OPASKA!

- Ovi materijali namijenjeni su isključivo studenticama/studentima koji su upisali predmet “Računala i procesi” na FER-u u šk. g. 2002/2003.
- Za svako drugo korištenje potrebna je pismena suglasnost autora!
- Materijali služe kao pomoć u praćenju predavanja, a ne kao njihova zamjena te se ne mogu tumačiti izvan konteksta predavanja!

M. Žagar, 2002-10-01

ⰮⰀⱃⰹⱁ ⰶⰀⰳⰀⱃ

ⱄⰲⰵⱆⱍⰹⰾⰹⱎⱅⰵ ⱆ ⰸⰀⰳⱃⰵⰱⱆ

ⱇⰀⰽⱆⰾⱅⰵⱅ ⰵⰾⰵⰽⱅⱃⱁⱅⰵⱈⱀⰹⰽⰵ ⰹ ⱃⰀⱍⱆⱀⰀⱃⱄⱅⰲⰀ

(ⱇⰵⱃ)

ⱃⰀⱍⱆⱀⰀⰾⰀ ⰹ ⱂⱃⱁⱌⰵⱄⰹ

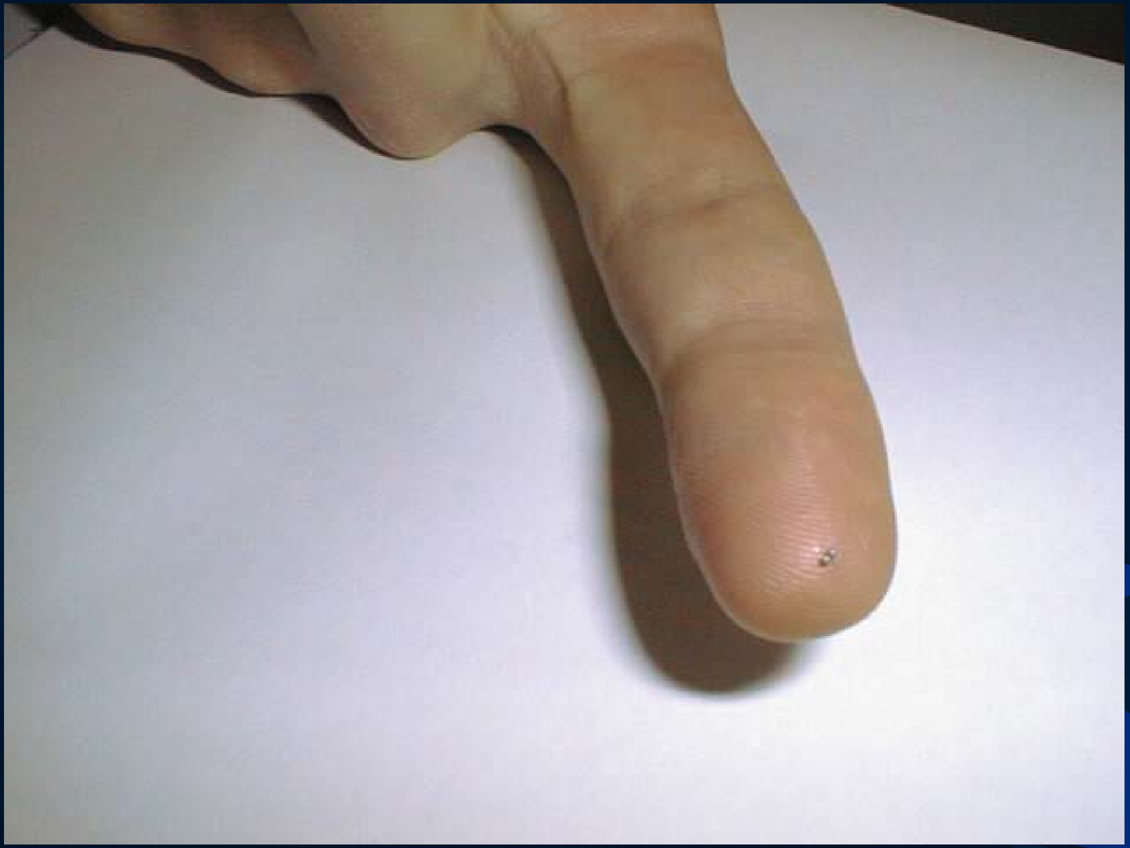
Mario Žagar

Sveučilište u Zagrebu

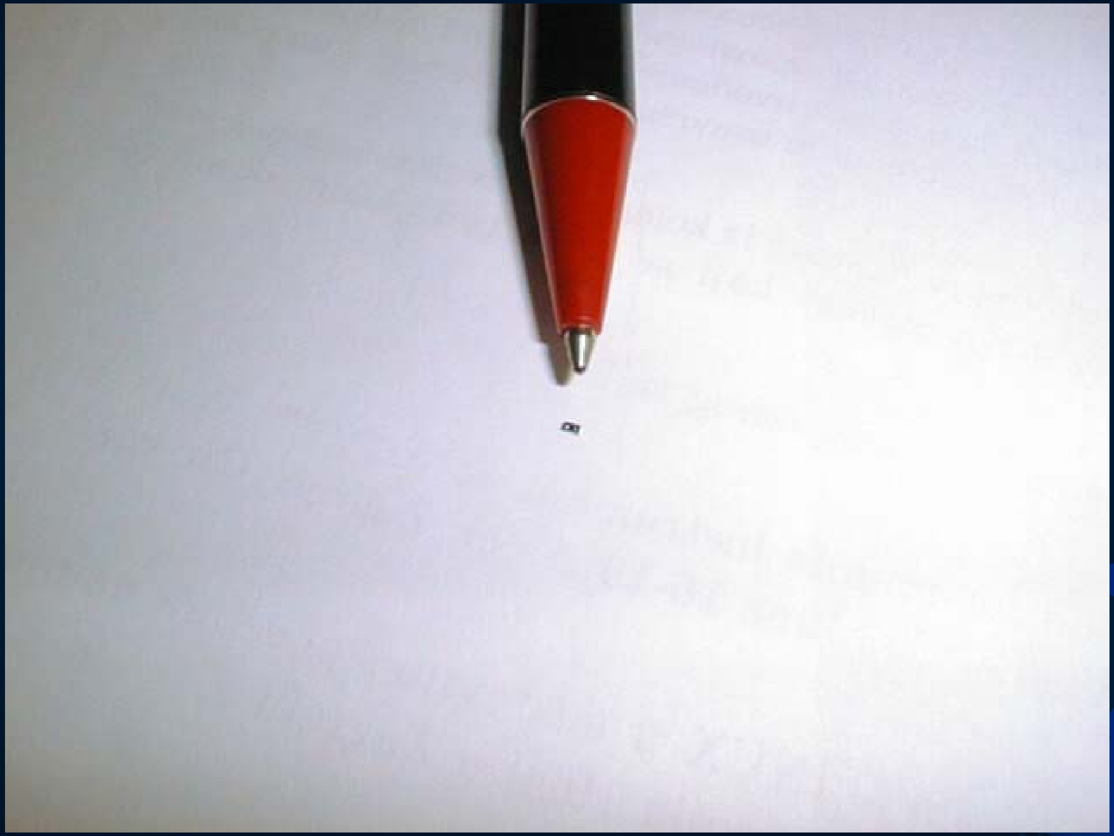
Fakultet elektrotehnike i računarstva (FER)

12. RIP - programiranje mikroračunala

Predgovor I



Predgovor II



Predgovor III

Programsko inženjerstvo
može biti i nešto vrlo beznačajno :-)

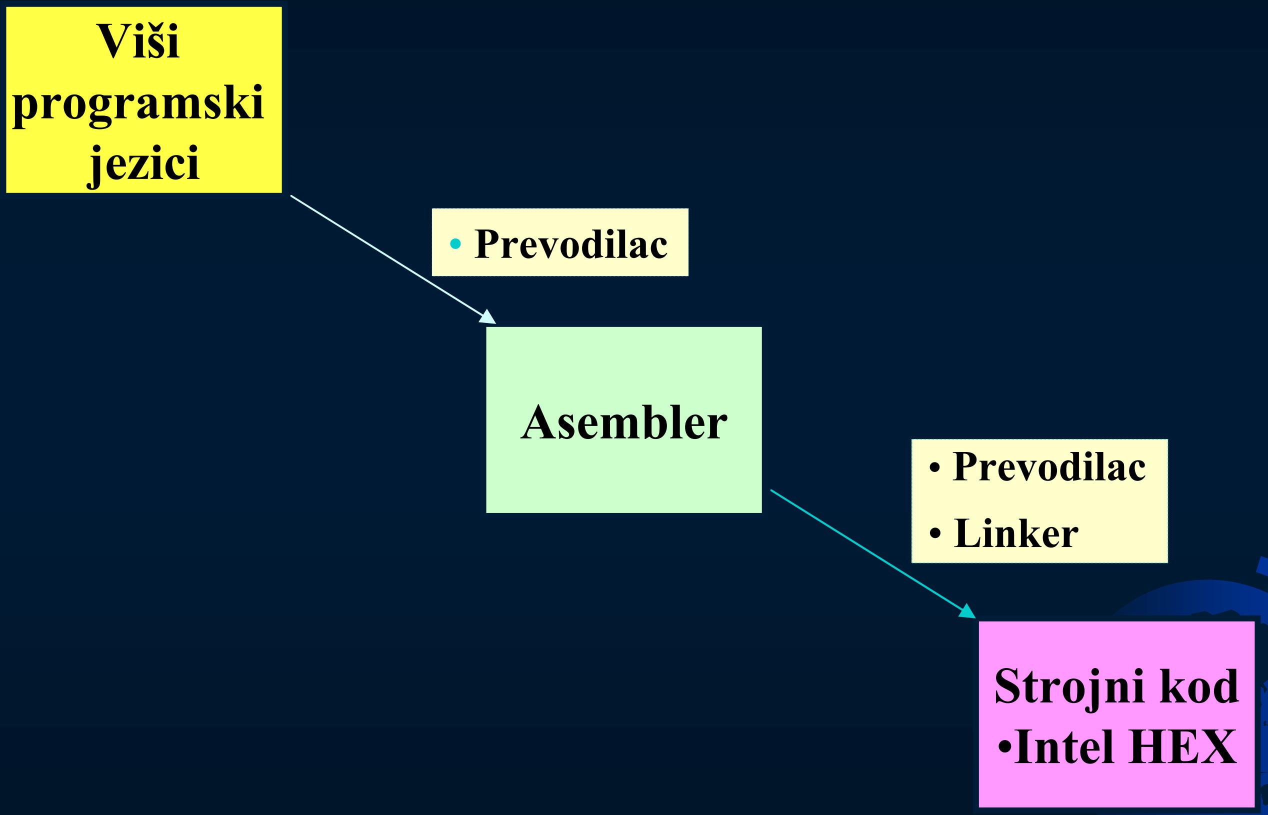
Klasično programiranje (I)

