PRODOTTO TRA SOMME DI NUMERI COMPLESSI

Relazione relativa al progetto d'esame ARCHITETTURA DEGLI ELABORATORI

sessione estiva — 2021/2022

Corso di Laurea in Informatica Applicata
Università di Urbino

DOCENTE:

Alessandro Bogliolo

STUDENTI:

Nicolas Barzotti
matricola: 313687

Gabriele Ramagnano
matricola: 315439

Indice

1 Specifica

1.1 Scopo del progetto

Nel seguente progetto si esegue il prodotto fra due somme di numeri complessi, con il primo fattore composto da due addendi, mentre il secondo da tre. Dati i numeri complessi z_1, z_2, z_3, z_4, z_5 dove

$z_1 = a + ib$

$z_2 = c + id$

$z_3 = e + if$

$z_4 = g + ih$

$z_5 = j + ik$

si esegue il prodotto della somma dei numeri

$S = (z_1 + z_2) * (z_3 + z_4 + z_5)$

ossia

$S = (z_1 * z_3) + (z_1 * z_4) + (z_1 * z_5) + (z_2 * z_3) + (z_2 * z_4) + (z_2 * z_5)$

1.2 Specifica algoritmica

```
typedef struct num_com
{
    double real,
           im;
}   num_com_t;


int main (void)
{
        int i,
            j;

        num_com_t num_res = {0, 0};

        num_com_t num_com_1[2]={{1, 1},
                                {2, 2}},

                  num_com_2[3]={{3, 3},
                                {4, 4},
                                {5, 5}},

                  num_com_3[2*3]={{0, 0},
                                  {0, 0},
                                  {0, 0},
                                  {0, 0},
                                  {0, 0},
                                  {0, 0}};

        for (i = 0;
             i < 2; i++)
        {
               for (j = 0;
                    j < 3; j++)
               {
                      num_res.real += num_com_3[3 * i + j].real =
                      (num_com_1[i].real * num_com_2[j].real) -
                      (num_com_1[i].im   * num_com_2[j].im);

                      num_res.im   += num_com_3[3 * i + j].im   =
                      (num_com_1[i].real * num_com_2[j].im)   +
                      (num_com_1[i].im   * num_com_2[j].real);


               }



        }


        printf("The resulting number is: %.2lf\t+ i%.2lf\n",
               num_res.real,
               num_res.im);

        return(0);
}
```

1.3 Benchmark

I dati ai quali si applica il codice sono rappresentati da un insieme di numeri in virgola mobile. Il programma non presenta particolari limitazioni per quanto riguarda la struttura dati utilizzata (array). La specifica illustrata utilizza un numero limitato di operandi. Al crescere di tale numero aumenta il numero di operazioni effettuate ed il numero di cicli di clock impiegato.

2 Prima Implementazione in Assembly

2.1 Descrizione delle scelte implementative

1. **Modello di strutture dati scelte**

   Linguaggio C (*righe 1,27*)

   Per la rappresentazione dei numeri complessi si è fatto ricorso al modello di struttura dati di tipo *struct* (*struct num_com*) che permette di selezionare con più facilità la parte reale o immaginaria del numero. Di conseguenza sono stati utilizzati tre array per il calcolo del prodotto fra numeri complessi, di cui uno si riferisce alla somma dei primi due numeri (*num_com_1*) , un altro alla somma dei successivi tre (*num_com_2*) , mentre il restante array viene utilizzato nel calcolo come una struttura dati d'appoggio impiegata nella somma delle rispettive parti reali e immaginarie (*num_com_3*). L'esito dell'operazione viene memorizzato in un quarto array (*num_res*).

   Linguaggio Assembly (*righe 1,20*)

   A differenza del linguaggio C sono stati utilizzati sei array per il calcolo del prodotto fra complessi. Gli array che si riferiscono alla somma dei numeri sono stati divisi (quindi ne sono stati impiegati il doppio) in un array contenente la parte reale (*reali_e/reali_i*) ed un array rappresentante la parte immaginaria (*immaginari_e/immaginari_i*). Anche il quarto array d'appoggio è stato diviso secondo il medesimo criterio (*res_r* e *res_i*). L'array servito per la sommatoria delle parti reali e immaginarie del numero non è stato tradotto in Assembly dal momento che ai fini di tale operazione s' impiega il quarto array d'appoggio e sarebbe dunque risultato ridondante trascrivere un elemento essenziale per il funzionamento di tale calcolo in C, ma superfluo e non necessario nel linguaggio assemblativo. Difatti risulta più semplice e naturale aggiornare ad ogni loop il registro interessato piuttosto che popolare la memoria di dati che andrebbero infine a sommarsi nel registro stesso.

2. **Algoritmi utilizzati**

   Linguaggio C (*righe 29,48*)

   Per il calcolo del prodotto fra due somme di numeri complessi è stato implementato un algoritmo di tipo iterativo che si compone di due istruzioni *for* annidate. Il ciclo più esterno seleziona gli addendi del primo fattore, mentre il ciclo più interno si occupa degli addendi del secondo fattore.

   Linguaggio Assembly (*righe 22,77*)

   L'algoritmo impiegato in C è stato riproposto nel linguaggio assemblativo. Il ciclo più esterno viene denominato *loope* mentre quello più interno *loopi*. Al fine di migliorare la leggibilità del codice si è deciso di aggiungere agli identificatori delle parti reali e immaginarie di ciascun numero complesso una lettera finale i/e che indica a quale dei due cicli appartengono (interno o esterno) e, rispettivamente, a quale dei due fattori del prodotto fanno parte.

3. **Calcolo fra numeri complessi**

   Linguaggio C (*righe 35,41*)

   Il calcolo di un numero complesso z_1 per un secondo numero complesso z_2 si traduce nella somma algebrica di due prodotti per quanto riguarda sia la parte reale sia la parte immaginaria. Tale

operazione viene reiterata sei volte, ossia tre volte per il primo addendo del primo fattore e tre volte per il secondo addendo del fattore medesimo. In C questo è reso possibile tramite l'operatore di indirizzo presente nella struttura dati statica di un array.

Linguaggio Assembly (*righe 22,77*)

Il meccanismo di calcolo presente in C viene riproposto pari pari nel linguaggio assemblativo, fatta eccezione per il terzo array d'appoggio che è stato eliminato per i motivi precedentemente argomentati. Ora, analizzando il seguente frammento di codice

file.s : ComplessiV_1

```
1           .data
2   res_r:            .double 0
3   res_i:            .double 0
4   reali_e:          .double 1, 2
5   immaginari_e:     .double 1, 2
6   reali_i:          .double 3, 4, 5
7   immaginari_i:     .double 3, 4, 5
8   ne:       .word   2
9   ni:       .word   3
10
11          .text
12  loope:
13  loopi:
14          ; INIZIO CALCOLO PARTE REALE
15
16          L.D     f0, 0(r5)       ; read an element (ex.loop)
17          L.D     f1, 0(r7)       ; read an element (in.loop)
18          MUL.D   f2, f0, f1      ; multiply (r)
19          L.D     f0, 0(r6)       ; read an element (ex.loop)
20          L.D     f1, 0(r8)       ; read an element (in.loop)
21          MUL.D   f3, f0, f1      ; multiply (r)
22          SUB.D   f3, f2, f3      ; subtraction (r)
23          ADD.D   f4, f4, f3      ; real summation
24          S.D     f4, 0(r3)       ; write the element back in memory (r)
25
26          ; FINE CALCOLO PARTE REALE
```

possiamo notare come le operazioni fra i registri rispettino le medesime regole di precedenza degli operandi utilizzati nel linguaggio C. Si eseguono difatti prima le due moltiplicazioni, poi la differenza e infine il dato ottenuto viene sommato e memorizzato nel registro f_4 che nel ciclo successivo verrà nuovamente aggiornato.

