Nome e Cognome _____ Matricola _____

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

Prova Scritta, 19/2/2019

- 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.
- 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 2019** 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. Illustrare i concetti di *speedup* ed *efficienza* di un programma parallelo, spiegando come si calcolano, quali proprietà hanno e che cosa significano.

Domanda 2. Si consideri il seguente frammento di codice in linguaggio C:

```
#define N 1000000
int a[N], b[N];
int i, s = 0, r = 1;

/* … */

for (i=1; i<N; i++) {
   a[i] = f(b[i]);
   s = s + a[i];
   r = r * b[i-1];
}
```

Si riscriva il codice in modo da sfruttare il parallelismo OpenMP. Si assuma che la funzione `f()` ritorni un risultato che dipende solo dal parametro, senza accedere né modificare variabili globali.

Risposta. Le uniche loop-carried dependencies presenti nel ciclo sono quelle relative alle variabili s ed r, che possono comunque essere rimosse tramite due clausole reduction:

```
#pragma omp parallel for reduction(+:s) reduction(*:r)
for (i=1; i<N; i++) {
   a[i] = f(b[i]);
   s = s + a[i];
   r = r * b[i-1];
}
```

(Come era stato detto a lezione, è possibile usare più clausole reduction nello stesso ciclo; di conseguenza non è necessario suddividere il ciclo). Si noti che non era necessario allineare le iterazioni dei loop; l'unica dipendenza di tipo RAW è tra la prima e la seconda istruzione nel corpo del loop, ma tale dipendenza non riguarda iterazioni diverse (prima si modifica a[i], e subito dopo si legge il valore di a[i]). Con la direttiva precedente, tutte le variabili sono shared di default, il che è corretto in questo caso.

Domanda 3. Commentare riga per riga il seguente programma, che compila ed esegue correttamente. Il programma viene compilato usando il compilatore GCC; assumere che l'architettura target sia un processore Intel che supporta tutte le estensioni SIMD, e in cui una variabile di tipo `int` occupi 4 byte (32 bit). Le righe da commentare sono quelle numerate; i numeri di riga NON fanno parte del programma.
Che valore viene stampato alla riga 5?

```
      #include <stdio.h>

1     typedef int v4i __attribute__((vector_size(16)));

      int main( void )
      {
2         v4i va = {1, 2, 3, 4};
3         v4i vb = {5, 6, 7, 8};
4         v4i vc = (2*va) + vb;
5         printf("%d\n", vc[0] + vc[1] + vc[2] + vc[3]);
          return 0;
      }
```

Domanda 4. Descrivere la gerarchia di memoria di una moderna GPU programmabile mediante CUDA; ogni tipo di memoria menzionato deve essere descritto in modo sintetico ma il più possibile esaustivo.