



**Universitatea POLITEHNICA din București**  
Facultatea de electronică, Telecomunicații și Tehnologia Informației

---



## **TEMA 1**

### **PARTEA 1.B**

#### **PROCESOARE DE SEMNAL IN COMUNICATII**

**Nume student: Stanescu Vlad-Constantin**

**Grupa: 444C**

## 1. Cerinta temei

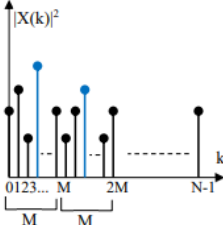
**I.B.** Folosind proiectul CodeWarrior realizat la **tema I.A.**, **optimizați funcția** implementată în C folosind tehnicile de optimizare prezentate în curs (loop merging, loop unroll, split computation, multisample, etc.) Precizați ce tehnici de optimizare ați aplicat și indicați modificările făcute în program (comentând instrucțiunile modificate din programul inițial urmate de modificări). Indicați în codul rezultat în limbaj de asamblare efectul optimizării (instrucțiuni realizate în paralel în același ciclu mașină, transfer multiplu, etc.). Se vor alege dimensiunile vectorilor/datelor de intrare multiplu de 4 sau valori corespunzătoare astfel încât să se obțină în fișierele .sl efectele de optimizare dorite (cîte 4 operații aritmetico-logice realizate în același ciclu mașină simultan cu 2 transferuri din/în memorie).

Se va verifica în Matlab corectitudinea rezultatelor după optimizare (NU se va optimiza un program aferent temei 1 neverificat sau incorect!).

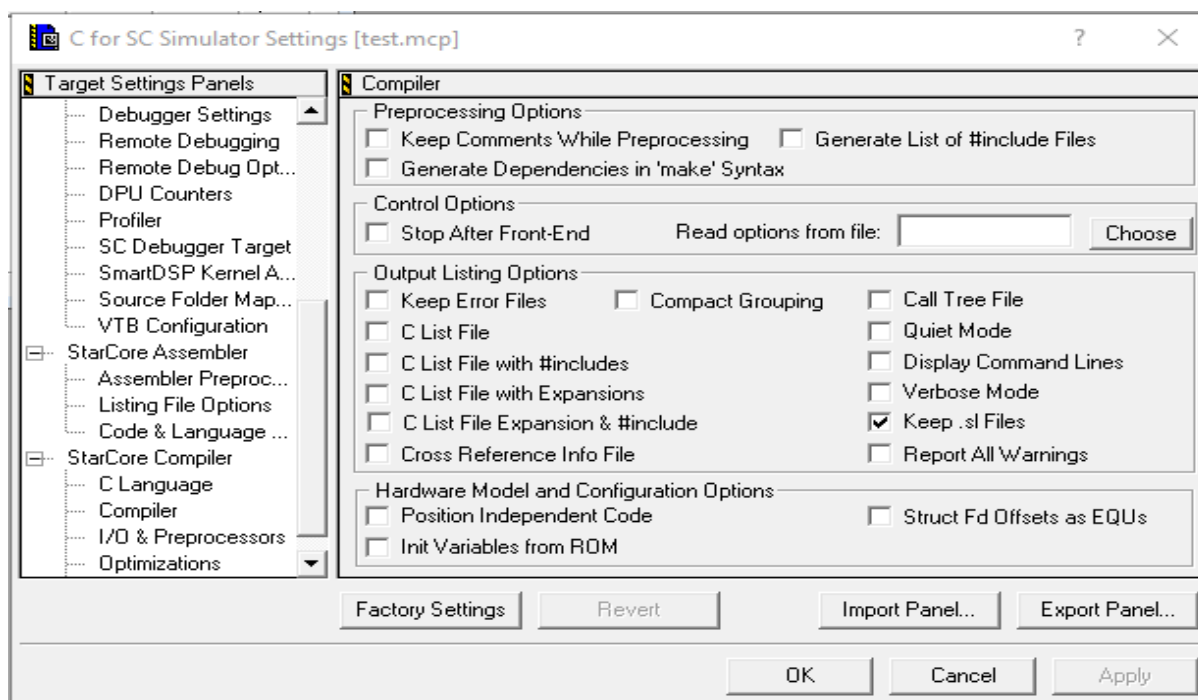
Tema I.B. se va salva într-un director (folder) cu numele **44gs\_IB\_Nume\_Prenume** (unde 'gs' e grupa și seria).

