DIGITALNI PROCESORI SIGNALA – Implementacija IIR filtara na DSP platformi

Vladimir Petrović, Aleksandra Lekić

Univerzitet u Beogradu – Elektrotehnički fakultet

2018/2019

IIR filtar

- Dizajnira se kao analogni filtar nakon čega se primenjuje odgovarajuća diskretizacija.
- Najčešće se koristi bilinearna transformacija za diskretizaciju.

$$s = \frac{2}{T}\,\frac{z-1}{z+1}$$

- Ima transfer funkciju oblika

$$H(z) = \frac{\sum_{l=0}^{L-1} b_l\, z^{-l}}{1+\sum_{m=1}^{M} a_m\, z^{-m}}.$$

- IIR filtar je opisan diferencnom jednačinom

$$y[n] = \sum_{l=0}^{L-1} b_l\, x[n-l] - \sum_{m=1}^{M} a_m\, y[n-m].$$

Realizacija IIR filtra

Može da se realizuje u više formi:

- Direktna forma I
- Direktna forma II
- Kaskadna struktura
- Paralelna struktura

Direktna forma

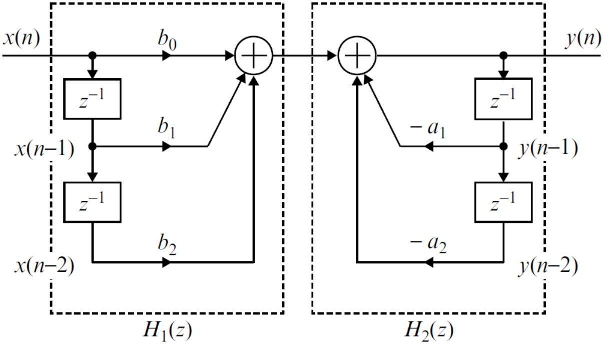
- Ima $L+M$ koeficijenata i treba $L+M+1$ memorijskih lokacija za čuvanje odbiraka $\{x[n-l];\ l \in \{0,\cdots,L-1\}\}$ i $\{y[n-m];\ m \in \{0,\cdots,M\}\}$.
- Zahteva $L+M$ množenja i $L+M-1$ sabiranja.

Direktna forma I

- Direktna forma I filtra drugog reda. Interpretira se kao kaskada dva filtra

$$H(z) = H_1(z)\, H_2(z).$$

$$H_1(z) = b_0 + b_1\, z^{-1} + b_2\, z^{-2}, \qquad H_2(z) = \frac{1}{1 + a_1\, z^{-1} + a_2\, z^{-2}}$$

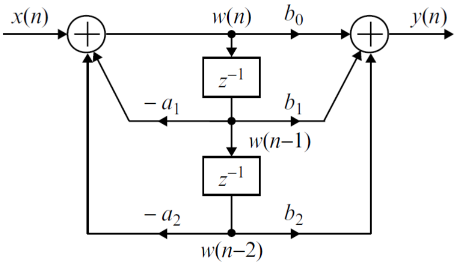


Direktna forma II - *biquad*

- Direktna forma II filtra drugog reda zahteva tri memorijske lokacije, za razliku od direktne forme I koja zahteva 6.
- Važi

$$y[n] = b_0\, w[n] + b_1\, w[n-1] + b_2\, w[n-2],$$

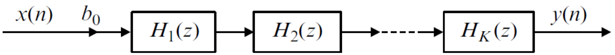
$$w[n] = x[n] - a_1\, w[n-1] - a_2\, w[n-2].$$



Kaskadna forma

- Realizuje se kao kaskada više IIR filtara drugog reda kada je M paran broj.

$$H(z) = b_0 \prod_{k=1}^{K} H_k(z) \qquad K = \frac{M}{2}$$

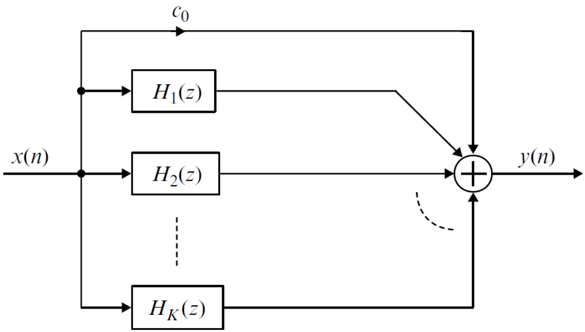


Paralelna forma

- Prenosna funkcija se razloži na parcijalne razlomke

$$H(z) = c_0 + H_1(z) + \cdots + H_K(z),$$

gde je c_0 konstanta, a $H_i(z)$ može biti IIR prvog ili drugog reda.