- UNIX, syst. prog., ljuske, filtari,....
- C, cc, ln, as, a.out, dbx, od, ...
- make, yacc, lex, time, prof, gprof,...
- sccs, nroff, bibl, vi, sort,.....
- HTML, WWW, cgi,.....
- Znanje koje se podrazumijeva !
- **lit.:**
  - **M.Ž., UNIX i kako ga koristiti,**
  - **M.Ž., UNIX i kako ga iskoristiti**

Klasično programiranje (II)

- WIN 3.11, Windows95, Windows98, NT, Windows 2000
- VisualBasic, C++, Java
- X11, Motif,
- Baze (SQL), G, ...

Razine programa



Viši programski jezici

- **Formalni opisni jezik kojim programer opisuje što mikroprocesor treba napraviti.**
- **Notacija kojom se programer služi lakša je za snalaženje nego u asembleru.**
- **Svaka naredba u višem programskom jeziku odgovara nizu naredaba u strojnom jeziku.**

U nastavku, primjeri programskog jezika C, C--

Asembler

- Programski jezik koji omogućava programeru pisanje programa koristeći mnemonike,
- Mnemonici - razina strojnog koda,
- Assembler je simbolički programski jezik niske razine (eng. LOW-LEVEL LANGUAGE),
- Omogućava korištenje mnemonika, makro naredaba, labela, ključnih riječi (npr.: ORG, DW, DB ...),
- Omogućava jednostavan pristup svim registrima mikroračunala.

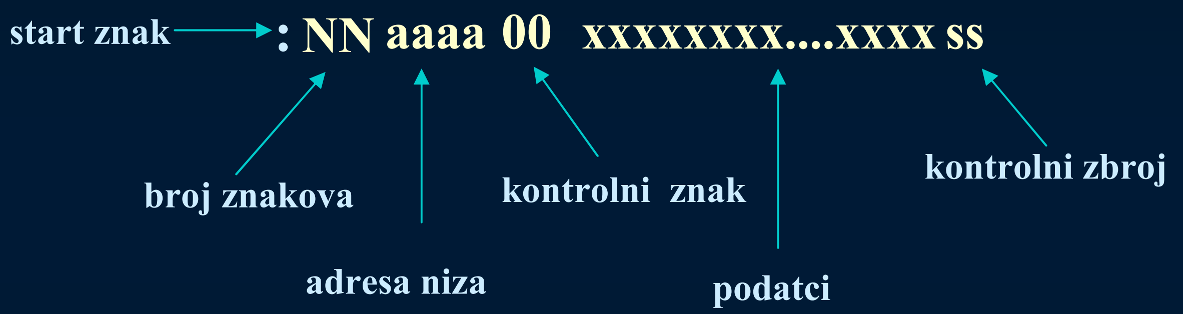
U nastavku primjeri za Z80, i8051 i PIC16c54

Strojni kod

- **Osnovni programski jezik koje mikroprocesor može izvršavati bez prevođenja.**
- **Sastoji se od niza brojeva koji predstavljaju naredbe mikroprocesora**
- **Za zapis strojnog koda često se koristi INTEL HEX FORMAT (ASCII)**

FORMAT INTEL HEX

format zapisa:



- ***start znak*** - **početak svakog retka**
- ***broj znakova*** - **broj znakova koje se upisuju na adresu *aaaa***
- ***adresa niza*** - **logička adresa niza u memoriji počevši od nule**
- ***kontrolni zbroj*** - **broj koji dodan ukupnom 8-bitnom zbroju niza daje zbroj nula**

FORMAT INTEL HEX

```
:20 01 00 00 F3 ...
:20 01 20 00 30 ...
:20 01 40 00 21 ...
:20 01 60 00 D6 ...
:01 01 80 00 87 F7
:00 01 00 01 FE
:BBAAAACCDDDDD...DDDDSS
```

```
  00
  01
  00
+01
-----
  02
  FD
+  1
-----
  FE
```

Z80 I8051 PIC16c54

```
c = 10;     /* a je tipa int*/
```

Z80:

```
LD      HL,0
ADD     HL,SP
EX      DE,HL;;
LD      HL,10
LD      A,L
LD      (DE),A
```

I8051:

```
MOV     R7,0x0A
```

PIC16c54:

```
MOVLW   0Ah
MOVWF   ?(a_main+4)
```

```
a++ ;
```

Z80:

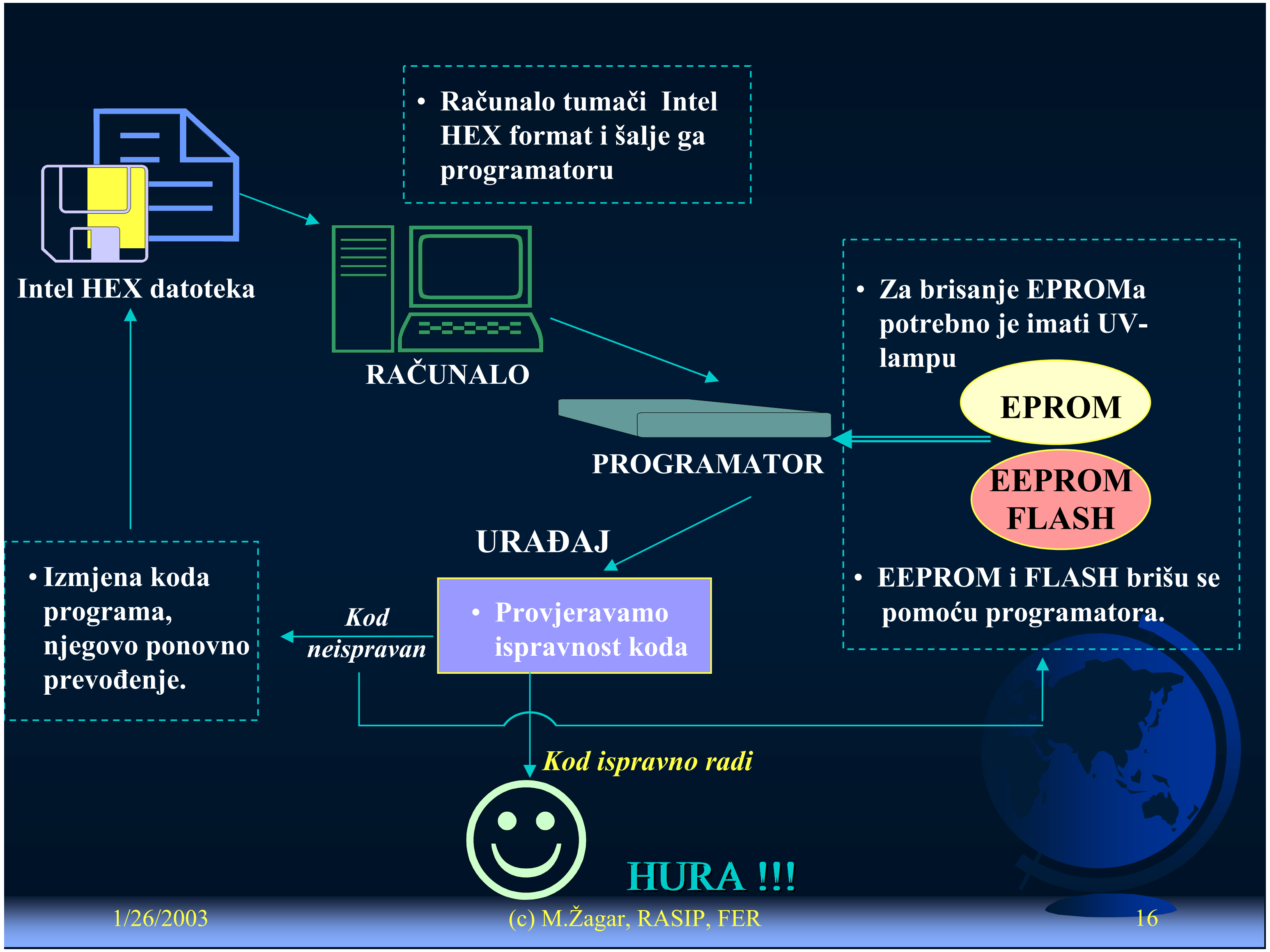
```
LD      HL,3
ADD     HL,SP
LD      D,H
LD      E,L
CALL    CCGINT##
INC     HL
CALL    CCPINT##
DEC     HL
```

I8051:

```
 INC    R5
 CJNE   R5,#0x00,L20
 INC    R4
L20:  NOP
```

PIC16c54:

```
INCF    ?(a_main+0)
BTFSC   status,2
INCF    ?(a_main+0+1)
```



- Računalo tumači Intel HEX format i šalje ga programatoru

- Za brisanje EPROMa potrebno je imati UV-lampu

- EEPROM i FLASH brišu se pomoću programatora.

- Izmjena koda programa, njegovo ponovno prevođenje.

