FIR Tiefpass Filter in Assembler

Ziel

Diese Übung hat zum Ziel ein FIR-Filter mit gegebener Grenzfrequenz und Länge als Funktion in Assembler zu implementieren. Die Filterkoeffizienten werden mit Matlab nach verschiedenen Methoden bestimmt.

Umfeld

Aus Effizienzgründen werden Filteralgorithmen immer in Assembler codiert. Grund hierfür sind die schlechten Optimierungsfähigkeiten der Compiler. So kann parallele Instruktionsausführung nur beschränkt verwendet werden. Dabei wird aber keinesfalls das gesamte Projekt in Assembler codiert, sondern nur die zeitkritischen Teile, in der Regel auf Stufe Funktion.

So verfügt man dann über eine Bibliothek von Funktionen, die zwar in Assembler codiert sind aber bequem aus der Hochsprache aufgerufen werden können.

Der Zugriff auf C-Variablen kann in Assembler direkt erfolgen indem dem C-Variablenname ein *F* voran gestellt wird. So kann ein FIR-Filter folgendermassen realisiert werden:

```
#include "FIR_COEFF.H"                        // Filterkoeffizienten fuer allgemeines FIR-Filter

_X _fract  _circ delay[NTAPS];                // NTAPS Speicherzellen fuer History und Eingangssignal
_X _fract  _circ *delayPtr=delay;             // NTAP wird in FIR_COEFF definiert

_Y _fract _circ *FIRCoeffPtr=FIRcoeff;

     /* Funktionsprototypen */
void codecInit(int, int, int, int);         // CS4215 Codec und zugehoerige Interruptroutinen initialisieren
void filterFIR(void);

/* FIR-Filterroutine
   Die Filterkoeffizienten und Historydaten werden in Ringpuffern gehalten. Der Zugriff erfolgt ueber Zeiger,
   die in den globalen Variablen delayPtr und FIRCoefPtr gehalten werden.
   Die Datenein- und Ausgabe erfolgt (aus Effizienzgruenden) ebenfalls ueber Seiteneffekt mit den globalen
   Variablen x und y. Man beachte, dass der Zugriff auf eine C-Variable in Assembler immer ein voran gestelltes 'F'
   benoetigt!
   Durch den umfangereichen Kontext-Save und Restore wird die maximale Filterlaenge bei hohen Sampleraten recht kurz
   (Groessenordnung < 60 Taps), da innerhalb 1/44kHz [s] auch noch die Datenaquisition und Ausgabe in den
   Interruptroutinen erfolgen muss.

 */
void filterFIR(void)
{
#pragma asm

     move      R0,x:(R7)+              ; Kontext save
     move      M0,x:(R7)+
     move      R4,x:(R7)+
     move      M4,x:(R7)+
     move      #(NTAPS-1),M0           ; History Puffer [R0]=z, [R0+1]=z^-1, [R0+2]=z^-2...
     move      #(NTAPS-1),M4           ;  Filterkoeffizienten ho=[R4], h1=[R4+1],...
     move      FdelayPtr,R0            ; Historydaten in X
     move      FFIRCoeffPtr,R4         ; Filterkoeffizienten Y

     move      Fx,X0             ; Eingangsdatum aus der C-Variable x in das Register x0 holen

     clr       A        X0,X:(R0)+    Y:(R4)+,Y0      ; A=0, x(n)->save ,h0->Y0
     rep    #NTAPS-1
     mac       X0,Y0,A  X:(R0)+,X0    Y:(R4)+,Y0      ; nach dem Rechenschema FIR-Filter bis N-1
     macr      X0,Y0,A  (R0)-                         ; Letzte Rechnung mit Rundung

     move      A,Fy                     ; Resultat in C-Variable y speichern

     move      R0,FdelayPtr           ; Zeiger zurueck speichern
     move      R4,FFIRCoeffPtr
     move      x:(R7)-,M4             ; Kontext restore
     move      x:(R7)-,R4
     move      x:(R7)-,M0
     move      x:(R7)-,R0

#pragma endasm

}
```

