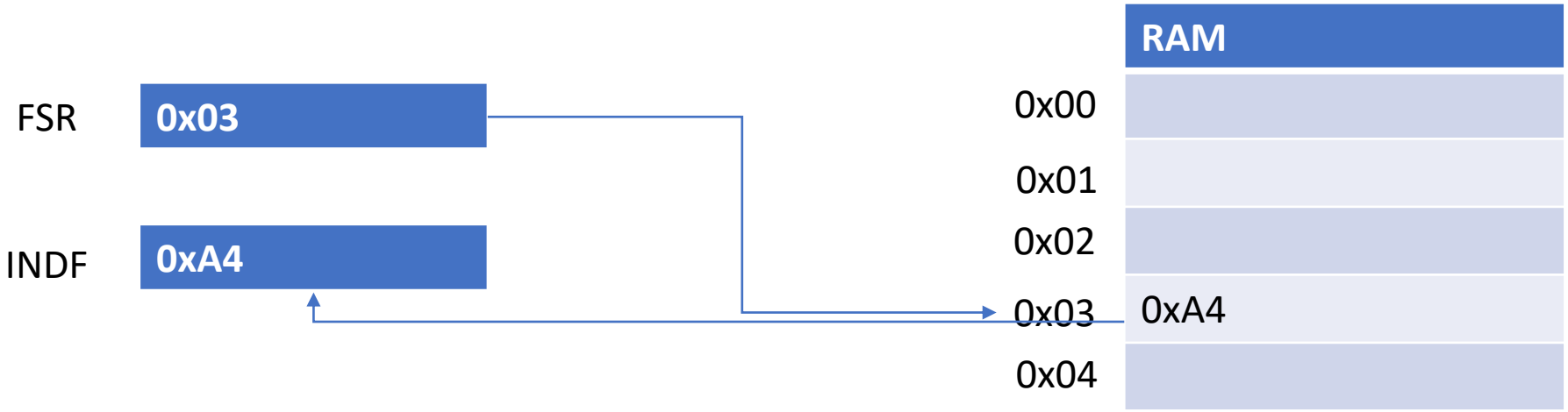
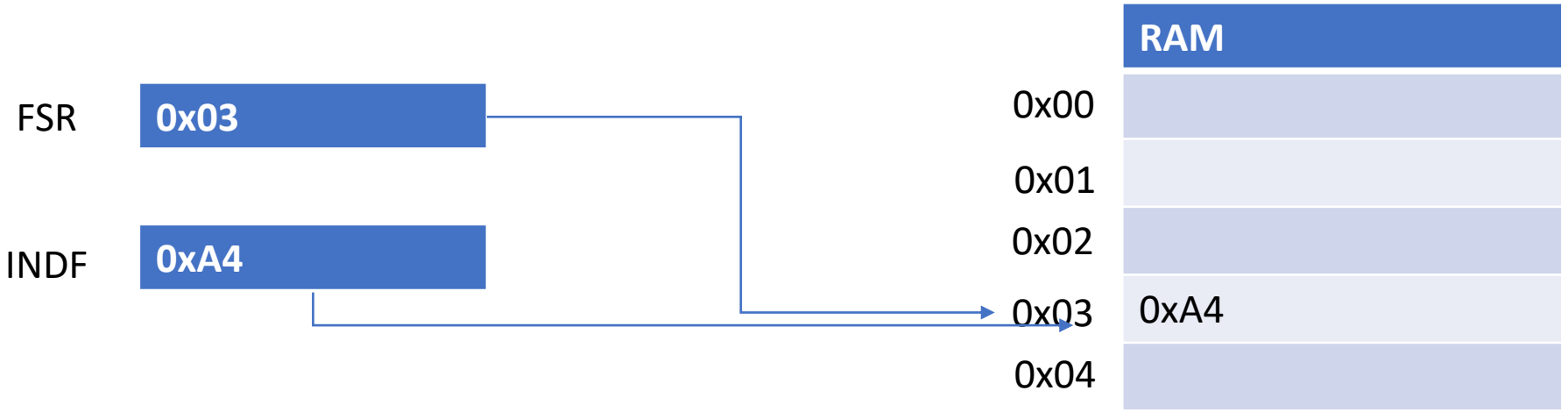
Indirektno adresiranje
Vremenske petlje
Zadatak 3 - Merenje vremena

Indirektno adresiranje - čitanje




```
movlw 0x03
movwf FSR
movf INDF,w
movwf CIFRA
```

Indirektno adresiranje - upis




```
movlw 0x03
movwf FSR
movlw 0xA4
movwf INDF
```

