F

Fondamenti di Informatica L-A Prof. Paolo Torroni

Prima prova parziale 5 Novembre 2005

1. [3.5]

Si consideri l'architettura di un Personal Computer.

A [**V**] Il registro Flag (o PSW) contiene informazioni riguardanti l'ultima operazione eseguita dalla ALU.

B [**V**] Ad ogni ciclo il registro IR contiene l'istruzione da eseguire nello stesso ciclo.

C [**V**] La memoria centrale puo' contenere i dati del programma correntemente in esecuzione.

D [**V**] L'accesso alla cache di secondo livello e' piu' veloce rispetto all'accesso alla RAM.

E [**F**] La memoria di massa e' volatile.

2. [3.5]

Linguaggi di programmazione.

A [**V**] Un programma espresso in linguaggio macchina e' una sequenza di bit.

B [**V**] Un compilatore traduce un programma scritto con un linguaggio di alto livello in uno scritto con un linguaggio a basso livello, ma non viceversa.

C [**V**] Lo sviluppo di programmi in linguaggio assembler necessita dell'uso di un traduttore.

D [**F**] Un compilatore per il linguaggio C e' un programma che e' indipendente dall'architettura HW/SW del computer utilizzato.

E [**V**] L'esecuzione di un programma compilato è tipicamente più veloce dell'esecuzione dello stesso programma interpretato.

3. [4]

Si consideri il seguente programma C:

```
#include <stdio.h>

main()
{ int A=2;
  float B=A--;
  char C=A;

  if (--B)
      if(--A)
          C='B';
      else
          C='A';
  else  C=A;
  /* punto 1 */
  B=(A+=3, A--); /*istruzione 1*/
  /* punto 2 */
  { float num;
    num=(C>'B'? B--: B/A);
    /* punto 3*/
  }
  /*punto 4*/
}
```

A [**F**] Al punto 1 la variabile C assume il valore 'B'.

B [**V**] L'istruzione 1 e' equivalente alla sequenza di istruzioni: `A=A+3; B=A; --A;`

C [**V**] Al punto 2 la variabile A ha valore 2.

D [**V**] Al punto 2 le variabili A e B hanno valore diverso.

E [**V**] Al punto 3 la variabile num ha valore 1.5.

4. [4]

Si consideri il seguente programma C:

```
#include <stdio.h>

main()
{int A; float B; char C, ch;
 A=4; B=-2; C='C'; ch='D';

 A=( A%(int)B ?++A: ch-C); /* istruzione 1 */
 B=A++/B; /* istruzione 2 */
 ch=C+=1; ch-=C;/* istruzione 3 */
}
```

A [**V**] Immediatamente dopo l'esecuzione dell'*istruzione 2*, la variabile A ha il valore 2.

B [**F**] Immediatamente dopo l'esecuzione dell'*istruzione 3*, la variabile C ha valore `'C'`.

C [**F**] L'operatore di casting nell'*istruzione 1* e' ininfluente.

D [**F**] Il programma puo' generare errori durante l'esecuzione.

E [**V**] Immediatamente dopo l'esecuzione dell'*istruzione 2*, la variabile B ha il valore -0.5.

5. [4]

Si consideri il seguente programma C:

```
#include <stdio.h>

main()
{int x[4]={0, 3, 4, 8};
 int j, A, B=0, C;
 for(j=3; j>=0; j--) /*istruzione 1*/
   switch(x[j]%3-1)  /*istruzione 2*/
     {case 0:     x[j]--;
      case 1:     A=x[j]; break;
      case 2:     C=0; break;
      default:    B=--A;
     }
 C=x[0]+x[2] - B; /* istruzione 3*/
}
```

A [**V**] Immediatamente dopo il ciclo for, la variabile B ha il valore 1.
B [**V**] L'esecuzione del ciclo for provoca 4 iterazioni.
C [**F**] Immediatamente dopo il ciclo for, la variabile C vale 0.
D [**V**] Il blocco di istruzioni corrispondente al `case 1:` viene eseguito 2 volte.
E [**F**] Immediatamente dopo l'istruzione 3, la variabile C ha il valore 3.

6. [4]

Si consideri il seguente programma C:

```
#include <stdio.h>
#define N 8

main()
{typedef float V1[N];
 typedef char V2[N];
 V1 x={0,0.5,1,1.5,2.0,2.5,3.0,3.5};
 V2 y={'C','i','a','o',0,1,2,3};
 int i;
 x[N-1]=0;
 for (i=1; i<N; i+=i)
   x[i]=y[N-i]/i;
 /* punto 1 */
 if (x[--i]?1:x[0])
   printf("Ciao!\n");
 else
   {i=x[0];
    for (;i<=4;) printf("%c", y[i++]);
   }
 /* punto 2*/
}
```

A [**V**] Al punto 1: il valore di x[1] e' 3.
B [**F**] Nel programma non vengono usati tipi di dato non primitivi.
C [**F**] Il programma contiene un errore di sintassi.
D [**V**] Al punto 2: x[i-1] ha valore 0.0.
E [**F**] Al punto 1: il valore di x[4] e' 1.5.

7. [3.5]

Si considerino i vettori nel linguaggio C:
A [**V**] E' possibile copiare gli elementi di un vettore in un altro vettore.
B [**F**] Gli elementi di uno stesso vettore possono essere di tipo diverso.
C [**V**] E' possibile stampare il contenuto di un vettore di 5 interi con una sola printf.
D [**V**] E' possibile ordinare un vettore di interi in ordine crescente.
E [**V**] Nella definizione di un vettore, la dimensione puo' essere espressa mediante una espressione matematica.

8. [3.5]

Si considerino i tipi di dato scalari primitivi nel linguaggio C:
A [**V**] L'*overloading* degli operatori permette di denotare operatori diversi con lo stesso simbolo.
B [**F**] E' possibile applicare l'operatore `%` (modulo, o resto) a una variabile `float` e una variabile `int`.
C [**F**] La sottrazione tra un `float` e un `char` produce un errore in compilazione.
D [**F**] Nel linguaggio C il vettore e' un tipo scalare primitivo.
E [**F**] L'applicazione della regola di conversione implicita produce una espressione in cui tutti gli operandi assumono uno stesso tipo.

9. [3.5]

Si considerino le istruzioni del linguaggio C:
A [**V**] Nell'istruzione `do..while` se la condizione non e' verificata, si esce dal ciclo.
B [**F**] Nell'istruzione `switch` può essere eseguito al più un solo blocco di istruzioni.
C [**V**] Ogni istruzione `switch` e' sempre esprimibile mediante uno o piu' if (eventualmente annidati).
D [**V**] Nell'istruzione `for` l'espressione di modifica viene sempre eseguita alla fine di ogni iterazione.
E [**V**] Ogni istruzione `for` e' sempre esprimibile mediante `while`.