Primeri

- *Direct-Form I IIR Filter Using Floating-Point C*
- *Direct-Form I IIR Filter Using Fixed-Point C*
- *Cascade IIR Filter Using Fixed-Point C*
- *Cascade IIR Filter Using Assembly Program*

Direct-Form I IIR Filter Using Floating-Point C

Programski fajlovi

Fajl	Opis
`floatPoint_directIIRTest.c`	program za testiranje *floating-point* IIR filtra
`floatPoint_directIIR.c`	C funkcija *floating-point* IIR filtra
`floatPointIIR.h`	C header fajl
`tistdtypes.h`	header fajl standardnih tipova podataka
`c5505.cmd`	linker fajl
`input.pcm`	ulazni signal

Zadatak

- Importovati projekat u CCS i pokrenuti program.
- Proveriti da li se na izlazu dobijaju 60 dB oslabljene sinusoide na 800 Hz i 3300Hz. Ulazni signal sadrži tri sinusoide na frekvencijama 800 Hz, 1800 Hz i 3300Hz.

```c
#include "tistdtypes.h"
#include "floatPointIIR.h"


void floatPoint_IIR(double in, double *x, double *y, double *b, short nb,
     double *a, Int16 na)
{
    double z1,z2;
    Int16 i;

    for(i=nb-1; i>0; i--)           // Update the delay line x[]
    {
        x[i] = x[i-1];
    }
    x[0] = in;                      // Insert new data to delay line x[0]

    for(z1=0, i=0; i<nb; i++)       // Filter the x[] with coefficient b[]
    {
        z1 += x[i] * b[i];
    }

    for(i=na-1; i>0; i--)           // Update the y delay line
    {
        y[i] = y[i-1];
    }

    for(z2=0, i=1; i<na; i++)       // Filter the y[] with coefficient a[]
    {
        z2 += y[i] * a[i];
    }
    y[0] = z1 - z2;                 // Place the result into y[0]

}
```

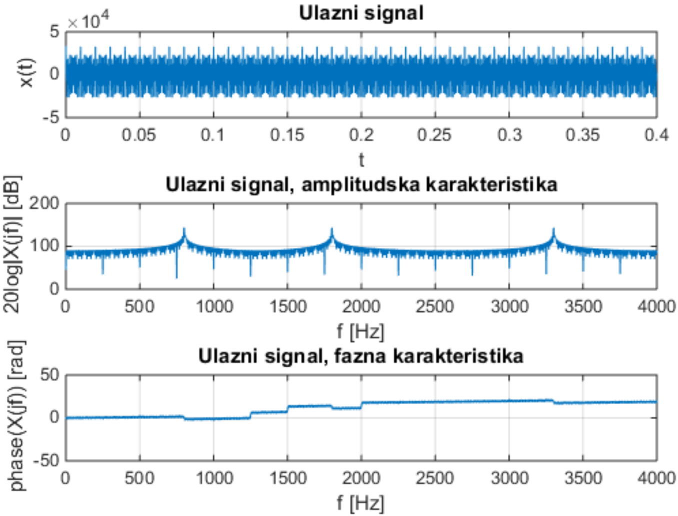
- Funkcija `floatPoint_IIR` se za računanje jednog odbirka izlaznog signala izvršava 4219 taktnih intervala.
- Podaci `input.pcm` i `output.pcm` se mogu prikazati u MATLAB-u korišćenjem skripte sa sledećeg slajda.

