**Tema 1**

**Iepure Albert**

**442G**

**Cerinta**: Se va scrie o funcţie care va returna energia calculată şi factorul de scalare. N este un număr multiplu de 4.

**Descrierea Programului:**

**Matlab:**

Se genereaza un semnal sinusoidal de amplitudine data avand 40 de esantioane. Valorile acestor esantioane sunt subunitare. Aceste esantioane se inmultesc cu 2^15 pentru a obtine numere intregi pe 16 biti si sunt scrise in fisierul „a.dat”, de unde vor fi citite de programul in C.

Se calculeaza energia semnalului in Matlab (matlab\_energy) pe baza esantioanelor generate si se afiseaza pe ecran. Se citesc apoi din fisier energia semnalului calculata de programul in C („signalenergy.dat”) si factorul de scalare („scalefactor.dat”). Deoarece energia calculata de programul in C ete un numar intreg cuprins intre 0 si 32767, impartim energia astfel obtinuta la 2^15=32768 pentru a obtine un numar fractionar, subunitar. Luand in considerare si factorul de scalare si aportul sau la calculul energiei semnalului, se afiseaza pe ecran energia semnalului calculata de Starcore140 (starcore\_energy) si se calculeaza si se afiseaza si diferenta ce rezulta intre cele doua valori (starcore\_energy si matlab\_energy).

**Programul C - main:**

Se citeste vectorul a din fisierul „a.dat”, apoi este apelata functia „signalenergy”. Aceasta este de tip void, are ca parametru de intrare un vector de elemente si are drept output

2 fisiere: scalefactor.dat si signalenergy.dat, fisiere care contin factorul de scalare, respectiv energia semnalului.

**Programul C - signalenergy:**

Primul pas il reprezinta calculul energiei intr-o bucla for folosind instructiunea „e=L\_mac(e,a[i],a[i]);” unde a[i] este vectorul cu esantioane.

Daca e>=1 (dar deoarece esantioanele subunitare ale vectorului a[i] au fost inmultite cu 32768=2^15, aceasta conditie se traduce prin e>=32767 ) se intra intr-o bucla while in care vectorul a este scalat cu 2, iar la fiecare impartire cu 2 variabila scalefactor se incrementeaza cu 1,dupa care energia este recalculata cu elementele scalate ale noului vector a. Bucla se inchide atunci cand valoarea energiei este mai mica decat 1 (adica mai mica decat 32767).

Dupa iesirea din bucla factorul de scalare este scris in fisierul „scalefactor.dat”, iar energia calculata in fisierul „signalenergy.dat”.

**Rezultatele Testarii:**

Vectorul a este de forma „x=A\*cos(2\*pi/5\*n)”, A fiind amplitudinea semnalului.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Amplitude** | **matlab\_energy** | **starcore\_energy** | **scalefactor** | **Error** |
| **0.1** | 0.2000 | 0.2000 | 0 | 0.0000 |
| **0.25** | 1.2500 | 1.2496 | 1 | 0.0004 |
| **0.5** | 5.0000 | 4.9985 | 2 | 0.0015 |
| **1** | 20.0000 | 19.9902 | 3 | 0.0098 |

**Explicatii suplimentare:**

Se obseerva ca eroarea creste direct proportional cu factorul de scalare, deoarece atunci cand se incrementeaza factorul de scalare (se impart elementele vectorului la 2) se fac rotunjiri care genereaza o eroare nenula.

**Optimizare**

Pentru a folosi in acelasi ciclu de ceas cele patru ALU ale DSP-ului pentru a efectua patru inmultiri se apeleaza la tehnica split computation.

Partea de cod care implementeaza in C aceasta tehnica este:

energy0=0;

energy1=0;

energy2=0;

energy3=0;

for(i=0;i<DataBlockSize;i+=4)

{

energy0=L\_mac(energy0,a[i],a[i]);

energy1=L\_mac(energy1,a[i+1],a[i+1]);

energy2=L\_mac(energy2,a[i+2],a[i+2]);

energy3=L\_mac(energy3,a[i+3],a[i+3]);

}

energy0 = L\_add(energy0, energy1);

energy2 = L\_add(energy2, energy3);

energy0 = L\_add(energy0, energy2);

energy0=round(energy0);

iar in assembler este:

LOOPSTART3

DW6

[

mac d8,d8,d1 ;[19,16] 1%=1

mac d9,d9,d2 ;[20,16] 1%=1

mac d10,d10,d3 ;[21,16] 1%=1

mac d11,d11,d4 ;[19,16] 1%=1

move.4f (r1)+,d8:d9:d10:d11 ;[19,16] 0%=0

]

LOOPEND3

Imbunatatirea performantei programului se traduce prin micsorarea timpului in care programul este executat.