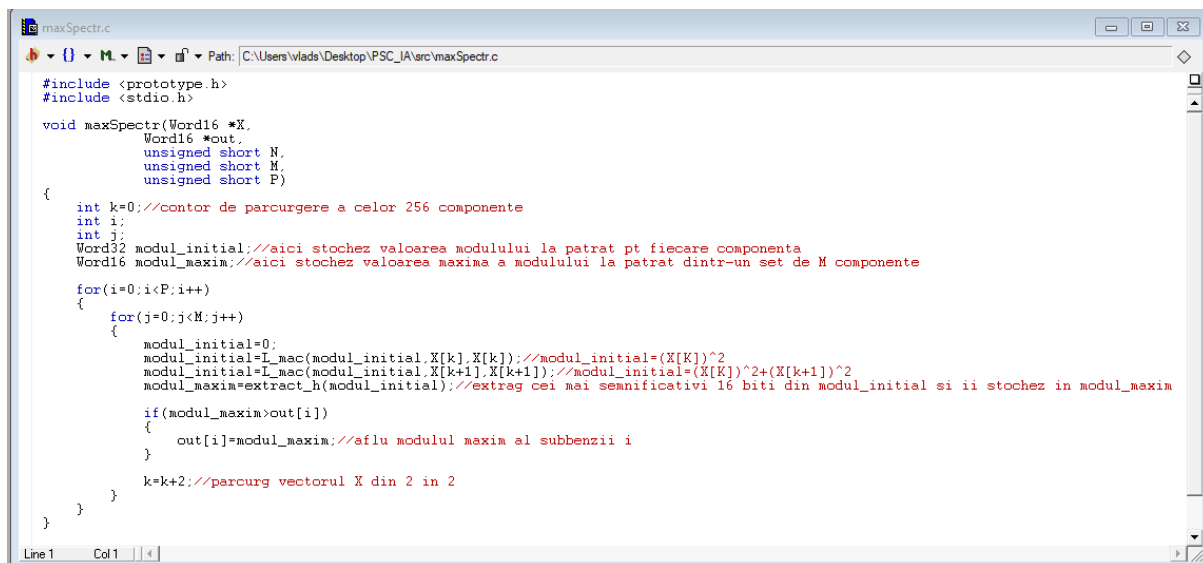
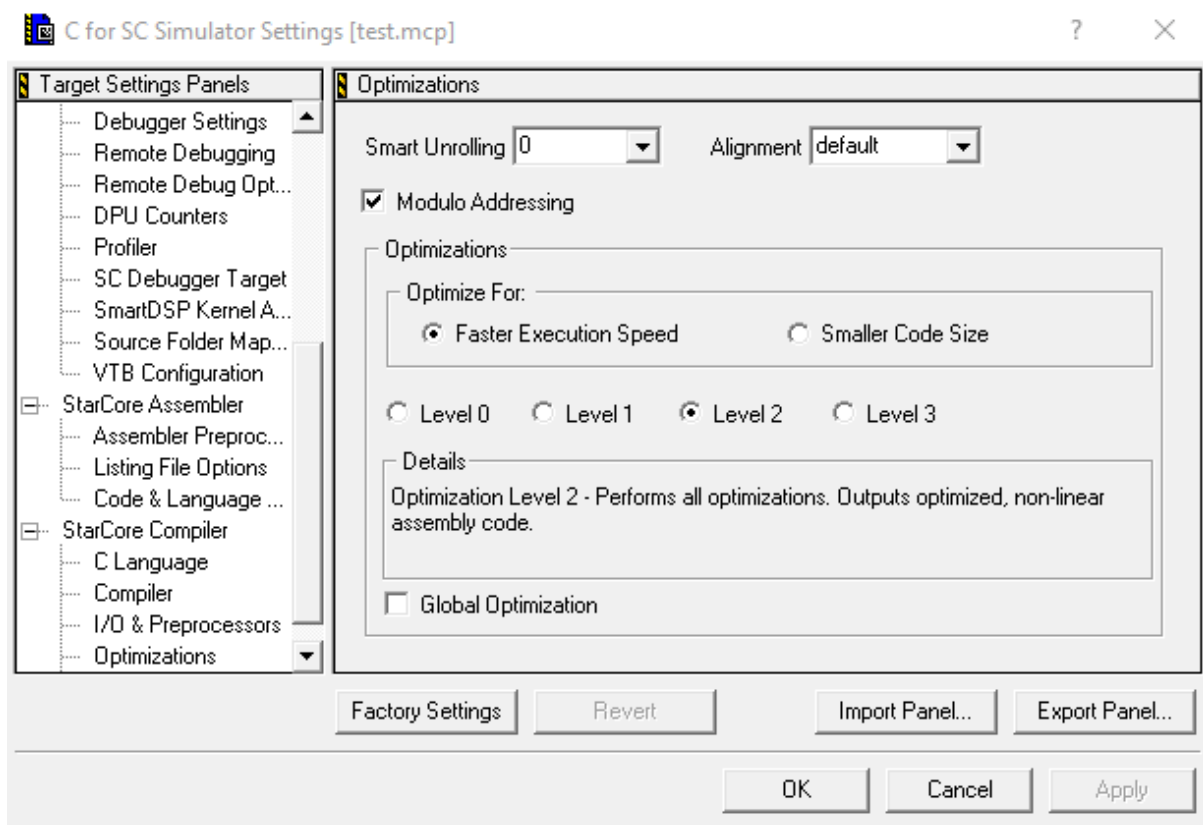
Tema I.B. va conține:

- un fișier Word cu: descrierea programului optimizat, toate detaliile de implementare (etapele de optimizare), rezultatele testării (erori de calcul evaluate în Matlab) și explicații suplimentare.
- un director cu proiectul StarCore integral (cu programele C și toate subfolderele pe care le are și proiectul dat ca model pe site, "test", inclusiv un subdirector cu programele Matlab de test).

