1506
UNIVERSITÀ DEGLI STUDI DI URBINO CARLO BO
DISPEA
DIPARTIMENTO DI SCIENZE PURE E APPLICATE

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

1506
UNIVERSITÀ DEGLI STUDI DI URBINO CARLO BO
DISPEA
DIPARTIMENTO DI SCIENZE PURE E APPLICATE

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):

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

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;
```

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;
```

(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
    .data
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)

    .text
start:
    LW r1, n(r0)        ; carica il numero di elementi dell'array -1 in r1 (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
    MTC1 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 mettilo in f3
                            ; stackoverflow
loop:
    BEQZ r1, exit       ; esci se il contatore è zero
    L.D f1, 8(r2)       ; carica in f1 il valore di a[1] indirizzato da r2 + 8
    ADD.D f2, f2, f1        ; carica in f2 la somma di tutti i valori fino all'i-esimo
    DADDI r2, r2, 8     ; r2 += 8 -> r2 contiene l'indirizzo di a[i+1]
    DADDI r1, r1, -1    ; decrementa contatore ciclo
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    BEQ r3, r4, loop    ; se sono uguali vai avanti nell'array (loop)
                        ; altrimenti r3==f0 = max > f1==r4 oppure il contrario
    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
    BEQZ r5, save       ; salva il nuovo max = f1==r4 se r5 = 0
    J loop              ; loop (while)

save:
    MOV.D f0, f1        ; sovrascrivi (sposta) f0 (max) con f1 (nuovo max)
    J loop              ; vai avanti nell'array
                        ; il massimo è in f0
exit:
    DIV.D f2, f2, f3        ; dividi la somma di tutti i valori per il numero di valori
    HALT                ; 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
    .data
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)

    .text
start:
    LW r1, n(r0)        ; carica il numero di elementi dell'array -1 in r1 (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
    MTC1 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 mettilo in f3

loop:
    BEQZ r1, exit       ; esci se il contatore è zero
    L.D f1, 8(r2)       ; carica in f1 il valore di a[1] indirizzato da r2 + 8
    DADDI r2, r2, 8     ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
    DADDI r1, r1, -1    ; decrementa contatore ciclo
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    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)
    BEQZ r5, save       ; salva il nuovo max = f1==r4 se r5 = 0
    J loop              ; loop (while)

save:
    MOV.D f0, f1        ; sovrascrivi (sposta) f0 (max) con f1 (nuovo max)
    J loop              ; vai avanti nell'array
                        ; il massimo è in f0
exit:
    DIV.D f2, f2, f3    ; dividi la somma di tutti i valori per il numero di valori
    HALT                ; la media aritmetica è in f2
```

Versione tre

```
1  ; ricerca del massimo in un array di numeri e media dei valori versione 3
2
3      .data
4  a:  .double 4.3, 2.1, 5.7, 8.4, 9.0, 2.7, 8.4
5  n:  .word 6             ; contatore dim-1, il primo elemento è il max, mancano 6 confronti (dim=7)
6
7      .text
8  start:
9      LW r1, n(r0)        ; carica il numero di elementi dell'array -1 in r1 (r0 == 0)
10     DADDI r2, r0, a     ; carica in r2 il puntatore al primo elemento dell'array
11     L.D f0, 0(r2)       ; carica in f0 il valore di a[0] == max
12     DADDI r6, r1, 1     ; salva per la divisione finale (media) la dimensione dell'array
13     MOV.D f2, f0        ; sposta a[0] in f2 per la somma di tutti i valori
14     MTC1 r6, f3         ; sposta il contenuto (dim array) di r6 (INT reg) in f3 (FP reg)
15     CVT.D.L f3, f3      ; (NB divisione) converti f3 (Long INT) in un Double FP e mettilo in f3
16
17 loop:
18     L.D f1, 8(r2)       ; carica in f1 il valore di a[1] indirizzato da r2 + 8
19     DADDI r2, r2, 8     ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
20     DADDI r1, r1, -1    ; decrementa contatore ciclo
21     MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
22     MFC1 r4, f1         ; sposta il contenuto di f1 in r4
23                         ; altrimenti errore nell'istruzione BEQ
24     SLT r5, r4, r3      ; r5 = 1 se r4==f1 < f0==r3 = max altrimenti r5 = 0
25                         ; se r5 = 0 allora r4==f1 > max quindi lo salvo
26     ADD.D f2, f2, f1    ; carica in f2 la somma di tutti i valori fino all'i-esimo
27     BEQ r3, r4, loop    ; se sono uguali vai avanti nell'array (loop)
28     BEQZ r5, save       ; salva il nuovo max = f1==r4 se r5 = 0
29     BNEZ r1, loop          ; loop finché r1 diverso da 0
30     J exit                 ; esci ciclo
31
32 save:
33     MOV.D f0, f1        ; sovrascrivi (sposta) f0 (max) con f1 (nuovo max)
34     J loop              ; vai avanti nell'array
35                         ; il massimo è in f0
36 exit:
37     DIV.D f2, f2, f3    ; dividi la somma di tutti i valori per il numero di valori
38     HALT                ; 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