2.2 Listato del codice Assembly

file.s : ComplessiV_1

```
1           .data
2   res_r:            .double 0
3   res_i:            .double 0
4   reali_e:          .double 1, 2
5   immaginari_e:     .double 1, 2
6   reali_i:          .double 3, 4, 5
7   immaginari_i:     .double 3, 4, 5
8   ne:       .word   2
9   ni:       .word   3
10
11          .text
12  start:
13          LW      r1, ne(r0)              ; number of elements (external loop)
14          LW      r2, ni(r0)              ; number of elements (internal loop)
15          DADDI   r3, r0, res_r           ; p. to the 1st support array element (r)
16          DADDI   r4, r0, res_i           ; p. to the 1st array element (im)
17          DADDI   r5, r0, reali_e         ; p. to the 1st array r. element (ex.loop)
18          DADDI   r6, r0, immaginari_e    ; p. to the 1st array i. element (ex.loop)
19          DADDI   r7, r0, reali_i         ; p. to the 1st array r. element (in.loop)
20          DADDI   r8, r0, immaginari_i    ; p. to the 1st array i. element (in.loop)
21
22  loope:
23  loopi:
24          ; INIZIO CALCOLO PARTE REALE
25
26          L.D     f0, 0(r5)               ; read an element (ex.loop)
27          L.D     f1, 0(r7)               ; read an element (in.loop)
28          MUL.D   f2, f0, f1              ; multiply (r)
29          L.D     f0, 0(r6)               ; read an element (ex.loop)
30          L.D     f1, 0(r8)               ; read an element (in.loop)
31          MUL.D   f3, f0, f1              ; multiply (r)
32          SUB.D   f3, f2, f3              ; subtraction (r)
33          ADD.D   f4, f4, f3              ; real summation
34          S.D     f4, 0(r3)               ; write the element back in memory (r)
35
36          ; FINE CALCOLO PARTE REALE
37
38          ; INIZIO CALCOLO PARTE IMMAGINARIA
39
40          L.D     f0, 0(r5)               ; read an element (ex.loop)
41          L.D     f1, 0(r8)               ; read an element (in.loop)
42          MUL.D   f2, f0, f1              ; multiply (im)
43          L.D     f0, 0(r6)               ; read an element (ex.loop)
44          L.D     f1, 0(r7)               ; read an element (in.loop)
45          MUL.D   f3, f0, f1              ; multiply (im)
46          ADD.D   f3, f3, f2              ; sum (im)
47          ADD.D   f5, f5, f3              ; imaginary summation
48          S.D     f5, 0(r4)               ; write the element back in memory (im)
49
50          ; FINE CALCOLO PARTE IMMAGINARIA
51
52          ; INIZIO AGGIORNAMENTO DEL LOOP INTERNO
53
54          DADDI   r7, r7, 8               ; move to next element
55          DADDI   r8, r8, 8               ; move to next element
56          DADDI   r2, r2, -1              ; decrement the counter (in.loop)
57          BNEZ    r2, loopi               ; jump if not equal zero
```

```
58
59          ; FINE AGGIORNAMENTO DEL LOOP INTERNO
60
61          ; INIZIO AGGIORNAMENTO DEL LOOP ESTERNO
62
63          DADDI   r7, r7, -24             ; move to next element
64          DADDI   r8, r8, -24             ; move to next element
65          DADDI   r2, r2, 3               ; increment the counter (in.loop)
66          DADDI   r5, r5, 8               ; move to next element
67          DADDI   r6, r6, 8               ; move to next element
68          DADDI   r1, r1, -1              ; decrement the counter (ex.loop)
69          BNEZ    r1, loope               ; jump if not equal zero
70
71          ; FINE AGGIORNAMENTO DEL LOOP ESTERNO
72
73  end:
74          HALT
```

```
Execution
340 Cycles
155 Instructions
2.194 Cycles Per Instruction (CPI)

Stalls
212 RAW Stalls
0 WAW Stalls
12 WAR Stalls
12 Structural Stalls
5 Branch Taken Stalls
0 Branch Misprediction Stalls

Code size
152 Bytes
```

Nel codice presente qui di seguito sono stati annotati i tipi di stalli presenti durante la sua esecuzione. Oltre agli ABORT e a stalli di tipo strutturale vi sono:

1. **Stalli di tipo Read After Write**

   Sono i tipi di stalli più frequenti nel codice. Un esempio di stallo Read After Write si può notare fra nell'istruzione MUL.D f2, f0, f1, dove l'esecuzione di tale operazione internamente pipeline si arresta per un ciclo di clock dato che la sua precedente (L.D f1, 0(r7)) può fornire i dati ai registri solo dopo la fase di memory access, poiché si tratta di un'istruzione di caricamento dalla memoria. A parte questo, i RAW più "consistenti" si verificano nelle istruzioni di SUB.D e di ADD.D dal momento che si crea una forte dipendenza fra i registri utilizzati.

2. **Stalli di tipo Write After Read**

   Tale stallo è presente solamente due volte all'interno di un ciclo (12 WAR in totale). L'istruzione di caricamento dalla memoria dovrebbe far uso di un registro che risulta "impiegato" dall'istruzione MUL.D. In realtà lo stallo è dovuto ad un effetto di propagazione causato dalla precedenteistruzione di caricamento dalla memoria che ancora non ha introdotto i dati nel registro.

file.s : ComplessiV_1

```
1           .data
2   res_r:            .double 0
3   res_i:            .double 0
4   reali_e:          .double 1, 2
5   immaginari_e:     .double 1, 2
6   reali_i:          .double 3, 4, 5
7   immaginari_i:     .double 3, 4, 5
8   ne:       .word   2
9   ni:       .word   3
10
11          .text
12  start:
13          LW      r1, ne(r0)              ; number of elements (external loop)
14          LW      r2, ni(r0)              ; number of elements (internal loop)
15          DADDI   r3, r0, res_r           ; p. to the 1st support array element (r)
16          DADDI   r4, r0, res_i           ; p. to the 1st array element (im)
17          DADDI   r5, r0, reali_e         ; p. to the 1st array r. element (ex.loop)
18          DADDI   r6, r0, immaginari_e    ; p. to the 1st array i. element (ex.loop)
19          DADDI   r7, r0, reali_i         ; p. to the 1st array r. element (in.loop)
20          DADDI   r8, r0, immaginari_i    ; p. to the 1st array i. element (in.loop)
21
22  loope:
23  loopi:
24          ; INIZIO CALCOLO PARTE REALE
25
26          L.D     f0, 0(r5)               ; read an element (ex.loop)
27          L.D     f1, 0(r7)               ; read an element (in.loop)
28          MUL.D   f2, f0, f1              ; multiply (r)
29          ***RAW
30          L.D     f0, 0(r6)               ; read an element (ex.loop)
31          ***WAR
32          L.D     f1, 0(r8)               ; read an element (in.loop)
33          MUL.D   f3, f0, f1              ; multiply (r)
34          ***RAW
35          SUB.D   f3, f2, f3              ; subtraction (r)
36          ***RAW
37          ***RAW
38          ***RAW
39          ***RAW
40          ***RAW
41          ***RAW
42          ADD.D   f4, f4, f3              ; real summation
43          ***RAW
44          ***RAW
45          ***RAW
46          S.D     f4, 0(r3)               ; write the element back in memory (r)
47          ***RAW
48          ***RAW
49          ***STR
```

```
50
51          ; FINE CALCOLO PARTE REALE
52
53          ; INIZIO CALCOLO PARTE IMMAGINARIA
54
55          L.D     f0, 0(r5)           ; read an element (ex.loop)
56          L.D     f1, 0(r8)           ; read an element (in.loop)
57          MUL.D   f2, f0, f1          ; multiply (im)
58          ***RAW
59          L.D     f0, 0(r6)           ; read an element (ex.loop)
60          ***WAR
61          L.D     f1, 0(r7)           ; read an element (in.loop)
62          MUL.D   f3, f0, f1          ; multiply (im)
63          ***RAW
64          ADD.D   f3, f3, f2          ; sum (im)
65          ***RAW
66          ***RAW
67          ***RAW
68          ***RAW
69          ***RAW
70          ***RAW
71          ADD.D   f5, f5, f3          ; imaginary summation
72          ***RAW
73          ***RAW
74          ***RAW
75          S.D     f5, 0(r4)           ; write the element back in memory (im)
76          ***RAW
77          ***RAW
78          ***STR
79
80          ; FINE CALCOLO PARTE IMMAGINARIA
81
82          ; INIZIO AGGIORNAMENTO DEL LOOP INTERNO
83
84          DADDI   r7, r7, 8               ; move to next element
85          DADDI   r8, r8, 8               ; move to next element
86          DADDI   r2, r2, -1              ; decrement the counter (in.loop)
87          BNEZ    r2, loopi               ; jump if not equal zero
88          ***RAW
89
90
91          ; FINE AGGIORNAMENTO DEL LOOP INTERNO
92
93          ; INIZIO AGGIORNAMENTO DEL LOOP ESTERNO
94
95          DADDI   r7, r7, -24             ; move to next element
96          ***ABORT
97          DADDI   r8, r8, -24             ; move to next element
98          DADDI   r2, r2, 3               ; increment the counter (in.loop)
99          DADDI   r5, r5, 8               ; move to next element
100         DADDI   r6, r6, 8               ; move to next element
101         DADDI   r1, r1, -1              ; decrement the counter (ex.loop)
102         BNEZ    r1, loope               ; jump if not equal zero
103         ***RAW
104
105
106         ; FINE AGGIORNAMENTO DEL LOOP ESTERNO
107
108 end:
109         HALT
110         ***ABORT
```

