A

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**] La memoria cache di primo livello e' volatile.

B [**F**] L'accesso ai registri della CPU e' piu' lento rispetto all'accesso alla cache di primo livello.

C [**F**] L'IR contiene il programma correntemente in esecuzione.

D [**V**] La ALU esegue operazioni matematiche.

E [**V**] Il bus consente il trasferimento di istruzioni dalla memoria alla CPU.

2. [3.5]

Linguaggi di programmazione.

A [**F**] Le istruzioni in linguaggio assembler sono espresse mediante una sequenza di bit.

B [**V**] Un programma di alto livello deve essere necessariamente tradotto per poter essere eseguito.

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

D [**F**] Il compilatore serve solo per rilevare gli errori sintattici in un programma.

E [**F**] L'esecuzione di un programma interpretato e' tipicamente piu' veloce dell'esecuzione dello stesso programma compilato.

3. [4]

Si consideri il seguente programma C:

```
#include <stdio.h>

main()
{ int x=2;
  float y=x--;
  char z='y';

  if (--y)
      if(--x)
          z='a';
      else
          z='b';
  else  z=x;
  /* punto 1 */
  y=(x+=3, x--); /*istruzione 1*/
  /* punto 2 */
  { float A;
    A=(z>'a'? --y: y/x);
    /* punto 3*/
  }
  /*punto 4*/
}
```

A [**V**] Al punto 1 la variabile z assume il valore 'b'.

B [**F**] Al punto 2 le variabili x e y hanno lo stesso valore.

C [**F**] Al punto 2 la variabile A vale 0.

D [**V**] Al punto 3 la variabile A ha valore 2.0.

E [**V**] Al punto 4 la variabile A non e' visibile.

4. [4]

Si consideri il seguente programma C:

```
#include <stdio.h>

main()
{int x; float y; char z, w;
 x=4; y=-2; z='B'; w='C';

 x=( x%(int)y ?++x: w-z); /* istruzione 1 */
 y=x++/y; /* istruzione 2 */
 w=z+=1; w-=z; /* istruzione 3 */
}
```

A [**V**] Il programma puo' essere eseguito correttamente.

B [**F**] Immediatamente dopo l'esecuzione dell'*istruzione 1*, la variabile x ha il valore corrispondente al codice ASCII del carattere 'A'.

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

D [**F**] Immediatamente dopo l'esecuzione dell'*istruzione 2*, la variabile y ha il valore -1.

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

5. [4]

Si consideri il seguente programma C:

```
#include <stdio.h>

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

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

6. [4]

Si consideri il seguente programma C:

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

main()
{typedef float f[N];
 typedef char c[N];
 f A={0,0.5,1,1.5,2.0,2.5,3.0,3.5};
 c B={'C','i','a','o',0,1,2,3};
 int i;
 A[N-1]=0;
 for (i=1; i<N; i+=i)
   A[i]=B[N-i]/i;
 /* punto 1 */
 if (A[--i]?1:A[0])
   printf("Ciao!\n");
 else
   {i=A[0];
    for (;i<5;) printf("%c", B[i++]);
   }
 /* punto 2*/
}
```

A [**V**] Il programma stampa "Ciao".
B [**V**] Al punto 1: il valore di A[1] e' 3.0.
C [**F**] Al punto 2: il valore di A[2] e' 2.0.
D [**F**] Al punto 1: tutti gli elementi di A tranne gli estremi sono reali strettamente positivi.
E [**V**] Nel programma vi sono tipi non primitivi.

7. [3.5]

Si considerino i vettori nel linguaggio C:
A [**V**] Gli elementi di un vettore sono tutti dello stesso tipo.
B [**F**] Un vettore di caratteri puo' essere considerato una stringa se non contiene il carattere '0'.
C [**F**] Nella definizione di un vettore, la dimensione puo' essere espressa mediante il nome di una variabile.
D [**V**] L'indice di un elemento puo' essere una variabile di tipo char.
E [**F**] L'indice di un elemento puo' essere una variabile di tipo float.

8. [3.5]

Si considerino i tipi di dato scalari primitivi nel linguaggio C:
A [**V**] E' possibile eseguire una somma tra una variabile `float` e una variabile `char`
B [**F**] E' possibile applicare l'operatore `%` (modulo, o resto) a una variabile `double` e una variabile `int`.
C [**F**] L'applicazione della regola di conversione implicita di tipo è sempre possibile.
D [**V**] La coercizione puo' trasformare un float in un char.
E [**V**] La sottrazione tra un `float` e un `char` produce un valore di tipo `float`.

9. [3.5]

Si considerino le istruzioni del linguaggio C:
A [**F**] Nell'istruzione `switch` almeno un blocco di istruzioni è sempre eseguito.
B [**V**] Ogni istruzione `while` e' sempre esprimibile mediante un'istruzione `for`.
C [**F**] Ogni istruzione `if` e' sempre esprimibile mediante un assegnamento.
D [**F**] Nell'istruzione `switch` il blocco di istruzioni associato all'etichetta `default` viene sempre eseguito.
E [**V**] Nell'istruzione `for` il blocco di istruzioni può non essere mai eseguito.