9	<pre>void maxim_spectr(Word16 *X,                   Word16 *out,                   unsigned short N,                   unsigned short M,                   unsigned short P);</pre>	<p>Considerând <math>N</math> componente complexe ale unui vector <math>X</math> (de exemplu componentele spectrale calculate cu TFD de lungime <math>N</math>) să se determine <b>valorile maxime</b> ale modulului la pătrat ale componentelor <u>din fiecare set</u> (subbandă) de <math>M</math> componente.</p> <p>Se consideră <math>N</math> multiplu de <math>M</math> și se va declara (pe lângă valorile <math>M</math> și <math>N</math>) și constanta <math>P=N/M</math> pentru a evita operația de împărțire în program.</p> <p><math>N</math>, <math>M</math> și <math>P</math> trebuie alese astfel încât să fie multiplu de 4 (de ex. <math>N=1024</math>, <math>M=16</math> și rezultă <math>P=64</math>).</p>  <p>Word16 *X → pointer la vectorul cu componente complexe. Părțile reale și imaginare ale elementelor vectorului sunt stocate astfel: <math>\{X\_re(0), X\_im(0), X\_re(1), X\_im(1), \dots, X\_re(N-1), X\_im(N-1)\}</math>. În total vectorul <math>X</math> are lungime <math>2N</math>.</p> <p>Word16 *out → pointer la vectorul de ieșire (conține cele <math>P</math> valori maxime).</p>
---	---	--

## 2. Functia maxSpectr





- Se setează în fereastra Settings (butonul cu același nume sau Alt+F7),
  - la secțiunea pentru Compilator, opțiunea Keep .sl Files.
  - la secțiunea Optimizations:
    - se modifică opțiunea Smart Unrolling la valoarea 0
    - se selectează Level 2 de optimizare.
- Se compilează și se observă codul rezultat în fișierul .sl.

```

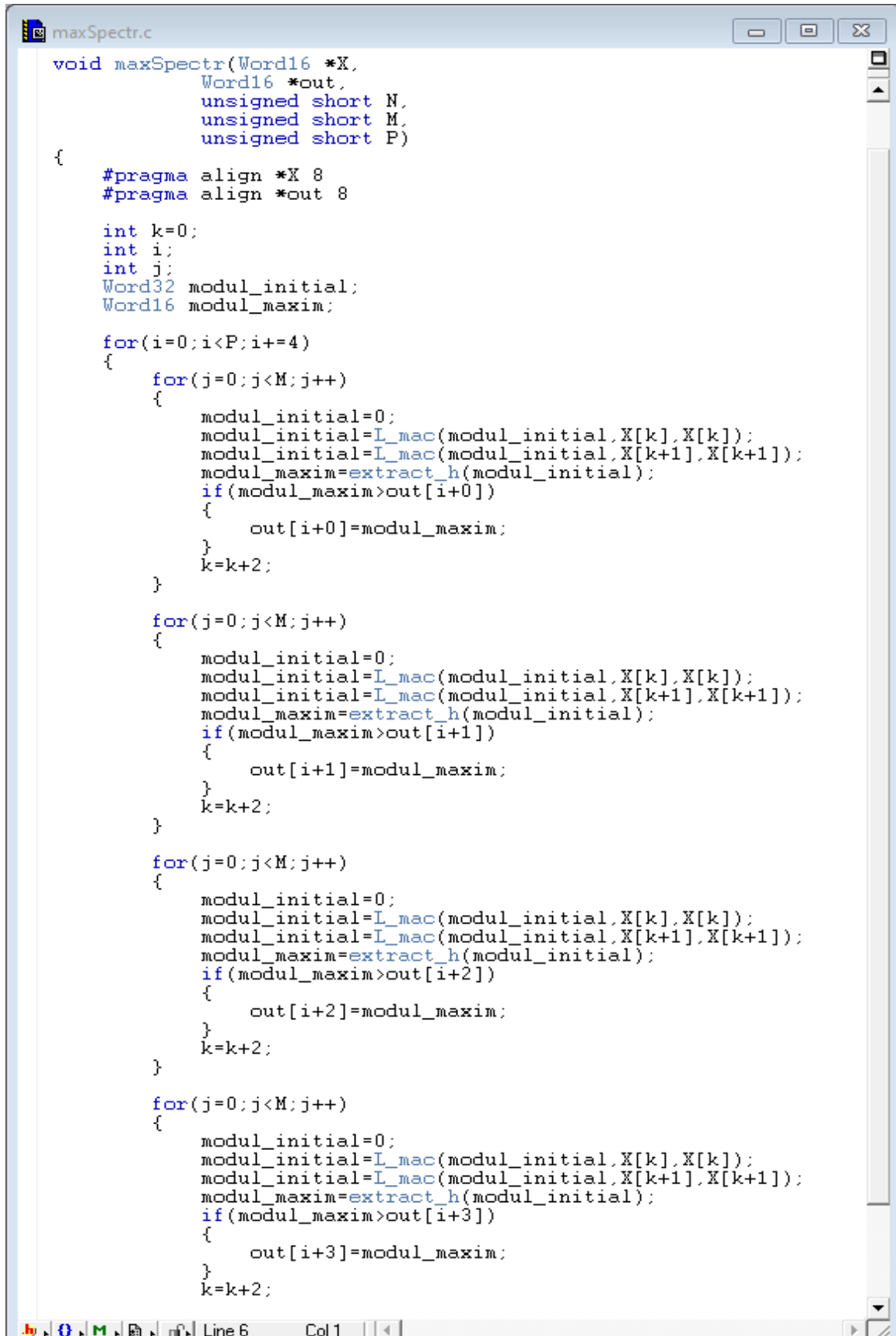
maxSpectr.sl
DW14
[
    move.f    (r3)+n3,d1          ;[21,1] 0%=0
    [
        mpy    d1,d1,d4          ;[21,1] 1%=0
        adda   r0,r2              ;[22,1]
    ]
]
DW15
    move.f    (r2)+n3,d6          ;[22,1] 0%=0
DW16
    mac       d6,d6,d4            ;[22,1] 2%=0
DW17
    asrw      d4,d1               ;[25,1] 3%=0
DW18
[
    aslw      d1,d1               ;[25,1] 4%=0
    skipls    PL000               ;[0,0]
]
    FALIGN
    LOOPSTART3
PL001
DW19
[
    max       d1,d5               ;[22,1] 5%=1
    move.f    (r3)+n3,d1          ;[21,1] 0%=0
    move.f    (r2)+n3,d6          ;[22,1] 0%=0
]
DW20
    mpy       d1,d1,d4            ;[21,1] 1%=0
DW21
    mac       d6,d6,d4            ;[22,1] 2%=0
DW22
    asrw      d4,d1               ;[25,1] 3%=0
DW23
    aslw      d1,d1               ;[25,1] 4%=0
    LOOPEND3
PL000
DW24
    max       d1,d5               ;[22,1] 5%=1
DW25
    moves.f   d5,(r1)             ;[0,1]
L4
DW26
    adda      #<2,r1              ;[25,1]
DW27