3 Ottimizzazione Statica

3.1 Versione 2

file.s : ComplessiV_2

```
1         .data
2  res_r:          .double 0
3  res_i:          .double 0
4  reali_e:        .double 1, 2
5  immaginari_e:   .double 1, 2
6  reali_i:        .double 3, 4, 5
7  immaginari_i:   .double 3, 4, 5
8  ne:     .word   2
9  ni:     .word   3
10
11        .text
12 start:
13        LW      r1, ne(r0)              ; number of elements (external loop)
14        LW      r2, ni(r0)              ; number of elements (internal loop)
15        DADDI   r3, r0, res_r           ; p. to the 1st support array element (r)
16        DADDI   r4, r0, res_i           ; p. to the 1st array element (im)
17        DADDI   r5, r0, reali_e         ; p. to the 1st array r. element (ex.loop)
18        DADDI   r6, r0, immaginari_e    ; p. to the 1st array i. element (ex.loop)
19        DADDI   r7, r0, reali_i         ; p. to the 1st array r. element (in.loop)
20        DADDI   r8, r0, immaginari_i    ; p. to the 1st array i. element (in.loop)
21
22
23 loop:
24        ; INIZIO CALCOLI REALE E IMMAGINARIO
25        L.D     f0, 0(r5)               ; read an element (ex.loop)
26        L.D     f1, 0(r7)               ; read an element (in.loop)
27        L.D     f2, 0(r6)               ; read an element (ex.loop)
28        L.D     f3, 0(r8)               ; read an element (in.loop)
29        MUL.D   f4, f0, f1              ; multiply (r)
30        MUL.D   f5, f2, f3              ; multiply (r)
31        MUL.D   f8, f0, f3              ; multiply (im)
32        MUL.D   f9, f2, f1              ; multiply (im)
33
34        SUB.D   f6, f4, f5              ; subtraction (r)
35        ADD.D   f7, f7, f6              ; real summation
36        ADD.D   f10, f9, f8             ; sum (im)
37        ADD.D   f11, f11, f10           ; imaginary summation
38
39        ; FINE CALCOLI REALE E IMMAGINARIO
40
41        ; INIZIO AGGIORNAMENTO DEL LOOP INTERNO
42
43        DADDI   r7, r7, 8               ; move to next element
44        DADDI   r8, r8, 8               ; move to next element
45        DADDI   r2, r2, -1              ; decrement the counter (in.loop)
46        BNEZ    r2, loop                ; jump if not equal zero
47
48        ; FINE AGGIORNAMENTO DEL LOOP INTERNO
49
50        ; INIZIO AGGIORNAMENTO DEL LOOP ESTERNO
51
52        DADDI   r7, r7, -24             ; move to starting element
53        DADDI   r8, r8, -24             ; move to starting element
54        DADDI   r2, r2, 3               ; reset the counter (in.loop)
```

```
55          DADDI   r5, r5, 8               ; move to next element
56          DADDI   r6, r6, 8               ; move to next element
57          DADDI   r1, r1, -1              ; decrement the counter (ex.loop)
58          BNEZ    r1, loop                ; jump if not equal zero
59
60          ; FINE AGGIORNAMENTO DEL LOOP ESTERNO
61
62          S.D     f7, 0(r3)               ; write the element back in memory (r)
63          S.D     f11, 0(r4)              ; write the element back in memory (im)
64
65  end:
66          HALT
```

```
Execution
192 Cycles
121 Instructions
1.587 Cycles Per Instruction (CPI)


Stalls
68 RAW Stalls
0 WAW Stalls
0 WAR Stalls
12 Structural Stalls
5 Branch Taken Stalls
0 Branch Misprediction Stalls


Code size
136 Bytes
```

1 **Tecnica applicata**: instruction reordering e register renaming.

2 **Principali accorgimenti adottati**:

(a) Sono state eliminate 4 istruzioni di caricamento dalla memoria (L.D) (righe 24,28).

(b) Il risultato di ogni operazione viene memorizzato su un registro differente. Nella versione precedente invece vi era un riuso più massiccio dei registri (righe 29,37).

(c) Le istruzioni di *store* nella memoria (S.D) sono state portate fuori dal ciclo (righe 62,63).

(d) Sono stati rimossi gli identificatori "loope" e "loopi" e sono stati sostituiti entrambi dall'identificatore "loop" che assolve alla medesima funzione. Questa implementazione poteva essere effettuata anche in precedenza ma, per facilitare una maggiore leggibilità del codice rispetto alla sua specifica in C, si è scelto di non riportarla (riga 23).

3 **Conflitti residui che danno ancora luogo a stalli**:

(a) Stalli di RAW fra le istruzioni di MUL.D e ADD.D.

(b) Stalli di RAW fra le istruzioni di ADD.D e ADD.D.

(c) Stalli di RAW fra le istruzioni di DADDI e BNEZ, ossia fra l'istruzione di decremento del contatore e quella di valutazione della condizione di salto.

(d) ABORT dell'istruzione causata dall'istruzione di salto.

(e) Stalli strutturali.

