Alcune problematiche alla base dello sviluppo di software efficiente

LOOP OPTIMIZATIONS

Esempio:

Assegnati due vettori x ed y di dimensione n ed uno scalare α si vuole effettuare la seguente operazione:

$$y \leftarrow y + \alpha x$$

Moltiplicazione di x per uno scalare α e somma componente per componente del risultato con y

operazione di tipo saxpy

A. Murli

Utilizzo di BLAS 1

Saxpy(n,alpha,x,incx,y,incy)

Come viene
implementata la saxpy in BLAS
?

L'algoritmo implementato in BLAS1 per la saxpy

```
…
nrest := mod(n,4);
for i = 1 to nrest
      y(i):= y(i)+a*x(i);
endfor
n1 := nrest+1;
for i= n1 to n step 4
      y(i):= y(i)+a*x(i);
      y(i+1):=y(i+1)+a*x(i+1);
      y(i+2):=y(i+2)+a*x(i+2);
      y(i+3):=y(i+3)+a*x(i+3);
endfor
…
```

A. Murli

Il nostro algoritmo…

```
for i = 1 to n

      y(i):= y(i)+a*x(i);

endfor
```

Perché questa Differenza ?

A. Murli

Analizziamo il nostro algoritmo

```
for i = 1 to n

      y(i):= y(i)+a*x(i);

endfor
```

In dettaglio…

1. Viene effettuato **n** volte il controllo del ciclo: $i \leq n$

2. Viene effettuato **n** volte l'incremento del contatore del ciclo: `i:=i+1`

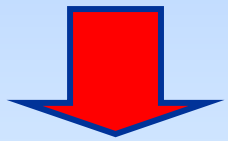
overhead

A. Murli

Overhead

Tutte le istruzioni di un programma del tipo:

- Chiamate a sottoprogrammi
- Aggiornamento del contatore di un ciclo for (i=i+1)
- Verifica della condizione di fine ciclo di iterazione (i < n ?)
- ...



Introducono un overhead rispetto alle operazioni sul vettore y

A. Murli

Dimezziamo la lunghezza del ciclo for…

```
for i = 1 to n step 2

     y(i):=y(i)+a*x(i);
     y(i+1):=y(i+1)+a*x(i+1);

endfor
```

In dettaglio…

1. Viene effettuato **n/2** volte il controllo del ciclo

2. Viene effettuato **n/2** volte l'incremento dell'indice del ciclo: **i:=i+2**

overhead

A. Murli

Riduciamo di $\frac{1}{4}$ la lunghezza del ciclo for…

```
for i = 1 to n step 4
      y(i):=y(i)+a*x(i);
      y(i+1):=y(i+1)+a*x(i+1);
      y(i+2):=y(i+2)+a*x(i+2);
      y(i+3):=y(i+3)+a*x(i+3);
endfor
```

In dettaglio…

1. Viene effettuato **n/4** volte il controllo del ciclo

2. Viene effettuato **n/4** volte l'incremento dell'indice del ciclo: `i:=i+4`

overhead

In generale…

L'overhead e
Si riduce
di un fattore proporzionale
alla nuova lunghezza del ciclo
(profondità dell'unrolling)

Lo "srotolamento" del ciclo consistente nel modificare il **controllo del ciclo** e nel **replicare** opportunamente le istruzioni all'interno del ciclo viene detto

Tecnica di
LOOP UNROLLING

A. Murli

Vantaggi del loop unrolling

1. **Riduzione dell'overhead** del ciclo di iterazione
2. Utilizzo **ottimale** dei processori con architettura pipelined
3. **Riduzione** del numero di trasferimenti fra i vari livelli memoria
4. **Aumento** delle operazioni concorrenti

A. Murli

Svantaggi del loop unrolling

1. **Aumento** della memoria destinata al programma
2. **Perdita** della generalità del codice

A. Murli

In quali casi
è vantaggioso utilizzare
il loop unrolling
?

