UNIVERZITET U BEOGRADU

ELEKTROTEHNIČKI FAKULTET

Katedra za elektroniku

Predmet: Računarska elektronika

Projekat: FIR filtar korišćenjem simetrije

Projekat radili:
Katarina Radinović Kapralović 0007/2014
Nikola Nešković 0217/2014

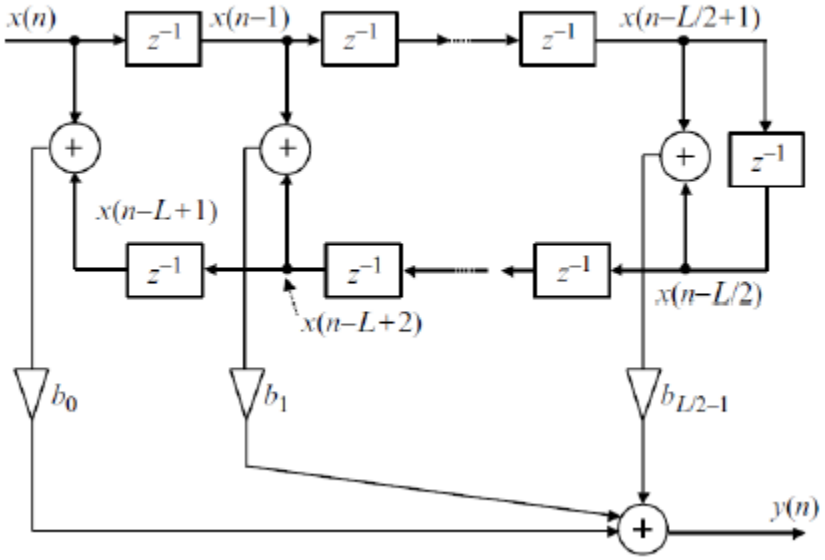
Beograd, jun 2017.

I) Tekst zadatka (projekat broj 28):

Potrebno je napraviti filtar niskih učestanosti. Projektovani filtar propušta niske učestanosti frekvencije 5 kHz, dok na visokim učestanostima slabi signal 80 dB. Potrebno je napraviti program koji filtrira ulazni signal u PCM formatu korišćenjem FIR filtra čiji su nam koeficijenti dati. Formula za računanje izlaznog signala je:

$$y[n] = \sum_{l=0}^{\frac{L}{2}-1} b_l(x[n-l] + x[n-L+1+l])$$

Gde je L – broj odbiraka FIR filtra, x[n] – ulazni signal, y[n] – izlazni signal i bl – koeficijent FIR filtra.



II) Izrada zadatka:

Istaknute funkcije:

1. **CreateOutputFile** – Funkciji se prosleđuje adresa na kojoj je smešten string - naziv datoteke koju treba da napraviti. Funkcija u EAX registar upisuje „handle“ vrednost datoteke, obično ceo broj koji služi za identifikaciju.

2. **OpenInputFile** – Funkciji se prosleđuje adresa naziva datoteke koja se otvara. Funkcija u EAX registar upisuje „handle“ vrednost datoteke, obično ceo broj koji služi za identifikaciju.

3. **ReadFromFile** – Funkcija iz registara EDX i ECX čita adresu i veličinu, u bajtima, dela memorije u koji smešta učitan podatak, iste veličine, iz zadate datoteke koju smo otvorili prethodnom funkcijom.

4. **WriteToFile** – Funkcija iz registara EAX, EDX i ECX redom čita „handle“ vrednost izlazne datoteke, adresu i veličinu dela memorije odakle uzima podatke koje smešta u izlaznu datoteku.

5. **GetMseconds** – Funkcija u registar EAX upisuje vreme dela izvršavanja programa u kom se funkcija nalazi. Iskorišćena je za merenje vremena izvršavanja programa.

6. **CloseFile** – Funkcija zatvara datoteku čija je „handle“ vrednost smeštena u registar EAX.