    .data
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)

    .text
start:
    LW r1, n(r0)        ; carica il numero di elementi dell'array -1 in r1 (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 r1, 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
    MTC1 r1, f3             ; sposta il contenuto (dim array) di r1 (INT reg) in f3 (FP reg)
    CVT.D.L f3, f3      ; (NB divisione) converti f3 (Long INT) in un Double FP e mettilo in f3

l0:
    L.D f1, 8(r2)       ; carica in f1 il valore di a[1] indirizzato da r2 + 8
    DADDI r2, r2, 8     ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    ADD.D f2, f2, f1    ; carica in f2 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, l1          ; se sono uguali vai avanti nell'array
    BEQZ r5, save0          ; salva il nuovo max = f1==r4 se r5 = 0
    J l1                    ; iterazione successiva

save0:
    MOV.D f0, f1        ; sovrascrivi (sposta) f0 (max) con f1 (nuovo max)
l1:
    L.D f1, 8(r2)       ; carica in f1 il valore di a[1] indirizzato da r2 + 8
    DADDI r2, r2, 8     ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    ADD.D f2, f2, f1    ; carica in f2 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, l2          ; se sono uguali vai avanti nell'array
    BEQZ r5, save1          ; salva il nuovo max = f1==r4 se r5 = 0
    J l2                    ; iterazione successiva

save1:
    MOV.D f0, f1        ; sovrascrivi (sposta) f0 (max) con f1 (nuovo max)

l2:
    L.D f1, 8(r2)       ; carica in f1 il valore di a[1] indirizzato da r2 + 8
    DADDI r2, r2, 8     ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    ADD.D f2, f2, f1    ; carica in f2 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, l3          ; se sono uguali vai avanti nell'array
    BEQZ r5, save2          ; salva il nuovo max = f1==r4 se r5 = 0
    J l3                    ; iterazione successiva

save2:
    MOV.D f0, f1        ; sovrascrivi (sposta) f0 (max) con f1 (nuovo max)
```

```
13:
    L.D f1, 8(r2)       ; carica in f1 il valore di a[1] indirizzato da r2 + 8
    DADDI r2, r2, 8     ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    ADD.D f2, f2, f1    ; carica in f2 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, 14          ; se sono uguali vai avanti nell'array
    BEQZ r5, save3          ; salva il nuovo max = f1==r4 se r5 = 0
    J 14                    ; iterazione successiva

save3:
    MOV.D f0, f1        ; sovrascrivi (sposta) f0 (max) con f1 (nuovo max)

14:
    L.D f1, 8(r2)       ; carica in f1 il valore di a[1] indirizzato da r2 + 8
    DADDI r2, r2, 8     ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    ADD.D f2, f2, f1    ; carica in f2 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, 15          ; se sono uguali vai avanti nell'array
    BEQZ r5, save4          ; salva il nuovo max = f1==r4 se r5 = 0
    J 15                    ; iterazione successiva

save4:
    MOV.D f0, f1        ; sovrascrivi (sposta) f0 (max) con f1 (nuovo max)