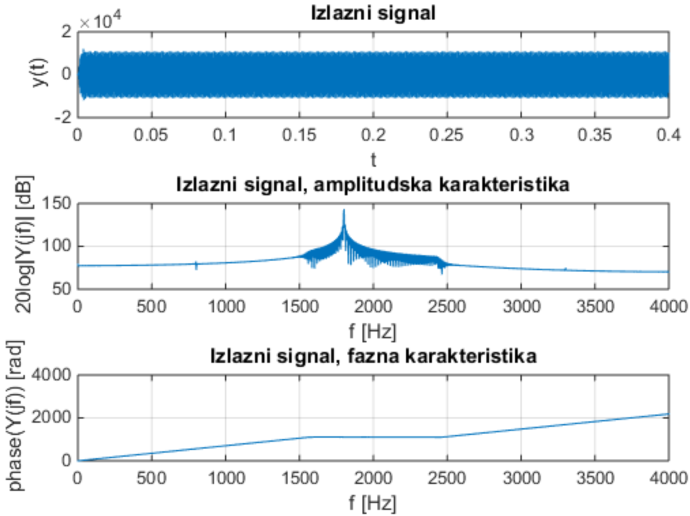
```
Fs = 8e3; % sampling frequency
input_file = fopen('input.pcm', 'r');
input = fread(input_file, inf, 'int16'); % opseg je od -32768 do 32767
output_file = fopen('output.pcm', 'r');
output = fread(output_file, inf, 'int16');
sound(input,Fs); sound(output, Fs);

sample = 1 / Fs; % sampling frequency - 8000 Hz
t = 0 : sample : (length(input) - 1) * sample;

Nfft = 4096; % 4096 je prvi stepen dvojke veci od broja odbiraka 3200
n = 0 : Nfft/2-1;
w = n * Fs / 2 /(Nfft/2 - 1);
Input = fft(input, Nfft);
figure(1)
subplot(311),plot(t,input),xlabel('t'),ylabel('x(t)'),grid on, title('Ulazni signal')
subplot(312), plot(w,20*log10(abs(Input(1:Nfft/2)))), grid on
xlabel('f [Hz]'),ylabel('20log|X(jf)| [dB]'),title('Ulazni signal, amplitudska karakteristika')
subplot(313),plot(w,unwrap(angle(Input(1:Nfft/2)))),grid on,
ylabel('phase(X(jf)) [rad]'),xlabel('f [Hz]'),title('Ulazni signal, fazna karakteristika')

Output = fft(output, Nfft);
figure(2)
subplot(311),plot(t,output),xlabel('t'),ylabel('y(t)'),grid on, title('Izlazni signal')
subplot(312),plot(w,20*log10(abs(Output(1:Nfft/2)))), grid on
xlabel('f [Hz]'),ylabel('20log|Y(jf)| [dB]'),title('Izlazni signal, amplitudska karakteristika')
subplot(313),plot(w,unwrap(angle(Output(1:Nfft/2)))),grid on,
ylabel('phase(Y(jf)) [rad]'),xlabel('f [Hz]'),title('Izlazni signal, fazna karakteristika')
```


Ulazni signal
×10^4
5
0
-5
x(t)
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
t
Ulazni signal, amplitudska karakteristika
20log|X(jf)| [dB]
200
100
0
0
500
1000
1500
2000
2500
3000
3500
4000
f [Hz]
Ulazni signal, fazna karakteristika
phase(X(jf)) [rad]
50
0
-50
0
500
1000
1500
2000
2500
3000
3500
4000
f [Hz]


Izlazni signal
×10^4
2
0
-2
y(t)
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
t
Izlazni signal, amplitudska karakteristika
150
100
50
20log|Y(jf)| [dB]
0
500
1000
1500
2000
2500
3000
3500
4000
f [Hz]
Izlazni signal, fazna karakteristika
4000
2000
0
phase(Y(jf)) [rad]
0
500
1000
1500
2000
2500
3000
3500
4000
f [Hz]

Direct-Form I IIR Filter Using Fixed-Point C

Programski fajlovi

Fajl	Opis
`fixPoint_directIIRTest.c`	program za testiranje *fixed-point* IIR filtra
`fixPoint_directIIR.c`	C funkcija *floating-point* IIR filtra
`fixPointIIR.h`	C header fajl
`tistdtypes.h`	header fajl standardnih tipova podataka
`c5505.cmd`	linker fajl
`input.pcm`	ulazni signal

