

Calcolo di massimo e media in un vettore di numeri reali.

Progetto d'esame di Architettura degli Elaboratori A.A. 2020/2021

Relatori:

Montanari Matteo Marco, 299166 Di Fabrizio Giacomo, 301591

Docente:

Prof. Alessandro Bogliolo



Specifica

Scopo del progetto

La specifica che si vuole presentare consiste nel realizzare un algoritmo che calcoli il valore massimo e la media aritmetica di un vettore di numeri reali rappresentati in virgola mobile a doppia precisione (double floating point). Tale algoritmo sarà realizzato in Assembly per architetture MIPS a 64 bit.



Algoritmo utilizzato

Per il calcolo del valore massimo in un array di numeri reali abbiamo utilizzato il seguente algoritmo (linguaggio C):

Per il calcolo della media abbiamo invece utilizzato il seguente algoritmo (linguaggio C):

```
somma = a[0];
for(i = 1; i < n; i++)

{
    somma += a[i];
}
media = somma / n;</pre>
```

Per cui l'algoritmo per il calcolo di media e massimo è dato dal seguente codice (linguaggio C):

```
somma = a[0];
massimo = a[0];
for(i = 1; i < n; i++)

{
    somma += a[i];
    if(massimo < a[i])

    massimo = a[i]
    }
media = somma / n;</pre>
```

(n è la dimensione del vettore e a indica il puntatore al primo elemento del vettore)



Versione uno

Passando al codice assembly per architettura MIPS a 64 bit otteniamo il seguente codice:

```
; ricerca del massimo in un array di numeri e media dei valori
     a: .double 4.3, 2.1, 5.7, 8.4, 9.0, 2.7, 8.4
                                   ; contatore dim-1, il primo elemento è il max, mancano 6 confronti (dim=7)
    n: .word 6
     start:
          LW rl, n(r0) ; carica il numero di elementi dell'array -l in rl (r0 == 0) DADDI r2, r0, a ; carica in r2 il puntatore al primo elemento dell'array
          L.D f0, 0(r2)
                                   ; carica in f0 il valore di a[0] == max
                                  ; salva per la divisione finale (media) la dimensione dell'array
          DADDI r6, r1, 1
         MOV.D f2, f0 ; sposta a[0] in f2 per la somma di tutti i valori
MTCl r6, f3 ; sposta il contenuto (dim array) di r6 (INT reg) in f3 (FP reg)
CVT.D.L f3, f3 ; (NB divisione) converti f3 (Long INT) in un Double FP e mettile
                                         ; (NB divisione) converti f3 (Long INT) in un Double FP e mettilo in f3
14
                             ; stackoverflow
          BEQZ rl, exit
                                   ; esci se il contatore è zero
18
          L.D f1, 8(r2)
ADD.D f2, f2, f1
                                  ; carica in fl il valore di a[l] indirizzato da r2 + 8
; carica in f2 la somma di tutti i valori fino all'i-esimo
19
          DADDI r2, r2, 8
                                   ; r2 += 8 -> r2 contiene l'indirizzo di a[i+1]
20
          DADDI rl, rl, -1
                                   ; decrementa contatore ciclo
                             ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
; sposta il contenuto di fl in r4
; altrimenti errore nell'istruzione BEQ
          MFC1 r3, f0
23
          MFC1 r4, f1
24
          BEQ r3, r4, loop ; se sono uguali vai avanti nell'array (loop)
                                   ; altrimenti r3==f0 = max > f1==r4 oppure il contrario
          , ... , ... - 1 se rq==fl < f0==r3 = max altrimenti r5
; se r5 = 0 allora r4==fl > max quindi lo salvo
BEQZ r5, save ; salva il nuovo may = flore.
                                   ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
28
29
30
                                   ; loop (while)
          J loop
33
          MOV.D f0, fl ; sovrascrivi (sposta) f0 (max) con fl (nuovo max)
                             ; vai avanti nell'array
; il massimo è in f0
34
          J loop
          DIV.D f2, f2, f3
                                        ; dividi la somma di tutti i valori per il numero di valori
                                 ; la media aritmetica è in f2
```

Nella sezione data, comune a tutte le versioni dell'algoritmo che presenteremo, sono presenti i dati di input costituiti dal numero di elementi del vettore meno uno (indirizzo n) e il vettore stesso (indirizzo a).

Per quanto riguarda il codice, nelle istruzioni prima del ciclo carichiamo nei registri le informazioni necessarie all'elaborazione tra cui la dimensione dell'array e il primo elemento del vettore che inizialmente sarà il valore massimo. Perciò nel registro r1 abbiamo il contatore del numero di cicli rimasti, nel registro r2 l'indirizzo del primo elemento del vettore, in f0 e in f2 il valore del primo elemento del vettore, in f3 la dimensione del vettore in rappresentazione in virgola mobile a doppia precisione.

All'interno del ciclo inizialmente verifichiamo che il contatore del numero di cicli rimasti, contenuto in r1, sia diverso da zero, altrimenti usciamo dal ciclo. Poi carichiamo in f1 l'elemento successivo del vettore, indicizzato da r2 + 8. Sommiamo f1 a f2, salvando il risultato in f2, per memorizzare la somma di tutti i valori, la quale verrà utilizzata per il calcolo della media aritmetica. Successivamente si decrementa il contatore del ciclo e si aggiorna l'indirizzo del prossimo elemento. Per effettuare i confronti dobbiamo spostare il contenuto di f0 e f1 rispettivamente in r3 e r4 tramite le istruzioni MFC1. Ora verifichiamo che non siano uguali, altrimenti si passa all'elemento successivo, poiché il massimo sarebbe tale valore. A questo punto tramite l'istruzione SLT memorizziamo in r5 il fatto che r4 sia minore di r3, ovvero che il massimo attuale, contenuto in f0, sia ancora il massimo. In caso contrario il registro r5 conterrà il valore zero. Prima di passare all'elemento successivo occorre salvare il nuovo massimo nel caso in cui r5 contenga zero, tramite l'istruzione MOV.D. Al termine del ciclo il massimo valore sarà contenuto in f0, mentre la media sarà calcolata facendo la divisione tra la somma di tutti i valori (f2) e il numero di elementi del vettore (f3) e sarà contenuta in f2.



