



**Politecnico di Milano**

**Facoltà di Ingegneria Civile, Ambientale e Territoriale**

**Informatica (ICA-LC) [083668] – Informatica B [079904]**

**Informatica B [060054]**

**Prof. P. Plebani**

**Allievi Ingegneria Civile e Ambientale**

### **SIMULAZIONE PRIMA PROVA IN ITINERE**

Il presente documento illustra una tipica prova d'esame. Lo svolgimento di questa prova richiede volutamente un tempo superiore a quello di solito concesso alle prove ufficiali (circa 2h) a causa della presenza di numerosi esercizi.

A complemento della presente simulazione di prova d'esame, alla pagina <http://home.dei.polimi.it/plebani/did-info0809ICA.htm> è possibile trovare i temi d'esame proposti lo scorso anno che, per la parte teorica, rimane pressoché identica.

### Domanda 1. Rappresentazione di valori numerici .

1. Si ordinino in modo decrescente i seguenti numeri:

- $N1 = -113$  in base 10
- $N2 = 0100\ 0111$  in base 2
- $N3 = 43$  in base 16

Converto tutto in base 10

$N1 = -113$

$N2 = 71$

$N3 = 67$

$N2 > N3 > N1$

2. Quanti bit occorrono per rappresentare  $N3$  ed  $N1$  in complemento a 2? Eseguire in complemento a 2 (mostrando i passaggi, indicando esplicitamente se si verifica overflow e motivando la risposta) le operazioni:

- $N1 + N2$
- $N1 - N3$ .

Per  $N1$  servono 8 bit

Per  $N3$  servono 8 bit

$N1$  in base 2 è 01110001 complementato è 10001111

$N2$  in complemento a 2 è 01000111

$N3$  in complemento a 2 è 01000011

$-N3$  in complemento a 2 è 10111101

$N1 + N2 = 10001111 + 01000111 = 11010110$  (-42)

$N1 - N3 = 10001111 + 10111101 = (1) 01001100$  (overflow)

**Domanda 2. Logica booleana.**

- A or C and not B
- not ( ( B or not C ) and not A )

A	B	C	Not(B)	C and NOT(B)	A or C and NOT (B)
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	0	1

A	B	C	NOT C	NOT A	B or NOT(C)	(B or NOT(C)) and NOT A	NOT((B or NOT(C)) and NOT A)
0	0	0	1	1	1	1	0
0	0	1	0	1	0	0	1
0	1	0	1	1	1	1	0
0	1	1	0	1	1	1	0
1	0	0	1	0	1	0	1
1	0	1	0	0	0	0	1
1	1	0	1	0	1	0	1
1	1	1	0	0	1	0	1

Le due espressioni producono la medesima tabella di verità quindi sono equivalenti

**Domanda 3. Architettura dei calcolatori.**

Illustrare le componenti dell'Unità di Elaborazione secondo il modello della macchina di Von Neumann e descrivere il ciclo fetch-decode-execute

#### Domanda 4. Tracing di programmi .

a) Dato il seguente frammento di codice MATLAB si riempia la tabella riportata di seguito con i valori delle variabili *a*, *b*, *ii* nel momento in cui sono stampate a video. N.B.: la tabella non deve essere necessariamente riempita tutta. In caso di loop infinito specificarlo chiaramente

```
a=[];  
b=[100:100:500]  
for ii=1:length(b)  
    a=[a,ii/10];  
    b(ii)=b(ii).*a(ii);  
    disp(['a(end) = ',num2str(a(end))])  
    disp(['b(ii) = ',num2str(b(ii))])  
end  
  
a(find(b)<5)=0;  
disp(a);
```