Das zugehörige Koeffizientenfile enthält die Filterkoeffizienten:

FIR_COEFF.H:

```
/* Allgemeine FIR Filter Koeffizientendefinition zur Benutzung mit FIR-Filter
   Funktion.

   Gerhard Krucker
   8.3.2001
*/

/* FIR Filterkoeffizienten, erzeugt mit Matlab 5.2
   Equiripple FIR Bandpass 51 Taps, fc1=500Hz, fc2=2000Hz fs=16kHz
   14.3.2001, G. Krucker
*/


#define  NTAPS 51
#pragma asm
NTAPS equ 51
#pragma endasm


_Y   _fract _circ FIRcoeff[NTAPS]=
{    0.0008,
     0.0008,
     0.0003,
    -0.0001,
     0.0003,
     0.0021,
     0.0048,
     0.0066,
     0.0054,
    -0.0000,
    -0.0081,
    -0.0150,
    -0.0164,
    -0.0110,
    -0.0023,
     0.0017,
    -0.0072,
    -0.0317,
    -0.0645,
    -0.0900,
    -0.0898,
    -0.0533,
     0.0156,
     0.0971,
     0.1628,
     0.1879,
     0.1628,
     0.0971,
     0.0156,
    -0.0533,
    -0.0898,
    -0.0900,
    -0.0645,
    -0.0317,
    -0.0072,
     0.0017,
    -0.0023,
    -0.0110,
    -0.0164,
    -0.0150,
    -0.0081,
    -0.0000,
     0.0054,
     0.0066,
     0.0048,
     0.0021,
     0.0003,
    -0.0001,
     0.0003,
     0.0008,
     0.0008};
```

Die Koeffizienten sind mit Matlab 5.2 synthetisiert worden, gemäss den Filteranforderungen:

Kaiser FIR-Bandpass
Samplefrequenz: 16kHz
Untere Grenzfrequenz: 500Hz
Obere Grenzfrequenz: 2kHz
Kaiser-Gamma: 4.93
Filterlänge: 51 Taps

Matlab:

```
%
% FIR Bandpass Filter Design mit Matlab
% Gerhard Krucker
%
fS=16000;            %Sample Frequenz [Hz]
fC1=500;             %Grenzfrequenz unten[Hz]
fC2=2000;            % Grenzfrequenz oben [Hz]
NTAPS=51;            %Filterlänge
order=NTAPS-1;
WN=[2*fC1/fS,2*fC2/fS]    % Normierte digitale Grenzfrequenz
w=kaiser(NTAPS,4.93);
h=fir1(order,WN,w);
transpose(h)              % Ausgabe der Filterkeoffizienten
```

Aufgaben

1. Synthetisieren der Filterkoeffizienten für ein FIR Hochpassfilter mit Blackman-Fenster. Die Grenzfrequenz soll 2kHz betragen, bei einer Samplerate von 48kHz. Die Filterlänge L sei 61 Taps.

2. Aufsetzen eines neuen, leeren C-Projektes (z.B. Uebung_FIR_HP2kHz, keine Leerschläge erlaubt) auf der Grundlage des Prototypen-Files. Kommentieren und Anpassen des Prototypen. Vgl. hierzu auch Übung 1.

3. Synthetisieren und Implementieren eines Bandpassfilter analog dem Beispiel aber für 48kHz Samplerate mit dem Remez-Exchange Algorithmus. Bei der doppelten (halben) Grenzfrequenz soll die Dämpfung 20dB (1/100) betragen, die Welligkeit im Durchlassbereich ≤ 0.1dB.

   Kann es mit dieser Hardware und Filter-Funktion implementiert werden? Wenn nein Vorschläge, welche Änderungen sinnvoll und notwendig wären.