3.2 Versione 3

file.s : ComplessiV_3

```
1         .data
2  res_r:          .double 0
3  res_i:          .double 0
4  reali_e:        .double 1, 2
5  immaginari_e:   .double 1, 2
6  reali_i:        .double 3, 4, 5
7  immaginari_i:   .double 3, 4, 5
8  ne:     .word   2
9
10        .text
11 start:
12        LW      r1, ne(r0)              ; number of elements (external loop)
13        DADDI   r3, r0, res_r           ; p. to the 1st support array element (r)
14        DADDI   r4, r0, res_i           ; p. to the 1st array element (im)
15        DADDI   r5, r0, reali_e         ; p. to the 1st array r. element (ex.loop)
16        DADDI   r6, r0, immaginari_e    ; p. to the 1st array i. element (ex.loop)
17        DADDI   r7, r0, reali_i         ; p. to the 1st array r. element (in.loop)
18        DADDI   r8, r0, immaginari_i    ; p. to the 1st array i. element (in.loop)
19
20
21 loop:
22
23        ; INIZIO 1   ITERAZIONE INTERNA
24        ; INIZIO CALCOLI REALE E IMMAGINARIO
25        L.D     f0, 0(r5)               ; read an element (ex.loop)
26        L.D     f1, 0(r7)               ; read an element (in.loop)
27        L.D     f2, 0(r6)               ; read an element (ex.loop)
28        L.D     f3, 0(r8)               ; read an element (in.loop)
29        MUL.D   f4, f0, f1              ; multiply (r)
30        MUL.D   f5, f2, f3              ; multiply (r)
31        MUL.D   f8, f0, f3              ; multiply (im)
32        MUL.D   f9, f2, f1              ; multiply (im)
33
34        SUB.D   f6, f4, f5              ; subtraction (r)
35        ADD.D   f7, f7, f6              ; real summation
36        ADD.D   f10, f9, f8             ; sum (im)
37        ADD.D   f11, f11, f10           ; imaginary summation
38
39        ; FINE CALCOLI REALE E IMMAGINARIO
40
41        ; INIZIO AGGIORNAMENTO DEL LOOP INTERNO
42
43        DADDI   r7, r7, 8               ; move to next element
44        DADDI   r8, r8, 8               ; move to next element
45
46        ; FINE AGGIORNAMENTO DEL LOOP INTERNO
47        ; FINE 1   ITERAZIONE INTERNA
48
49
50        ; INIZIO 2   ITERAZIONE INTERNA
51        ; INIZIO CALCOLI REALE E IMMAGINARIO
52        L.D     f0, 0(r5)               ; read an element (ex.loop)
53        L.D     f1, 0(r7)               ; read an element (in.loop)
54        L.D     f2, 0(r6)               ; read an element (ex.loop)
55        L.D     f3, 0(r8)               ; read an element (in.loop)
56        MUL.D   f4, f0, f1              ; multiply (r)
```

```
57          MUL.D   f5, f2, f3              ; multiply (r)
58          MUL.D   f8, f0, f3              ; multiply (im)
59          MUL.D   f9, f2, f1              ; multiply (im)
60
61          SUB.D   f6, f4, f5              ; subtraction (r)
62          ADD.D   f7, f7, f6              ; real summation
63          ADD.D   f10, f9, f8             ; sum (im)
64          ADD.D   f11, f11, f10           ; imaginary summation
65
66          ; FINE CALCOLI REALE E IMMAGINARIO
67
68          ; INIZIO AGGIORNAMENTO DEL LOOP INTERNO
69
70          DADDI   r7, r7, 8               ; move to next element
71          DADDI   r8, r8, 8               ; move to next element
72
73          ; FINE AGGIORNAMENTO DEL LOOP INTERNO
74          ; FINE 2   ITERAZIONE INTERNA
75
76
77          ; INIZIO 3   ITERAZIONE INTERNA
78          ; INIZIO CALCOLI REALE E IMMAGINARIO
79          L.D     f0, 0(r5)               ; read an element (ex.loop)
80          L.D     f1, 0(r7)               ; read an element (in.loop)
81          L.D     f2, 0(r6)               ; read an element (ex.loop)
82          L.D     f3, 0(r8)               ; read an element (in.loop)
83          MUL.D   f4, f0, f1              ; multiply (r)
84          MUL.D   f5, f2, f3              ; multiply (r)
85          MUL.D   f8, f0, f3              ; multiply (im)
86          MUL.D   f9, f2, f1              ; multiply (im)
87
88          SUB.D   f6, f4, f5              ; subtraction (r)
89          ADD.D   f7, f7, f6              ; real summation
90          ADD.D   f10, f9, f8             ; sum (im)
91          ADD.D   f11, f11, f10           ; imaginary summation
92
93          ; FINE CALCOLI REALE E IMMAGINARIO
94
95          ; INIZIO AGGIORNAMENTO DEL LOOP INTERNO
96
97          DADDI   r7, r7, 8               ; move to next element
98          DADDI   r8, r8, 8               ; move to next element
99
100         ; FINE AGGIORNAMENTO DEL LOOP INTERNO
101         ; FINE 3   ITERAZIONE INTERNA
102
103         ; INIZIO AGGIORNAMENTO DEL LOOP ESTERNO
104
105         DADDI   r7, r7, -24             ; move to starting element
106         DADDI   r8, r8, -24             ; move to starting element
107         DADDI   r2, r2, 3               ; reset the counter (in.loop)
108         DADDI   r5, r5, 8               ; move to next element
109         DADDI   r6, r6, 8               ; move to next element
110         DADDI   r1, r1, -1              ; decrement the counter (ex.loop)
111         BNEZ    r1, loop                ; jump if not equal zero
112
113         ; FINE AGGIORNAMENTO DEL LOOP ESTERNO
114
115         S.D     f7, 0(r3)               ; write the element back in memory (r)
116         S.D     f11, 0(r4)              ; write the element back in memory (im)
117
```

```
118 end:
119         HALT
```

```
Execution

175 Cycles
108 Instructions
1.620 Cycles Per Instruction (CPI)


Stalls

62 RAW Stalls
0 WAW Stalls
0 WAR Stalls
18 Structural Stalls
1 Branch Taken Stall
0 Branch Misprediction Stalls


Code size

236 Bytes
```

1 **Tecnica applicata**: applicato loop unrolling al loop interno. *Nota bene*: Da qui il numero di complessi che si possono inserire nel secondo vettore viene fissato a 3.

2 **Principali accorgimenti adottati**:

   (a) Rimossi BNEZ per il loop interno.

   (b) Rimossi tutti i dati corrispondenti al loop interno: .word 3, BNEZ, DADDI per ridurre il contatore interno

3 **Conflitti residui che danno ancora luogo a stalli**:

   (a) Stalli di RAW fra le istruzioni di MUL.D e ADD.D.

   (b) Stalli di RAW fra le istruzioni di ADD.D e ADD.D.

   (c) Stalli di RAW fra le istruzioni di DADDI e BNEZ, ossia fra l'istruzione di decremento del contatore e quella di valutazione della condizione di salto (solo nel loop esterno).

   (d) ABORT dell'istruzione causata dall'istruzione di salto solo nel loop esterno.

   (e) Stalli strutturali.

