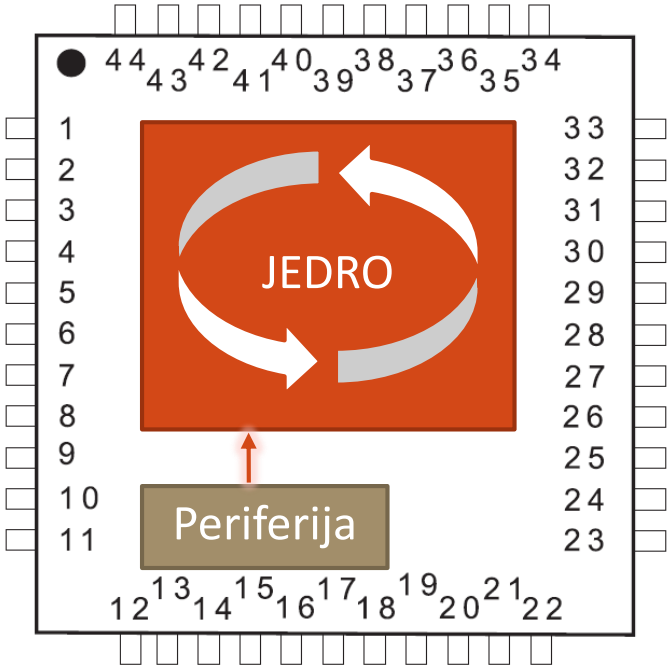
Prekinitve

# Kaj je to prekinitev (interrupt)?

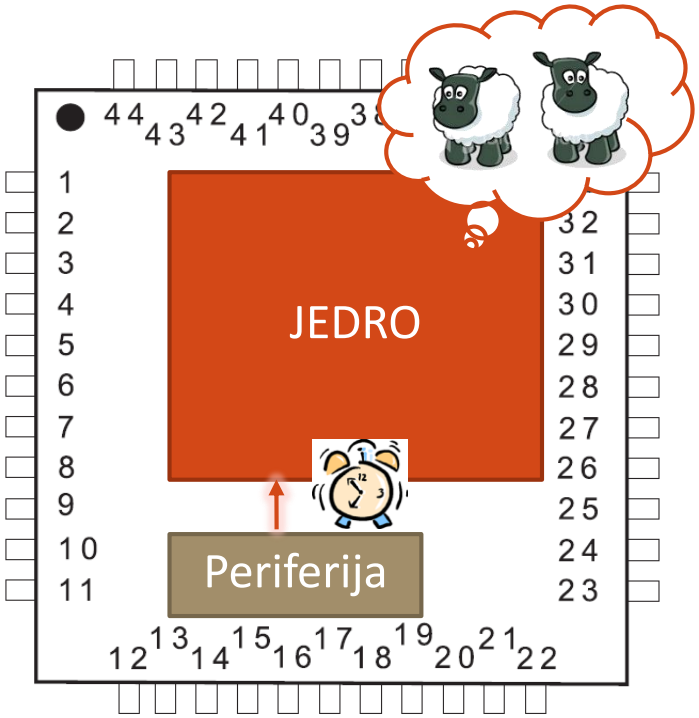


# Pozivanje (polling) in prekinitve (interrupts)

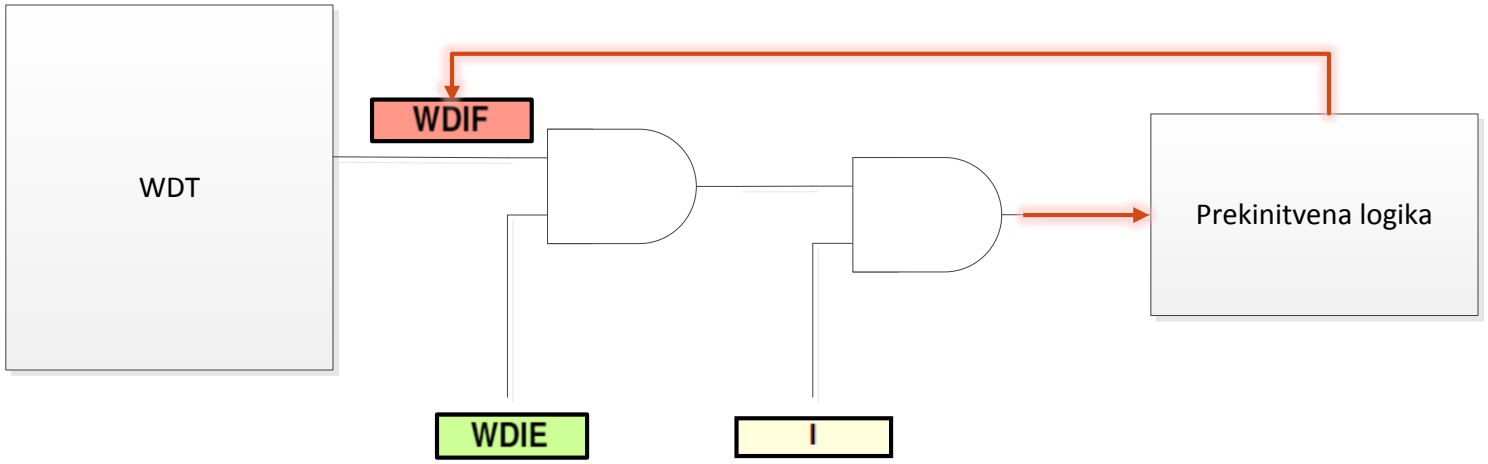
## Pozivanje (polling)




## Prekinitve (interrupts)



# Omogočanje in indikacija prekinitev (primer WDT)




| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x60) | WDIF | WDIE | WDP3 | WDCE | WDE | WDP2 | WDP1 | WDP0 | WDTCSR |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | |

| 0x3F (0x5F) | I | T | H | S | V | N | Z | C | SREG |
|---|---|---|---|---|---|---|---|---|---|
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

# Sklad (Stack)

| Word | Program Memory |
|---|---|
| 000 | ..... |
| 001 | push r15 |
| 002 | push r22 |
| 003 | nop |
| 004 | nop |
| 005 | nop |
| 006 | ..... |
| 007 | ..... |
| 008 | ..... |

Slice of SRAM space