Due versioni ottimizzate dell'algoritmo sopra descritto sono le seguenti:

Versione due

```
; ricerca del massimo in un array di numeri e media dei valori versione 2
     ; instruction reordering
    a: .double 4.3, 2.1, 5.7, 8.4, 9.0, 2.7, 8.4
                                  ; contatore dim-1, il primo elemento è il max, mancano 6 confronti (dim=7)
    n: .word 6
          .text
8
    start:
                               ; carica il numero di elementi dell'array -l in rl (r0 == 0)
; carica in r2 il puntatore al primo elemento dell'array
; carica in f0 il valore di a[0] == max
; salva per la divisione finale (media) la dimensione dell'array
         LW rl, n(r0)
10
         DADDI r2, r0, a
11
         L.D f0, 0(r2)
         DADDI r6, r1, 1
MOV.D f2, f0
12
13
                                 ; sposta a[0] in f2 per la somma di tutti i valori
                                 ; sposta il contenuto (dim array) di r6 (INT reg) in f3 (FP reg)
14
        MTCl r6, f3
                                 ; (NB divisione) converti f3 (Long INT) in un Double FP e mettilo in f3
15
         CVT.D.L f3, f3
16
    loop:
         BEQZ rl, exit
                                 ; esci se il contatore è zero
                                ; carica in fl il valore di a[l] indirizzato da r2 + 8; i += 8 -> r2 contiene l'indirizzo di a[i+1]; decrementa contatore ciclo; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
         L.D fl, 8(r2)
         DADDI r2, r2, 8
         DADDI rl, rl, -1
         MFC1 r3, f0
                                 ; sposta il contenuto di fl in r4
23
         MFC1 r4, f1
                               ; altrimenti errore nell'istruzione BEQ
24
25
         SLT r5, r4, r3
                                 ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti <math>r5 = 0
                               ; se r5 = 0 allora r4==fl > max quindi lo salvo
26
         ADD.D f2, f2, f1 ; carica in f2 la somma di tutti i valori fino all'i-esimo BEQ r3, r4, loop ; se sono uguali vai avanti nell'array (loop)
27
28
29
         BEQZ r5, save
                                  ; salva il nuovo max = fl==r4 se r5 = 0
30
         J loop
                                 ; loop (while)
31
32
    save:
        MOV.D f0, f1
                                ; sovrascrivi (sposta) f0 (max) con fl (nuovo max)
33
                      ; vai avanti nell'array
; il massimo è in f0
34
          J loop
35
36
          DIV.D f2, f2, f3 ; dividi la somma di tutti i valori per il numero di valori
                                  : la media aritmetica è in f2
```



Versione tre

```
1 ; ricerca del massimo in un array di numeri e media dei valori versione 3
   a: .double 4.3, 2.1, 5.7, 8.4, 9.0, 2.7, 8.4
                         ; contatore dim-1, il primo elemento è il max, mancano 6 confronti (dim=7)
        .text
   start:
       ; carica in f0 il valore di a[0] == max
       L.D f0, 0(r2)
11
       DADDI r6, r1, 1
                         ; salva per la divisione finale (media) la dimensione dell'array
12
       MOV.D f2, f0
                         ; sposta a[0] in f2 per la somma di tutti i valori
13
                      ; (NB divisione) converti f3 (Long INT) in un Double FP e mettilo in f3
       MTC1 r6. f3
14
       CVT.D.L f3, f3
15
16
   loop:
      L.D fl, 8(r2)
                        ; carica in fl il valore di a[l] indirizzato da r2 + 8
18
                        ; i += 8 -> r2 contiene l'indirizzo di a[i+l] ; decrementa contatore ciclo
       DADDI r2, r2, 8
19
20
       DADDI rl, rl, -1
       MFC1 r3, f0
21
                         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
22
       MFC1 r4, f1
                         ; sposta il contenuto di fl in r4
23
                     ; altrimenti errore nell'istruzione BEQ
                         ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
       SLT r5, r4, r3
24
25
                         ; se r5 = 0 allora r4==fl > max quindi lo salvo
       ADD.D f2, f2, f1
26
                         ; carica in f2 la somma di tutti i valori fino all'i-esimo
       BEQ r3, r4, loop ; se sono uguali vai avanti nell'array (loop)
27
       BEQZ r5, save
BNEZ r1, loop
28
                         ; salva il nuovo max = fl==r4 se r5 = 0
                            ; loop finché rl diverso da 0
29
30
       J exit
                             ; esci ciclo
31
     MOV.D f0, f1
                         ; sovrascrivi (sposta) f0 (max) con fl (nuovo max)
                          ; vai avanti nell'array
                ; il massimo è in f0
      DIV.D f2, f2, f3 ; dividi la somma di tutti i valori per il numero di valori
                          : la media aritmetica è in f2
```

In queste due soluzioni abbiamo adottato la tecnica dell'instruction reordering (riordinamento delle istruzioni) per ridurre gli stalli RAW dovuti alla vicinanza delle istruzioni L.D e ADD.D (righe 18-19 versione 1). Abbiamo dunque spostato l'istruzione ADD.D al termine del ciclo e l'istruzione BEQ dopo di essa in modo da ridurre gli stalli dovuti alla latenza dell'istruzione di somma floating point e quelli dovuti ai possibili salti (Branch Taken Stalls). Infine, nella terza soluzione, abbiamo sostituito la logica di controllo del ciclo (inizialmente simile al costrutto while del C) tramite l'istruzione BNEZ che verifica la condizione su r1 solo al termine del ciclo e non prima (simile al costrutto do-while del C).