Vremenske petlje

- Ponekad je neophodno da se implementira fiksno kašnjenje u PIC programima (pritisci tastera, vremenska ograničenja)
- To se postiže izvođenjem “beskorisnih” instrukcija, pri čemu su često organizovane u vidu ugnježdenih petlji
- Za ovu svrhu koristi se **decfsz** instrukcija, čiji je efekat da dekrementira sadržaj registra i preskoči sledeću instrukciju ako je rezultat nakon dekrementiranja nula
- Takođe, koristi se i NOP da dopuni kašnjenje do određene vrednosti
- U XC8 postoji bibliotečka funkcija **__delay_ms(broj milisekundi)**

Primer petlje bez ugnježđavanja

```
                        ;counter cnt_1 is given a value
                        ;before the execution of the delay loop
lbl:    decfsz  cnt_1   ;decrement cnt_1 skip next instruction if result was 0
        goto    lbl     ;loop

                        ;counter cnt_1 is zero at this point
```

```
                ;in this example cnt_1 = 4
decfsz  cnt_1   ; cnt_1 = 3,  microcyles =  1 } 3
goto            ;       = 3,              =  2
decfsz  cnt_1   ;       = 2,              =  1 } 3
goto            ;       = 2,              =  2
decfsz  cnt_1   ;       = 1,              =  1 } 3
goto            ;       = 1,              =  2
decfsz  cnt_1   ;       = 0,              =  2
                                    total = 11
```

$$t = 2 + 3(cnt_1 - 1)$$

Primer sa ugnježđenim petljama

```
                        ;counters cnt_1, cnt_2 are given values
                        ;before the execution of the delay loops

lbl:    decfsz  cnt_1   ;decrement cnt_1 skip next instruction if result was 0
        goto    lbl     ;loop
        decfsz  cnt_2   ;decrement cnt_2 skip next instruction if result was 0
        goto    lbl     ;loop

                        ;counters cnt_1, cnt_2 are zero at this point
```

```
<t0_(n-1)>      ; cnt_n =  00, microcycles = t0_(n-1)
decfsz  cnt_1   ;       = 255,             =  1
goto            ;       = 255,             =  2
<t0_(n-1)>      ; cnt_n = 255, microcycles = t0_(n-1)
decfsz  cnt_1   ;       = 254,             =  1
goto            ;       = 254,             =  2
     .
     .
     .
<t0_(n-1)>      ; cnt_n =  01, microcycles = t0_(n-1)
decfsz  cnt_1   ;       =  01,             =  1
goto            ;       =  01,             =  2
decfsz  cnt_1   ;       =  00,             =  2
```

$t = t^0_{n-1} + 3$

$t = t^0_{n-1} + 3$

$t = t^0_{n-1} + 3$

$$t^0_n = 2 + \left(t^0_{n-1} + 3\right)(256 - 1) = 255\, t^0_{n-1} + 767$$

Proizvoljan početni indeks

```
                ;counters have arbitrary values at start
                ;for this example cnt_n = 4

<t1_(n-1)>      ; cnt_n =  00, microcycles = t1_(n-1)
decfsz  cnt_1   ;       =  03,             =  1
goto            ;       =  03,             =  2
<t0_(n-1)>      ; cnt_n =  03, microcycles = t0_(n-1)
decfsz  cnt_1   ;       =  02              =  1
goto            ;       =  02,             =  2
<t0_(n-1)>      ; cnt_n =  02, microcycles = t0_(n-1)
decfsz  cnt_1   ;       =  01,             =  1
goto            ;       =  01,             =  2
<t0_(n-1)>      ; cnt_n =  01, microcycles = t0_(n-1)
decfsz  cnt_1   ;       =  00,             =  2
```

$t = t_{n-1}^{0} + 3$

$t = t_{n-1}^{0} + 3$

$t = t_{n-1}^{0} + 3$

$$t_n^1 = t_{n-1}^1 + \left(cnt_n - 1\right)\left(t_{n-1}^0 + 3\right) + 2$$

Formule

- Bez ugnježdavanaja
  - $t = 2 + 3 * (cnt1 - 1)$
- Sa ugnježdenim petljama i početnim indeksom 0
  - $t_n^0 = 255 t_{n-1}^0 + 767$
- Sa ugnježdenim petljama i proizvoljnim početnim indeksom
  - $t_n^1 = t_{n-1}^1 + \left(cnt_n - 1\right)(t_{n-1}^0 + 3) + 2$
- Ukupno trajanje
  - $T_{ukupno} = 4T * t_n^1 = \frac{4}{F} * t_n^1, F - frekvencija,\ t\text{-}broj\ ciklusa$

Primer

- Koliko traje kašnjenje prouzrokovano petljom, ako se koristi oscilator 4MHz?

```
movlw .203
movwf C1
movlw .8
movwf C2
loop:
    decfsz C1,1
    goto loop
    decfsz C2,1
    goto loop
```

