

<p>1. [3.5]</p>	<p>Si consideri l'architettura di un Personal Computer.</p> <p>A [V] Il bus consente il trasferimento di istruzioni dalla memoria alla CPU.</p> <p>B [F] Il bus consente esclusivamente la trasmissione unidirezionale tra memoria e CPU.</p> <p>C [F] Il clock comunica l'ora corrente ai dispositivi che la richiedono.</p> <p>D [V] Durante un ciclo di esecuzione il registro PC contiene l'indirizzo dell'istruzione da eseguire nel ciclo successivo.</p> <p>E [V] La memoria cache di primo livello e' volatile.</p>	<p>2. [3.5]</p>	<p>Linguaggi di programmazione.</p> <p>A [F] L'esecuzione di un programma interpretato e' tipicamente piu' veloce dell'esecuzione dello stesso programma compilato.</p> <p>B [V] Un programma scritto in linguaggio assembler necessita di un traduttore.</p> <p>C [F] Un programma compilato e linkato e sicuramente privo di errori.</p> <p>D [V] Il debugger e' utile nella fase di sviluppo di un programma.</p> <p>E [F] Le istruzioni in linguaggio assembler sono espresse mediante una sequenza di bit.</p>
<p>3. [4]</p>	<p>Si consideri il seguente programma C:</p> <pre>#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*/ }</pre> <p>A [V] Al punto 4 la variabile A non e' visibile.</p> <p>B [F] Al punto 1 la variabile x ha valore 1.</p> <p>C [F] L'istruzione 1 e' equivalente alla sequenza di istruzioni: $x=x+3$; $x--$; $y=x$;</p> <p>D [V] L'istruzione 1 effettua 3 assegnamenti.</p> <p>E [V] Al punto 1 la variabile z assume il valore 'b'.</p>	<p>4. [4]</p>	<p>Si consideri il seguente programma C:</p> <pre>#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 */ }</pre> <p>A [V] Immediatamente dopo l'esecuzione dell'istruzione 2, la variabile x ha il valore 2.</p> <p>B [F] Immediatamente dopo l'esecuzione dell'istruzione 3, le variabili z e w hanno lo stesso valore.</p> <p>C [V] Immediatamente dopo l'esecuzione dell'istruzione 3, la variabile z ha valore 'C'.</p> <p>D [F] Immediatamente dopo l'esecuzione dell'istruzione 3, la variabile w ha valore '0'.</p> <p>E [V] Il programma puo' essere eseguito correttamente.</p>

<p>5. [4]</p>	<p>Si consideri il seguente programma C:</p> <pre>#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*/ }</pre> <p>A [F] Immediatamente dopo il ciclo for, la variabile z vale 0. B [V] Immediatamente dopo l'istruzione 3, la variabile x ha il valore 1. C [V] Al termine dell'esecuzione, il valore di uno solo tra gli elementi di V risulta modificato rispetto al suo valore iniziale. D [F] Al termine dell'esecuzione, il valore di i e' uguale a 0. E [V] Immediatamente dopo l'istruzione 3, la variabile z ha il valore 2.</p>	<p>Si consideri il seguente programma C:</p> <pre>#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*/ }</pre> <p>6. [4]</p> <p>A [V] Nel programma vi sono tipi non primitivi. B [V] Al punto 2: la variabile i ha valore 5. C [F] Il programma contiene un errore di sintassi. D [V] Al punto 2: A[i-1] ha valore 0.0. E [V] Il programma stampa "Ciao".</p>
<p>7. [3.5]</p>	<p>Si considerino i vettori nel linguaggio C:</p> <p>A [F] L'indice di un elemento puo' essere una variabile di tipo float. B [F] L'indice di un elemento di un vettore deve essere minore della sua dimensione logica. C [V] Non e' possibile applicare l'operatore di assegnamento a variabili di tipo vettore. D [F] Non e' possibile stampare il contenuto di un vettore di 3 interi con una sola printf. E [V] Gli elementi di un vettore sono tutti dello stesso tipo.</p>	<p>8. [3.5]</p> <p>Si considerino i tipi di dato scalari primitivi nel linguaggio C:</p> <p>A [V] La sottrazione tra un float e un char produce un valore di tipo float. B [V] Il criterio di equivalenza strutturale puo' consentire l'assegnamento del valore di una variabile di tipo non primitivo a una variabile di tipo scalare primitivo. C [F] L'overloading degli operatori permette di elaborare con lo stesso operatore tipi di dati altrimenti incompatibili . D [V] L'overloading degli operatori permette di denotare operatori diversi con lo stesso simbolo. E [V] E' possibile eseguire una somma tra una variabile float e una variabile char</p>
<p>9. [3.5]</p>	<p>Si considerino le istruzioni del linguaggio C:</p> <p>A [V] Nell'istruzione for il blocco di istruzioni puo' non essere mai eseguito. B [V] Ogni istruzione switch e' sempre esprimibile mediante altre istruzioni. C [F] Nell'istruzione for l'espressione di modifica viene eseguita nel momento in cui la condizione e' falsa. D [F] Nell'istruzione do..while se la condizione e' verificata, si esce dal ciclo. E [F] Nell'istruzione switch almeno un blocco di istruzioni è sempre eseguito.</p>	