Versione quattro

Un'altra tecnica di ottimizzazione, che funziona solo se si conosce a priori la dimensione (costante) del vettore, è il loop unrolling totale. Questo procedimento prevede di eliminare il ciclo e copiare le istruzioni da ripetere dell'algoritmo una di seguito all'altra, un numero di volte pari alla dimensione del vettore meno uno.

```
; ricerca del massimo in un array di numeri e media dei valori versione 4
    ; loop unrolling
3
   a: .double 4.3, 2.1, 5.7, 8.4, 9.0, 2.7, 8.4
   n: .word 6
                           ; contatore dim-1, il primo elemento è il max, mancano 6 confronti (dim=7)
   start:
10
       LW rl, n(r0)
                           ; carica il numero di elementi dell'array -1 in rl (r0 == 0)
                         ; carica in r2 il puntatore al primo elemento dell'array
; carica in f0 il valore di a[0] == max
       DADDI r2, r0, a
       L.D f0, 0(r2)
13
14
       DADDI rl, rl, 1
                               ; salva per la divisione finale (media) la dimensione dell'array
      MOV.D f2, f0
15
                          ; sposta a[0] in f2 per la somma di tutti i valori
16
       MTCl rl, f3
                               ; sposta il contenuto (dim array) di rl (INT reg) in f3 (FP reg)
17
        CVT.D.L f3, f3
                          ; (NB divisione) converti f3 (Long INT) in un Double FP e mettilo in f3
18
19 10:
                          ; carica in fl il valore di a[1] indirizzato da r2 + 8
       L.D fl. 8(r2)
20
       DADDI r2, r2, 8
                           ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
21
        MFC1 r3, f0
                           ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
22
23
       MFC1 r4. fl
                           : sposta il contenuto di fl in r4
24
                           ; altrimenti errore nell'istruzione BEQ
2.5
       SLT r5, r4, r3
                           ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
                           ; se r5 = 0 allora r4==f1 > max quindi lo salvo
26
                           ; carica in f2 la somma di tutti i valori fino all'i-esimo
27
       ADD.D f2, f2, f1
       BEQ r3, r4, 11
                              ; se sono uguali vai avanti nell'array
28
29
       BEQZ r5, save0
                               ; salva il nuovo max = fl==r4 se r5 = 0
        J 11
30
                               ; iterazione successiva
31
32 save0:
       MOV.D f0, f1
                           ; sovrascrivi (sposta) f0 (max) con fl (nuovo max)
33
35 11:
                           ; carica in fl il valore di a[l] indirizzato da r2 + 8
36
        L.D fl. 8(r2)
                           ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
       DADDI r2, r2, 8
37
       MFC1 r3, f0
                           ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
38
39
       MFC1 r4, f1
                           ; sposta il contenuto di fl in r4
40
                           ; altrimenti errore nell'istruzione BEQ
41
        SLT r5, r4, r3
                            ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
42
                           ; se r5 = 0 allora r4==fl > max quindi lo salvo
43
        ADD.D f2, f2, f1
                           ; carica in f2 la somma di tutti i valori fino all'i-esimo
                              ; se sono uguali vai avanti nell'array
        BEQ r3, r4, 12
44
45
        BEQZ r5, savel
                               ; salva il nuovo max = fl==r4 se r5 = 0
46
                                ; iterazione successiva
48 savel:
49
        MOV.D f0, f1
                            ; sovrascrivi (sposta) f0 (max) con fl (nuovo max)
50
51
52
       L.D fl. 8(r2)
                           : carica in fl il valore di a[l] indirizzato da r2 + 8
        DADDI r2, r2, 8
                           ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
53
       MFC1 r3, f0
54
                           ; sposta il contenuto di f0 (FP reg) in r3 (INT reg)
5.5
       MFC1 r4, f1
                           ; sposta il contenuto di fl in r4
56
                            ; altrimenti errore nell'istruzione BEQ
57
        SLT r5, r4, r3
                           ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti <math>r5 = 0
58
                            ; se r5 = 0 allora r4==f1 > max quindi lo salvo
59
        ADD.D f2, f2, f1
                          ; carica in f2 la somma di tutti i valori fino all'i-esimo
        BEQ r3, r4, 13
                             ; se sono uguali vai avanti nell'array
; salva il nuovo max = fl==r4 se r5 = 0
60
61
        BEQZ r5, save2
62
                                ; iterazione successiva
63
   save2:
       MOV.D f0, fl ; sovrascrivi (sposta) f0 (max) con fl (nuovo max)
```

```
67
                            ; carica in fl il valore di a[1] indirizzato da r2 + 8
; i += 8 -> r2 contiene l'indirizzo di a[i+1]
; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
         L.D fl. 8(r2)
 68
 69
         DADDI r2, r2, 8
         MFC1 r3, f0
 70
                             ; sposta il contenuto di fl in r4
 71
         MFC1 r4, f1
 72
                             ; altrimenti errore nell'istruzione BEQ
                             ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
 73
         SLT r5, r4, r3
         ; se r5 = 0 allora r4==f1 > max quindi lo salvo
ADD.D f2, f2, f1 ; carica in f2 la somma di tutti i valori fino all'i-esimo
 75
         BEQ r3, r4, 14
                              ; se sono uguali vai avanti nell'array
 76
         BEQZ r5, save3
                                  ; salva il nuovo max = fl==r4 se r5 = 0
 78
                                 ; iterazione successiva
         J 14
 79
 80
     save3:
         MOV.D f0, f1 ; sovrascrivi (sposta) f0 (max) con fl (nuovo max)
 81
 82
 83
 84
         L.D fl, 8(r2)
                            ; carica in fl il valore di a[1] indirizzato da r2 + 8
                             ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
 85
         DADDI r2, r2, 8
         MFC1 r3, f0
 86
                             ; sposta il contenuto di fl in r4
 87
         MFC1 r4, f1
                             ; altrimenti errore nell'istruzione BEQ
 88
                             ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
 89
         SLT r5, r4, r3
                            ; se r5 = 0 allora r4==fl > max quindi lo salvo
 90
                            ; carica in f2 la somma di tutti i valori fino all'i-esimo
 91
         ADD.D f2, f2, f1
         BEQ r3, r4, 15
                                 ; se sono uguali vai avanti nell'array
 92
                                 ; salva il nuovo max = fl==r4 se r5 = 0
 93
         BEQZ r5, save4
 94
         J 15
                                  : iterazione successiva
 95
 96
     save4:
         MOV.D f0, fl ; sovrascrivi (sposta) f0 (max) con fl (nuovo max)
 97
98
 99
     15:
                            ; carica in fl il valore di a[l] indirizzato da r2 + 8
         L.D fl. 8(r2)
                            ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
         DADDI r2, r2, 8
102
         MFC1 r3, f0
                             ; sposta il contenuto di f0 (FP reg) in r3 (INT reg)
                             ; sposta il contenuto di fl in r4
103
        MFC1 r4, f1
104
                              ; altrimenti errore nell'istruzione BEQ
                              ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
105
         SLT r5, r4, r3
106
                             ; se r5 = 0 allora r4==fl > max quindi lo salvo
107
         ADD.D f2, f2, f1 ; carica in f2 la somma di tutti i valori fino all'i-esimo
         BEQ r3, r4, 16
                                 ; se sono uguali vai avanti nell'array
108
109
         BEQZ r5, save5
                                  ; salva il nuovo max = fl==r4 se r5 = 0
         J 16
                                   ; fine iterazione
110
111
112 save5:
113
         MOV.D f0, f1
                            ; sovrascrivi (sposta) f0 (max) con fl (nuovo max)
114
115 16:
116
         DIV.D f2, f2, f3 ; dividi la somma di tutti i valori per il numero di valori
117
          HALT
                               ; la media aritmetica è in f2
```