| 0060 | 0061 | 0062 | 0063 | 0064 | 0065 | 0066 | 0067 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| 0068 | 0069 | 006A | 006B | 006C | 006D | 006E | 006F |
| | | | | | | | |
| 0070 | 0071 | 0072 | 0073 | 0074 | 0075 | 0076 | 0077 |
| | | | | | | | |
| 0078 | 0079 | 007A | 007B | 007C | 007D | 007E | 007F |
| | | | | | | | 47 |
| 0080 | 0081 | 0082 | 0083 | | | | |

| Word | Program Memory |
|---|---|
| 000 | ..... |
| 001 | pop r22 |
| 002 | pop r15 |
| 003 | nop |
| 004 | nop |
| 005 | nop |
| 006 | ..... |
| 007 | ..... |
| 008 | ..... |

| General Purpose Registers | | | | | | | |
|---|---|---|---|---|---|---|---|
| R0 | | R8 | | R16 | | R24 | |
| R1 | | R9 | | R17 | | R25 | |
| R2 | | R10 | | R18 | | R26 ($X_L$) | |
| R3 | | R11 | | R19 | | R27 ($X_H$) | |
| R4 | | R12 | | R20 | | R28 ($Y_L$) | |
| R5 | | R13 | | R21 | | R29 ($Y_H$) | |
| R6 | | R14 | | R22 | A3 | R30 ($Z_L$) | |
| R7 | | R15 | 47 | R23 | | R31 ($Z_H$) | |