3.3 Versione 4

file.s : ComplessiV_4

```
1         .data
2  res_r:          .double 0
3  res_i:          .double 0
4  reali_e:        .double 1, 2
5  immaginari_e:   .double 1, 2
6  reali_i:        .double 3, 4, 5
7  immaginari_i:   .double 3, 4, 5
8
9         .text
10 start:
11
12        DADDI   r3, r0, res_r           ; p. to the 1st support array element (r)
13        DADDI   r4, r0, res_i           ; p. to the 1st array element (im)
14        DADDI   r5, r0, reali_e         ; p. to the 1st array r. element (ex.loop)
15        DADDI   r6, r0, immaginari_e    ; p. to the 1st array i. element (ex.loop)
16        DADDI   r7, r0, reali_i         ; p. to the 1st array r. element (in.loop)
17        DADDI   r8, r0, immaginari_i    ; p. to the 1st array i. element (in.loop)
18
19        ; INIZIO 1   ITERAZIONE ESTERNA
20        ; INIZIO 1   ITERAZIONE INTERNA
21        ; INIZIO CALCOLI REALE E IMMAGINARIO
22
23        L.D     f0, 0(r5)               ; read an element (ex.loop)
24        L.D     f1, 0(r7)               ; read an element (in.loop)
25        L.D     f2, 0(r6)               ; read an element (ex.loop)
26        L.D     f3, 0(r8)               ; read an element (in.loop)
27        MUL.D   f4, f0, f1              ; multiply (r)
28        MUL.D   f5, f2, f3              ; multiply (r)
29        MUL.D   f8, f0, f3              ; multiply (im)
30        MUL.D   f9, f2, f1              ; multiply (im)
31
32        SUB.D   f6, f4, f5              ; subtraction (r)
33        ADD.D   f7, f7, f6              ; real summation
34        ADD.D   f10, f9, f8             ; sum (im)
35        ADD.D   f11, f11, f10           ; imaginary summation
36
37        ; FINE CALCOLI REALE E IMMAGINARIO
38
39        ; INIZIO AGGIORNAMENTO DEL LOOP INTERNO
40
41        DADDI   r7, r7, 8               ; move to next element
42        DADDI   r8, r8, 8               ; move to next element
43
44        ; FINE AGGIORNAMENTO DEL LOOP INTERNO
45        ; FINE 1   ITERAZIONE INTERNA
46
47
48        ; INIZIO 2   ITERAZIONE INTERNA
49        ; INIZIO CALCOLI REALE E IMMAGINARIO
50        L.D     f0, 0(r5)               ; read an element (ex.loop)
51        L.D     f1, 0(r7)               ; read an element (in.loop)
52        L.D     f2, 0(r6)               ; read an element (ex.loop)
53        L.D     f3, 0(r8)               ; read an element (in.loop)
54        MUL.D   f4, f0, f1              ; multiply (r)
55        MUL.D   f5, f2, f3              ; multiply (r)
56        MUL.D   f8, f0, f3              ; multiply (im)
```

```
57          MUL.D   f9, f2, f1              ; multiply (im)
58
59          SUB.D   f6, f4, f5              ; subtraction (r)
60          ADD.D   f7, f7, f6              ; real summation
61          ADD.D   f10, f9, f8             ; sum (im)
62          ADD.D   f11, f11, f10           ; imaginary summation
63
64          ; FINE CALCOLI REALE E IMMAGINARIO
65
66          ; INIZIO AGGIORNAMENTO DEL LOOP INTERNO
67
68          DADDI   r7, r7, 8               ; move to next element
69          DADDI   r8, r8, 8               ; move to next element
70
71          ; FINE AGGIORNAMENTO DEL LOOP INTERNO
72          ; FINE 2   ITERAZIONE INTERNA
73
74
75          ; INIZIO 3   ITERAZIONE INTERNA
76          ; INIZIO CALCOLI REALE E IMMAGINARIO
77          L.D     f0, 0(r5)               ; read an element (ex.loop)
78          L.D     f1, 0(r7)               ; read an element (in.loop)
79          L.D     f2, 0(r6)               ; read an element (ex.loop)
80          L.D     f3, 0(r8)               ; read an element (in.loop)
81          MUL.D   f4, f0, f1              ; multiply (r)
82          MUL.D   f5, f2, f3              ; multiply (r)
83          MUL.D   f8, f0, f3              ; multiply (im)
84          MUL.D   f9, f2, f1              ; multiply (im)
85
86          SUB.D   f6, f4, f5              ; subtraction (r)
87          ADD.D   f7, f7, f6              ; real summation
88          ADD.D   f10, f9, f8             ; sum (im)
89          ADD.D   f11, f11, f10           ; imaginary summation
90
91          ; FINE CALCOLI REALE E IMMAGINARIO
92
93          ; FINE 3   ITERAZIONE INTERNA
94
95          ; INIZIO AGGIORNAMENTO DEL LOOP ESTERNO
96
97          DADDI   r7, r7, -16             ; move to starting element
98          DADDI   r8, r8, -16             ; move to starting element
99          DADDI   r5, r5, 8               ; move to next element
100         DADDI   r6, r6, 8               ; move to next element
101
102         ; FINE AGGIORNAMENTO DEL LOOP ESTERNO
103         ; FINE 1   ITERAZIONE ESTERNA
104
105         ; INIZIO 2   ITERAZIONE ESTERNA
106         ; INIZIO 1   ITERAZIONE INTERNA
107         ; INIZIO CALCOLI REALE E IMMAGINARIO
108
109         L.D     f0, 0(r5)               ; read an element (ex.loop)
110         L.D     f1, 0(r7)               ; read an element (in.loop)
111         L.D     f2, 0(r6)               ; read an element (ex.loop)
112         L.D     f3, 0(r8)               ; read an element (in.loop)
113         MUL.D   f4, f0, f1              ; multiply (r)
114         MUL.D   f5, f2, f3              ; multiply (r)
115         MUL.D   f8, f0, f3              ; multiply (im)
116         MUL.D   f9, f2, f1              ; multiply (im)
117
```

```
118         SUB.D   f6, f4, f5              ; subtraction (r)
119         ADD.D   f7, f7, f6              ; real summation
120         ADD.D   f10, f9, f8             ; sum (im)
121         ADD.D   f11, f11, f10           ; imaginary summation
122
123         ; FINE CALCOLI REALE E IMMAGINARIO
124
125         ; INIZIO AGGIORNAMENTO DEL LOOP INTERNO
126
127         DADDI   r7, r7, 8               ; move to next element
128         DADDI   r8, r8, 8               ; move to next element
129
130         ; FINE AGGIORNAMENTO DEL LOOP INTERNO
131         ; FINE 1   ITERAZIONE INTERNA
132
133
134         ; INIZIO 2   ITERAZIONE INTERNA
135         ; INIZIO CALCOLI REALE E IMMAGINARIO
136         L.D     f0, 0(r5)               ; read an element (ex.loop)
137         L.D     f1, 0(r7)               ; read an element (in.loop)
138         L.D     f2, 0(r6)               ; read an element (ex.loop)
139         L.D     f3, 0(r8)               ; read an element (in.loop)
140         MUL.D   f4, f0, f1              ; multiply (r)
141         MUL.D   f5, f2, f3              ; multiply (r)
142         MUL.D   f8, f0, f3              ; multiply (im)
143         MUL.D   f9, f2, f1              ; multiply (im)
144
145         SUB.D   f6, f4, f5              ; subtraction (r)
146         ADD.D   f7, f7, f6              ; real summation
147         ADD.D   f10, f9, f8             ; sum (im)
148         ADD.D   f11, f11, f10           ; imaginary summation
149
150         ; FINE CALCOLI REALE E IMMAGINARIO
151
152         ; INIZIO AGGIORNAMENTO DEL LOOP INTERNO
153
154         DADDI   r7, r7, 8               ; move to next element
155         DADDI   r8, r8, 8               ; move to next element
156
157         ; FINE AGGIORNAMENTO DEL LOOP INTERNO
158         ; FINE 2   ITERAZIONE INTERNA
159
160
161         ; INIZIO 3   ITERAZIONE INTERNA
162         ; INIZIO CALCOLI REALE E IMMAGINARIO
163         L.D     f0, 0(r5)               ; read an element (ex.loop)
164         L.D     f1, 0(r7)               ; read an element (in.loop)
165         L.D     f2, 0(r6)               ; read an element (ex.loop)
166         L.D     f3, 0(r8)               ; read an element (in.loop)
167         MUL.D   f4, f0, f1              ; multiply (r)
168         MUL.D   f5, f2, f3              ; multiply (r)
169         MUL.D   f8, f0, f3              ; multiply (im)
170         MUL.D   f9, f2, f1              ; multiply (im)
171
172         SUB.D   f6, f4, f5              ; subtraction (r)
173         ADD.D   f7, f7, f6              ; real summation
174         ADD.D   f10, f9, f8             ; sum (im)
175         ADD.D   f11, f11, f10           ; imaginary summation
176
177         ; FINE CALCOLI REALE E IMMAGINARIO
178
```

```
179         ; FINE 3   ITERAZIONE INTERNA
180         ; FINE 2   ITERAZIONE ESTERNA
181
182         S.D     f7, 0(r3)              ; write the element back in memory (r)
183         S.D     f11, 0(r4)             ; write the element back in memory (im)
184
185 end:
186         HALT
```

```
Execution

159 Cycles
93 Instructions
1.710 Cycles Per Instruction (CPI)


Stalls

62 RAW Stalls
0 WAW Stalls
0 WAR Stalls
18 Structural Stalls
0 Branch Taken Stalls
0 Branch Misprediction Stalls


Code size

372 Bytes
```

1 **Tecnica applicata**: applicato loop unrolling al loop esterno. *Nota bene*: ora i calcoli sono fissati alla specifica, prima si potevano modificare.

2 **Principali accorgimenti adottati**:

   (a) Rimosse tutte le istruzioni relative al loop: .word ne, BNEZ, DADDI .

3 **Conflitti residui che danno ancora luogo a stalli**:

   (a) Stalli di RAW fra le istruzioni di MUL.D e ADD.D.

   (b) Stalli di RAW fra le istruzioni di ADD.D e ADD.D.