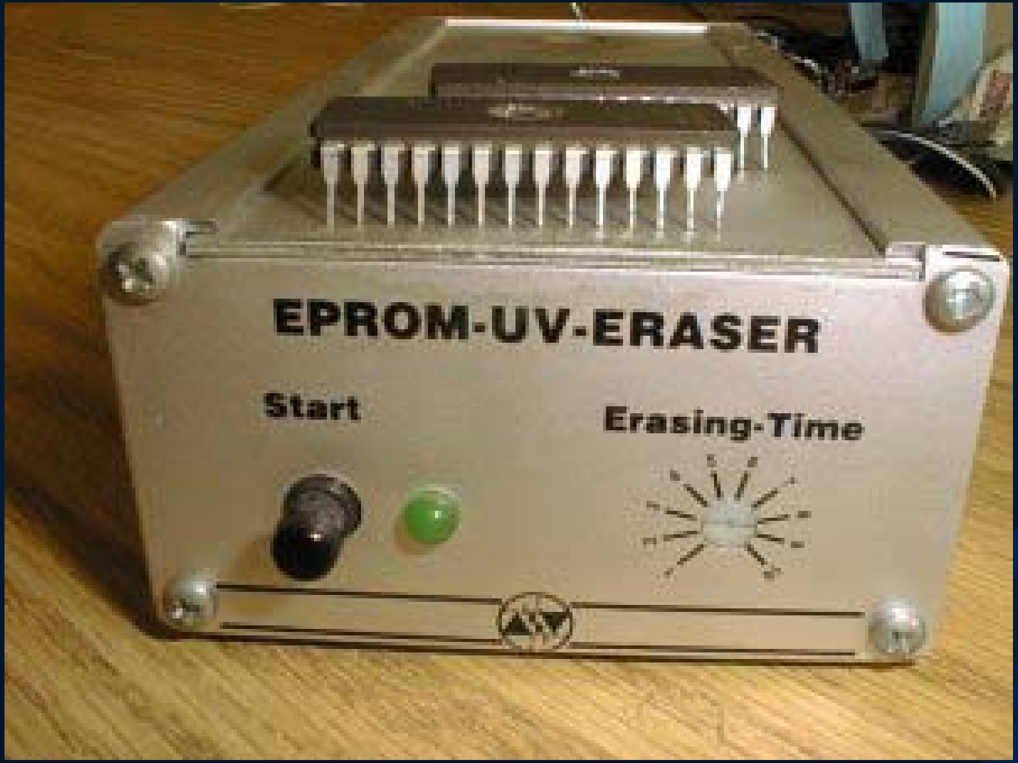
- Provjeravamo ispravnost koda

PROGRAMIRANJE PIC-16c54

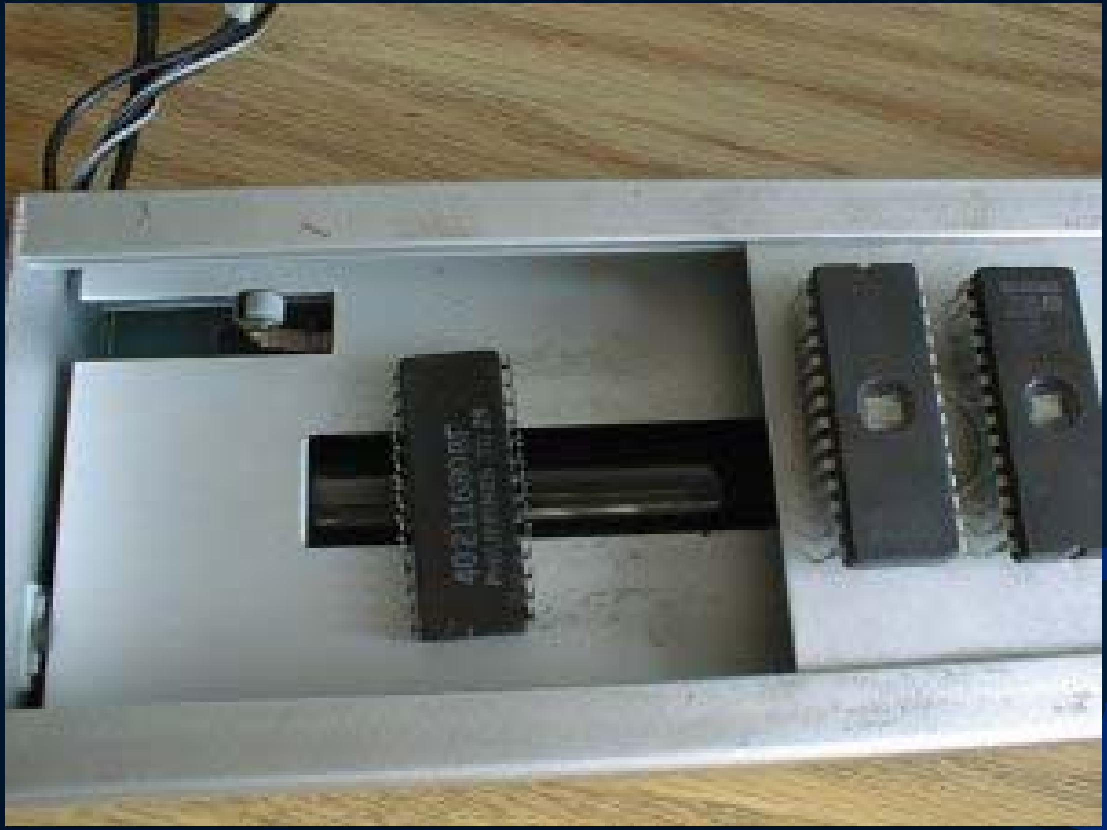
•*Za programiranje se koriste sljedeće linije*