Le istruzioni iniziali sono le stesse degli algoritmi precedenti tranne che per le istruzioni a righe 14 e 16 in cui il registro r6 di destinazione è stato sostituito con r1, il cui contenuto non è più necessario (contatore del ciclo). Dato che il primo elemento è inizialmente il massimo sono necessari 6 confronti per il calcolo del massimo in tutto il vettore (dimensione 7). Perciò i vari passaggi sono indicizzati da 0 a 5, le istruzioni BEQ di ogni passaggio salteranno al passaggio successivo come anche l'istruzione J che permette di saltare l'esecuzione dell'istruzione MOV.D nel caso in cui non sia necessaria. In caso contrario invece sarà necessario eseguire un'istruzione MOV.D per ogni passaggio che sarà dunque indicizzata dal numero del passaggio nel quale dovrà essere eventualmente eseguita (passaggio corrente). Al termine delle iterazioni si esegue la divisione come per gli algoritmi precedenti. L'istruzione DADDI che decrementa il contatore del ciclo ad ogni passaggio è stata eliminata poiché non più necessaria.



Versione cinque

Per ottimizzare ulteriormente questa versione dell'algoritmo utilizziamo la tecnica del register renaming che consiste nell'utilizzare più registri per memorizzare i vari elementi del vettore.

```
; ricerca del massimo in un array di numeri e media dei valori versione 5
    ; loop unrolling e register renaming
   a: .double 4.3, 2.1, 5.7, 8.4, 9.0, 2.7, 8.4
6 n: .word 6
                            ; contatore dim-1, il primo elemento è il max, mancano 6 confronti (dim=7)
        .text
   start:
        LW rl, n(r0)
                             ; carica il numero di elementi dell'array -1 in rl (r0 == 0)
                          ; carica in r2 il puntatore al primo elemento dell'array
       DADDI r2, r0, a
        L.D f0, 0(r2)
                                 ; carica in f0 il valore di a[0] == max
                                ; carica in f1 il valore di a[1]; carica in f2 il valore di a[2]; carica in f3 il valore di a[3]
14
        L.D fl, 8(r2)
15
        L.D f2, 16(r2)
16
        L.D f3, 24(r2)
        L.D f4, 32(r2)
                                ; carica in f4 il valore di a[4]
                                 ; carica in f5 il valore di a[5]
18
        L.D f5, 40(r2)
        L.D f6, 48(r2)
                                 : carica in f6 il valore di a[6]
19
20
21
        DADDI rl, rl, l ; salva per la divisione finale (media) la dimensione dell'array
       MOV.D f7, f0 ; sposta a[0] in f7 per la somma di tutti i valori
MTCl rl, f8 ; sposta il contenuto (dim array) di rl (INT reg) :
                                 ; sposta il contenuto (dim array) di rl (INT reg) in f8 (FP reg)
        CVT.D.L f8, f8
                                 ; (NB divisione) converti f8 (Long INT) in un Double FP e mettilo in f8
25
26
   ; analogo alla versione precedente
   10:
27
        MFC1 r3. f0
28
                             : all'inizio max == f0
        MFC1 r4, f1
                            ; sposta il contenuto di fl in r4
29
                            ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
; se r5 = 0 allora r4==f1 > max quindi lo salvo
        SLT r5, r4, r3
30
31
        ADD.D f7, f7, f1
                                 ; carica in f7 la somma di tutti i valori fino all'i-esimo
        BEQ r3, r4, 11
                            ; se sono uguali vai avanti nell'array (loop)
33
                             ; salva il nuovo max = fl==r4 se r5 = 0
34
        BEQZ r5, save0
   J 11
save0:
                            ; esci ciclo
        MOV.D f0, f1
                             ; sovrascrivi (sposta) f0 (max) con fl (nuovo max)
39
40
                      ; il massimo attuale è in fl4
41
       MFC1 r3, f0
42
        MFC1 r4, f2
                                  ; sposta il contenuto di f2 in r4
                            ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0 ; se r5 = 0 allora r4==f1 > max quindi lo salvo
43
       SLT r5, r4, r3
44
        ADD D f7, f7, f2
                                 ; carica in f7 la somma di tutti i valori fino all'i-esimo
45
       BEQ r3, r4, 12
                             ; se sono uguali vai avanti nell'array (loop)
46
                            ; salva il nuovo max = fl==r4 se r5 = 0
; esci ciclo
        BEOZ r5. savel
47
48
        J 12
49
   savel:
        MOV.D f0, f2
                              ; sovrascrivi (sposta) f0 (max) con f2 (nuovo max)