Delovi koda (labele):

1. **file_ok –** nakon provere da li je došlo do greške prilikom otvaranja ulazne i izlazne datoteke čita se odbirak u bafer, veličine 2 bajta, iz datoteke „input.pcm“ koristeći funkciju ReadFromFile. Posle toga proverom ispravnosti funkcije prelazi se na sledeći deo koda.

2. **buf_ok** – Na početku su vrednosti osnovnih registara postavljene na vrednost 0.

   Prvom naredbom grananja – IF – vrši se provera da li je niz x – niz ulaznih odbiraka, dužine L reči – „popunjen do kraja“, tj. Da li je pročitano više od L odbiraka. Ako nije vrednost promenljive K uvećava se za 2 pošto se iz datoteke čita po 2 bajta po odbirku. Samim tim, promenljiva K pamti koliko bajtova je upisano u niz x. Ovo služi da se kasnije ne bi trošilo vreme na pomeranje odbiraka koji još uvek nisu dobili smislenu vrednost.

Sledeća IF naredba služi prvenstveno za pomeranje niza učitanih podataka. Ceo niz pomera se za 2 bajta udesno i tako se oslobađa mesto za učitani odbirak koji će biti ubačen u niz na prvo mesto. Ovaj deo koda se nalazi pod naredbom IF opet zbog uštede vremena izvršavanja – jer nije potrebno obaviti pomeranje niza pri čitanju prvog odbirka.
Nakon toga, prebacuje se prethodno učitana vrednost iz bafera u x.

3. **while petlja** – Ova petlja nam služi za generisanje formule iz postavke zadatka. Petlja se vrti sve dok brojač, ECX, ne dodje do polovine niza odbiraka. Na kraju se u akumulatoru nalazi vrednist:

$$EAX = b[L - C] * (x[C * 2] + x[(L - C) * 2])$$

Takođe, i u ovom delu se svi indeksi množe sa 2, tj. binarne predstave vrednosti pomeraju se za jedno mesto ulevo zbog toga što odbirci zauzimaju 2 bajta memorije. Dobijena vrednost iz EAX registra se sabira, na kraju petlje, sa y, čime se formira zbir koji će biti upisan u izlazni fajl. Petlja se za svaki odbirak ponavlja 140 puta.

4. **Upis u fajl** – Upisuje se vrednost y u izlaznu datoteku i proverava se da li je upis pravilno izvršen. Ako nije ispisuje se poruka o greški.

5. **dalje** – Y se postavlja na 0 - priprema se njegova vrednost za sledeću petlju. Učitava se novi odbirak u bafer. Pri ovom čitanju se proverava i eventualni dolazak do kraja datoteke. Proverava se da li je došlo do greške i ako nije skače se na labelu „buf_ok“ i isti procesi od te labele se ponavljaju dok se ne dođe do kraja ulazne datoteke.

6. **close_file** – Zatvaraju se sve datoteke koje su korišćene. Funkcijom GetMseconds učitava se vreme kraja programa. Oduzimanjem od te vrednosti vreme dobijeno istom funkcijom na početku programa računa se vreme izvršavanja programa, u ms, koje se potom ispisuje kao poruka u prozoru konzole.

7. **Kraj programa**

Matlab provera:

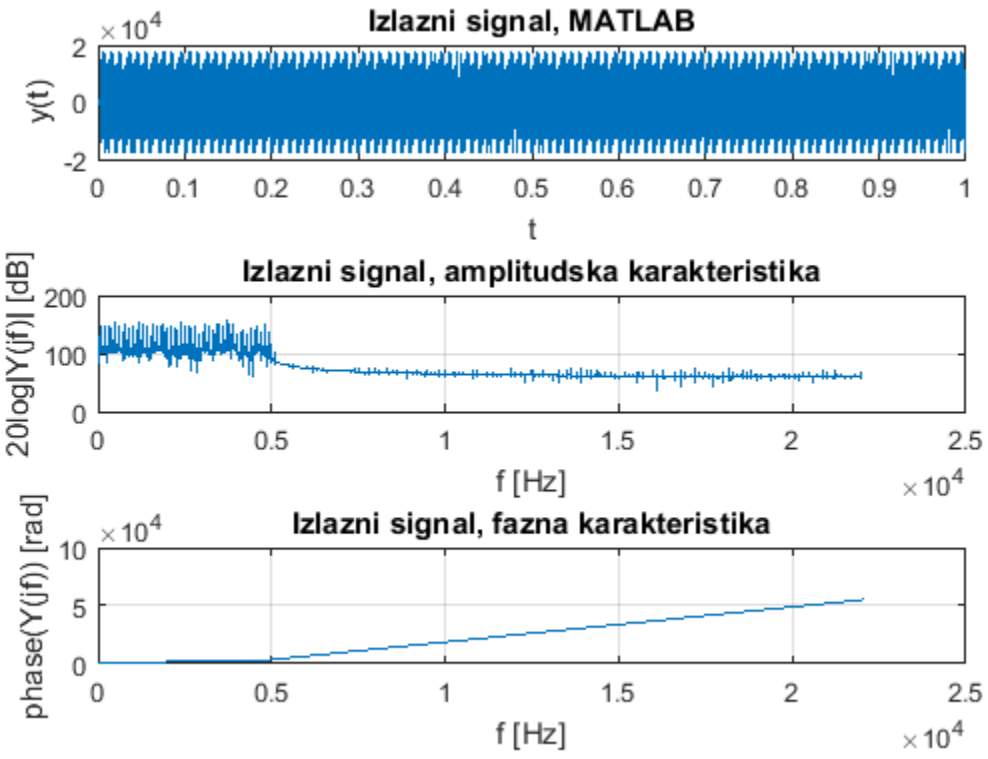


Figure 1 Izlaznog signal dobijen korišćenjem matlaba

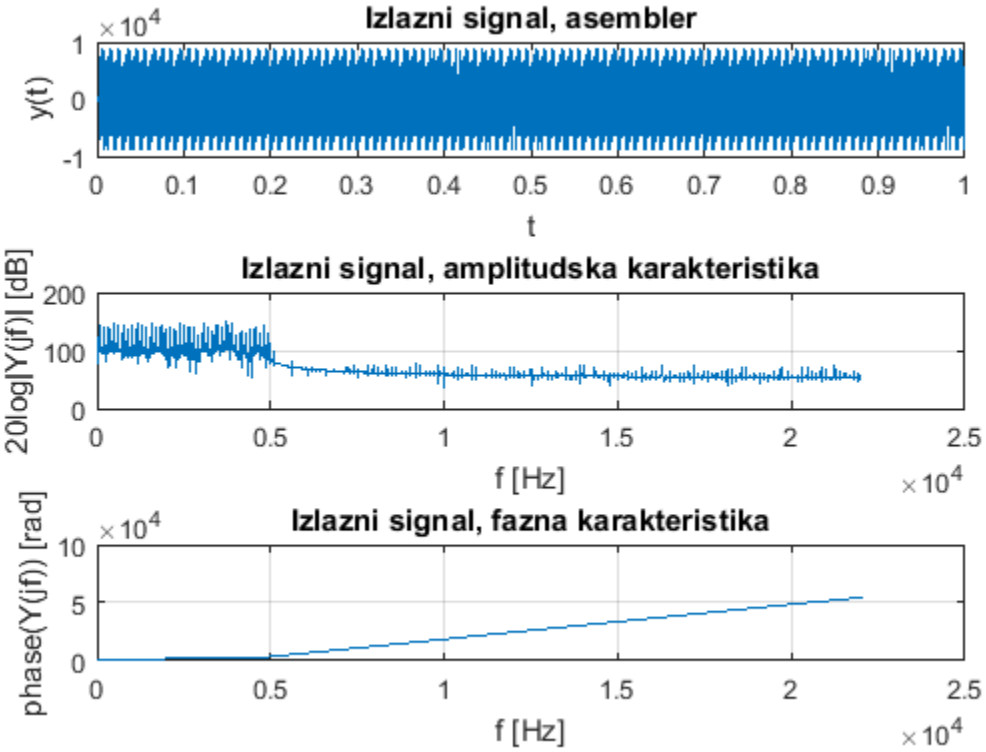


Figure 2 Izlazni signal dobijen korišćenjem asemblera