Naziv pina	Tijekom Programiranja		
	Naziv pina	Tip pina	Opis pina
T0CKI	/PROG-VER	I	
RA0 – RA3	D0 – D3	I/O	Ulaz izlaz podataka
RB0 – RB7	D4 – D11	I/O	Ulaz izlaz podataka
OSC1	INCPC	I	Uvećavanje prog. brojila
/MCLR	V_{PP}	P	Napon programiranja
V_{DD}	V_{DD}	P	Napon napajanja
V_{SS}	V_{SS}	P	Masa

Brisalica EPROM-a



Brisalica EPROM-a iznutra



Programator EPROM-a (HI-LO)

Programator EPROM-a (domaći)

PROGRAMIRANJE PIC-16c54

•***Postupak programiranja:***

- **Podižemo napon na pinu /MCLR od 0V do 13V i držimo pin T0CKI na 5V.**
- **Programski brojač se postavlja u “0xFFF”, zato što je na početku /MCLR na 5V što predstavlja reset procesora.**
- **Pulsiranjem OSC1 pina povećavamo programsko brojilo**
- **Spuštanjem pina T0CKI na 0V pohranjujemo podatak na pinovima D0 - D11 u EPROM**
- **Nakon što programsko brojilo dođe do zadnje lokacije “0x1FF”, njegovo daljnje povećavanje znači adresiranje funkcijskog dijela EPROMA**

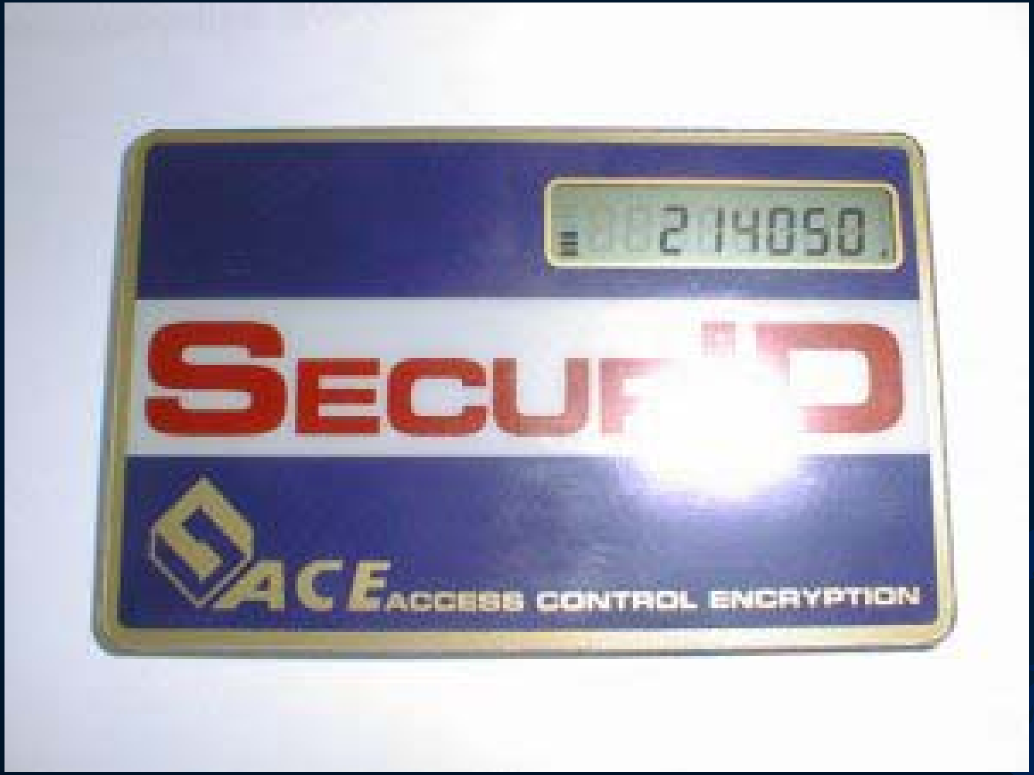
PIC16C54 - u kutiji šibica

Mikrokontroleri na sve strane (I)

Mikrokontroleri na sve strane (II)

Mikrokontroleri na sve strane (III)

Mikrokontroleri na sve strane (IV)



PIC16C7X (I)

PIC16C7X - 8-Bit CMOS Microcontrollers with A/D Converter

PIC16C7X Microcontroller Core Features:

- High-performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
- DC - 200 ns instruction cycle
- Up to 8K x 14 words of Program Memory, up to 368 x 8 bytes of Data Memory (RAM)
- Interrupt capability
- Eight level deep hardware stack
- Direct, indirect, and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation

PIC16C7X (II)