Esempio 1...:

```
for i=1 to n
    a:= a*x(i) + y(i);
endfor
for j=1 to n
    b:= b*x(j)+ z(j);
endfor
```

Segmento di programma in cui i cicli di for hanno lo stesso numero di iterazioni e non presentano dipendenze tra le operazioni che effettuano (Segmento **loop fusion**)

1. Si effettuano **n** controlli ed n aggiornamenti su i
2. Si effettuano **n** controlli ed n aggiornamenti su j

A. Murli

Esempio 1...:

```
for i=1 to n
    a:= a*x(i) + y(i);
    b:= b*x(i)+ z(i);
endfor
```

1. Si effettuano **n** controlli ed **n** aggiornamenti su i

A. Murli

Esempio 2…:

```
for i=1 to n
    if (flag > 0) then
        y(i):=y(i) + a*x(i);
    else
        y(i):=y(i) - a*x(i);
    endif
endfor
```

Segmento di programma in cui la condizione dell'if non viene modificata all'interno del ciclo di iterazione (**segmento loop invariant**)

1. Viene effettuato **n** volte il controllo

A. Murli

…Esempio 2:

```
if (flag > 0) then
     for i=1 to n
          y(i):=y(i) + a*x(i);
     endfor
else
     for i=1 to n
          y(i):=y(i) - a*x(i);
     endfor
endif
```

1. Viene effettuato **1** volta il controllo

A. Murli

Esempio 3…:

```
for i=1 to n
    if (mod(i,2)== 0) then
        y(i):=y(i) + a*x(i);
    else
        y(i):=y(i) - a*x(i);
    endif
endfor
```

Segmento di programma in cui la condizione dell'if viene modificata all'interno del ciclo di iterazione (**segmento control dependent**)

1. Viene effettuato **n** volte il controllo del ciclo

A. Murli

…Esempio 3:

```
for i=2 to n step 2
        y(i):=y(i) + a*x(i);
endfor
for i=1 to n step 2
        y(i):=y(i) - a*x(i);
endfor
```

1. Non viene effettuato **alcun** controllo del ciclo

A. Murli

Esempio 4…:

```
for i=1 to n
    if(x(i)< 0) then
        y(i):=y(i) - a*x(i);
    else
        y(i):=y(i) + a*x(i);
    endif
endfor
```

Segmento di programma in cui la condizione dell'if viene utilizzata all'interno del ciclo di iterazione (**segmento control dependent**)

1. Viene effettuato **n** volte il controllo del ciclo

A. Murli

…Esempio 4:

```
for i=1 to n
      y(i):=y(i) + a* abs(x(i));
endfor
```

1. Non viene effettuato **alcun** controllo del ciclo

A. Murli

Esempio 5…:

```
sum:=0;
for i=1 to n
        sum:=sum + y(i)*x(i);
endfor
```

Conviene modificare il passo del ciclo?

COME ?

A. Murli

...Esempio 5:

```
sum:=0;
for i=1 to n step 2
      sum:=sum + y(i)*x(i);
      sum:=sum + y(i+1)*x(i+1);
endfor
```

1. Si effettuano **n/2** controlli ed n/2 aggiornamenti su i

Le operazioni non possono essere eseguite in modo concorrente !!!
(dipendenza tra le operazioni nel loop)

Murli

…Esempio 5:

```
s1:=0;
s2:=0;
for i=1 to n step 2
      s1:=s1 + y(i)*x(i);
      s2:=s2 + y(i+1)*x(i+1);
endfor
sum:= s1 + s2;
```

1. Si effettuano **n/2** controlli ed n/2 aggiornamenti su i

Le operazioni ora possono essere eseguite in modo concorrente !!

...Esempio 5:

OSSERVAZIONI

1. Si è utilizzata la proprietà associativa della somma: a+b+c+d=(a+b)+(c+d)

2. La somma floating-point non gode della proprietà associativa !!

A. Murli

Esempio 6…:

```
for j = 1 to m
    for i = 1 to n
        y(i,j) := a*x(i,j)+z(i);
    endfor
endfor
```

Cicli di iterazione innestati

Si può modificare il passo del ciclo esterno

(Unrolling Outer loop)

1. Per **m** volte il vettore **z** passa da un livello più basso di memoria in un registro

2. Si effettuano **m** controlli ed **m** aggiornamenti su j

A. Murli

…Esempio 6:

```
for j = 1 to m step 2
    for i = 1 to n
        y(i,j) := a*x(i,j)+z(i);
        y(i,j+1) := a*x(i,j+1)+z(i);
    endfor
endfor
```

1. Per **m/2** volte il vettore **z** passa da un livello più basso di memoria in un registro

2. Si effettuano **m/2** controlli ed m/2 aggiornamenti su j

A. Murli

Esempio 7…:

```
for i = 1 to m
    for j = 1 to n

        y(i,j):= a*y(i,j)+b;

    endfor
endfor
```

A. Murli

Cicli di iterazione innestati

L'ordine dei cicli può essere permutato (loop interchange)