Rešenje

- Potrebno nam je t_2^1
- cnt1=203, cnt2=8
- $t_n^1 = t_{n-1}^1 + \left(cnt_n - 1\right)(t_{n-1}^0 + 3) + 2$
- $t_n^0 = 255t_{n-1}^0 + 767$
- $t_2^1 = t_1^1 + (cnt2 - 1) * (t_1^0 + 3) + 2$
- $t_1^0 = 255t_0^0 + 767 = 767\ , jer\ je\ (t_0^0 = 0)$
- $t_1^1 = t_0^1 + (cnt1 - 1)(t_0^0 + 3) + 2 = (203 - 1) * 3 + 2 = 608\ , jer\ je\ (t_0^1 = 0)$
- $t_2^1 = 608+(8-1)(767+3)+2=6000$
- S obzirom da je F=4Mhz, a jedan ciklus traje 4 takta, pri čemu je T=1/F, onda je konačno rešenje
  - $T_{ukupno} = 4T * t_2^1 = \frac{4}{F} * t_2^1$ =6000μs=6ms
  - Ako uzmemo u obzir i inicjalizaciju broajča (4 instrukcije), treba dodati +4 na t_2^1

Zadatak 3

- Projektovati sistem sa šest 7s displeja baziran na procesoru Microchip PIC16F84a.
- Sistem realizovati bez korišćenja pomoćnih logičkih kola na sledeći način:
  - Linije B0-B6 sa mikrokontrolera iskoristiti za kontrolu pojedinačnih segmenata na displejima (a-g).
  - Linije B7 i A0-A4 iskoristiti za kontrolu tranzistora koji su povezani na displeje.
  - Napisati proceduru koja realizuje tehniku osvežavanja displeja.
  - Cifre koje se ispisuju nalaze se na adresama 35h-3Ah.
- U glavnom delu programa je potrebno ciklično rotirati poruku EF2019 ulevo za jednu poziciju brzinom jedne rotiacije u sekundi. Tablicu def. cifara realizovati programski. Takt procesora je 3.2768MHz, a Displej je potrebno osvežavati sa 50Hz.

EF2019 —1s→ F2019E —1s→ 2019EF —1s→ 019EF2 —1s→ 19EF20 —1s→ 9EF201 —1s→ EF2019

Podešavanje prescaler-a

- $\frac{2^{15}*100Hz}{2^2*2^{n+1}*2^8} = 6 * 50Hz$
- $\frac{2^{15}}{2^{n+11}} = 3$
- $2^{4-n} = 3?$
- Bolje u ovom slučaju da fiksiramo n, a tražimo X
- Uzmimo n=7
- $\frac{2^{15}*100Hz}{2^2*2^{n+1}*X} = 6 * 50Hz$
- $\frac{2^{15}}{2^2*2^8*X} = 3$
- $\frac{2^5}{X} = 3$
- $32 = 3 * \mathrm{X}$
- X=10.67
- TMR0:=256-11=245

XC8 rešenje

```
#include <stdio.h>
#include <stdlib.h>

#include <xc.h>
#include <htc.h>
#define _XTAL_FREQ 3276800

// Configuration Byte
#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code protection off)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = ON      // Power-up Timer Enable bit (PWRT disabled)
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)

#define DISPLAY PORTB

void osvezi();
void rotiraj();

unsigned char brojac = 0;

unsigned char poruka [] = {'E','F','2','0','1','9'};

unsigned char kod(unsigned char karakter){
    static unsigned char kodovi10[10] = {0x7D, 0x30, 0x6D, 0x79, 0x33, 0x5B, 0x5F, 0x70, 0x7F, 0x7B};
      static unsigned char kodoviaf[6]={0x77, 0x1F, 0x1E, 0x3D, 0x4F, 0x0F};

      if(karakter>='0' && karakter<='9')
            return kodovi10[karakter-'0'];
      else if(karakter>='A' && karakter<='F')
      return kodoviaf[karakter-'A'];
}
```

```
void main(int argc, char** argv) {
    TRISB = 0; // port B output
    TRISA = 0; // RA0 RA1 output

    DISPLAY = 0;

    OPTION_REG = 7;
    TMR0 = 245;
      //256-11
    INTCON = 0x00;

    while (1){
        while (!INTCONbits.T0IF);
        osvezi();
        if (brojac == 50)
            rotiraj();
    }
}
```

```
void osvezi(){
    static unsigned char displej = 1;

    if (displej > 6)
        displej = 1;

      PORTBbits.RB7 = 0;
    switch (displej){
        case 1:
            brojac++;
            DISPLAY = kod(poruka[0]);
            PORTBbits.RB7 = 1;
            PORTA = 0;
            break;
        case 2:
            DISPLAY = kod(poruka[1]);
            PORTA = 1;
            break;
        case 3:
            DISPLAY = kod(poruka[2]);
            PORTA = 2;
            break;
        case 4:
            DISPLAY = kod(poruka[3]);
            PORTA = 4;
            break;
        case 5:
            DISPLAY = kod(poruka[4]);
            PORTA = 8;
            break;
        case 6:
            DISPLAY = kod(poruka[5]);
            PORTA = 16;
            break;
        default:
            break;
    }

    displej++;
}
```

```
void rotiraj(){
    unsigned char pm = poruka[0];
    for (int i=0; i<5; i++)
        poruka[i] = poruka[i+1];
    poruka[5] = pm;
}
```