Zadatak

- Importovati projekat u CCS i pokrenuti program.
- Proveriti da li se na izlazu dobijaju 60 dB oslabljene sinusoide na 800 Hz i 3300Hz. Ulazni signal sadrži tri sinusoide na frekvencijama 800 Hz, 1800 Hz i 3300Hz.

```c
#include "tistdtypes.h"
#include "fixPointIIR.h"

void fixPoint_IIR(Int16 in, Int16 *x, Int16 *y, Int16 *b, Int16 nb, Int16 *a,
        Int16 na)
{
    Int32 z1,z2;
    Int16 i;

    for(i=nb-1; i>0; i--)          // Update the delay line x[]
    {
        x[i] = x[i-1];
    }
    x[0] = in;                     // Insert new data to delay line x[0]

    for(z1=0, i=0; i<nb; i++)      // Filter the x[] with coefficient b[]
    {
        z1 += (Int32)x[i] * b[i];
    }

    for(i=na-1; i>0; i--)          // Update the y delay line
    {
        y[i] = y[i-1];
    }

    for(z2=0, i=1; i<na; i++)      // Filter the y[] with coefficient a[]
    {
        z2 += (Int32)y[i] * a[i];
    }

    z1 = z1 - z2;                  // Q15 data filtered using Q11 coefficients
    z1 += 0x400;                   // Rounding
    y[0] = (Int16)(z1>>11);        // Place the Q15 result into y[0]
}
```

Cascade IIR Filter Using Fixed-Point C

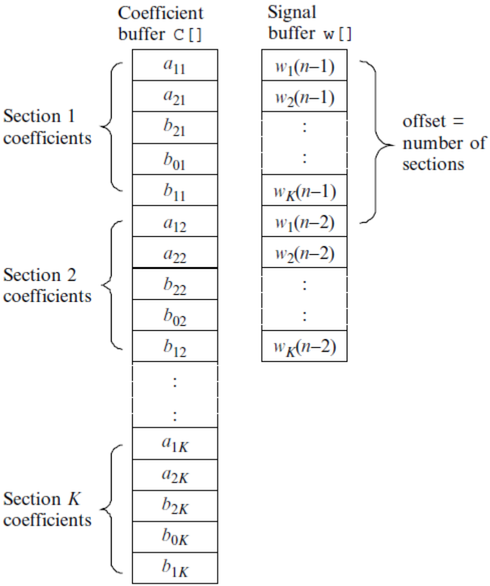
Programski fajlovi

Fajl	Opis
`fixPoint_cascadeIIRTest.c`	program za testiranje *floating-point* IIR filtra
`fixPoint_cascadeIIR.c`	C funkcija *fixed-point* IIR filtra drugog reda
`cascadeIIR.h`	C header fajl
`tistdtypes.h`	header fajl standardnih tipova podataka
`fdacoefsMATLAB.h`	FDATool-MATLAB fajl
`tmwtypes.h`	definicija podataka za MATLAB C header fajl
`c5505.cmd`	linker fajl
`in.pcm`	ulazni signal

Zadatak

- Importovati projekat u CCS i pokrenuti program.
- Proveriti da li se na izlazu dobijaju 60 dB oslabljene sinusoide na 800 Hz i 3300Hz. Ulazni signal sadrži tri sinusoide na frekvencijama 800 Hz, 1800 Hz i 3300Hz odabiran sa 8000 Hz.

- Koristi IIR filtre direktne forme II reda koje su implementirane kao *fixed-point*.
- Koeficijentni i signalni nizovi su konfigurisani kao kružni baferi.
- Signalni bafer za svaku sekciju filtra drugog reda sadrži dva elementa $w_k[n-1]$ i $w_k[n-2]$. Organizovan je tako što su prvo dati elementi sa indeksom $n-1$, a zatim sa indeksom $n-2$. Ofset je jednak broju sekcija.
- Pokazivači kružnog bafera se *update*-uju sa `j = (j+1) % m` i `l = (l+1) % s`.
- U ovom primeru se koristi IIR sa $N_S = 4$ sekcije, pa je $m = 5\,N_S = 20$ i $s = 2\,N_S = 8$.


