Nome e Cognome _____ Matricola _____

Corso di *High Performance Computing*
Ingegneria e Scienze Informatiche—Università di Bologna

Prova Scritta, 6/2/2020

- La prova dura 60 minuti
- Durante la prova non è consentito consultare libri, appunti o altro materiale.
- Non è consentito l'uso di dispositivi elettronici (ad esempio, cellulari, tablet...), né interagire in alcun modo con gli altri studenti pena l'esclusione dalla prova, che potrà avvenire anche dopo il termine della stessa.
- Le risposte devono essere scritte **a penna** su questi fogli, in modo **leggibile**. Le parti illeggibili o scritte a matita saranno ignorate.
- Eventuali altri fogli possono essere utilizzati per la brutta copia ma non verranno valutati.
- Si risponda in modo sintetico ma il più possibile esaustivo.
- I voti saranno pubblicati su AlmaEsami e ne verrà data comunicazione all'indirizzo mail di Ateneo (@studio.unibo.it).
- I voti restano validi fino alla sessione d'esame di **settembre 2020** inclusa. Dopo tale data tutti i voti in sospeso saranno persi.

NON SCRIVERE NELLA TABELLA SOTTOSTANTE

D. 1	D. 2	D. 3	D. 4
/ 8	/ 8	/ 8	/ 8

Domanda 1.

1. Che cosa si intende per *sbilanciamento del carico* riferito alla programmazione parallela?
2. Illustrare in quali modi è possibile porre rimedio allo sbilanciamento del carico.

Domanda 2. Si consideri il seguente ciclo:

```
#define N ...un qualche valore "grande"...
double a[N];
const double f = 1.000001;
double val = 1.0;

1. for (i=0; i<N; i++) {
2.    a[i] = val;
3.    val = val * f
4. }
```

Tralasciando ogni considerazione di natura numerica (quindi supponendo che non si verifichino overflow alla riga 3, ad esempio), e supponendo di lavorare su una architettura a memoria condivisa (es., usando OpenMP), rispondere alle seguenti domande:

1. Dire se nel ciclo "for" sono presenti *loop-carried dependencies*;
2. In caso affermativo, si spieghi nel modo più preciso possibile come sia possibile rimuoverle, anche ristrutturando il codice. **Non è richiesto di fornire codice funzionante** (anche perché la soluzione potrebbe essere laboriosa).

Risposta. È presente una loop-carried dependency sulla variabile "val", il cui valore viene modificato ad ogni iterazione del ciclo e utilizzato all'iterazione successiva.
La dipendenza non può essere rimossa riarrangiando il corpo del ciclo, né dichiarando la variabile val "firstprivate" (in quest'ultimo caso infatti i thread si ritroverebbero tutti con una variabile locale inizializzata a 1.0 che utilizzano per iniziare a riempire i valori della porzione di array loro assegnata). Occorre invece esaminare cosa calcola il ciclo e individuare una strategia di parallelizzazione adeguata.
È facile rendersi conto che al termine del ciclo $a[i] = f^i$, per ogni $i = 0, \ldots N - 1$. Pertanto, una prima soluzione potrebbe essere quella di riscrivere il ciclo come:

```
for (i=0; i<N; i++) {
   a[i] = pow(f, i);
}
```

e osservare come non siano presenti loop-carried dependencies. Tuttavia, questa versione non sarebbe molto efficiente dato che il tempo necessario a calcolare `pow(f, i)` dipenderà da i (qualcuno ha correttamente segnalato come il calcolo della potenza intera possa essere effettuato in tempo logaritmico su i), per cui questa soluzione potrebbe non essere work-efficient.
La soluzione work-efficient consiste nell'osservare che il ciclo calcola una *scan esclusiva* utilizzando l'operatore prodotto (anziché la somma). Una implementazione della primitiva *scan* usando OpenMP è stata proposta in laboratorio (si veda l'esercizio 6 del primo lab OpenMP, in cui si usa la somma; l'uso del prodotto è del tutto analogo); come da testo del problema, non era richiesto di riportare i dettagli dell'implementazione della scan vista in laboratorio.

Domanda 3.

1. Si spieghi che cosa si intende per computazione di tipo *stencil*.
2. Discutere i problemi connessi all'implementazione di computazioni di tipo *stencil* su architetture a memoria distribuita.

Domanda 4. Spiegare riga per riga il seguente programma, che compila ed esegue correttamente. Si assuma di usare il compilatore GCC, che l'architettura target sia un processore Intel che supporta tutte le estensioni SIMD, e che una variabile di tipo `int` occupi 4 byte (32 bit). Le righe da commentare sono solo quelle numerate. In particolare, si spieghi quali valori contiene la variabile `tmp` al termine della riga 4, e cosa stampa la riga 6.

```
      #include <stdio.h>

1.    typedef int vec_t __attribute__((vector_size(16)));

      int main( void )
      {
2.        vec_t va = {10, 20, 30, 40};
3.        vec_t vb = { 2, 55, 47,  8};
4.        vec_t tmp = (va > vb);
5.        vec_t r = (tmp & va) | (~tmp & vb);
6.        printf("%d %d %d %d\n", r[0], r[1], r[2], r[3]);
          return 0;
      }
```