ASM

```
#include <p16f84a.inc>

    errorlevel -302
    __CONFIG   _CP_OFF & _WDT_OFF & _PWRTE_ON & _HS_OSC  ;ovo
je za PIC16F84

    CBLOCK 0x20
    CODE1
    CODE2
    CODE3
    CODE4
    CODE5
    CODE6
    ENDC

    CBLOCK 0x35
    CIFRA1
    CIFRA2
    CIFRA3
    CIFRA4
    CIFRA5
    CIFRA6
    PROLAZ
    OSVCIF
    PCIFRA
    BROJAC
    WREG_TEMP           ;storage for WREG during interrupt
    STATUS_TEMP         ;storage for STATUS during interrupt
    PCLATH_TEMP         ;storage for PCLATH during interrupt
    FSR_TEMP            ;storage for FSR during interrupt
    ENDC

    ORG 0x0000
    goto ResetCode

ResetCode:
    clrf    PCLATH            ;select program memory page 0
      goto  Main        ;go to beginning of program
```

```
Main:
    banksel TRISA
    clrf TRISA
    clrf TRISB
    movlw b'00000111'
    movwf OPTION_REG
    banksel PORTA
    clrf PORTA
    clrf PORTB
    clrf INTCON
      ;cuvanje potrebnih kodova u RAM-u,
      ;RB6-a...RB0-g
    movlw b'01111110' ; '0'
    movwf CODE1
    movlw b'00110000' ; '1'
    movwf CODE2
    movlw b'01101101' ; '2'
    movwf CODE3
    movlw b'01011011' ; '9'
    movwf CODE4
    movlw b'01001111' ; 'E'
    movwf CODE5
    movlw b'01000111' ; 'F'
    movwf CODE6
      ;IND.  012345
      ;CODE: 0129EF

      ;EF2019
      ;452013

    movlw .4
    movwf CIFRA1
    movlw .5
    movwf CIFRA2
    movlw .2
    movwf CIFRA3
    movlw .0
    movwf CIFRA4
    movlw .1
    movwf CIFRA5
    movlw .3
    movwf CIFRA6
    clrf OSVCIF
    bsf OSVCIF, 7 ;RB'7'
    clrf PROLAZ
```

```
petlja:
    movlw .245
    movwf TMR0
cekaj:
    btfss INTCON, T0IF
    goto cekaj


    call osvezi
    bcf INTCON, T0IF

    movlw .50
    xorwf PROLAZ, W
    btfsc STATUS, Z
    call pomeri
    goto petlja
```

```
osvezi:
    ;10000000
    btfsc OSVCIF, 7
    goto c1
      ;00000001
    btfsc OSVCIF, 0
    goto c2
      ;00000010
    btfsc OSVCIF, 1
      goto c3
      ;00000100
    btfsc OSVCIF, 2
    goto c4
    ;00001000
    btfsc OSVCIF, 3
    goto c5
c6:
    movf CIFRA6, W
    goto lportb
c5:
    movf CIFRA5, W
    goto lportb
c4:
    movf CIFRA4, W
    goto lportb
c3:
    movf CIFRA3, W
    goto lportb
c2:
    movf CIFRA2, W
    goto lportb
c1:
      incf PROLAZ, F
    movf CIFRA1, W
```

```
lportb:
    addlw 0x20
    movwf FSR
    movf INDF, W
    movwf PORTB

    movf OSVCIF, W
    movwf PORTA
    ;1000 0000
      andlw 0x80
    iorwf PORTB, F

    rlf OSVCIF, F
    btfsc STATUS, C
    bsf OSVCIF, 0
    btfss OSVCIF, 5
    return
    bcf OSVCIF, 5
    bsf OSVCIF, 7
    return
```

```
pomeri:
    clrf PROLAZ
    movf CIFRA1, W
    movwf PCIFRA
    movf CIFRA2, W
    movwf CIFRA1
    movf CIFRA3, W
    movwf CIFRA2
    movf CIFRA4, W
    movwf CIFRA3
    movf CIFRA5, W
    movwf CIFRA4
    movf CIFRA6, W
    movwf CIFRA5
    movf PCIFRA, W
    movwf CIFRA6
    return

    END
```

EF2019->F2019E->2019EF->019EF2->19EF20->9EF201->EF2019