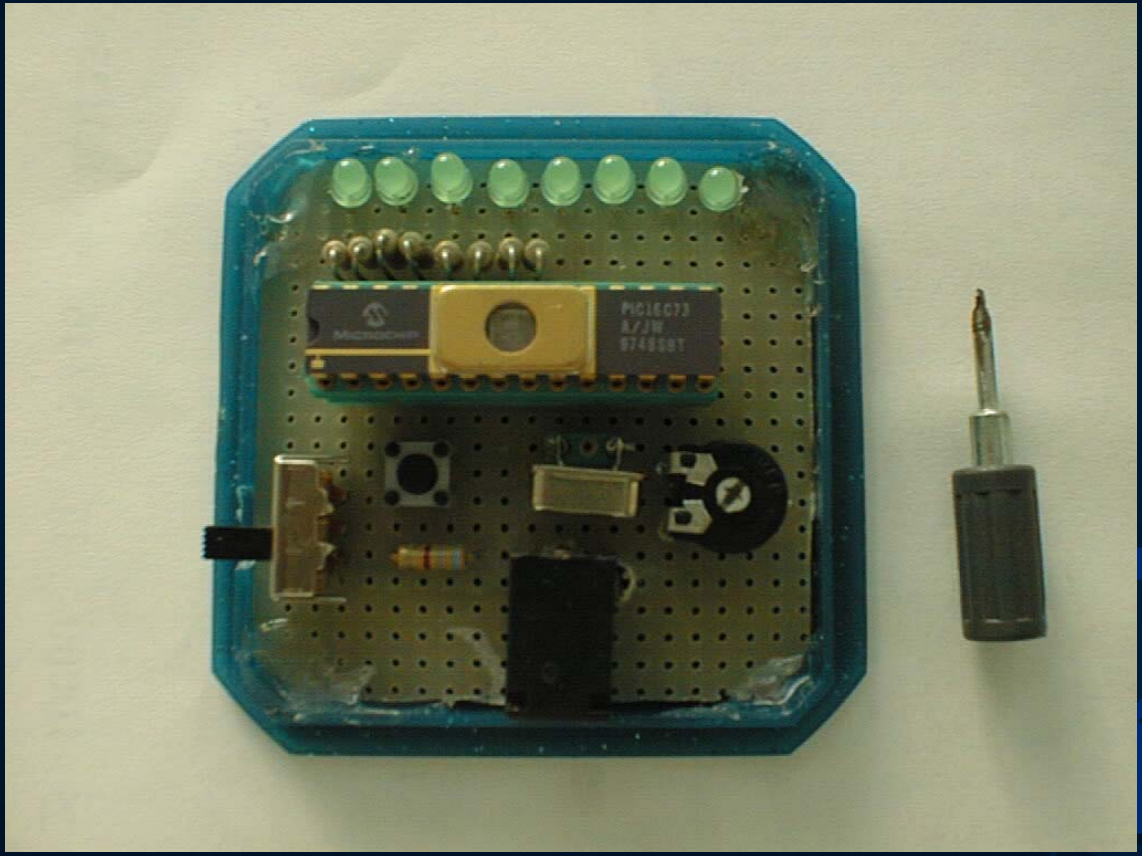
- Programmable code-protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low-power, high-speed CMOS EPROM technology
- Fully static design
- PIC16C72, PIC16C73, PIC16C73A , PIC16C74, PIC16C74A, PIC16C76, PIC16C77
- Wide operating voltage range: 2.5V to 6.0V
- High Sink/Source Current 25/25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
    - < 2 mA @ 5V, 4 MHz
    - 15 mA typical @ 3V, 32 kHz
    - < 1 mA typical standby current

PIC16C7X (III)