```

Dupa recompilare, in fisierul .sl observam ca programul functiei, la nivelul de optimizare 2, avem un numar mic de operatii de asamblare in fiecare ciclu de ceas, asadar programul nu foloseste toti acesti ciclii efficient si de asta folosim optimizarea. Procesorul de semnal StarCore 140 poate efectua in acelasi ciclu 4 operatii aritmetico-logice si 2 transferuri/adresari de memorie.

### 3. Optimizarea functiei maxSpectr

#### 3.1. LOOP UNROLLING pentru bucla exterioara



```
void maxSpectr(Word16 *X,
               Word16 *out,
               unsigned short N,
               unsigned short M,
               unsigned short P)
{
    #pragma align *X 8
    #pragma align *out 8

    int k=0;
    int i;
    int j;
    Word32 modul_initial;
    Word16 modul_maxim;

    for(i=0; i<P; i+=4)
    {
        for(j=0; j<M; j++)
        {
            modul_initial=0;
            modul_initial=L_mac(modul_initial, X[k], X[k]);
            modul_initial=L_mac(modul_initial, X[k+1], X[k+1]);
            modul_maxim=extract_h(modul_initial);
            if(modul_maxim>out[i+0])
            {
                out[i+0]=modul_maxim;
            }
            k=k+2;
        }

        for(j=0; j<M; j++)
        {
            modul_initial=0;
            modul_initial=L_mac(modul_initial, X[k], X[k]);
            modul_initial=L_mac(modul_initial, X[k+1], X[k+1]);
            modul_maxim=extract_h(modul_initial);
            if(modul_maxim>out[i+1])
            {
                out[i+1]=modul_maxim;
            }
            k=k+2;
        }

        for(j=0; j<M; j++)
        {
            modul_initial=0;
            modul_initial=L_mac(modul_initial, X[k], X[k]);
            modul_initial=L_mac(modul_initial, X[k+1], X[k+1]);
            modul_maxim=extract_h(modul_initial);
            if(modul_maxim>out[i+2])
            {
                out[i+2]=modul_maxim;
            }
            k=k+2;
        }

        for(j=0; j<M; j++)
        {
            modul_initial=0;
            modul_initial=L_mac(modul_initial, X[k], X[k]);
            modul_initial=L_mac(modul_initial, X[k+1], X[k+1]);
            modul_maxim=extract_h(modul_initial);
            if(modul_maxim>out[i+3])
            {
                out[i+3]=modul_maxim;
            }
            k=k+2;
        }
    }
}
```

Line 6 Col 1

- Pentru a se genera un cod optimizat, trebuie efectuate următoarele modificări în programul C:

- alinierea datelor în memorie pentru a permite transferul datelor în paralel. Se introduce pragma de aliniere după declararea vectorului X.

```
#pragma align *X 8
#pragma align *out 8
```

- definirea a 4 variabile pentru calculul separat al celor 4 sume și eliminarea dependenței rezultatelor intermediare

- în buclă se repetă de 4 ori operația de înmulțire și acumulare, contorul buclei FOR se incrementează din 4 în 4.