15:
    L.D f1, 8(r2)       ; carica in f1 il valore di a[1] indirizzato da r2 + 8
    DADDI r2, r2, 8     ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    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
    BEQZ r5, save5          ; salva il nuovo max = f1==r4 se r5 = 0
    J 16                    ; fine iterazione

save5:
    MOV.D f0, f1        ; sovrascrivi (sposta) f0 (max) con f1 (nuovo max)

16:
    DIV.D f2, f2, f3    ; dividi la somma di tutti i valori per il numero di valori
    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

    .data
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)

    .text
start:
    LW r1, n(r0)        ; carica il numero di elementi dell'array -1 in r1 (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
    L.D f1, 8(r2)           ; carica in f1 il valore di a[1]
    L.D f2, 16(r2)          ; carica in f2 il valore di a[2]
    L.D f3, 24(r2)          ; carica in f3 il valore di a[3]
    L.D f4, 32(r2)          ; carica in f4 il valore di a[4]
    L.D f5, 40(r2)          ; carica in f5 il valore di a[5]
    L.D f6, 48(r2)          ; carica in f6 il valore di a[6]

    DADDI r1, r1, 1     ; 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
    MTC1 r1, f8             ; sposta il contenuto (dim array) di r1 (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

; analogo alla versione precedente
l0:
    MFC1 r3, f0         ; all'inizio max == f0
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
    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
    ADD.D f7, f7, f1        ; carica in f7 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, l1      ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save0      ; salva il nuovo max = f1==r4 se r5 = 0
    J l1                ; esci ciclo
save0:
    MOV.D f0, f1        ; sovrascrivi (sposta) f0 (max) con f1 (nuovo max)

l1:
    MFC1 r3, f0         ; il massimo attuale è in f14
    MFC1 r4, f2             ; sposta il contenuto di f2 in r4
    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
    ADD.D f7, f7, f2        ; carica in f7 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, l2      ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save1      ; salva il nuovo max = f1==r4 se r5 = 0
    J l2                ; esci ciclo

save1:
    MOV.D f0, f2            ; sovrascrivi (sposta) f0 (max) con f2 (nuovo max)

l2:
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f3             ; sposta il contenuto di f3 in r4
    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
    ADD.D f7, f7, f3        ; carica in f7 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, l3      ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save2      ; salva il nuovo max = f1==r4 se r5 = 0
    J l3                ; esci ciclo

save2:
    MOV.D f0, f3            ; sovrascrivi (sposta) f0 (max) con f3 (nuovo max)
```

```
l3:
    MFC1 r3, f0          ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f4             ; sposta il contenuto di f4 in r4
    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
    ADD.D f7, f7, f4        ; carica in f7 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, l4       ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save3       ; salva il nuovo max = f1==r4 se r5 = 0
    J l4                 ; esci ciclo

save3:
    MOV.D f0, f4            ; sovrascrivi (sposta) f0 (max) con f4 (nuovo max)

l4:
    MFC1 r3, f0          ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f5             ; sposta il contenuto di f5 in r4
    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
    ADD.D f7, f7, f5        ; carica in f7 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, l5       ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save4       ; salva il nuovo max = f1==r4 se r5 = 0
    J l5                 ; esci ciclo

save4:
    MOV.D f0, f5            ; sovrascrivi (sposta) f0 (max) con f5 (nuovo max)

l5:
    MFC1 r3, f0          ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f6             ; sposta il contenuto di f6 in r4
    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
    ADD.D f7, f7, f6        ; carica in f7 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, l6       ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save5       ; salva il nuovo max = f1==r4 se r5 = 0
    J l6

save5:
    MOV.D f0, f6            ; sovrascrivi (sposta) f0 (max) con f6 (nuovo max)

l6:
    DIV.D f7, f7, f8        ; dividi la somma di tutti i valori per il numero di valori
                         ; la media è in f7
    HALT                 ; 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

    .data
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)