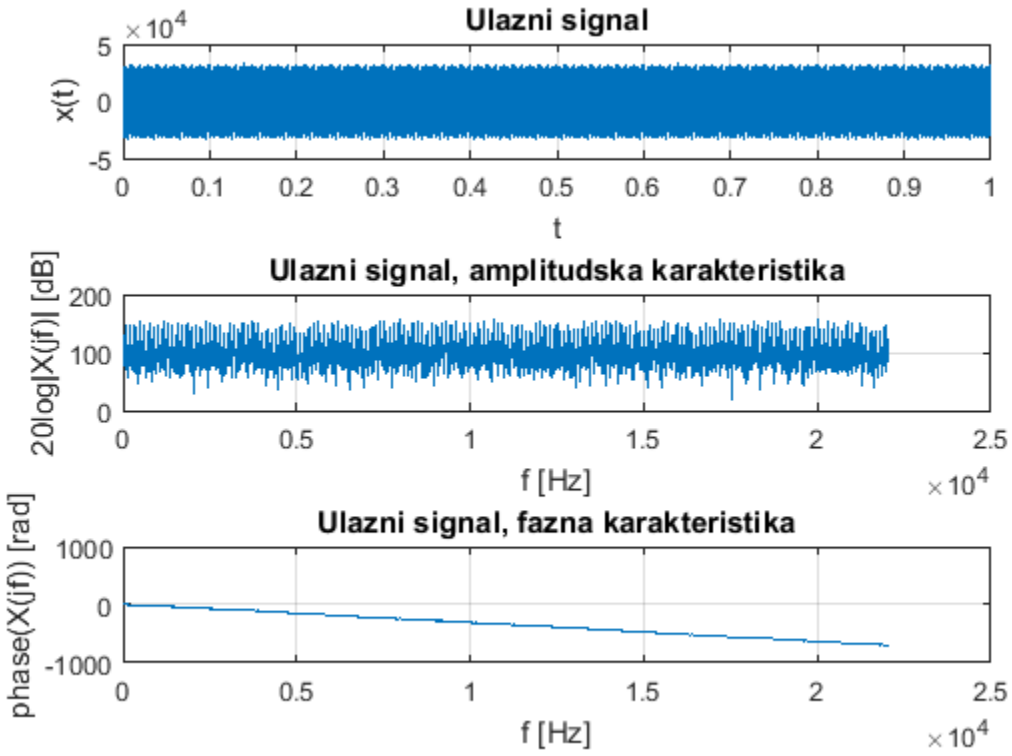


Figure 3 Ulazni signal

Uz pomoć Matlaba proverena je i potvrđena funkcionalnost programa, što se vidi iz priloženih grafika.