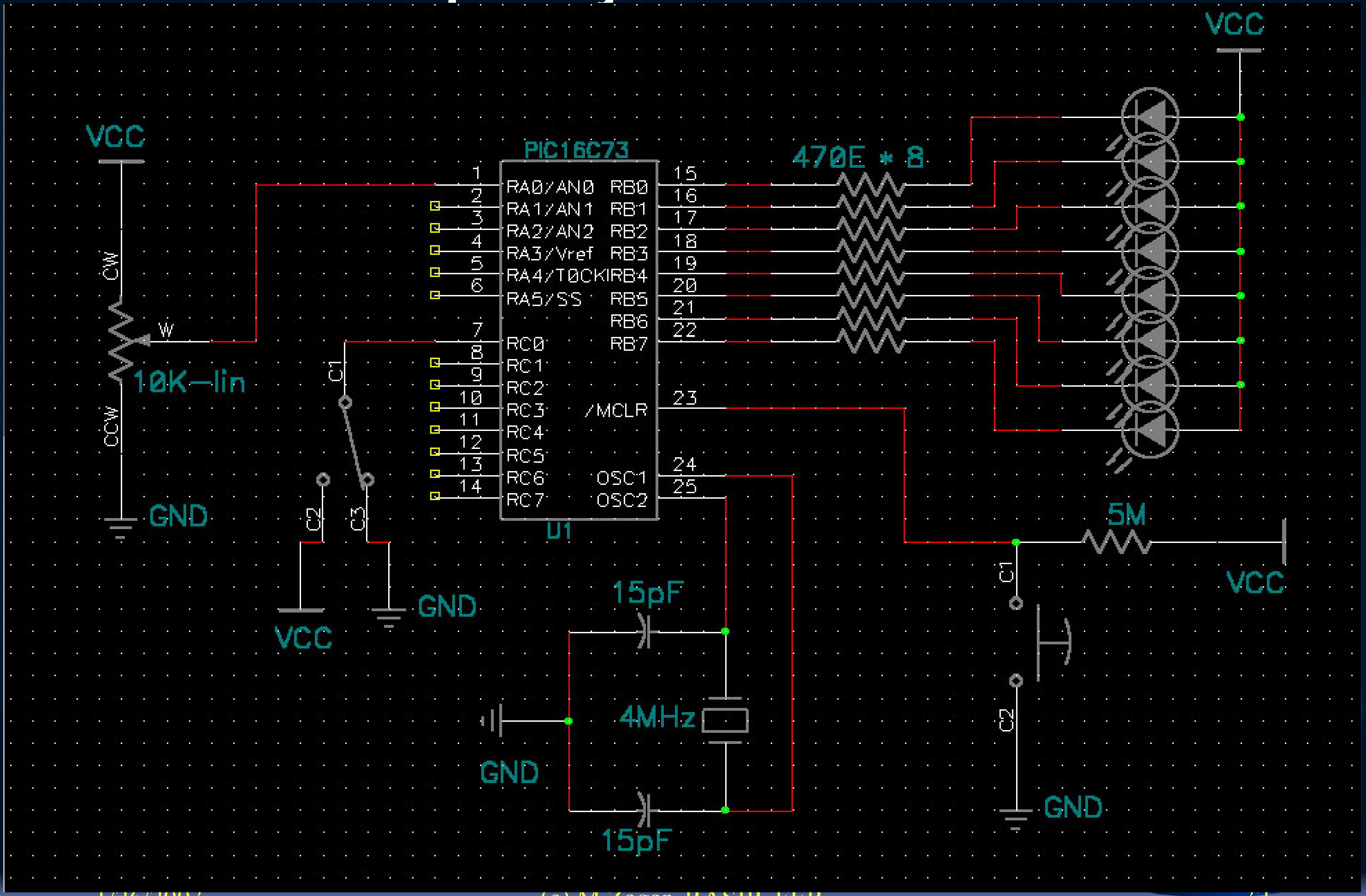
PIC16C7X Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Capture, Compare, PWM module(s)
    - Capture is 16-bit, max. resolution is 12.5 ns,
    - Compare is 16-bit, max. resolution is 200 ns,
    - PWM max. resolution is 10-bit
- 8-bit multichannel analog-to-digital converter
- Synchronous Serial Port (SSP)
- Universal Synchronous Asynchronous Receiver Transmitter (USART)
- Parallel Slave Port (PSP) 8-bits wide, with ext. RD, WR and CS controls
- Brown-out detection circuitry for Brown-out Reset (BOR)

PIC16C73 - u kutiji od nakita



Sklopovlje PIC16C73



Asembler (I)

```
        LIST    p=16c73 ; PIC16c73 is the target processor
        #include "P16c73.INC" ; Include header file
TEMP1   EQU     0x3A
TEMP2   EQU     0x3B
TEMP3   EQU     0x3D

        GOTO    MAIN
WAIT:
        MOVWF   TEMP1
        MOVLW   0xFF
        MOVWF   TEMP2
        MOVWF   TEMP3
WAIT1:
        DECFSZ  TEMP2
        GOTO    WAIT1
        DECFSZ  TEMP3
        GOTO    WAIT1
        DECFSZ  0x3A
        GOTO    WAIT1
        RETURN
```

Asembler (II)

```
RUNLED:
            MOVLW   0
            MOVF    TRISB
LOOP:
            INCF    PORTB
            MOVLW   1
            CALL    WAIT
            GOTO    LOOP
AD2LED:
            MOVLW   0
            MOVF    TRISB
            MOVLW   0x0C1
            MOVF    ADCON0
            CLRF    ADCON1
            MOVLW   0xFF
            MOVF    PORTB
            MOVLW   20
            CALL    WAIT
            MOVLW   0x00
            MOVF    PORTB
            BSF     ADCON1,2
LOOP1:
            BTFSC   ADCON1,2
            GOTO    LOOP1
            MOVFW   ADRES
            MOVWF   PORTB
            BSF     ADCON1,2
            GOTO    LOOP1
MAIN:
            BTFSC   PORTC,0
            GOTO    AD2LED
            GOTO    RUNLED
   END
```

C

```
#include "pic1673.h"
/* Mala Pauza */
void wait(int a){
   int i,j;
   for (j=0;j<a;j++){
      for (i=0;i<2000;i++){
         i=i;
      }
   }
}
/* Trcece diode*/
void RunLED(){
   TRISB=0;
   while(1){
      PORTB=PORTB+1;
      wait(1);
   }
}
/*    AD pretvorba */
void AD2LED(){
   TRISB=0;      // init AD
   ADCON0=0xC1;
   ADCON1=0x00;
   PORTB=0xFF;
   wait(20);
   PORTB=0x00;
   ADGO=1;
   while(1){
      if (ADGO==0){
         PORTB=ADRES;
         ADGO=1;
      }
   }
}
/* Glavni Program */
void main(){
   if (RC0==0)   {
      RunLED();
   }else {
      AD2LED();
   }
}
```

Veza C--, Z80-ASM

- ccz--
- telnet ccz--