Stack Pointer
$007F

Program Counter
00000001

Microschematic

# Prekinitveni vektorji

PRIORITETA IZVAJANJA

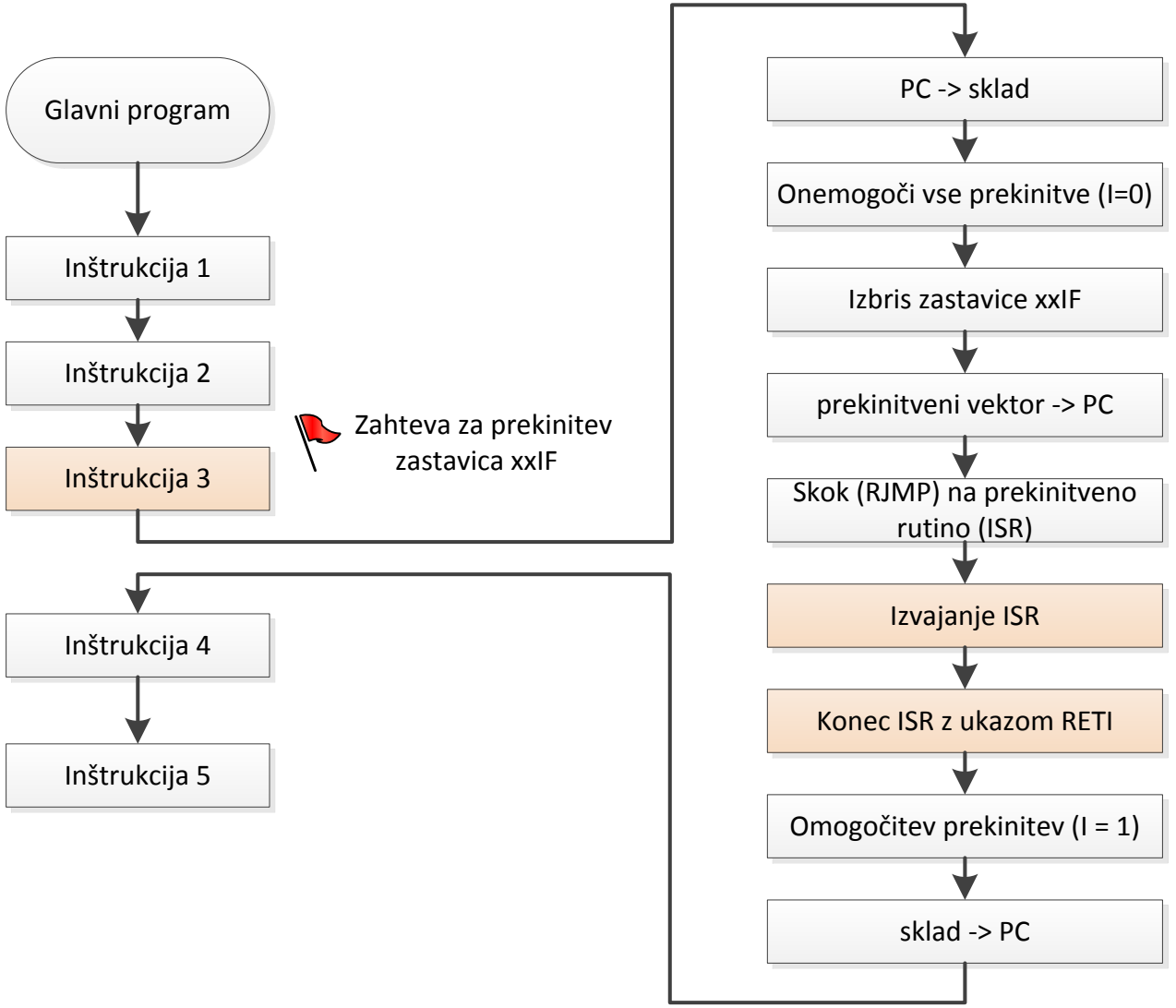
| Naslov | Vektor | Vir |
|---|---|---|
| 0x0000 | RESET | Zunanji RESET, POR, BOR, WDT RESET, JTAG RESET |
| 0x0002 | EXT_INT0 | Zunanja prekinitev 0 |
| 0x0004 | EXT_INT1 | Zunanja prekinitev 1 |
| 0x0006 | EXT_INT2 | Zunanja prekinitev 2 |
| 0x0008 | PCINT0 | Prekinitev ob spremembi stanja pina na PORTA |
| 0x000A | PCINT1 | Prekinitev ob spremembi stanja pina na PORTB |
| 0x000C | PCINT2 | Prekinitev ob spremembi stanja pina na PORTC |
| 0x000E | PCINT3 | Prekinitev ob spremembi stanja pina na PORTD |
| 0x0010 | WDT | Prekinitev WDT |
| 0x0012 | TIMER2_COMPA | Prekinitev TC2 ob dogodku COMPARE MATCH A |
| 0x0014 | TIMER2_COMPB | Prekinitev TC2 ob dogodku COMPARE MATCH B |
| … | … | … |