Coefficient buffer C[]
Signal buffer w[]
Section 1 coefficients
Section 2 coefficients
Section K coefficients
a_{11}
a_{21}
b_{21}
b_{01}
b_{11}
a_{12}
a_{22}
b_{22}
b_{02}
b_{12}
a_{1K}
a_{2K}
b_{2K}
b_{0K}
b_{1K}
$w_1(n-1)$
$w_2(n-1)$
$w_K(n-1)$
$w_1(n-2)$
$w_2(n-2)$
$w_K(n-2)$
offset = number of sections

```c
#include "tistdtypes.h"

void cascadeIIR(Int16 *x, Int16 Nx, Int16 *y, Int16 *coef, Int16 Ns,Int16 *w)
{
    Int16 i,j,n,m,l,s;
    Int16 temp16;
    Int32 w_0, temp32;
    m=Ns*5;                             // Setup for circular buffer coef[]
    s=Ns*2;                             // Setup for circular buffer w[]
    for (j=0,l=0,n=0; n<Nx; n++)        // IIR filter begin
    {
        w_0 = (Int32)x[n]<<12;          // Scale input to prevent overflow
        for (i=0; i<Ns; i++)
        {
            temp32 = (Int32)(*(w+l)) * *(coef+j); j++; l=(l+Ns)%s;
            w_0   -= temp32<<1;
            temp32 = (Int32)(*(w+l)) * *(coef+j); j++;
            w_0   -= temp32<<1;
            w_0   += 0x4000;            // Rounding

            temp16 = *(w+l);
            *(w+l) = (Int16)(w_0>>15); // Save in Q15

            w_0    = (Int32)temp16 * *(coef+j); j++;
            w_0  <<= 1;
            temp32 = (Int32)*(w+l) * *(coef+j); j++; l=(l+Ns)%s;
            w_0   += temp32<<1;
            temp32 = (Int32)*(w+l) * *(coef+j); j=(j+1)%m; l=(l+1)%s;
            w_0   += temp32<<1;
            w_0   += 0x800;             // Rounding
        }
        y[n] = (Int16)(w_0>>12);        // Output in Q15 format
    }
}
```

Cascade IIR Filter Using Assembly Program

Programski fajlovi

Fajl	Opis
`asmIIRTest.c`	program za testiranje *floating-point* IIR filtra
`asmIIR.asm`	asemblerska implementacija IIR filtra drugog reda
`asmIIR.h`	C header fajl
`tistdtypes.h`	header fajl standardnih tipova podataka
`fdacoefsMATLAB.h`	FDATool-MATLAB fajl
`tmwtypes.h`	definicija podataka za MATLAB C header fajl
`c5505.cmd`	linker fajl
`in.pcm`	ulazni signal

Zadatak

- Importovati projekat u CCS i pokrenuti program.
- Proveriti da li se na izlazu dobijaju 60 dB oslabljene sinusoide na 800 Hz i 3300Hz. Ulazni signal sadrži tri sinusoide na frekvencijama 800 Hz, 1800 Hz i 3300Hz.

```
        or      #0x340,mmap(ST1_55)       ; Set FRCT,SXMD,SATD
        bset    SMUL                      ; Set SMUL
        sub     #1,T0                     ; Number of samples - 1
        mov     T0,BRC0                   ; Set up outer loop counter
        sub     #1,T1,T0                  ; Number of sections -1
        mov     T0,BRC1                   ; Set up inner loop counter

        mov     T1,T0                     ; Set up circular buffer sizes
        sfts    T0,#1
        mov     mmap(T0),BK03             ; BK03=2*number of sections
        sfts    T0,#1
        add     T1,T0
        mov     mmap(T0),BK47             ; BK47=5*number of sections
        mov     mmap(AR3),BSA23           ; Initial delay buffer base
        mov     mmap(AR2),BSA67           ; Initial coefficient base
        amov    #0,AR3                    ; Initial delay buffer entry
        amov    #0,AR7                    ; Initial coefficient entry
        or      #0x88,mmap(ST2_55)
        mov     #1,T0                     ; Used for shift left
||      rptblocal sample_loop-1         ; Start IIR filter loop
        mov     *AR0+ <<#12,AC0           ; AC0 = x(n)/8 (i.e. Q12)
||      rptblocal filter_loop-1           ; Loop for each section
        masm    *(AR3+T1),*AR7+,AC0       ; AC0-=ai1*wi(n-1)
        masm    T3=*AR3,*AR7+,AC0       ; AC0-=ai2*di(n-2)
        mov     rnd(hi(AC0<<T0)),*AR3     ; wi(n-2)=wi(n)
||      mpym    *AR7+,T3,AC0              ; AC0+=bi2*wi(n-2)
        macm    *(AR3+T1),*AR7+,AC0       ; AC0+=bi0*wi(n-1)
        macm    *AR3+,*AR7+,AC0           ; AC0+=bi1*wi(n)
filter_loop
        mov     rnd(hi(AC0<<#4)),*AR1+    ; Store result in Q15
sample_loop
```