    .text
start:
    LW r1, n(r0)        ; carica il numero di elementi dell'array -1 in r1 (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
    L.D f1, 8(r2)       ; carica in f1 il valore di a[1]
    L.D f2, 16(r2)      ; carica in f2 il valore di a[2]
    L.D f3, 24(r2)      ; carica in f3 il valore di a[3]
    L.D f4, 32(r2)      ; carica in f4 il valore di a[4]
    L.D f5, 40(r2)      ; carica in f5 il valore di a[5]
    L.D f6, 48(r2)      ; carica in f6 il valore di a[6]

    ADD.D f9, f1, f0        ; NB somma FP latenza 3
    ADD.D f10, f3, f2
    ADD.D f11, f5, f4
    ADD.D f12, f9, f6
    ADD.D f13, f11, f10     ; somma tutti valori tranne f12 = f1+f0+f6

    DADDI r1, r1, 1     ; salva per la divisione finale (media) la dimensione dell'array
    MTC1 r1, f8         ; sposta il contenuto (dim array) di r1 (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

    ADD.D f13, f13, f12     ; carica in f13 la somma di tutti i valori

    DIV.D f13, f13, f8      ; dividi la somma di tutti i valori per il numero di valori
                            ; la media è in f13

; analogo alla versione precedente senza ADD.D
l0:
    MFC1 r3, f0         ; all'inizio max == f0
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
    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
    BEQ r3, r4, l1      ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save0      ; salva il nuovo max = f1==r4 se r5 = 0

    J l1                ; esci ciclo

save0:
    MOV.D f0, f1        ; sovrascrivi (sposta) f0 (max) con f1 (nuovo max)

l1:
    MFC1 r3, f0         ; il massimo attuale è in f14
    MFC1 r4, f2         ; sposta il contenuto di f1 in r4
    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
    BEQ r3, r4, l2      ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save1      ; salva il nuovo max = f1==r4 se r5 = 0
    J l2                ; esci ciclo

save1:
    MOV.D f0, f2        ; sovrascrivi (sposta) f0 (max) con f2 (nuovo max)

l2:
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f3         ; sposta il contenuto di f1 in r4
    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
    BEQ r3, r4, l3      ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save2      ; salva il nuovo max = f1==r4 se r5 = 0
    J l3                ; esci ciclo
```

```
save2:
    MOV.D f0, f3        ; sovrascrivi (sposta) f0 (max) con f3 (nuovo max)

l3:
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f4         ; sposta il contenuto di f1 in r4
    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
    BEQ r3, r4, l4      ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save3      ; salva il nuovo max = f1==r4 se r5 = 0
    J l4                ; esci ciclo

save3:
    MOV.D f0, f4        ; sovrascrivi (sposta) f0 (max) con f4 (nuovo max)

l4:
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f5         ; sposta il contenuto di f1 in r4
    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
    BEQ r3, r4, l5      ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save4      ; salva il nuovo max = f1==r4 se r5 = 0
    J l5                ; esci ciclo

save4:
    MOV.D f0, f5        ; sovrascrivi (sposta) f0 (max) con f5 (nuovo max)

l5:
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f6         ; sposta il contenuto di f1 in r4
    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
    BEQ r3, r4, l6      ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save5      ; salva il nuovo max = f1==r4 se r5 = 0
    HALT

save5:
    MOV.D f0, f6        ; sovrascrivi (sposta) f0 (max) con f6 (nuovo max)
l6:
    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

    .data
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)

    .text
start:
    LW r1, n(r0)        ; carica il numero di elementi dell'array -1 in r1 (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
    MTC1 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 mettilo in f3

loop0:
    SLTI r6, r1, 2          ; se mancano meno di due elementi allora r6 = 1 altrimenti r6 = 0
    BEQZ r1, end            ; esci se il contatore è 0 (NB ultimi due elementi uguali)
    BNEZ r6, exit           ; r6 == 1 allora ultimo loop "a mano"

