## Esercitazione del 13 maggio

**Domanda 1**

Il seguente frammento di codice Scala genera errori di compilazione:

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

**VERO**

**Domanda 2**

Il seguente frammento di codice Scala genera errori di compilazione:

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

**VERO**

**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.

**FALSO**

**Domanda 4**

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

**VERO**

**Domanda 5**

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

**FALSO**

**Domanda 6**

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

**VERO**

**Domanda 7**

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

**FALSO**

**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)
}
```

**FALSO**

**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];
}
```

**FALSO**

**Domanda 10**

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

**FALSO**

## Esercitazione del 20 maggio

**Domanda 1**

Se si riduce di un fattore 2 il tempo di esecuzione di una porzione di codice che prende il 90% del tempo di esecuzione si ottiene uno speedup complessivo pari a 1.33.

**FALSO**

**Domanda 2**

Se l è una lista di Int, il seguente frammento di codice Scala restituisce un valore di tipo Any:

```
l match {
   case Nil => List(0)
   case h::t => h
}
```

**VERO**

**Domanda 3**

Secondo la tassonomia di Flynn, un computer convenzionale è un'istanza del modello MISD.

**FALSO**

**Domanda 4**

Un codice computation-intensive riesce a sfruttare meglio il parallelismo di un codice data-intensive.

**VERO**

**Domanda 5**

La classe Option[T] viene usata quando serve modellare nel dominio di output di una funzione l'assenza di un valore.

**VERO**

**Domanda 6**

Una classe case fornisce automaticamente un metodo compareTo che consente di confrontare se un elemento è maggiore di un altro.

**FALSO**

**Domanda 7**

Una classe case fornisce automaticamente un metodo equals che consente di confrontare l'uguaglianza di due elementi della classe.

**VERO**

**Domanda 8**

Un metodo implicit serve tipicamente per convertire un oggetto da un tipo a un altro.

**VERO**

**Domanda 9**

In Scala non è possibile avere variabili static in una classe come in Java.

**VERO**

**Domanda 10**

L'espressione Scala a zip b combina gli elementi di a con quelli corrispondenti di b, purché abbiano lo stesso tipo.

**FALSO**

**Domanda 11**

L’espressione l.sum funziona su qualsiasi tipo di lista l.

**FALSO**

**Domanda 12**

Una funzione che restituisce un’altra funzione è una funzione di ordine inferiore.

**FALSO**

**Domanda 13**

Una chiusura è una funzione in cui il corpo fa riferimento ad almeno un argomento che non appare nella lista dei parametri.

**VERO**

**Domanda 14**

L’espressione:

```
for {
        i <- 0 until 10
} yield i
```

denota una collezione.

**VERO**

**Domanda 15**

La funzione apply non può essere definita in un object.

**FALSO**

**Domanda 16**

La funzione apply consente di applicare argomenti tra parentesi direttamente su un riferimento a oggetto istanza di una classe.

**VERO**

**Domanda 17**

Una classe case definisce automaticamente un metodo apply.

**VERO**

**Domanda 18**

Un trait in Scala è l’analogo di una classe Java.

**FALSO**

**Domanda 19**

L’espressione if (true) 3.14 else false ha come tipo Any.

**FALSO**

**Domanda 20**

L’espressione (x:Int) => 2*x+y è una chiusura, assumendo che y sia visibile nel punto in un cui appare.

**VERO**

**Domanda 21**

In OpenCL, un NDRange è una griglia logica su cui vengono istanziati work item che sono a loro volta istanze di kernel.

**VERO**

**Domanda 22**

In OpenCL, un kernel è una particolare funzione scritta in un dialetto del C che specifica il comportamento parallelo di un’applicazione.

**VERO**

**Domanda 23**

In OpenCL, non è necessario copiare i dati dalla memoria host alla memoria device e viceversa se il device in questione è la CPU.

**FALSO**

**Domanda 24**

In OpenCL, il codice compilato di un kernel può essere generato a tempo di esecuzione.

**VERO**

**Domanda 25**

In AVX, è sempre possibile caricare dati da memoria in un packed integer __m256i usando _mm256_load_si256.

**FALSO**

**Domanda 26**

In SSE è possibile sommare due vettori contenenti 4 interi di 32 bit ciascuno con una singola istruzione _mm_add_epi16.