53
54
       MFC1 r3, f0
                            ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
                                  ; sposta il contenuto di f3 in r4
55
        MFC1 r4, f3
                             ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
56
        SLT r5, r4, r3
57
                             ; se r5 = 0 allora r4==fl > max quindi lo salvo
        ADD.D f7, f7, f3
58
                                 ; carica in f7 la somma di tutti i valori fino all'i-esimo
                             ; se sono uguali vai avanti nell'array (loop)
59
        BEQ r3, r4, 13
                             ; salva il nuovo max = fl==r4 se r5 = 0
60
        BEQZ r5, save2
61
                            ; esci ciclo
62
63 save2:
       MOV.D f0, f3
                                ; sovrascrivi (sposta) f0 (max) con f3 (nuovo max)
```

```
13:
                            ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
 67
        MFC1 r3, f0
                              ; sposta il contenuto di f4 in r4
 68
         MFC1 r4, f4
                            ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
 69
         SLT r5, r4, r3
 70
                            ; se r5 = 0 allora r4==fl > max quindi lo salvo
 71
        ADD.D f7, f7, f4
                               ; carica in f7 la somma di tutti i valori fino all'i-esimo
 72
         BEQ r3, r4, 14
                           ; se sono uguali vai avanti nell'array (loop)
 73
         BEQZ r5, save3
                            ; salva il nuovo max = fl==r4 se r5 = 0
 74
         J 14
                            ; esci ciclo
 75
 76
     save3:
        MOV.D f0, f4
                                ; sovrascrivi (sposta) f0 (max) con f4 (nuovo max)
 78
 79
         MFC1 r3, f0
                           ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
 80
         MFC1 r4, f5
 81
                                ; sposta il contenuto di f5 in r4
         SLT r5, r4, r3
                            ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
 82
 83
                           ; se r5 = 0 allora r4==fl > max quindi lo salvo
 84
         ADD.D f7, f7, f5
                               ; carica in f7 la somma di tutti i valori fino all'i-esimo
 85
         BEQ r3, r4, 15
                            ; se sono uguali vai avanti nell'array (loop)
 86
         BEQZ r5, save4
                            ; salva il nuovo max = fl==r4 se r5 = 0
         J 15
 87
                            ; esci ciclo
 88
 89
     save4:
 90
        MOV.D f0, f5
                                ; sovrascrivi (sposta) f0 (max) con f5 (nuovo max)
 91
 92
     15:
        MFC1 r3, f0
 93
                            ; sposta il contenuto di f0 (FP reg) in r3 (INT reg)
 94
         MFC1 r4, f6
                                ; sposta il contenuto di f6 in r4
 95
         SLT r5, r4, r3
                            ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
                            ; se r5 = 0 allora r4==fl > max quindi lo salvo
 96
         ADD.D f7, f7, f6
 97
                               ; carica in f7 la somma di tutti i valori fino all'i-esimo
                          ; se sono uguali vai avanti nell'array (loop)
 98
         BEQ r3, r4, 16
 99
         BEQZ r5, save5
                            ; salva il nuovo max = fl==r4 se r5 = 0
100
        J 16
102 save5:
        MOV.D f0, f6
103
                           ; sovrascrivi (sposta) f0 (max) con f6 (nuovo max)
104
105 16:
106
         DIV.D f7, f7, f8
                               ; dividi la somma di tutti i valori per il numero di valori
                            ; la media è in f7
107
108
                            ; il massimo è in f0
```

In questa versione i vari elementi del vettore sono caricati a inizio elaborazione tramite istruzioni L.D con offset opportuni e caricati nei registri da f0 a f6. La somma di tutti i valori e poi la media saranno memorizzati nel registro f7 mentre la dimensione del vettore sarà contenuta in f8. Per la restante parte il codice è analogo alla versione precedente senza istruzioni DADDI (non più necessarie) e L.D all'interno dei vari passaggi (spostate all'inizio). Da notare il fatto che ora i vari elementi del vettore non si trovano più in f1 ad ogni passaggio ma sono su registri diversi e vanno richiamati opportunamente nelle istruzioni MFC1, ADD.D e MOV.D.