a(end)	B(ii)	ii
0.1	10	1
0.2	40	2
0.3	90	3
0.4	160	4
0.5	250	5

b) Dato il seguente frammento di codice MATLAB si riempia la tabella riportata di seguito con i valori delle variabili *A*, *b*, *ii* nel momento in cui sono stampate a video. N.B.: la tabella non deve essere necessariamente riempita tutta. In caso di loop infinito specificarlo chiaramente

```
row=[1:10:40];
disp(['row = ',num2str(row)]);
A= repmat(row,[5,1]);
A(1:3,1:3)=-A(1:3,1:3);
disp('A= ')
disp(A);

ii=1;
b=[];

while (ii<5)
    if (A(ii,ii)<10)
        b=[b,A(ii,ii)];
    else
        b=[b,3*A(ii,ii)];
    end
    ii=ii+1;
    disp(['ii = ',num2str(ii)]);
    disp(['b = ',num2str(b)]);
end
```

A	b	ii
-1 -11 -21 31 -1 -11 -21 31 -1 -11 -21 31 1 11 21 31 1 11 21 31		
	2	-1
	3	-1 -11
	4	-1 -11 -21
	5	-1 -11 -21 93

### Domanda 5. Manipolazione vettori e matrici

Dati i seguenti frammenti di codice indicare il contenuto dello schermo al termine della sua esecuzione (vedi istruzione disp)

1)

```
a=ones(1,13)
b=[1:2:30]
c=b(1:2:14)
a=a>0
k=a(1:length(c)).*b(1:length(c)).*c(1:length(c))
h=k(end:-2:1)
disp(h(end-1))
```

```
a = 1 1 1 1 1 1 1 1 1 1 1 1 1
b = 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29
c = 1 5 9 13 17 21 25
a = 1 1 1 1 1 1 1 1 1 1 1 1 1
k = 1 15 45 91 153 231 325
h = 325 153 45 1
la disp visualizza quindi 45
```

2)

```
a=zeros(10,1)
a(5)=1
a(6:end)=7:-1:3
b=ones(10,1)
b(find(a<5))=5
c=[a;b]
h=a.*b.*c(5:14)
disp(h(5))
```

```
a = 0;0;0;0;0;0;0;0;0;0
a = 0;0;0;0;1;0;0;0;0;0
a = 0;0;0;0;1;7;6;5;4;3
b = 1;1;1;1;1;1;1;1;1;1
b = 5;5;5;5;5;1;1;1;5;5
c = 0;0;0;0;1;7;6;5;4;3;5;5;5;5;5;1;1;1;5;5
h = 0;0;0;0;20;21;30;25;100;75

la disp visualizza 20
```

## Domanda 6. Algoritmi in MATLAB

Dato un array  $v$  di 10 numeri interi, si codifichi un programma in linguaggio MATLAB in grado di fornire le seguenti funzionalità:

- inserimento dei 10 valori da parte dell'utente. I valori ammessi devono essere compresi tra 1 e 100 (in caso l'utente inserisca un valore fuori dall'intervallo, il programma deve richiedere di re-inserire il valore);
- stampa a video del valore minimo e della posizione all'interno dell'array (senza utilizzare la funzione min);
- stampa a video del valore ottenuto moltiplicando i 10 valori tra di loro;
- stampa a video dell'array dopo aver invertito la posizione dei valori al suo interno (il valore in 10a posizione messo in 1a posizione, il valore in 9a posizione messo in 2a posizione, ecc..)

```
v=[];
i=1
while (i<=10)
    a = 0
    while (a<1 | a >100)
        a = input(['inserire valore in posizione ' num2str(i) ' ']);
    end
    v = [v a];
    i = i+1;
end

v= [1 2 3 4 5 6 7 8 9 100];
min = 101;
max = 0;
for i=1:1:10
    if (v(i) > max)
        max = v(i);
    end
    if (v(i) < min)
        min = v(i);
    end
end
disp(['min ' num2str(min)])
disp(['max ' num2str(max)])

mult = 1;
for i=1:1:10
    mult = mult * v(i);
end
disp(['moltiplicazione ' num2str(mult)])

for i=1:1:5
    temp = v(i);
    v(i) = v(length(v)-(i-1));
    v(length(v)-(i-1)) = temp;
end

disp(v)
```