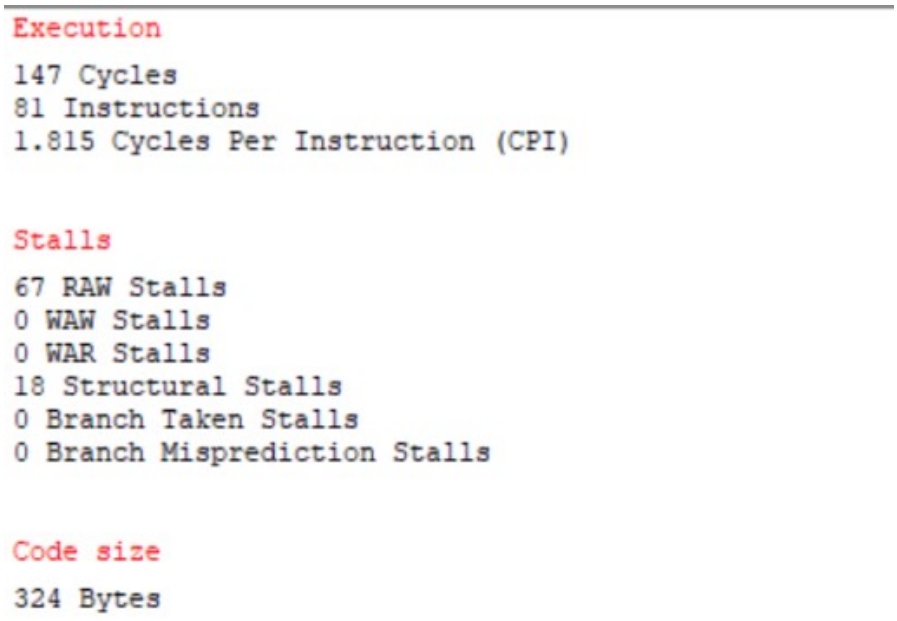
3.4 Versione 5

file.s : ComplessiV_5

```
1         .data
2  res_r:                  .double 0
3  res_i:                  .double 0
4  reali_e:                .double 1, 2
5  immaginari_e:   .double 1, 2
6  reali_i:        .double 3, 4, 5
7  immaginari_i:   .double 3, 4, 5
8
9         .text
10 start:
11        DADDI   r3, r0, res_r           ; p. to the 1st support array element (r)
12        DADDI   r4, r0, res_i           ; p. to the 1st array element (im)
13        DADDI   r5, r0, reali_e         ; p. to the 1st array r. element (ex.loop)
14        DADDI   r6, r0, immaginari_e    ; p. to the 1st array i. element (ex.loop)
15        DADDI   r7, r0, reali_i         ; p. to the 1st array r. element (in.loop)
16        DADDI   r8, r0, immaginari_i    ; p. to the 1st array i. element (in.loop)
17
18        ; INIZIO 1   ITERAZIONE ESTERNA
19        ; INIZIO 1   ITERAZIONE INTERNA
20        ; INIZIO CALCOLI REALE E IMMAGINARIO
21        L.D     f0, 0(r5)               ; read an element (ex.loop)
22        L.D     f1, 0(r7)               ; read an element (in.loop)
23        L.D     f2, 0(r6)               ; read an element (ex.loop)
24        L.D     f3, 0(r8)               ; read an element (in.loop)
25        MUL.D   f4, f0, f1              ; multiply (r)
26        MUL.D   f5, f2, f3              ; multiply (r)
27        MUL.D   f7, f0, f3              ; multiply (im)
28        MUL.D   f8, f2, f1              ; multiply (im)
29
30        SUB.D   f6, f4, f5              ; subtraction (r)
31        ADD.D   f30, f30, f6            ; real summation
32        ADD.D   f9, f8, f7              ; sum (im)
33        ADD.D   f31, f31, f9            ; imaginary summation
34
35        ; FINE CALCOLI REALE E IMMAGINARIO
36
37        ; FINE 1   ITERAZIONE INTERNA
38        ; INIZIO 2   ITERAZIONE INTERNA
39
40        ; INIZIO CALCOLI REALE E IMMAGINARIO
41        L.D     f10, 0(r5)              ; read an element (ex.loop)
42        L.D     f11, 8(r7)              ; read an element (in.loop)
43        L.D     f12, 0(r6)              ; read an element (ex.loop)
44        L.D     f13, 8(r8)              ; read an element (in.loop)
45        MUL.D   f14, f10, f11           ; multiply (r)
46        MUL.D   f15, f12, f13           ; multiply (r)
47        MUL.D   f17, f10, f13           ; multiply (im)
48        MUL.D   f18, f12, f11           ; multiply (im)
49
50        SUB.D   f16, f14, f15           ; subtraction (r)
51        ADD.D   f30, f30, f16           ; real summation
52        ADD.D   f19, f18, f17           ; sum (im)
53        ADD.D   f31, f31, f19           ; imaginary summation
54
55        ; FINE CALCOLI REALE E IMMAGINARIO
56
```

```
57          ; FINE 2   ITERAZIONE INTERNA
58          ; INIZIO 3   ITERAZIONE INTERNA
59
60          ; INIZIO CALCOLI REALE E IMMAGINARIO
61          L.D     f20, 0(r5)              ; read an element (ex.loop)
62          L.D     f21, 16(r7)             ; read an element (in.loop)
63          L.D     f22, 0(r6)              ; read an element (ex.loop)
64          L.D     f23, 16(r8)             ; read an element (in.loop)
65          MUL.D   f24, f20, f21           ; multiply (r)
66          MUL.D   f25, f22, f23           ; multiply (r)
67          MUL.D   f27, f20, f23           ; multiply (im)
68          MUL.D   f28, f22, f21           ; multiply (im)
69
70          SUB.D   f26, f24, f25           ; subtraction (r)
71          ADD.D   f30, f30, f26           ; real summation
72          ADD.D   f29, f28, f27           ; sum (im)
73          ADD.D   f31, f31, f29           ; imaginary summation
74
75          ; FINE CALCOLI REALE E IMMAGINARIO
76
77          ; FINE 3   ITERAZIONE INTERNA
78          ; FINE 1   ITERAZIONE ESTERNA
79          ; INIZIO 2   ITERAZIONE ESTERNA
80          ; INIZIO 1   ITERAZIONE INTERNA
81
82          ; INIZIO CALCOLI REALE E IMMAGINARIO
83          L.D     f0, 8(r5)               ; read an element (ex.loop)
84          L.D     f1, 0(r7)               ; read an element (in.loop)
85          L.D     f2, 8(r6)               ; read an element (ex.loop)
86          L.D     f3, 0(r8)               ; read an element (in.loop)
87          MUL.D   f4, f0, f1              ; multiply (r)
88          MUL.D   f5, f2, f3              ; multiply (r)
89          MUL.D   f7, f0, f3              ; multiply (im)
90          MUL.D   f8, f2, f1              ; multiply (im)
91
92          SUB.D   f6, f4, f5              ; subtraction (r)
93          ADD.D   f30, f30, f6            ; real summation
94          ADD.D   f9, f8, f7              ; sum (im)
95          ADD.D   f31, f31, f9            ; imaginary summation
96
97          ; FINE CALCOLI REALE E IMMAGINARIO
98
99          ; FINE 1   ITERAZIONE INTERNA
100         ; INIZIO 2   ITERAZIONE INTERNA
101
102         ; INIZIO CALCOLI REALE E IMMAGINARIO
103         L.D     f10, 8(r5)              ; read an element (ex.loop)
104         L.D     f11, 8(r7)              ; read an element (in.loop)
105         L.D     f12, 8(r6)              ; read an element (ex.loop)
106         L.D     f13, 8(r8)              ; read an element (in.loop)
107         MUL.D   f14, f10, f11           ; multiply (r)
108         MUL.D   f15, f12, f13           ; multiply (r)
109         MUL.D   f17, f10, f13           ; multiply (im)
110         MUL.D   f18, f12, f11           ; multiply (im)
111
112         SUB.D   f16, f14, f15           ; subtraction (r)
113         ADD.D   f30, f30, f16           ; real summation
114         ADD.D   f19, f18, f17           ; sum (im)
115         ADD.D   f31, f31, f19           ; imaginary summation
116
117         ; FINE CALCOLI REALE E IMMAGINARIO
```

```
118
119         ; FINE 2   ITERAZIONE INTERNA
120         ; INIZIO 3   ITERAZIONE INTERNA
121
122         ; INIZIO CALCOLI REALE E IMMAGINARIO
123         L.D     f20, 8(r5)              ; read an element (ex.loop)
124         L.D     f21, 16(r7)             ; read an element (in.loop)
125         L.D     f22, 8(r6)              ; read an element (ex.loop)
126         L.D     f23, 16(r8)             ; read an element (in.loop)
127         MUL.D   f24, f20, f21           ; multiply (r)
128         MUL.D   f25, f22, f23           ; multiply (r)
129         MUL.D   f27, f20, f23           ; multiply (im)
130         MUL.D   f28, f22, f21           ; multiply (im)
131
132         SUB.D   f26, f24, f25           ; subtraction (r)
133         ADD.D   f30, f30, f26           ; real summation
134         ADD.D   f29, f28, f27           ; sum (im)
135         ADD.D   f31, f31, f29           ; imaginary summation
136
137         ; FINE CALCOLI REALE E IMMAGINARIO
138
139         ; FINE 3   ITERAZIONE INTERNA
140         ; FINE 2   ITERAZIONE ESTERNA
141
142
143         S.D     f30, 0(r3)              ; write the element back in memory (r)
144         S.D     f31, 0(r4)              ; write the element back in memory (im)
145
146 end:
147         HALT
```



1 **Tecnica applicata**: applicato register renaming così da utilizzare in modo più efficiente tutti i registri.