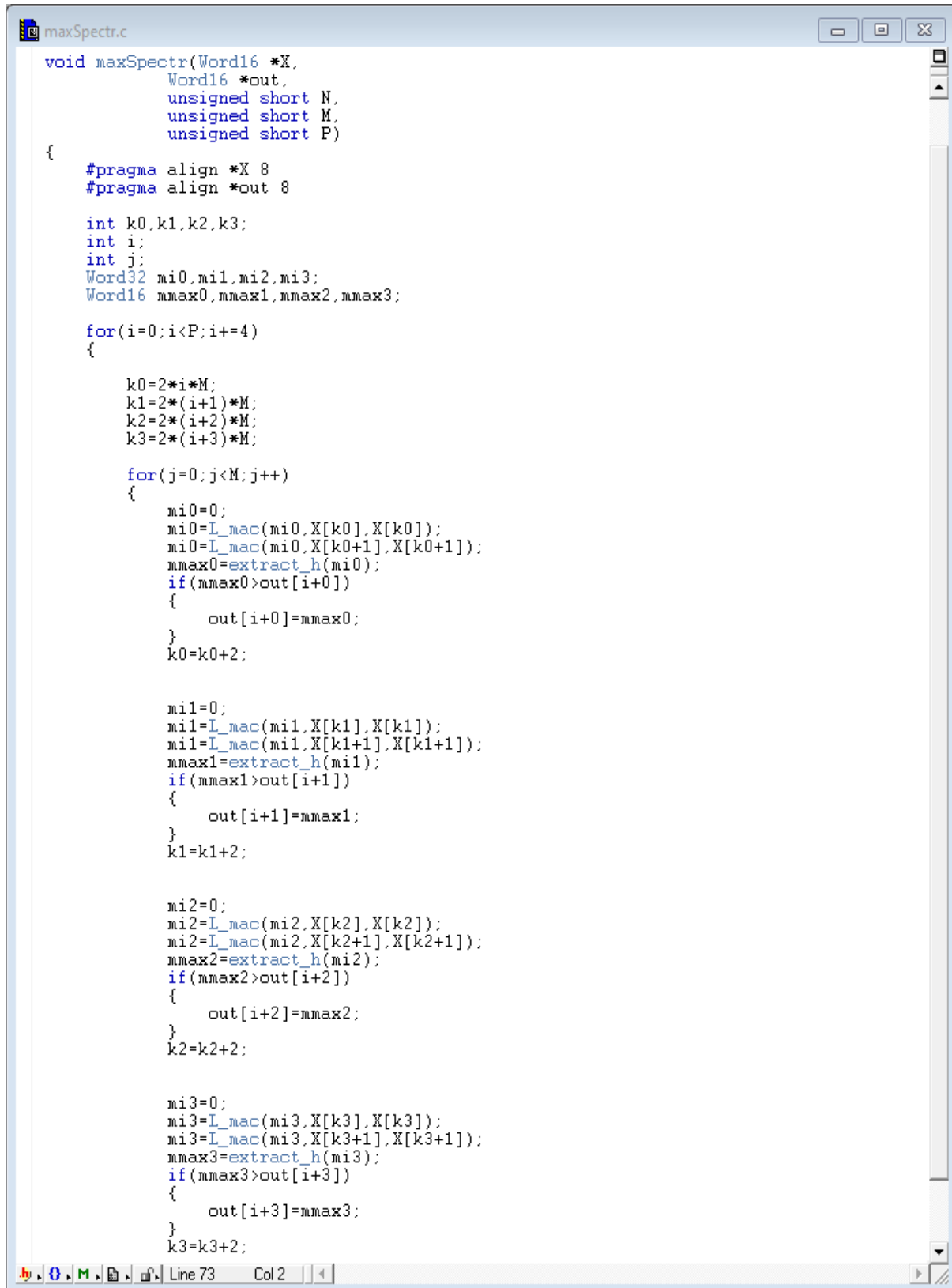
- Se recompilează și se observă codul rezultat în fișierul .sl.

Se observa ca functia maxSpectr are in componenta sa 2 bucle FOR. Tehnica multisample permite transformarea codului pentru bucle efectuate una în interiorul alteia, inasa instructiunile din cadrul celor doua bucle FOR nu depind de ambii contori i si j, ci doar de i (este index-ul pentru fiecare element din vectorul de iesire out). Contorul j a fost folosit doar pentru a parcurge cele M componente, iar prelucrarea acestora nu depinde direct de valoarea acestuia. Asadar am folosit pe rand tehnica de optimizare LOOP UNROLLING si LOOP MERGING. Bucla exterioara a fost modificata astfel incat sa se calculeze 4 valori maxime in loc de una, iar dupa recompilare nu se observa o schimbare foarte mare, lucru ce era de asteptat.

```
for(i=0;i<P;i+=4)
{
    for(j=0;j<M;j++)
    {
        modul_initial=0;
        modul_initial=L_mac(modul_initial,X[k],X[k]);
        modul_initial=L_mac(modul_initial,X[k+1],X[k+1]);
        modul_maxim=extract_h(modul_initial);
        if(modul_maxim>out[i+0])
        {
            out[i+0]=modul_maxim;
        }
        k=k+2;
    }
}
```

```
maxSpectr.sl
DW24
[
    move.2f    (r5)+,d6:d7          ;[24,1] 0%=0
    doensh3    d3                   ;[0,0]
]
DW25
    mpy        d6,d6,d8             ;[24,1] 1%=0
DW26
[
    mac        d7,d7,d8             ;[25,1] 2%=1
    move.2f    (r5)+,d6:d7          ;[24,1] 0%=0
]
DW27
[
    asrw       d8,d0                ;[27,1] 3%=1
    mpy        d6,d6,d8             ;[24,1] 1%=0
]
    LOOPSTART3
DW28
[
    mac        d7,d7,d8             ;[25,1] 2%=1
    aslw       d0,d0                ;[27,1] 4%=2
    move.2f    (r5)+,d6:d7          ;[24,1] 0%=0
]
DW29
[
    max        d0,d4                ;[24,1] 5%=2
    asrw       d8,d0                ;[27,1] 3%=1
    mpy        d6,d6,d8             ;[24,1] 1%=0
]
    LOOPEND3
DW30
```