**FALSO**

**Domanda 27**

Il loop unrolling è una tecnica di ottimizzazione che viene usata in modo preparatorio per agevolare la vettorizzazione.

**VERO**

**Domanda 28**

In OpenCL la global dimension non deve essere necessariamente un multiplo della local dimension

**FALSO**

**Domanda 29**

Con gli opportuni livelli di ottimizzazione, compilatori come gcc emettono automaticamente codice SSE per ottimizzare i programmi.

**VERO**

**Domanda 30**

E’ necessario includere l'header intrin.h per programmare in SSE/AVX.

**FALSO**

## Versione Stampabile ottimizzata

1 Il seguente frammento di codice Scala genera errori di compilazione: V

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

2 Il seguente frammento di codice Scala genera errori di compilazione: V

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

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. F

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

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

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

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

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

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

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

```
#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];
}
```

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

1 Se si riduce di un fattore 2 il tempo di esecuzione di una porzione di codice che prende il 90% del tempo di esecuzione si ottiene uno speedup complessivo pari a 1.33. F

2 Se l è una lista di Int, il seguente frammento di codice Scala restituisce un valore di tipo Any: V

```
l match {
   case Nil => List(0)
   case h::t => h
}
```

3 Secondo la tassonomia di Flynn, un computer convenzionale è un'istanza del modello MISD. F

4 Un codice computation-intensive riesce a sfruttare meglio il parallelismo di un codice data-intensive. V

5 La classe Option[T] viene usata quando serve modellare nel dominio di output di una funzione l'assenza di un valore. V

6 Una classe case fornisce automaticamente un metodo compareTo che consente di confrontare se un elemento è maggiore di un altro. F

7 Una classe case fornisce automaticamente un metodo equals che consente di confrontare l'uguaglianza di due elementi della classe. V

8 Un metodo implicit serve tipicamente per convertire un oggetto da un tipo a un altro. V

9 In Scala non è possibile avere variabili static in una classe come in Java. V

10 L'espressione Scala a zip b combina gli elementi di a con quelli corrispondenti di b, purché abbiano lo stesso tipo.
F11 L'espressione l.sum funziona su qualsiasi tipo di lista l. F

12 Una funzione che restituisce un'altra funzione è una funzione di ordine inferiore. F

13 Una chiusura è una funzione in cui il corpo fa riferimento ad almeno un argomento che non appare nella lista dei parametri. V

14 L'espressione:

```
for {
        i <- 0 until 10
} yield i
```

denota una collezione. V

15 La funzione apply non può essere definita in un object. F

16 La funzione apply consente di applicare argomenti tra parentesi direttamente su un riferimento a oggetto istanza di una classe. V

17 Una classe case definisce automaticamente un metodo apply. V

18 Un trait in Scala è l'analogo di una classe Java. F

19 L'espressione if (true) 3.14 else false ha come tipo Any. F

20 L'espressione (x:Int) => 2*x+y è una chiusura, assumendo che y sia visibile nel punto in un cui appare. V

21 In OpenCL, un NDRange è una griglia logica su cui vengono istanziati work item che sono a loro volta istanze di kernel. V

22 In OpenCL, un kernel è una particolare funzione scritta in un dialetto del C che specifica il comportamento parallelo di un'applicazione. V

23 In OpenCL, non è necessario copiare i dati dalla memoria host alla memoria device e viceversa se il device in questione è la CPU. F

24 In OpenCL, il codice compilato di un kernel può essere generato a tempo di esecuzione V

25 In AVX, è sempre possibile caricare dati da memoria in un packed integer __m256i usando _mm256_load_si256. F

26 In SSE è possibile sommare due vettori contenenti 4 interi di 32 bit ciascuno con una singola istruzione _mm_add_epi16. F

27 Il loop unrolling è una tecnica di ottimizzazione che viene usata in modo preparatorio per agevolare la vettorizzazione V

28 In OpenCL la global dimension non deve essere necessariamente un multiplo della local dimension F

29 Con gli opportuni livelli di ottimizzazione, compilatori come gcc emettono automaticamente codice SSE per ottimizzare i programmi. V

30 E' necessario includere l'header intrin.h per programmare in SSE/AVX. F