2 **Principali accorgimenti adottati**: ogni ciclo utilizza 4 registri per i Load, 6 registri per i valori di lavoro e 2 registri per il risultato. Dato che i risultati si possono sommare direttamente nello stesso registro, i registri per ogni blocco si riducono a 10. Essendo 10, ciò permette di utilizzare tutti i registri disponibili in modo da avere 3 blocchi che non hanno registri condivisi se non quello per il risultato. Il prossimo passo è ridurre le istruzioni ridondanti, ovvero i Load di valori già presenti.

3 **Conflitti residui che danno ancora luogo a stalli**:

(a) Stalli di RAW fra le istruzioni di MUL.D e ADD.D.

(b) Stalli di RAW fra le istruzioni di ADD.D e ADD.D.

(c) Stalli strutturali.

3.5 Versione 6

file.s : ComplessiV_6

```
1         .data
2  res_r:          .double 0
3  res_i:          .double 0
4  reali_e:        .double 1, 2
5  immaginari_e:   .double 1, 2
6  reali_i:        .double 3, 4, 5
7  immaginari_i:   .double 3, 4, 5
8
9         .text
10 start:
11        DADDI   r3, r0, res_r           ; p. to the 1st support array element (r)
12        DADDI   r4, r0, res_i           ; p. to the 1st array element (im)
13        DADDI   r5, r0, reali_e         ; p. to the 1st array r. element (ex.loop)
14        DADDI   r6, r0, immaginari_e    ; p. to the 1st array i. element (ex.loop)
15        DADDI   r7, r0, reali_i         ; p. to the 1st array r. element (in.loop)
16        DADDI   r8, r0, immaginari_i    ; p. to the 1st array i. element (in.loop)
17
18        ; INIZIO 1   ITERAZIONE ESTERNA
19        ; INIZIO 1   ITERAZIONE INTERNA
20        ; INIZIO CALCOLI REALE E IMMAGINARIO
21        L.D     f0, 0(r5)               ; read an element (ex.loop)
22        L.D     f1, 0(r7)               ; read an element (in.loop)
23        L.D     f2, 0(r6)               ; read an element (ex.loop)
24        L.D     f3, 0(r8)               ; read an element (in.loop)
25        MUL.D   f4, f0, f1              ; multiply (r)
26        MUL.D   f5, f2, f3              ; multiply (r)
27        MUL.D   f7, f0, f3              ; multiply (im)
28        MUL.D   f8, f2, f1              ; multiply (im)
29        SUB.D   f6, f4, f5              ; subtraction (r)
30        ADD.D   f9, f8, f7              ; sum (im)
31
32        L.D     f11, 8(r7)              ; read an element (in.loop)
33        L.D     f13, 8(r8)              ; read an element (in.loop)
34        ADD.D   f30, f30, f6            ; real summation
35        ADD.D   f31, f31, f9            ; imaginary summation
36
37        ; FINE CALCOLI REALE E IMMAGINARIO
38
39        ; INIZIO AGGIORNAMENTO DEL LOOP INTERNO
40
41
42        ; FINE AGGIORNAMENTO DEL LOOP INTERNO
43        ; FINE 1   ITERAZIONE INTERNA
44
45
46        ; INIZIO 2   ITERAZIONE INTERNA
47        ; INIZIO CALCOLI REALE E IMMAGINARIO
48
49        MUL.D   f14, f0, f11            ; multiply (r)
50        MUL.D   f15, f2, f13            ; multiply (r)
51        MUL.D   f17, f0, f13            ; multiply (im)
52        MUL.D   f18, f2, f11            ; multiply (im)
53
54        SUB.D   f16, f14, f15           ; subtraction (r)
55        ADD.D   f19, f18, f17           ; sum (im)
56
```

```
57          L.D     f21, 16(r7)             ; read an element ((in.loop)
58          L.D     f23, 16(r8)             ; read an element (in.loop)
59          ADD.D   f30, f30, f16           ; real summation
60          ADD.D   f31, f31, f19           ; imaginary summation
61
62          ; FINE CALCOLI REALE E IMMAGINARIO
63
64          ; FINE 2   ITERAZIONE INTERNA
65          ; INIZIO 3   ITERAZIONE INTERNA
66
67          ; INIZIO CALCOLI REALE E IMMAGINARIO
68
69          MUL.D   f24, f0, f21            ; multiply (r)
70          MUL.D   f25, f2, f23            ; multiply (r)
71          MUL.D   f27, f0, f23            ; multiply (im)
72          MUL.D   f28, f2, f21            ; multiply (im)
73
74          SUB.D   f26, f24, f25           ; subtraction (r)
75          ADD.D   f29, f28, f27           ; sum (im)
76
77          L.D     f0, 8(r5)               ; read an element (ex.loop)
78          L.D     f1, 0(r7)               ; read an element (in.loop)
79          L.D     f2, 8(r6)               ; read an element (ex.loop)
80          L.D     f3, 0(r8)               ; read an element (in.loop)
81          ADD.D   f30, f30, f26           ; real summation
82          ADD.D   f31, f31, f29           ; imaginary summation
83
84          ; FINE CALCOLI REALE E IMMAGINARIO
85
86          ; FINE 3   ITERAZIONE INTERNA
87          ; FINE 1   ITERAZIONE ESTERNA
88          ; INIZIO 2   ITERAZIONE ESTERNA
89          ; INIZIO 1   ITERAZIONE INTERNA
90
91          ; INIZIO CALCOLI REALE E IMMAGINARIO
92
93          MUL.D   f4, f0, f1              ; multiply (r)
94          MUL.D   f5, f2, f3              ; multiply (r)
95          MUL.D   f7, f0, f3              ; multiply (im)
96          MUL.D   f8, f2, f1              ; multiply (im)
97
98          SUB.D   f6, f4, f5              ; subtraction (r)
99          ADD.D   f9, f8, f7              ; sum (im)
100
101         L.D     f11, 8(r7)              ; read an element (in.loop)
102         L.D     f13, 8(r8)              ; read an element (in.loop)
103
104         ADD.D   f30, f30, f6            ; real summation
105         ADD.D   f31, f31, f9            ; imaginary summation
106
107         ; FINE CALCOLI REALE E IMMAGINARIO
108
109         ; FINE 1   ITERAZIONE INTERNA
110         ; INIZIO 2   ITERAZIONE INTERNA
111
112         ; INIZIO CALCOLI REALE E IMMAGINARIO
113
114         MUL.D   f14, f0, f11            ; multiply (r)
115         MUL.D   f15, f2, f13            ; multiply (r)
116         MUL.D   f17, f0, f13            ; multiply (im)
117         MUL.D   f18, f2, f11            ; multiply (im)
```

```
118
119         SUB.D   f16, f14, f15           ; subtraction (r)
120         ADD.D   f19, f18, f17           ; sum (im)
121
122         L.D     f21, 16(r7)             ; read an element (in.loop)
123         L.D     f23, 16(r8)             ; read an element (in.loop)
124
125         ADD.D   f30, f30, f16           ; real summation
126         ADD.D   f31, f31, f19           ; imaginary summation
127
128         ; FINE CALCOLI REALE E IMMAGINARIO
129
130         ; FINE 2   ITERAZIONE INTERNA
131         ; INIZIO 3   ITERAZIONE INTERNA
132
133         ; INIZIO CALCOLI REALE E IMMAGINARIO
134
135         MUL.D   f24, f0, f21            ; multiply (r)
136         MUL.D   f25, f2, f23            ; multiply (r)
137         MUL.D   f27, f0, f23            ; multiply (im)
138         MUL.D   f28, f2, f21            ; multiply (im)
139
140         SUB.D   f26, f24, f25           ; subtraction (r)
141         ADD.D   f29, f28, f27           ; sum (im)
142         ADD.D   f30, f30, f26           ; real summation
143         ADD.D   f31, f31, f29           ; imaginary summation
144
145         ; FINE CALCOLI REALE E IMMAGINARIO
146
147         ; FINE 3   ITERAZIONE INTERNA
148         ; FINE 2   ITERAZIONE ESTERNA
149
150
151         S.D     f30, 0(r3)               ; write the element back in memory (r)
152         S.D     f31, 0(r4)               ; write the element back in memory (im)
153
154 end:
155         HALT
```

```
Execution
108 Cycles
73 Instructions
1.479 Cycles Per Instruction (CPI)

Stalls
37 RAW Stalls
0 WAW Stalls
0 WAR Stalls
8 Structural Stalls
0 Branch Taken Stalls
0 Branch Misprediction Stalls

Code size
292 Bytes
```

1 **Tecnica applicata**: applicato register reordering. Rispetto alle versioni precedenti si nota una diminuizione della CPI più considerevole.