## 3.2. LOOP MERGING pentru bucla din interior



```
maxSpectr.c

void maxSpectr(Word16 *X,
               Word16 *out,
               unsigned short N,
               unsigned short M,
               unsigned short P)
{
    #pragma align *X 8
    #pragma align *out 8

    int k0,k1,k2,k3;
    int i;
    int j;
    Word32 mi0,mi1,mi2,mi3;
    Word16 mmax0,mmax1,mmax2,mmax3;

    for(i=0;i<P;i+=4)
    {
        k0=2*i*M;
        k1=2*(i+1)*M;
        k2=2*(i+2)*M;
        k3=2*(i+3)*M;

        for(j=0;j<M;j++)
        {
            mi0=0;
            mi0=L_mac(mi0,X[k0],X[k0]);
            mi0=L_mac(mi0,X[k0+1],X[k0+1]);
            mmax0=extract_h(mi0);
            if(mmax0>out[i+0])
            {
                out[i+0]=mmax0;
            }
            k0=k0+2;

            mi1=0;
            mi1=L_mac(mi1,X[k1],X[k1]);
            mi1=L_mac(mi1,X[k1+1],X[k1+1]);
            mmax1=extract_h(mi1);
            if(mmax1>out[i+1])
            {
                out[i+1]=mmax1;
            }
            k1=k1+2;

            mi2=0;
            mi2=L_mac(mi2,X[k2],X[k2]);
            mi2=L_mac(mi2,X[k2+1],X[k2+1]);
            mmax2=extract_h(mi2);
            if(mmax2>out[i+2])
            {
                out[i+2]=mmax2;
            }
            k2=k2+2;

            mi3=0;
            mi3=L_mac(mi3,X[k3],X[k3]);
            mi3=L_mac(mi3,X[k3+1],X[k3+1]);
            mmax3=extract_h(mi3);
            if(mmax3>out[i+3])
            {
                out[i+3]=mmax3;
            }
            k3=k3+2;
        }
    }
}
```

Line 73 Col 2



Modificarea cea mai importanta consta in crearea de noi variabile ce permit calculul a 4 valori maxime in paralel. Am separat modulele initiale in 4 variabile, acelasi lucru facandu-l si in cazul valorilor maxime. S-au folosit de asemenea contori k diferiti pentru a parcurge fiecare subbanda (de P/4 ori = de 2 ori). Aceasta etapa a optimizarii este necesara pentru ca instructiunile din interiorul buclei FOR interioare sa fie executate in acelasi timp, la fiecare parcurgere a buclei.

```

maxSpectr.sl
DW17
[
    asla    r7                ;[52,1]
    asla    r8                ;[63,1]
]
DW18
[
    adda    r0,r5             ;[30,1]
    adda    r0,r6             ;[41,1]
]
DW19
[
    adda    r0,r7             ;[52,1]
    adda    r0,r8             ;[63,1]
]
    FALIGN
    LOOPSTART3
L21
DW20
[
    move.2f (r5),d6:d7        ;[30,1]
    move.w  (r1),d10          ;[33,1]
]
DW21
[
    mpy     d6,d6,d8          ;[30,1]
    adda    #<4,r5            ;[30,1]
]
DW22
    mac     d7,d7,d8          ;[31,1]
DW23
    asrw    d8,d6             ;[33,1]
DW24
    cmpgt   d10,d6            ;[33,1]
DW25
    nop                                           ;[0,0] TBIT stall
DW26
    IFT moves.f d8,(r1)        ;[35,1]
DW27
[
    move.2f (r6),d12:d13       ;[41,1]
    move.w  (r4),d6            ;[44,1]
]
DW28
[
    mpy     d12,d12,d14        ;[41,1]
    adda    #<4,r6             ;[41,1]
]
DW29
  
```

Dupa recompilare se observa ca pasii de optimizare nu au avut un efect semnificativ, motivul fiind necesitatea gruparii instructiunilor astfel incat sa fie executate simultan.

