Programmazione Funzionale e Parallela

Corso di Laurea in Ingegneria Informatica e Automatica - A.A. 2019-2020

Esercitazione del 13 maggio 2018

Istruzioni per l'esercitazione:

- Aprite il form di consegna in un browser e loggatevi con le vostre credenziali `uniroma1`.
- Scaricate e decomprimete sulla scrivania il codice dell'esercitazione. Vi sarà una sotto-directory separata per ogni esercizio. Non modificate in alcun modo i programmi di test `*Main.scala` e `*main.c`.
- Rinominare la directory chiamandola `cognome.nome`.
- **Finiti gli esercizi**, e non più tardi della fine della giornata:
  - **zippate la directory di lavoro** in `cognome.nome.zip` (`zip -r cognome.nome.zip cognome.nome/`).
- **Per consegnare**:
  - inserite nel form di consegna come autovalutazione il punteggio di ciascuno dei test forniti (inserite zero se l'esercizio non è stato svolto, non compila, o dà errore di esecuzione)
  - fate **upload** del file `cognome.nome.zip`.

Per maggiori informazioni fate riferimento al regolamento delle esercitazioni.

Esercizio 1 (shear orizzontale di 45 gradi di un'immagine - OpenCL)

Il seguente esempio illustra l'applicazione di uno shear orizzontale con angolo di shear pari a 45° a un'immagine a 256 toni di grigio (0=nero, 255=bianco):

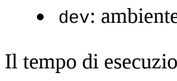
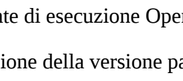

Immagine di input Immagine di output

Scrivere nel file `shear45.c` una versione OpenCL `shear45` della seguente funzione `shear45_host` che converte un'immagine di input in una di output ottenuta mediante uno shear di 45 gradi:

```
void shear45_host(unsigned char* in, unsigned char** out,
                  int h, int w, int* oh, int* ow,
                  unsigned char gray, double* t) {

    int x, y;

    // set size of output matrix
    *ow = w+h;
    *oh = h;

    // allocate output matrix
    *out = malloc((*oh)*(*ow)*sizeof(unsigned char));
    if (*out == NULL)
        clut_panic("failed to allocate output matrix on host");

    // get initial time
    double start = clut_get_real_time();

    // set pixels of output image
    for (y=0; y < *oh; ++y)
        for (x=0; x < *ow; ++x)
            (*out)[IDX(x, y, w+h)] = (x < y || x >= w+y) ? gray : in[IDX(x-y, y, w)];

    // get elapsed time
    *t = clut_get_real_time() - start;
}
```

La funzione `shear45_host` ha i seguenti parametri:

- `in`: puntatore all'immagine di input rappresentata in formato row-majow (cioè con le righe disposte consecutivamente in memoria)
- `out`: indirizzo di una variabile che conterrà il puntatore all'immagine di output rappresentata in formato row-major
- `h` e `w`: altezza e larghezza in pixel dell'immagine di input
- `oh` e `ow`: indirizzi di variabili che conterranno l'altezza e la larghezza in pixel dell'immagine di output
- `gray`: colore dello sfondo nell'immagine di output
- `t`: indirizzo di una variabile che conterrà il tempo richiesto per inizializzare l'immagine di output

La funzione `shear45` da realizzare ha gli stessi parametri di `shear45_host`, con l'aggiunta di:

- `dev`: ambiente di esecuzione OpenCL di tipo `clut_device*` da usare per il calcolo

Il tempo di esecuzione della versione parallela deve essere quello richiesto dall'operazione `clEnqueueNDRangeKernel`.

Usare il main di prova nella directory di lavoro digitando `make` per compilare e `./shear45` per eseguire il programma. Il risultato sarà presente nella directory `results`, ottenuto a partire dall'immagine di input nella directory `images`.

Inserire nel form di consegna 1 (test passato) se il risultato è corretto e 0 altrimenti.

Suggerimento: ispirarsi all'esempio di codice OpenCL su immagini contenuto nella directory `esempio`.

Esercizio 2 (verifica se un albero binario è di ricerca)

Un albero binario è di ricerca se il valore contenuto in ogni nodo è compreso tra il massimo del suo sottoalbero sinistro e il minimo del suo sottoalbero destro. Usare <= e >= per fare il test. Si richiede di implementare un metodo Scala `treeTest` che restituisce true se e solo se l'albero su cui viene applicato è di ricerca:

```
E2.scala

sealed abstract class Tree {
    def treeTest:Boolean = ??? // completare il metodo
}

// albero non vuoto
case class T(l:Tree, e:Int, r:Tree) extends Tree

// albero vuoto
case class E() extends Tree
```

Compilare con `scalac E2.scala E2Main.scala` ed eseguire il programma di prova con `scala E2Main`.

Domande

Rispondere alle seguenti domande con vero=V o falso=F.

Domanda 1

Il seguente frammento di codice Scala genera errori di compilazione:

```
def f(x:Int) = x
val v = f
```

Domanda 2

Il seguente frammento di codice Scala genera errori di compilazione:

```
def f[T](l:List[T]):List[T] = {
    l.sorted
}
```

Domanda 3

Dimezzare il tempo di esecuzione di una porzione di codice che richiede la metà del tempo di esecuzione di un programma porta a uno speedup complessivo pari a 2x per quel programma.

Domanda 4

Il tipo vettoriale `__m128i` permette di fare 16 operazioni in parallelo su valori di 8 bit.

Domanda 5

La vettorizzazione è un tipo di computazione MIMD secondo la tassonomia di Flynn.

Domanda 6

Uno dei problemi principali nella manutenzione di un data center è tenerne bassa la temperatura con un opportuno sistema di raffreddamento.

Domanda 7

In Scala il concetto di metodo e quello di funzione sono equivalenti.

Domanda 8

Il seguente metodo Scala viene compilato correttamente e calcola una copia della lista in ingresso:

```
def f[T](l:List[T]) = l match {
    case Nil => Nil
    case h::Nil => List(h)
    case h::t => h::f(t)
}
```

Domanda 9

La seguente funzione SSE calcola la somma dei numeri di un vettore di int di lunghezza arbitraria:

```
#include <immintrin.h>

int array_sum(int *v, int n) {
    int i, res[4];
    __m128i s = _mm_set_epi32(0,0,0,0);
    for (i=0; i+3<n; i+=4) {
        __m128i vv = _mm_loadu_si128((const __m128i*)(v+i));
        s = _mm_add_epi32(s, vv);
    }
    _mm_storeu_si128((__m128i*)res, s);
    return res[0]+res[1]+res[2]+res[3];
}
```

Domanda 10

In OpenCL la memoria host e quella device risiedono sempre in memorie fisiche distinte.

Valid XHTML 1.0 Transitional Valid CSS!