Kod:

```
; RE projekat broj 28
; jul 2017
; Projekat radili:
; Katarina Radinovic Kapralovic 2014 / 0007
; Nikola Neskovic 2014/0217

INCLUDE Irvine32.inc
INCLUDE macros.inc

ExitProcess proto, dwExitCode:dword

.const
L = 279
duzina = 280
L2 = 139
BUFFER_SIZE = 2

.data
barray		SWORD - 2, 2, 12, 31, 60, 99, 141, 177, 197, 192
		SWORD 158, 100, 30, -37, -83, -97, -77, -34, 18, 58
		SWORD 73, 58, 21, -23, -55, -62, -43, -5, 33, 57
		SWORD 54, 27, -12, -46, -59, -45, -10, 30, 56, 57
		SWORD 31, -11, -48, -64, -50, -12, 33, 64, 65, 35
		SWORD -12, -56, -74, -58, -14, 39, 76, 77, 42, -15
		SWORD -67, -89, -70, -16, 48, 92, 94, 50, -19, -83
		SWORD -109, -85, -18, 60, 114, 115, 61, -25, -103, -135
		SWORD -104, -21, 76, 142, 142, 74, -34, -131, -170, -129
		SWORD -23, 99, 180, 179, 91, -46, -169, -217, -163, -26
		SWORD 132, 236, 231, 114, -67, -227, -288, -213, -28, 184
		SWORD 323, 313, 150, -101, -324, -407, -297, -29, 279, 481
		SWORD 466, 216, -171, -522, -657, -479, -30, 507, 880, 869
		SWORD 402, -389, -1184, -1581, -1240, -31, 1890, 4102, 6037, 7164


buffer SWORD ?
filenameIN BYTE "input.pcm",0
filenameOUT BYTE "input_out.pcm"
fileHandleIN HANDLE ?
fileHandleOUT HANDLE ?
x SWORD duzina DUP(0)
K WORD 0
y SDWORD 0
time1 dword 0
time2 dword 0


.code
main proc

	call GetMseconds		; vreme starta
	mov time1, eax

;Otvaranje izlaznog fajla
	mov	edx, OFFSET filenameOUT
	call	CreateOutputFile
	mov	fileHandleOUT, eax

; Otvaranje ulaznog fajla
	mov	edx,OFFSET filenameIN
	call	OpenInputFile
```

```
        mov     fileHandleIN,eax


; Provera da li je doslo do greske
        cmp     eax,INVALID_HANDLE_VALUE
        jne     file_ok
        mWrite <"Cannot open file",0dh,0ah>
        jmp     quit

file_ok:
; Read the file into a buffer.
        mov     edx,OFFSET buffer
        mov     ecx,BUFFER_SIZE
        call    ReadFromFile
        jnc     buf_ok
        mWrite "Error reading file. "
        call    WriteWindowsMsg
        jmp     close_file


buf_ok:
        xor eax, eax
        xor edx, edx
        xor ecx, ecx
        xor ebx, ebx

        movsx ebx, K                            ;broj bajtova koji je do sada upisan u x
        .IF(ebx < L + L)                        ; max K = 2 * (280 - 1)
                add K, 2
        .ENDIF
        .IF(ebx > 0)                            ; preskace se samo pri upisu prvog podatka
                mov esi, ebx
                sub ebx,2
                mov eax,esi
                shr eax,1
                mov ecx, eax
        pomeranje:                                      ;siftovanje ulaznog niza-bafera
                movsx eax, x[ebx]
                mov x[esi], ax
                mov esi, ebx
                sub ebx, 2
                dec ecx
                jnz pomeranje
        .ENDIF

        mov ax, buffer
        mov x, ax
        mov ecx, L

        .while (ecx > 139)                      ; racunanje y(n)
                mov eax, ecx
                shl eax, 1
                mov ebx, eax
                movsx edx, x[ebx]
                mov eax, L
                sub eax, ecx
                shl eax, 1
                mov ebx, eax
                movsx eax, x[ebx]
                add edx, eax
                mov eax, L
                sub eax, ecx
                shl eax, 1
                mov ebx, eax
```

```
            movsx eax, barray[ebx]
            imul eax, edx
            add y, eax
            dec ecx
        .endw

; Upis u fajl
        mov eax, fileHandleOUT
        mov     edx,OFFSET y +2                         ;upis visa dva bajta
        mov     ecx,BUFFER_SIZE
        call    WriteToFile
        cmp     eax, BUFFER_SIZE
        jz      dalje                                   ; greska?
        mWrite "Error writing to file. "
        call    WriteWindowsMsg
        jmp     close_file

dalje:
        mov y, 0                                        ;priprema za sledecu petlju
        mov eax, fileHandleIN
        mov     edx,OFFSET buffer
        mov     ecx,BUFFER_SIZE
        call    ReadFromFile
        cmp     eax, BUFFER_SIZE
        jz buf_ok                                       ;povratak na pocetak


close_file:
        mov     eax,fileHandleIN
        call    CloseFile
        mov     eax, fileHandleOUT
        call    CloseFile

        call GetMseconds                            ; vreme zavrsetka programa
        mov time2, eax
        sub eax,time1
        call WriteDec
        mWrite " ms passed "
        call Crlf
        call WaitMsg

quit:
        exit

main endp
end main
```