### 3.3 Rearanjarea codului functiei

```
maxSpectr.c
#include <prototype.h>
#include <stdio.h>

void maxSpectr(Word16 *X,
               Word16 *out,
               unsigned short N,
               unsigned short M,
               unsigned short P)
{
    #pragma align *X 8
    #pragma align *out 8

    int k0,k1,k2,k3;
    int i;
    int j;
    Word32 mi0,mi1,mi2,mi3;
    Word16 mmax0,mmax1,mmax2,mmax3;
    Word16 maxim0,maxim1,maxim2,maxim3;

    for(i=0;i<P;i+=4)
    {
        k0=2*i*M;
        k1=2*(i+1)*M;
        k2=2*(i+2)*M;
        k3=2*(i+3)*M;

        maxim0=0,maxim1=0,maxim2=0,maxim3=0;

        for(j=0;j<M;j++)
        {
            mi0=0,mi1=0,mi2=0,mi3=0;

            mi0=L_mac(mi0,X[k0],X[k0]);
            mi1=L_mac(mi1,X[k1],X[k1]);
            mi2=L_mac(mi2,X[k2],X[k2]);
            mi3=L_mac(mi3,X[k3],X[k3]);

            mi0=L_mac(mi0,X[k0+1],X[k0+1]);
            mi1=L_mac(mi1,X[k1+1],X[k1+1]);
            mi2=L_mac(mi2,X[k2+1],X[k2+1]);
            mi3=L_mac(mi3,X[k3+1],X[k3+1]);

            mmax0=extract_h(mi0);
            mmax1=extract_h(mi1);
            mmax2=extract_h(mi2);
            mmax3=extract_h(mi3);

            if(mmax0>maxim0)
            {
                maxim0=mmax0;
            }

            if(mmax1>maxim1)
            {
                maxim1=mmax1;
            }

            if(mmax2>maxim2)
            {
                maxim2=mmax2;
            }

            if(mmax3>maxim3)
            {
                maxim3=mmax3;
            }

            k0=k0+2;
            k1=k1+2;
            k2=k2+2;
            k3=k3+2;
        }

        out[i]=maxim0;
        out[i+1]=maxim1;
        out[i+2]=maxim2;
        out[i+3]=maxim3;
    }
}
```

```

]
DW23
    adda        r0,r5                      :[38,1]
    FALIGN
    LOOPSTART3
L21
DW24
[
    move.2f     (r3)+,d12:d13              :[36,1]
    move.2f     (r2)+,d6:d7                :[35,1]
]
DW25
[
    mpy         d12,d12,d11                :[36,1]
    mpy         d6,d6,d10                  :[35,1]
    moves.f     d13,(sp-8)                  :[42,1]
    move.2f     (r4)+,d12:d13              :[37,1]
]
DW26
[
    mpy         d12,d12,d6                  :[37,1]
    mac         d7,d7,d10                  :[41,1]
    moves.f     d13,(sp-6)                  :[43,1]
    move.2f     (r5)+,d12:d13              :[38,1]
]
DW27
[
    mpy         d12,d12,d15                :[38,1]
    move.f      (sp-6),d8                   :[43,1]
    move.f      (sp-8),d7                   :[42,1]
]
DW28
[
    mac         d13,d13,d15                :[44,1]
    mac         d8,d8,d6                   :[43,1]
    mac         d7,d7,d11                   :[42,1]
    move.f      (sp-16),d12                 :[65,1]
]
DW29
[
    asrw        d6,d7                      :[65,1]
    and         #-65536,d11,d11            :[60,1]
    move.l      d15,(sp-12)                 :[44,1]
    move.f      (sp-20),d15                 :[60,1]
]
DW30
[
    aslw        d7,d8                      :[65,1]
    max         d11,d15                     :[60,1]
    move.f      (sp-14),d13                 :[70,1]
    move.f      (sp-12),d9                  :[70,1]
]
DW31
[
    and         #-65536,d10,d10            :[55,1]
    max         d8,d12                      :[65,1]
    max         d9,d13                      :[70,1]
    moves.f     d15,(sp-20)                 :[60,1]
]
DW32
[
    max         d10,d14                     :[55,1]
    moves.f     d12,(sp-16)                 :[65,1]
    moves.f     d13,(sp-14)                 :[38,1]
]
    LOOPEND3
DW33