# Primer vsebine programskega pomnilnika

```
0x0000      jmp  0x003E    ;RESET
0x0002      jmp  0x????    ;INT0
…           …
0x003A      jmp  0x????    ;USART1 UDRE
0x003C      jmp  0x????    ;USART1 TXC

0x003E      ldi  r16, high(RAMEND)
0x003F      out  SPH, r16
0x0040      ldi  r16, low(RAMEND)
0x0041      out  SPL, r16
0x0042      sei
…                 …
```

# Podroben potek izvajanja prekinitve



# Primer prekinitve

```
 ldi R16, 0xFF
loop:
 dec R16
 brne loop
 …
```

ISR

```
 in R0, PIND
 cpi R0,0x00
 brne konec
 out PORTB,0xFF
konec:
 reti
```

Vpliva na zastavico „z“

Vrednost zastavice „z“ ni prava

Ali:

- V ISR ne uporabljati inštrukcij, ki vplivajo na SREG
- Shraniti SREG začasno na sklad

# Latenca prekinitev

- Čas od zahteve za prekinitev do izvedbe koristnega dela ISR:
  - Izvedba trenutne inštrukcije, ki se prekine (1 .. 2 cikla)
  - 5 ciklov za skok na prekinitveno rutino
  - Shranitev registrov na sklad
    - Samo za SREG potrebujemo 5 ciklov

```
push R0      ;2 cikla
in R0,SREG   ;1 cikel
push R0      ;2 cikla
```

# Brisanje zastavic xxIF

- zastavice xxIF lahko zbrišemo programsko tako, da vanjo napišemo logično „1“
  - zato, da za izbris zastavice ni potrebnega maskiranja
  - izvede se v enem ciklu (True read-while-write operation)

```
WDTCSR=0X80;
```

```
ldi R16, 0x80
sts WDTCSR, R16
```

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x60) | WDIF | WDIE | WDP3 | WDCE | WDE | WDP2 | WDP1 | WDP0 | WDTCSR |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | |

# Neželeno brisanje zastavic xxIF

- Maskiranje
  - Sprememba enega bita v registru
  - Zastavico xxIF brišemo, če vpišemo logično „1“

```
WDTCSR|=0X10;
```

```
lds R16, WDTCSR
ori R16,0x10   ;postavi 4 bit (WDCE)
sts WDTCSR, R16
```

WDT postavi zastavico za prekinitev WDIF

Tukaj bi jo nevede zbrisali

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x60) | WDIF | WDIE | WDP3 | WDCE | WDE | WDP2 | WDP1 | WDP0 | WDTCSR |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | |

# Rešitev neželenega brisanja zastavic xxIF

```
char x;
x = WDTCSR;
x |= 0x10;
x &= ~0x80;
WDTCSR = x;
```

WDT je postavil zastavico za prekinitev WDIF.

```
lds R16, WDTCSR
ori R16,0x10  ;postavi 4 bit (WDCE)
andi R16,0x7F ;izbriši 7 bit (WDIF)
sts WDTCSR, R16
…
```

Tukaj jo ne zbrišemo.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x60) | WDIF | WDIE | WDP3 | WDCE | WDE | WDP2 | WDP1 | WDP0 | WDTCSR |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | |

# Zakaj onemogočiti vse prekinitve?

- Če gre za časovno kritično operacijo, ki se je ne sme prekiniti
- Nastavljanje perifernih enot lahko proži neželene zahteve po prekinitvi
- Ob branju/pisanju 16-bitnih I/O registrov
  - za branje/pisanje vrednosti potrebujemo dva urina cikla
  - vmes se vrednost registra ne sme spremeniti (race condition)

# Primer: pisanje v register WDTCSR

- Najprej vpišemo WDCE=1, ostale 0
- Nato moramo vsaj v času 4 urinih ciklov vpisati vsebino

```
unsigned char sreg;
sreg = SREG;  //shrani vrednost SREG
cli();     //onemogoči prekinitve
WDTCSR = 0x10; //postavi bit WDCE
WDTCST= 0x29; //vpiši želeno vsebino registra
SREG = sreg;  //omogoči prekinitve, če so bile že prej omogočene
```

```
ldi    r16, 0x10      ; postavi WDCE
ldi    r17, 0x29      ; želena vsebina registra, 1 cikel
in     r18,SREG       ; shrani SREG (zastavico I)
cli                   ; onemogoči prekinitve
sts    WDTCSR, r16    ; vpiši vsebino, 2 cikla
sts    WDTCSR, r17    ; vpiši vsebino, 2 cikla
out    SREG,r18       ; naloži nazaj vsebino SREG
```

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x60) | WDIF | WDIE | WDP3 | WDCE | WDE | WDP2 | WDP1 | WDP0 | WDTCSR |

