VERIFICA DI INFORMATICA (SOLUZIONE)

Liceo Scientifico “A. Volta”
classe 5° B, 10/05/2008
prof. Magni Claudio

Quesito 1 (2.5 punti)

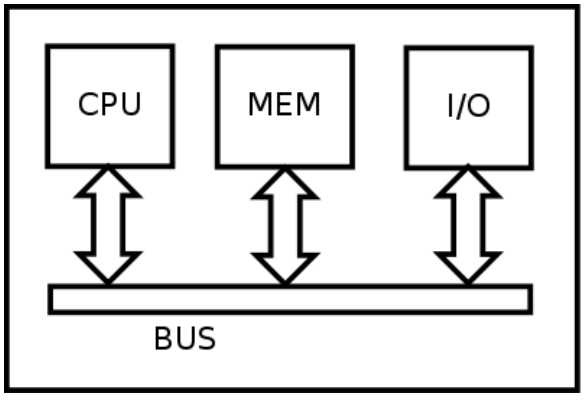
$100101_2 = 1*2^5 + 1*2^2 + 1*2^0 = 32 + 4 + 1 = 37_{10}$

$16_{16} = 1*16^1 + 6*16^0 = 16 + 6 = 22_{10}$

52	0 ↑
26	0
13	1
6	0
3	1
1	1
0	

$52_{10} = 110100_2$

Quesito 2 (2 punti)
La macchina di Von Neumann è un modello logico che descrive in modo astratto il funzionamento di un calcolatore. Nonostante i moderni calcolatori si siano evoluti parecchio dai tempi in cui questo modello fu inventato, i principi fondamentali sono rimasti invariati fino ad oggi.
Il modello può essere schematizzato come segue:



Il BUS collega i vari dispositivi tra loro: permette quindi la comunicazione.
La CPU è la parte attiva del calcolatore, svolge i calcoli matematici e gestisce gli altri componenti tramite BUS.
La MEMORIA (principale) contiene i dati e le istruzioni che la CPU necessita per svolgere i suoi compiti. A seconda degli ordini che riceve dalla CPU attraverso il BUS di controllo, può scrivere in una cella o leggerne il contenuto. Può anche scambiare dati direttamente con i dispositivi di I/O.
I dispositivi di I/O permettono la comunicazione della macchina con il mondo esterno. Operano quindi una trasformazione dalla logica binaria alla realtà fisica (continua). In genere prendono ordini dalla CPU.

Quesito 3 (3 punti + 1 bonus)
Riporto il codice completo, commentando a lato ciò che viene svolto.

```
 1 int n, x, resto;                                  /* Dichiaro 3 variabili intere */
 2 printf("Inserisci il numero: ");                  /* Stampo sullo schermo */
 3 scanf("%d", &n);                                  /* Ricevo il numero e lo metto in n */
 4 printf("\n");                                     /* Stampo sullo schermo (a capo) */
 5 x = 2;                                            /* Assegno a x il valore 2 */
 6 while ( x < n ) {                                 /* Ciclo finché x è minore di n */
 7    resto = n%x;                                   /* Calcolo il resto di n diviso x */
 8    if ( resto == 0 ) {                            /* Se resto è 0 (divisione intera) */
 9       printf("Il numero inserito non è primo.");  /* Stampo sullo schermo */
10       exit(0);                                    /* Termino il programma */
11    }                                              /* Fine if */
12    x = x + 1;                                     /* Incremento x di 1 */
13 }                                                  /* Fine ciclo while */
14 printf("Il numero inserito è primo.");           /* Stampo sullo schermo */
```

a) Quante divisioni svolge al massimo l'algoritmo, dato un numero **n**?
Al massimo **n – 2**. Infatti la variabile **x** parte da 2 e arriva ad **n – 1**.

b) Che tipo di numero deve inserire l'utente perché ciò avvenga?
Deve inserire un numero primo, l'unico caso in cui il programma arriva fino alla riga 14.
Si noti che il primo numero primo è **2**, per cui il programma si comporta in modo corretto.

c) Quante divisioni svolge come minimo?
0, quando il numero inserito è 2 (0 e 1 sono casi particolari).
In generale invece il minimo è **1** sola divisione, ogni volta che il numero inserito è pari.

Domanda facoltativa:
Due possibili miglioramenti:

1. Evitare di dividere per tutti i numeri pari dopo il 2. Quindi si divide per 2 all'inizio, dopodiché **x** parte da 3 e viene incrementato di 2 ogni volta (riga 12: `x = x + 2;`). Questo permette di dimezzare le divisioni.
2. Fermare il ciclo una volta che **x** raggiunge la metà del numero **n**, in quanto dopo quel punto non ci saranno più divisioni intere (riga 6: `while ( x <= (n/2) )`). Questo permette di dimezzare le divisioni.

Implementando i metodi sopracitati, il numero max di divisioni diventa circa $\frac{n}{4}$.

Curiosità: l'algoritmo più efficiente per decidere la primalità di un numero è stato inventato nel 2002 ed impiega circa $\log(n)$ operazioni (vedere en.wikipedia.org/wiki/AKS_primality_test).

Quesito 4 (2 punti)