```

```

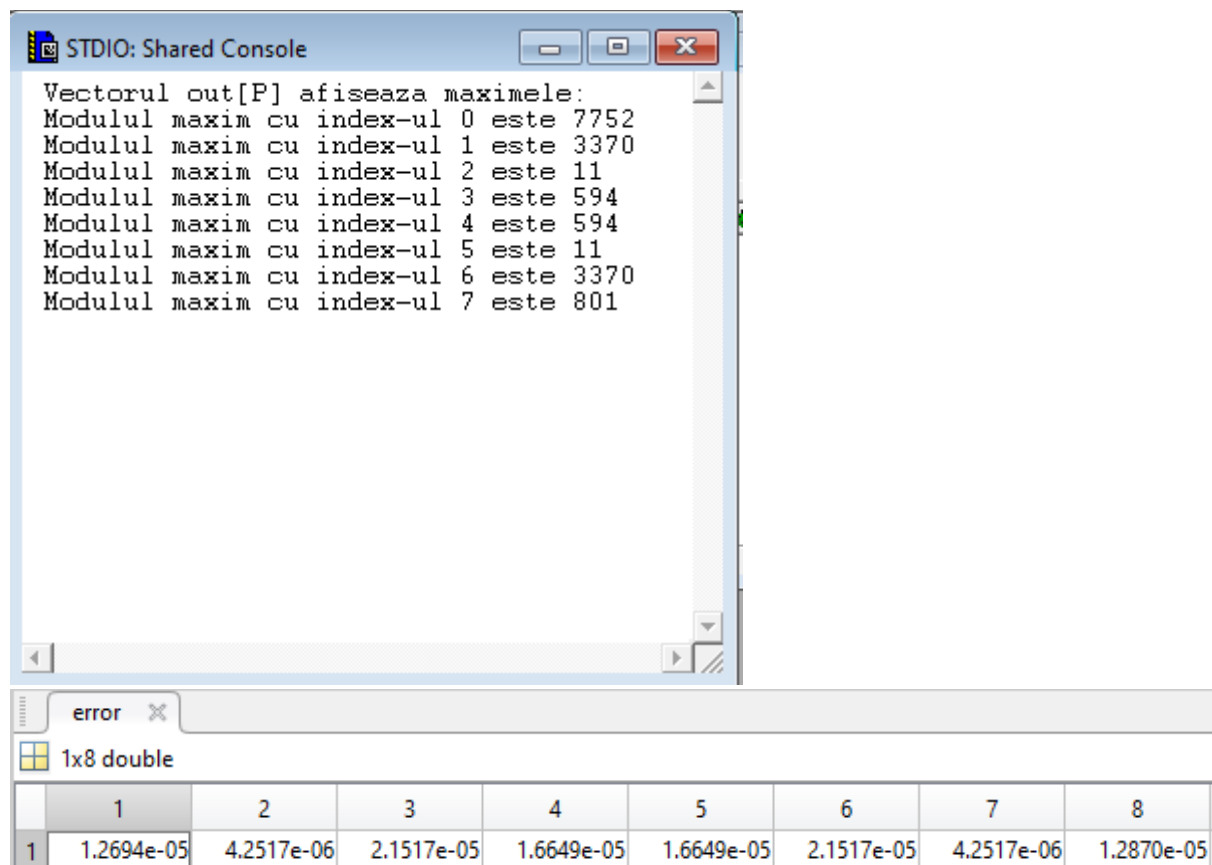
]
DW23
    adda        r0,r5                      :[38,1]
    FALIGN
    LOOPSTART3
L21
DW24
[
    move.2f     (r3)+,d12:d13              :[36,1]
    move.2f     (r2)+,d6:d7                :[35,1]
]
DW25
[
    mpy         d12,d12,d11                :[36,1]
    mpy         d6,d6,d10                  :[35,1]
    moves.f     d13,(sp-8)                  :[42,1]
    move.2f     (r4)+,d12:d13              :[37,1]
]
DW26
[
    mpy         d12,d12,d6                  :[37,1]
    mac         d7,d7,d10                  :[41,1]
    moves.f     d13,(sp-6)                  :[43,1]
    move.2f     (r5)+,d12:d13              :[38,1]
]
DW27
[
    mpy         d12,d12,d15                :[38,1]
    move.f      (sp-6),d8                   :[43,1]
    move.f      (sp-8),d7                   :[42,1]
]
DW28
[
    mac         d13,d13,d15                :[44,1]
    mac         d8,d8,d6                    :[43,1]
    mac         d7,d7,d11                   :[42,1]
    move.f      (sp-16),d12                 :[65,1]
]
DW29
[
    asrw        d6,d7                      :[65,1]
    and         #-65536,d11,d11            :[60,1]
    move.l      d15,(sp-12)                 :[44,1]
    move.f      (sp-20),d15                 :[60,1]
]
DW30
[
    aslw        d7,d8                      :[65,1]
    max         d11,d15                     :[60,1]
    move.f      (sp-14),d13                 :[70,1]
    move.f      (sp-12),d9                  :[70,1]
]
DW31
[
    and         #-65536,d10,d10            :[55,1]
    max         d8,d12                      :[65,1]
    max         d9,d13                      :[70,1]
    moves.f     d15,(sp-20)                 :[60,1]
]
DW32
[
    max         d10,d14                     :[55,1]
    moves.f     d12,(sp-16)                 :[65,1]
    moves.f     d13,(sp-14)                 :[38,1]
]
    LOOPEND3
DW33

```

În fisierul .sl, observăm o îmbunătățire a eficienței programului, deși nu am obținut o performanță ideală (4 operații aritmetico-logice realizate în același ciclu masina simultan cu 2 transferuri din/in memorie). În cadrul buclei interioare observăm că într-un singur ciclu de ceas se efectuează 2 sau 3 operații aritmetico-logice și 1 sau 2 de transfer a datelor.

În programul neoptimizat, inițial al funcției putem calcula numărul de cicluri de ceas efectuate în cadrul celor 2 bucle FOR și vom obține un număr de 5 cicluri efectuate în interiorul buclei interioare, și fiind parcursă de 16 de ori vom avea 80 în total. Buclea exterioară FOR conține 15 cicluri, deci vom avea în total 95 cicluri de ceas, executate de  $P(8 \text{ ori})$ , deci  $95 \cdot 8 = 760$  cicluri de ceas, un număr destul de mare.

Dacă analizăm varianta rearanjată a codului funcției maxSpectr, observăm că în cadrul buclei interioare FOR se efectuează 9 cicluri, iar în bucla exterioară se efectuează 22 deci un total de  $9 \cdot 16 + 22 = 166$  cicluri. Dar bucla exterioară a fost optimizată în cadrul primei tehnici prezentate, aceasta fiind parcursă doar de  $P/4 = 8/4 = 2$  ori. Asadar în total cele două bucle prezintă  $166 \cdot 2 = 332$  cicluri de ceas, un număr mai mic comparativ cu cel al codului neoptimizat. Asadar cu ajutorul optimizării am reușit să reducem numărul de cicluri masina.



Pentru verificare, am obținut aceleași valori ale maximelor și în funcția optimizată, la fel ca în funcția neoptimizată și o eroare foarte mică.