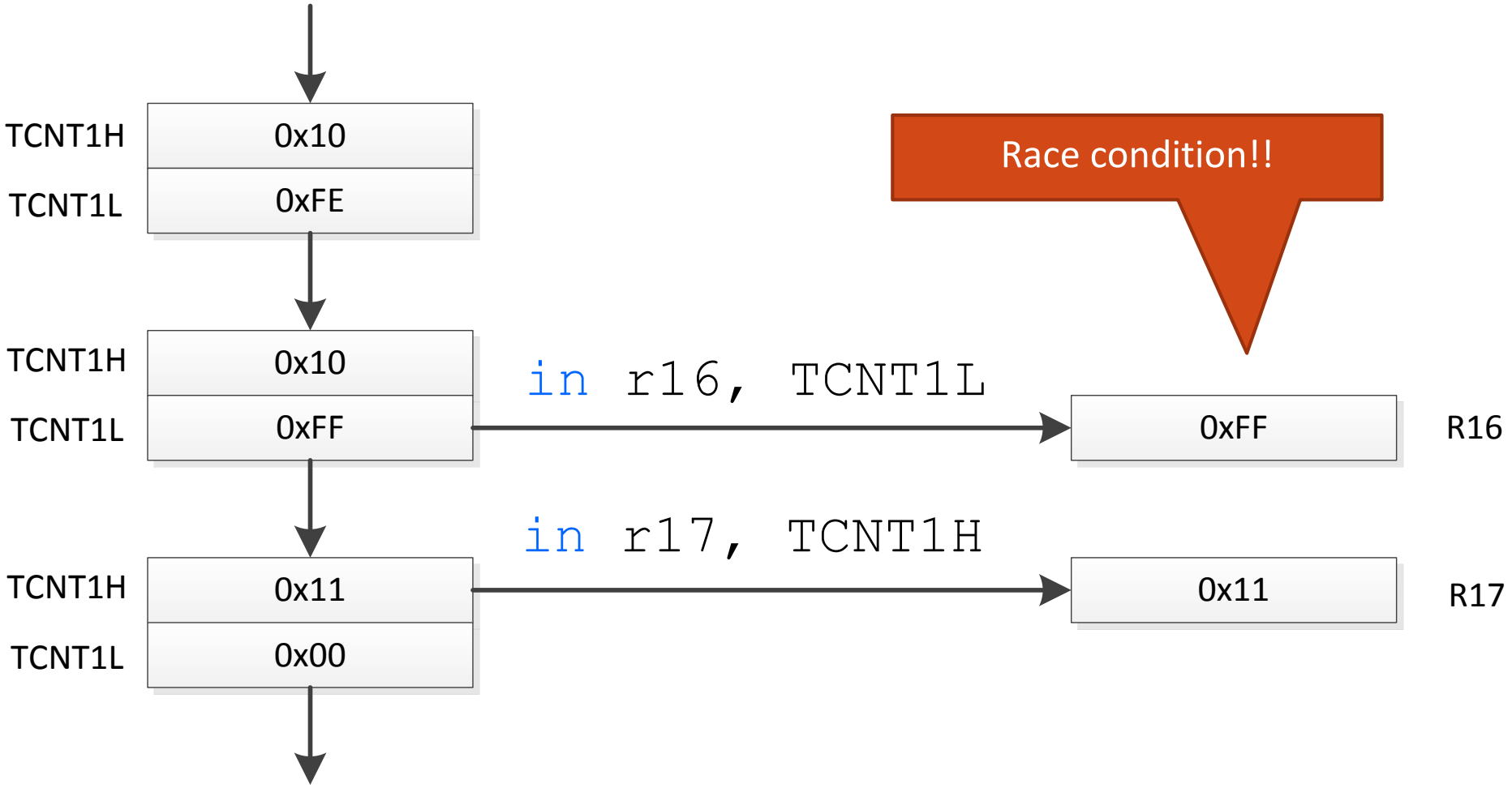
# Sprememba načina delovanja zunanje prekinitve

```
unsigned char sreg;
sreg = SREG;  //shrani vrednost SREG
cli();     //onemogoči prekinitve
EICFRA = 0x02; //nastavi drug način proženja
EIFR = 0x01; //če se je sprožila zahteva za prekinitev, pobriši IF
SREG = sreg;  //omogoči prekinitve, če so bile že prej omogočene
```

```
in     r18,SREG       ; shrani vrednost SREG
cli                   ; onemogoči prekinitve
ldi    r16, 0x02      ; ISC00=1 in ISC01=1 (rising edge)
out    EICRA, r16     ; vpiši vsebino v EICRA
ldi    r16, 0x01      ; INTF0 = 1 (pobriši zastavico)
out    EIFR, r16      ; vpiši vsebino v EIFR
out    SREG,r18       ; naloži nazaj vsebino SREG
```

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x69) | – | – | ISC21 | ISC20 | ISC11 | ISC10 | ISC01 | ISC00 | EICRA |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| 0x1C (0x3C) | – | – | – | – | – | INTF2 | INTF1 | IINTF0 | EIFR |

# Branje vrednosti 16-bitnega števca TCNT1



# Branje vrednosti 16-bitnega števca TCNT1

```
unsigned char sreg;
unsigned int count;
sreg = SREG;  //shrani vrednost SREG
cli();     //onemogoči prekinitve
i=TCNT1;
SREG = sreg;  //omogoči prekinitve, če so bile že prej omogočene
```

```
in    r18,SREG     ; shrani vrednost SREG
cli                ; onemogoči prekinitve
in    r16, TCNT1L  ; preberi spodnji byte
in    r17, TCNT1H  ; preberi zgornji byte
out   SREG,r18     ; naloži nazaj vsebino SREG
```