    L.D f1, 8(r2)       ; carica in f1 il valore di a[1] indirizzato da r2 + 8
    DADDI r2, r2, 8     ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
    DADDI r1, r1, -1    ; decrementa contatore ciclo
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    ADD.D f2, f2, f1    ; carica in f2 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, loop1       ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save0          ; salva il nuovo max = f1==r4 se r5 = 0

loop1:
    L.D f1, 8(r2)       ; carica in f1 il valore di a[1] indirizzato da r2 + 8
    DADDI r2, r2, 8     ; i += 8 -> r2 contiene l'indirizzo di a[i+1]
    DADDI r1, r1, -1    ; decrementa contatore ciclo
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    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)
    BEQZ r5, save1          ; salva il nuovo max = f1==r4 se r5 = 0

    BNEZ r1, loop0          ; loop finché r1 diverso da 0
    J end                   ; fine array pari (r1 == 0)

save0:
    MOV.D f0, f1        ; sovrascrivi (sposta) max = f0 con f1 = nuovo max
    J loop1                 ; vai avanti nell'array
                        ; il massimo è in f0
save1:
    MOV.D f0, f1        ; sovrascrivi (sposta) max = f0 con f1 = nuovo max
    J loop0                 ; vai avanti nell'array

exit:                       ; ultimo loop
    L.D f1, 8(r2)       ; carica in f1 il valore di a[1] indirizzato da r2 + 8
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    ADD.D f2, f2, f1    ; carica in f2 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, end         ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save2          ; salva il nuovo max = f1==r4 se r5 = 0
    J end                   ; fine array
```

```
save2:
    MOV.D f0, f1        ; sovrascrivi (sposta) max = f0 con f1 = nuovo max

end:
    DIV.D f2, f2, f3    ; dividi la somma di tutti i valori per il numero di valori
    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 segue per caricare correttamente di valori.

```
; ricerca del massimo in un array di numeri e media dei valori versione 8
; loop unrolling parziale passo 2

    .data
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)

    .text
start:
    LW r1, n(r0)        ; carica il numero di elementi dell'array -1 in r1 (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
    MTC1 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 mettilo in f3

loop0:
    SLTI r6, r1, 2      ; se mancano meno di due elementi allora r6 = 1 altrimenti r6 = 0
    BEQZ r1, end        ; esci se il contatore è 0 (NB ultimi due elementi uguali)
    BNEZ r6, exit       ; r6 == 1 allora ultimo loop "a mano"

    DADDI r2, r2, 16        ; i += 16 -> r2 contiene l'indirizzo di a[i+2]
    DADDI r1, r1, -2        ; decrementa contatore ciclo di passo 2

    L.D f1, -8(r2)          ; carica in f1 il valore di a[i+1] 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 f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    ADD.D f2, f2, f1    ; carica in f2 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, loop1   ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save0      ; salva il nuovo max = f1==r4 se r5 = 0

loop1:
    L.D f1, 0(r2)           ; carica in f1 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)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    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)
    BEQZ r5, save1      ; salva il nuovo max = f1==r4 se r5 = 0

    BNEZ r1, loop0      ; loop finché r1 diverso da 0
    J end               ; fine array pari (r1 == 0)

save0:
    MOV.D f0, f1        ; sovrascrivi (sposta) max = f0 con f1 = nuovo max
    J loop1             ; vai avanti nell'array
                        ; il massimo è in f0
save1:
    MOV.D f0, f1        ; sovrascrivi (sposta) max = f0 con f1 = nuovo max
    J loop0             ; vai avanti nell'array

exit:                   ; ultimo loop
    L.D f1, 8(r2)       ; carica in f1 il valore di a[n-1] indirizzato da r2 + 8
    MFC1 r3, f0         ; sposta il contenuto di f0 (FP reg)in r3 (INT reg)
    MFC1 r4, f1         ; sposta il contenuto di f1 in r4
                        ; altrimenti errore nell'istruzione BEQ
    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
    ADD.D f2, f2, f1    ; carica in f2 la somma di tutti i valori fino all'i-esimo
    BEQ r3, r4, end     ; se sono uguali vai avanti nell'array (loop)
    BEQZ r5, save2      ; salva il nuovo max = f1==r4 se r5 = 0
    J end               ; fine array

save2:

end:
    DIV.D f2, f2, f3    ; dividi la somma di tutti i valori per il numero di valori
    HALT                ; 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.

Termine della relazione.