Versione sei

Una versione ulteriormente ottimizzata di questo codice può essere ottenuta spostando tutte le somme a inizio elaborazione come pure il calcolo della media.

```
; ricerca del massimo in un array di numeri e media dei valori versione 6
    ; loop unrolling e register renaming
   a: .double 4.3, 2.1, 5.7, 8.4, 9.0, 2.7, 8.4
n: .word 6 ; contatore dim-1, il
                            ; contatore dim-1, il primo elemento è il max, mancano 6 confronti (dim=7)
        .text
    start:
10
      LW rl, n(r0)
                           ; carica il numero di elementi dell'array -1 in rl (r0 == 0)
                            ; carica in r2 il puntatore al primo elemento dell'array
        DADDI r2, r0, a
        L.D f0, 0(r2)
                           ; carica in f0 il valore di a[0] == max
       L.D fl, 8(r2)
                           ; carica in fl il valore di a[l]
                           ; carica in f2 il valore di a[2]
; carica in f3 il valore di a[3]
15
        L.D f2, 16(r2)
       L.D f3, 24(r2)
                           ; carica in f4 il valore di a[4]
; carica in f5 il valore di a[5]
; carica in f6 il valore di a[6]
17
        L.D f4, 32(r2)
18
        L.D f5, 40(r2)
19
        L.D f6, 48(r2)
20
21
       ADD.D f9, f1, f0
                                 ; NB somma FP latenza 3
22
       ADD.D f10, f3, f2
23
        ADD.D fl1, f5, f4
24
        ADD.D f12, f9, f6
       ADD.D f13, f11, f10
                                 : somma tutti valori tranne fl2 = fl+f0+f6
25
26
27
       DADDI rl, rl, 1 ; salva per la divisione finale (media) la dimensione dell'array
                             ; sposta il contenuto (dim array) di rl (INT reg) in f8 (FP reg)
       MTCl rl, f8
28
        CVT.D.L f8, f8
                             : (NB divisione) converti f8 (Long INT) in un Double FP e mettilo in f8
29
30
31
       ADD.D fl3, fl3, fl2
                                 ; carica in fl3 la somma di tutti i valori
32
       DIV.D f13, f13, f8
                                ; dividi la somma di tutti i valori per il numero di valori
33
                                 ; la media è in fl3
34
36
   ; analogo alla versione precedente senza ADD.D
38
        MFC1 r3, f0
                            ; all'inizio max == f0
        MFC1 r4, f1
                            ; sposta il contenuto di fl in r4
                            ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
40
       SLT r5, r4, r3
41
                            ; se r5 = 0 allora r4==f1 > max quindi lo salvo
                           ; se sono uguali vai avanti nell'array (loop)
; salva il nuovo max = fl==r4 se r5 = 0
42
       BEQ r3, r4, 11
43
       BEQZ r5, save0
44
        J 11
45
                             ; esci ciclo
46
47
   save0:
        MOV.D f0, f1 ; sovrascrivi (sposta) f0 (max) con f1 (nuovo max)
48
49
   11:
       MFC1 r3. f0
                           : il massimo attuale è in fl4
51
        MFC1 r4. f2
                             ; sposta il contenuto di fl in r4
                             ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti <math>r5 = 0
53
        SLT r5, r4, r3
                             ; se r5 = 0 allora r4==fl > max quindi lo salvo
54
       BEQ r3, r4, 12
55
                             ; se sono uguali vai avanti nell'array (loop)
       BEQZ r5, savel
                            ; salva il nuovo max = fl==r4 se r5 = 0
56
57
        J 12
                             ; esci ciclo
58
   savel:
       MOV.D f0, f2
60
                           ; sovrascrivi (sposta) f0 (max) con f2 (nuovo max)
61
      MFC1 r3, f0
                           ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
64
        MFC1 r4, f3
                             ; sposta il contenuto di fl in r4
65
                            ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
        SLT r5, r4, r3
66
                            ; se r5 = 0 allora r4==fl > max quindi lo salvo
67
        BEQ r3, r4, 13
                            ; se sono uguali vai avanti nell'array (loop)
68
        BEQZ r5, save2
                            ; salva il nuovo max = fl==r4 se r5 = 0
69
        J 13
                             ; esci ciclo
```

```
save2:
         MOV.D f0, f3
                            ; sovrascrivi (sposta) f0 (max) con f3 (nuovo max)
    13:
 74
         MFC1 r3, f0
MFC1 r4, f4
 75
                             ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
 76
                             ; sposta il contenuto di fl in r4
 77
         SLT r5, r4, r3
                             ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
 78
                             ; se r5 = 0 allora r4==fl > max quindi lo salvo
 79
        BEQ r3, r4, 14
                            ; se sono uguali vai avanti nell'array (loop)
 80
         BEQZ r5, save3
                            ; salva il nuovo max = fl==r4 se r5 = 0
 81
         J 14
                             ; esci ciclo
 82
 83 save3:
 84
         MOV.D f0, f4
                            ; sovrascrivi (sposta) f0 (max) con f4 (nuovo max)
 85
 86
     14:
         MFC1 r3, f0 ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
 87
         MFCl r4, f5
 88
                             ; sposta il contenuto di fl in r4
         SLT r5, r4, r3
                             ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
 89
 90
                             ; se r5 = 0 allora r4==fl > max quindi lo salvo
         BEQ r3, r4, 15
 91
                            ; se sono uguali vai avanti nell'array (loop)
         BEQZ r5, save4
 92
                            ; salva il nuovo max = fl==r4 se r5 = 0
 93
         J 15
                             ; esci ciclo
 94
 95 save4:
         MOV.D f0, f5
 96
                            ; sovrascrivi (sposta) f0 (max) con f5 (nuovo max)
 98 15:
         MFC1 r3, f0 ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
MFC1 r4, f6 ; sposta il contenuto di f1 in r4
 99
                            ; sposta il contenuto di fl in r4
100
                             ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti <math>r5 = 0
101
         SLT r5, r4, r3
         BEQ r3, r4, 16
102
         BEQ r3, r4, 16 ; se sono uguali vai avanti nell'array (loop)
BEQZ r5, save5 ; salva il nuovo may = fi
                             ; se r5 = 0 allora r4==fl > max quindi lo salvo
103
104
105
         HALT
106
107 save5:
MOV.D f0, f6 ; sovrascrivi (sposta) f0 (max) con f6 (nuovo max)
109 16:
110
         HALT
```

Perciò le istruzioni ADD.D sono state spostate opportunamente a inizio elaborazione cercando di ridurre gli stalli RAW e successivamente si effettua la divisione per il calcolo della media. Dato che questa operazione è effettuata parallelamente alle istruzioni successive (stesse del codice precedente senza ADD.D) utilizziamo molti meno cicli di clock per l'elaborazione (circa pari alla latenza dell'istruzione DIV.D).