1. Per **m×n** volte il vettore **y** passa da un livello più basso di memoria in un registro

…Esempio 7:

```
for i = 1 to m
 for j = 1 to n

   y(i,j):= a*y(i,j)+b;

 endfor
endfor
```

C

```
for j = 1 to n
 for i = 1 to m

   y(i,j):= a*y(i,j)+b;

 endfor
endfor
```

Fortran

Scorrimento dell'array bidimensionale y riga per riga

Scorrimento dell'array bidimensionale y colonna per colonna

A seconda delle caratteristiche del linguaggio di programmazione può essere più utile per applicare il Loop unrolling procedere per righe o per colonne

Esempio 8…:

```
for i = 1 to l
    for j = 1 to m
        for k = 1 to n

            c(i,j):= c(i,j)+ a(i,k)*b(k,j);

        endfor
    endfor
endfor
```

A. Murli

Cicli di iterazione innestati

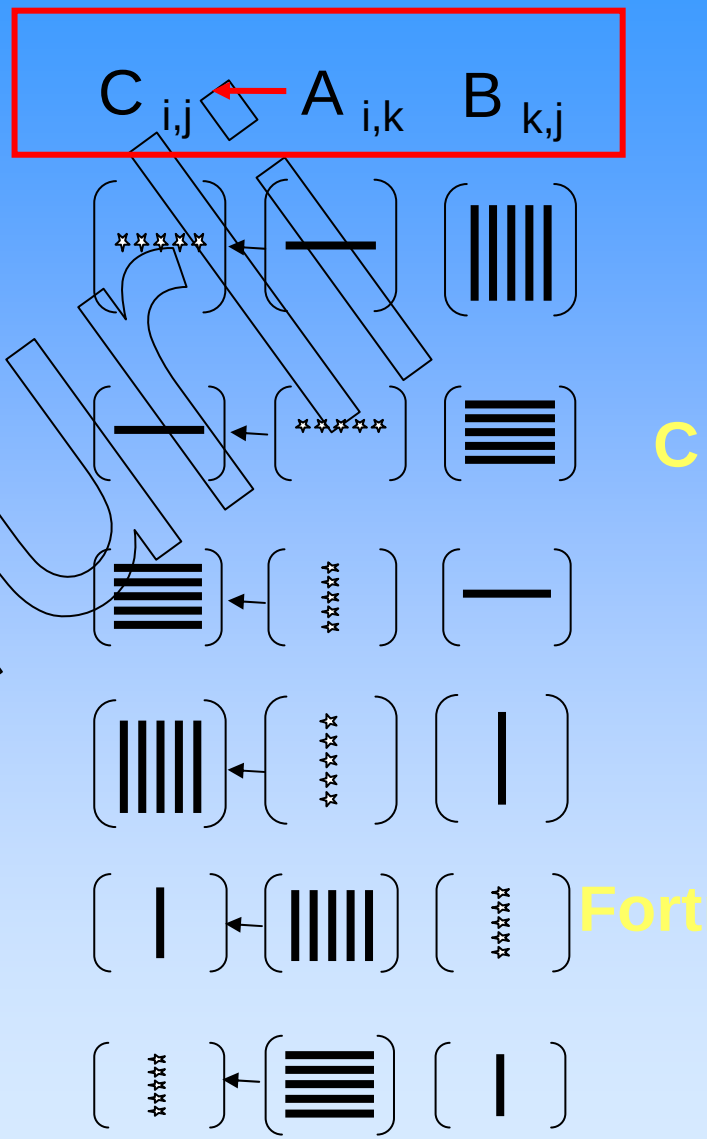
1. Per **l×m×n** volte il vettore c passa da un livello più basso di memoria in un registro

...Esempio 8:

```
for i=1 to l
 for j=1 to m
  for k=1 to n

   c(i,j):= c(i,j)+ a(i,k)*b(k,j);

  endfor
 endfor
endfor
```



Per modificare questo ciclo c'è la possibilità di scegliere fra 6 permutazioni degli indici **i,j,k**

Per ottenere i vantaggi del loop unrolling bisogna tener conto del linguaggio di programmazione e dell'ambiente di calcolo in cui si opera.

A. Murli

Conclusioni…

Il loop unrolling ed altre tecniche di loop optimizations vengono realizzate

da tutti i compilatori in maniera automatica con l'aggiunta di opportune opzioni al comando di compilazione

A. Murli