Intrinsic komande 1

C Compiler Intrinsic (a,b,c are 16-bit and d,e,f are 32-bit data)	Description
`c = _sadd(int a, int b);`	Adds 16-bit integers a and b, with SATA set, producing a saturated 16-bit result c.
`f = _lsadd(long d, long e);`	Adds 32-bit integers d and e, with SATD set, producing a saturated 32-bit result f.
`c = _ssub(int a, int b);`	Subtracts 16-bit integer b from a with SATA set, producing a saturated 16-bit result c.
`f = _lssub(long d, long e);`	Subtracts 32-bit integer e from d with SATD set, producing a saturated 32-bit result f.
`c = _smpy(int a, int b);`	Multiplies a and b, and shifts the result left by 1. Produces a saturated 16-bit result c. (upper 16-bit, SATD and FRCT set)
`f = _lsmpy(int a, int b);`	Multiplies a and b, and shifts the result left by 1. Produces a saturated 32-bit result f. (SATD and FRCT set)
`f = _smac(long d, int a, int b);`	Multiplies a and b, shifts the result left by 1, and adds it to d. Produces a saturated 32-bit result f. (SATD, SMUL and FRCT set)
`f = _smas(long d, int a, int b);`	Multiplies a and b, shifts the result left by 1, and subtracts it from d. Produces a 32-bit result f. (SATD, SMUL and FRCT set)
`c = _abss(int a);`	Creates a saturated 16-bit absolute value. $c = \|a\|$, _abss(0x8000) => 0x7FFF (SATA set)
`f = _labss(long d);`	Creates a saturated 32-bit absolute value. $f = \|d\|$, _labss(0x8000000) => 0x7FFFFFFF (SATD set)
`c = _sneg(int a);`	Negates the 16-bit value with saturation. $c = -a$, _sneg(0xffff8000) => 0x00007FFF

Intrinsic komande 2

C Compiler Intrinsic (a,b,c are 16-bit and d,e,f are 32-bit data)	Description
`f = _lsneg(long d);`	Negates the 32-bit value with saturation. $f = -d$, _lsneg(0x80000000) => 0x7FFFFFFF
`c = _smpyr(int a, int b);`	Multiplies a and b, shifts the result left by 1, and rounds the result c. (SATD and FRCT set)
`c = _smacr(long d, int a, int b);`	Multiplies a and b, shifts the result left by 1, adds the result to d, and then rounds the result c. (SATD, SMUL and FRCT set)
`c = _smasr(long d, int a, int b);`	Multiplies a and b, shifts the result left by 1, subtracts the result from d, and then rounds the result c. (SATD, SMUL and FRCT set)
`c = _norm(int a);`	Produces the number of left shifts needed to normalize a and places the result in c.
`c = _lnorm(long d);`	Produces the number of left shifts needed to normalize d and places the result in c.
`c = _rnd(long d);`	Rounds d to produces the 16-bit saturated result c. (SATD set)
`c = _sshl(int a, int b);`	Shifts a left by b and produces a 16-bit result c. The result is saturated if b is greater than or equal to 8. (SATD set)
`f = _lsshl(long d, int a);`	Shifts a left by b and produces a 32-bit result f. The result is saturated if a is greater than or equal to 8. (SATD set)
`c = _shrs(int a, int b);`	Shifts a right by b and produces a 16-bit result c. (SATD set)
`f = _lshrs(long d, int a);`	Shifts d right by a and produces a 32-bit result f. (SATD set)
`c = _addc(int a, int b);`	Adds a, b, and Carry bit and produces a 16-bit result c.
`f = _laddc(long d, int a);`	Adds d, a, and Carry bit and produces a 32-bit result f.

Funkcija za računanje jednog bloka izlaznog signala kaskadne realizacije IIR filtra se izvršava

- $187563 \times$ clk pisano u C,
- $101004 \times$ clk intrinsic C,
- oko $5000 \times$ clk asembler.