Versione sette

Un'ultima tecnica che può essere utilizzata per ottimizzare il codice è il loop unroll parziale. Per fare ciò reintroduciamo la versione del codice con ciclo (valida per qualsiasi vettore) e utilizziamo un loop unroll di passo 2.

```
; ricerca del massimo in un array di numeri e media dei valori versione 7
       ; loop unrolling parziale passo 2
               .double 4.3, 2.1, 5.7, 8.4, 9.0, 2.7, 8.4
       n: .word 6
                                                       ; contatore dim-1, il primo elemento è il max, mancano 6 confronti (dim=7)
                 .text
       start:
               LW rl, n(r0) ; carica il numero di elementi dell'array -1 in rl (r0 == 0)

DADDI r2, r0, a ; carica in r2 il puntatore al primo elemento dell'array

L.D f0, 0(r2) ; carica in f0 il valore di a[0] == max

DADDI r6, r1, 1 ; salva per la divisione finale (media) la dimensione dell'array

MOV.D f2, f0 ; sposta a[0] in f2 per la somma di tutti i valori

MTCl r6, f3 ; sposta il contenuto (dim array) di r6 (INT reg) in f3 (FP reg)
11
13
               MTCl r6, f3
16
                CVT.D.L f3, f3
                                                        ; (NB divisione) converti f3 (Long INT) in un Double FP e mettilo in f3
18
       loop0:
                                                             ; se mancano meno di due elementi allora r6 = 1 altrimenti r6 = 0 ; esci se il contatore è 0 (NB ultimi due elementi uguali)
19
                SLTI r6, r1, 2
20
                BEQZ rl, end
                                                              ; r6 == 1 allora ultimo loop "a mano"
21
               BNEZ r6. exit
22
                                                     ; carica in fl il valore di a[1] indirizzato da r2 + 8; i += 8 -> r2 contiene l'indirizzo di a[i+1]; decrementa contatore ciclo; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
                L.D fl. 8(r2)
23
24
               DADDI r2, r2, 8
25
                DADDI rl, rl, -1
26
               MFC1 r3, f0
               MFC1 r4, f1
                                                        ; sposta il contenuto di fl in r4
                                                     ; altrimenti errore nell'istruzione BEQ
; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
; se r5 = 0 allora r4==f1 > max quindi lo salvo
28
29
               SLT r5, r4, r3
               ADD.D f2, f2, f1 ; carica in f2 la somma di tutti i valori fino all'i-esimo BEQ r3, r4, loopl ; se sono uguali vai avanti pall'arressimo di tutti i valori fino all'i-esimo per rationale di se sono uguali vai avanti pall'arressimo di tutti i valori fino all'i-esimo per rationale di se sono uguali vai avanti pall'arressimo di tutti i valori fino all'i-esimo per rationale di tutti all'i-esimo per rationale di tutti all'i-esimo per rationale di tutti all'i-esimo per rationale d
30
32
33
                BEQZ r5, save0
                                                                 ; salva il nuovo max = fl==r4 se r5 = 0
       loop1:
                                                  ; carica in fl il valore di a[1] indirizzato da r2 + 8
              L.D fl, 8(r2)
               DADDI r2, r2, 8 ; i += 8 -> r2 contiene l'indirizzo di a[i+l]
DADDI r1, r1, -1 ; decrementa contatore ciclo
39
               MFC1 r3, f0
                                                       ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
40
              MFC1 r4, f1
                                                       ; sposta il contenuto di fl in r4
                                                       ; altrimenti errore nell'istruzione BEO
41
               SLT r5, r4, r3
                                                       ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
42
                                                       ; se r5 = 0 allora r4==fl > max quindi lo salvo
43
               ADD.D f2, f2, f1
                                                       ; carica in f2 la somma di tutti i valori fino all'i-esimo
                                                            ; se sono uguali vai avanti nell'array (loop)
               BEQ r3, r4, loop0
46
               BEQZ r5, savel
                                                               ; salva il nuovo max = fl==r4 se r5 = 0
47
48
              BNEZ rl, loop0
                                                             ; loop finché rl diverso da 0
49
               J end
                                                               ; fine array pari (rl == 0)
50
51
       save0:
             MOV.D f0, f1
                                                     ; sovrascrivi (sposta) max = f0 con f1 = nuovo max
                J loopl
                                                                ; vai avanti nell'arrav
                                     ; il massimo è in f0
55
              MOV.D f0, f1
                                                    ; sovrascrivi (sposta) max = f0 con f1 = nuovo max
57
               J loop0
                                                                : vai avanti nell'arrav
                                                               ; ultimo loop
59
               L.D fl, 8(r2)
                                                       ; carica in fl il valore di a[l] indirizzato da r2 + 8
60
                                                       ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
                MFC1 r3, f0
                                                        ; sposta il contenuto di fl in r4
               MFC1 r4, f1
63
                                                       ; altrimenti errore nell'istruzione BEQ
64
                SLT r5, r4, r3
                                                       ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
                                                        ; se r5 = 0 allora r4==f1 > max quindi lo salvo
65
               ADD.D f2, f2, f1
                                                       ; carica in f2 la somma di tutti i valori fino all'i-esimo
66
                BEQ r3, r4, end
67
                                                       ; se sono uguali vai avanti nell'array (loop)
; salva il nuovo max = fl==r4 se r5 = 0
68
                BEQZ r5, save2
69
                                                                ; fine array
```



```
70
71
72
73
74
end:
75
DIV.D f2, f2, f3; dividi la somma di tutti i valori per il numero di valori
76
HALT; la media aritmetica è in f2
```

Assembly language source file length: 3.594 lines: 7

Le istruzioni prima del ciclo sono le stesse della prima versione mentre all'interno del ciclo questa volta effettuiamo due passaggi anziché uno solo (loop0 e loop1). Se il vettore ha una dimensione pari allora l'esecuzione è analoga a quella vista in precedenza, altrimenti l'ultimo passaggio deve essere fatto fuori dal ciclo. Per verificare ciò prima di effettuare le consuete operazioni a inizio ciclo verifichiamo che il contatore dei cicli rimasti sia minore di 2. In caso affermativo esso sarà 0 oppure 1. Nel primo caso l'elaborazione è terminata e manca solo di effettuare la divisione (salto etichetta end), nel secondo caso è necessario effettuare un ulteriore passaggio prima di fare ciò (salto etichetta exit).