Nota per il prossimo aggiornamento: controllare il register reordering per migliorare ulteriormente il CPI.

3 **Conflitti residui che danno ancora luogo a stalli**:

(a) Stalli di RAW fra le istruzioni di MUL.D e ADD.D.

(b) Stalli di RAW fra le istruzioni di ADD.D e ADD.D.

(c) Stalli strutturali.

3.6 Versione 7

file.s : ComplessiV_7

```
1           .data
2   res_r:          .double 0
3   res_i:          .double 0
4   reali_e:        .double 1, 2
5   immaginari_e:   .double 1, 2
6   reali_i:        .double 3, 4, 5
7   immaginari_i:   .double 3, 4, 5
8
9           .text
10  start:
11          DADDI   r3, r0, res_r           ; p. to the 1st support array element (r)
12          DADDI   r4, r0, res_i           ; p. to the 1st array element (im)
13          DADDI   r5, r0, reali_e         ; p. to the 1st array r. element (ex.loop)
14          DADDI   r6, r0, immaginari_e    ; p. to the 1st array i. element (ex.loop)
15          DADDI   r7, r0, reali_i         ; p. to the 1st array r. element (in.loop)
16          DADDI   r8, r0, immaginari_i    ; p. to the 1st array i. element (in.loop)
17
18          ; INIZIO 1   ITERAZIONE ESTERNA
19          ; INIZIO 1   ITERAZIONE INTERNA
20          ; INIZIO CALCOLI REALE E IMMAGINARIO
21          L.D     f0,  0(r5)              ; read an element (ex.loop)
22          L.D     f1,  0(r7)              ; read an element (in.loop)
23          L.D     f2,  0(r6)              ; read an element (ex.loop)
24          L.D     f3,  0(r8)              ; read an element (in.loop)
25          L.D     f11, 8(r7)              ; read an element (in.loop)
26          L.D     f13, 8(r8)              ; read an element (in.loop)
27          L.D     f21, 16(r7)             ; read an element (in.loop)
28          L.D     f23, 16(r8)             ; read an element (in.loop)
29
30          MUL.D   f4,  f0, f1             ; multiply (r)
31          MUL.D   f5,  f2, f3             ; multiply (r)
32          MUL.D   f7,  f0, f3             ; multiply (im)
33          MUL.D   f8,  f2, f1             ; multiply (im)
34          MUL.D   f14, f0, f11            ; multiply (r)
35          MUL.D   f15, f2, f13            ; multiply (r)
36          MUL.D   f17, f0, f13            ; multiply (im)
37          MUL.D   f18, f2, f11            ; multiply (im)
38          MUL.D   f24, f0, f21            ; multiply (r)
39          MUL.D   f25, f2, f23            ; multiply (r)
40          MUL.D   f27, f0, f23            ; multiply (im)
41          MUL.D   f28, f2, f21            ; multiply (im)
42
43          SUB.D   f6,  f4,  f5            ; subtraction (r)
44          ADD.D   f9,  f8,  f7            ; sum (im)
45          SUB.D   f16, f14, f15           ; subtraction (r)
46          ADD.D   f19, f18, f17           ; sum (im)
47          SUB.D   f26, f24, f25           ; subtraction (r)
48          ADD.D   f29, f28, f27           ; sum (im)
49
50          L.D     f0, 8(r5)               ; read an element (ex.loop)
51          L.D     f2, 8(r6)               ; read an element (ex.loop)
52
53          MUL.D   f4,  f0,  f1            ; multiply (r)
54          MUL.D   f5,  f2,  f3            ; multiply (r)
55          MUL.D   f7,  f0,  f3            ; multiply (im)
56          MUL.D   f8,  f2,  f1            ; multiply (im)
```

```
57          ADD.D   f30, f30, f6          ; real summation
58          ADD.D   f31, f31, f9          ; imaginary summation
59          MUL.D   f14, f0,  f11         ; multiply (r)
60          MUL.D   f15, f2,  f13         ; multiply (r)
61          MUL.D   f17, f0,  f13         ; multiply (im)
62          MUL.D   f18, f2,  f11         ; multiply (im)
63
64          SUB.D   f6, f4, f5            ; subtraction (r)
65          ADD.D   f9, f8, f7            ; sum (im)
66
67          ADD.D   f30, f30, f16         ; real summation
68          ADD.D   f31, f31, f19         ; imaginary summation
69
70          MUL.D   f24, f0, f21          ; multiply (r)
71          MUL.D   f25, f2, f23          ; multiply (r)
72
73          ADD.D   f30, f30, f6          ; real summation
74          ADD.D   f31, f31, f9          ; imaginary summation
75
76          MUL.D   f27, f0, f23          ; multiply (im)
77          MUL.D   f28, f2, f21          ; multiply (im)
78
79          SUB.D   f16, f14, f15         ; subtraction (r)
80          ADD.D   f19, f18, f17         ; sum (im)
81
82          ADD.D   f30, f30, f26         ; real summation
83          ADD.D   f31, f31, f29         ; imaginary summation
84
85          SUB.D   f26, f24, f25         ; subtraction (r)
86          ADD.D   f29, f28, f27         ; sum (im)
87
88
89
90          ADD.D   f30, f30, f16         ; real summation
91          ADD.D   f31, f31, f19         ; imaginary summation
92          ADD.D   f30, f30, f26         ; real summation
93          ADD.D   f31, f31, f29         ; imaginary summation
94
95          S.D     f30, 0(r3)            ; write the element back in memory (r)
96          S.D     f31, 0(r4)            ; write the element back in memory (im)
97
98  end:
99          HALT
```

```
Execution

87 Cycles
67 Instructions
1.299 Cycles Per Instruction (CPI)


Stalls

5 RAW Stalls
0 WAW Stalls
0 WAR Stalls
16 Structural Stalls
0 Branch Taken Stalls
0 Branch Misprediction Stalls


Code size

268 Bytes
```

1 **Tecnica applicata**: applicato ulteriore instruction reordering.

Nota: In totale ci sono 6 registri per la parte reale e immaginaria di z_3, z_4 e z_5. I registri z_1 e z_2 invece vengono aggiornati una seconda volta.

3 **Conflitti residui che danno ancora luogo a stalli**:

(a) Stalli di RAW fra le istruzioni di MUL.D e ADD.D.

(b) Stalli di RAW fra le istruzioni di ADD.D e ADD.D.

(c) Stalli strutturali

4 Conclusioni

4.1 Confronto dei risultati ottenuti

Dalla tabella dei valori qui di seguito riportata, si può notare immediatamente il netto miglioramento in termini di cicli di clock, istruzioni eseguite e CPI fra la versione iniziale del codice (V1) e quella finale (V7). Difatti il numero dei cicli di clock si è ridotto del 75%, le istruzioni eseguite si sono abbassate di circa il 60% e il CPI ha subito un notevole calo del 40%. D'altra parte però la dimensione del codice stesso è aumentata del 76% con un risultato dunque diametralmente opposto a quello rilevato per il numero di cicli di clock. Nonostante ciò la dimensione del codice rispetto alle versione 4 (V4) risulta migliorata dell' 81% grazie all'uso massiccio di tutti i registri e all'impiego efficiente delle tecniche di instruction reordering e register renaming utilizzati di seguito al loop unrolling delle istruzioni.

Versione	Cycles	Instructions	CPI	Code size
V1	340	155	2.194	152 Bytes
V2	192	121	1.587	136 Bytes
V3	175	108	1.620	236 Bytes
V4	159	93	1.710	372 Bytes
V5	147	81	1.815	324 Bytes
V6	108	73	1.497	292 Bytes
V7	87	67	1.299	268 Bytes

4.2 Commenti conclusivi

Sulla base dei risultati ottenuti dall'esecuzione di ogni versione e dal confronto operato in precedenza, possiamo concludere che c'è stato sostanziale miglioramento prestazionale del codice.