Versione otto

Infine è possibile ottimizzare leggermente questo codice incorporando le istruzioni DADDI per l'aggiornamento di r1 ed r2 in due sole istruzioni anziché quattro e aggiustare le L.D come seque per caricare correttamente di valori.

```
; ricerca del massimo in un array di numeri e media dei valori versione 8
     ; loop unrolling parziale passo 2
    a: .double 4.3, 2.1, 5.7, 8.4, 9.0, 2.7, 8.4
n: .word 6 : contatore dim-1 :1
                             : contatore dim-1, il primo elemento è il max, mancano 6 confronti (dim=7)
         .text
        start:
        LW rl, n(r0)
        L.D f0, 0(r2)
DADDI r6, r1, 1
                            ; salva per la divisione finale (media) la dimensione dell'array
        MOV.D f2, f0
                            ; sposta a[0] in f2 per la somma di tutti i valori
                            ; sposta il contenuto (dim array) di r6 (INT reg) in f3 (FP reg)
15
       MTCl r6, f3
        CVT.D.L f3, f3
                            ; (NB divisione) converti f3 (Long INT) in un Double FP e mettilo in f3
18 loop0:
       SLTI r6, r1, 2
                            ; se mancano meno di due elementi allora r6 = 1 altrimenti r6 = 0
19
        BEQZ rl, end
                            ; esci se il contatore è 0 (NB ultimi due elementi uguali)
20
        BNEZ r6. exit
                            ; r6 == 1 allora ultimo loop "a mano"
21
        DADDI r2, r2, 16
                               ; i += 16 -> r2 contiene l'indirizzo di a[i+2]
        DADDI rl, rl, -2
                                ; decrementa contatore ciclo di passo 2
25
        L.D fl, -8(r2)
26
                                 ; carica in fl il valore di a[i+l] indirizzato da r2 + (16-8)
        MFC1 r3, f0
                            ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
        MFC1 r4, f1
                            ; sposta il contenuto di fl in r4
30
                            ; altrimenti errore nell'istruzione BEQ
                            ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
; se r5 = 0 allora r4==f1 > max guindi lo salvo
31
        SLT r5, r4, r3
32
        ADD.D f2, f2, f1
                             ; carica in f2 la somma di tutti i valori fino all'i-esimo
33
        BEQ r3, r4, loop1 ; se sono uguali vai avanti nell'array (loop)
        BEQZ r5, save0
                             ; salva il nuovo max = fl==r4 se r5 =
37 loop1:
         L.D f1, 0(r2)
                               ; carica in fl il valore di a[i+2] indirizzato da r2 + (16-0)
        MFC1 r3, f0
                         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
                            ; sposta il contenuto di fl in r4
40
        MFC1 r4, f1
41
                        ; altrimenti errore nell'istruzione BEQ
                          ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
; se r5 = 0 allora r4==f1 > max quindi lo salvo
        SLT r5, r4, r3
42
43
        ADD.D f2, f2, f1 ; carica in f2 la somma di tutti i valori fino all'i-esimo BEQ r3, r4, loop0 ; se sono uguali vai avanti nell'array (loop)
45
46
        BEQZ r5, savel
                            ; salva il nuovo max = fl==r4 se r5 = 0
47
        BNEZ rl, loop0
                            ; loop finché rl diverso da 0
48
                            ; fine array pari (rl == 0)
50
51
       MOV.D f0, f1
                            ; sovrascrivi (sposta) max = f0 con f1 = nuovo max
52
53
                            ; vai avanti nell'array
        J loopl
                    ; il massimo è in f0
       MOV.D f0, f1
56
                            ; sovrascrivi (sposta) max = f0 con f1 = nuovo max
57
        J loop0
                            : vai avanti nell'arrav
59
                          ; ultimo loop
                        ; carica in fl il valore di a[n-1] indication; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
        L.D fl, 8(r2)
                            ; carica in fl il valore di a[n-1] indirizzato da r2 + 8
        MFC1 r3, f0
61
62
        MFC1 r4, f1
                        ; altrimenti errore nell'istruzione BEQ
63
64
       SLT r5, r4, r3
                            ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti <math>r5 = 0
                            ; se r5 = 0 allora r4==fl > max quindi lo salvo
        ADD.D f2, f2, f1 ; carica in f2 la somma di tutti i valori fino all'i-esimo
                          ; se sono uguali vai avanti nell'array (loop)
67
       BEQ r3, r4, end
        BEQZ r5, save2
                           ; salva il nuovo max = fl==r4 se r5 = 0
; fine array
68
        J end
69
74 end:
         DIV.D f2, f2, f3 ; dividi la somma di tutti i valori per il numero di valori
75
76
                               : la media aritmetica è in f2
```



Testing e Performance

Il dato sul quale l'algoritmo ha eseguito è un vettore costituito da sette elementi (4.3,2.1,5.7,8.4,9.0,2.7,8.4).

L'algoritmo implementato nelle varie versioni restituisce correttamente il valore massimo e la media aritmetica, che sono rispettivamente 9.0 e 5.8.

Si riportano le varie informazioni statistiche per ogni versione implementata.

Versione	Cycles	Instructions	CPI	RAW Stalls	WAW Stalls	WAR Stalls	Structural Stalls	Branch Taken Stalls	Branch Misprediction Stalls	Code Size
1)	133	79	1,684	18	0	0	6	10	0	88
2)	115	79	1,456	0	0	0	3	10	0	88
3)	109	73	1,493	0	0	0	3	9	0	88
4)	95	63	1,508	0	0	0	3	6	0	276
5)	89	57	1,561	0	0	0	3	6	0	252
6)	68	55	1,236	5	0	0	4	5	0	248
7)	116	80	1,45	0	0	0	3	7	0	180
8)	110	74	1,486	0	0	0	3	7	0	172

Dai risultati ottenuti concludiamo che le versioni di codice più efficienti, in termini di numero di cicli di clock di esecuzione, sono quelle con il loop unroll totale (versioni 4,5,6), le quali però occupano una dimensione maggiore rispetto alle altre versioni oltre ad essere applicabili solamente con vettori di dimensione pari a sette.