Dipartimento di Informatica e Sistemistica
Antonio Ruberti

“Sapienza” Università di Roma

Esercitazione 4

Corso di Fondamenti di Informatica

Laurea in Ingegneria Informatica

(Canale di Ingegneria delle Reti e dei Sistemi Informatici - Polo di Rieti)

Anno Accademico 2007/2008

Tutor: Ing. Diego Rughetti

Somma tra Numeri Binari in Complemento a 2

$0010_2+1101_2=1111_2$

riporto 0000

0010+

1101=

1111

$0110_2+0111_2=$?

riporto 0110

0110

0111

1101

Somma tra Numeri Binari in Complemento a 2

1111_2+1000_2= ?

riporto 1000

1111+

1000=

0111

$0110_2+1111_2=0111$

riporto 1110

0110

1111

0101

Rappresentazione in virgola mobile

1. 32 bit totali
2. 1 bit di segno
3. 8 bit per l'esponente => in complemento a 2
4. 23 bit per la mantissa => in modulo e segno (NB il segno della mantissa è dato dal primo dei 32 bit, quello al punto 2)

Rappresentazione in virgola mobile, capacità di rappresentazione

Dopo lo 0. ci sono 23 cifre pari ad 1

32 bit:

1. Numeri positivi: tra 0.1*2^-128 e 0.1…..1*2^127
2. Numeri negativi: tra –0.1…1*2^127 e –0.1*2^-128
3. Errore $\leq$ 2^-23

Esempio – somma e prodotto

Dati due numeri N1 ed N2, rappresentati in virgola mobile a 32 bit:

N1 = 1 00000010 10111111100000000000000

N2 = 0 00000011 10100110000000000000000

Eseguirne la somma

Soluzione - somma

Gli esponenti sono diversi, devo manipolare i numeri in modo che abbiano entrambi lo stesso esponente. Per poter sommare direttamente le mantisse e diminuire la probabilità di dover normalizzare il risultato di tale somma, vado a manipolare il numero con esponente minore, nel nostro caso N1:

Faccio lo shift a destra della mantissa di N1 di una posizione e sommo 1 al suo esponente:

N1 = 1 00000010 10111111100000000000000

N1 = 1 00000011 01011111110000000000000

Ora sommo le mantisse ottenendo la mantissa del risultato (NB tenere conto del fatto che N1 è negativo!):

$M_{N1} + M_{N2} = 01000110010000000000000$

Notare che la mantissa ottenuta come somma deve essere normalizzata! Faccio lo shift a sinistra e sottraggo uno all'esponente:

Risultato = 0 00000010 10001100100000000000000

Esercizio 1

Definire una funzione che prenda in ingresso un numero n e restituisca in uscita il numero di cifre che compongono n.

Soluzione

```
unsigned short int inc(unsigned long int i){
        unsigned short int nc;
        nc = 1;
        while (i > 9){
                nc++;
                i = i / 10;
        }
        return nc;
}

unsigned short int inc1(unsigned long int i){
        unsigned short int nc;
        for (nc = 1;i > 9; nc++,i /= 10);
        return nc;
}
```

Esercizio 2

Definire una funzione che prenda in ingresso un numero n e stampi a video l’insieme dei divisori di n

Soluzione

```
void divisori(unsigned long int n){
        unsigned long int d = 1;
        while (d*d < n){
                if (n%d == 0)
                        printf("%lu %lu\n",d, n/d);
                d += 1;
        }
        if (d == n/d)
                printf("%lu\n",d);
}
```

Esercizio 3

Scrivere un programma C che prende in ingresso tre numeri a, b, c, i quali rappresentano i tre coefficienti di una equazione di secondo grado, del tipo:

$$a*x^2+b*x+c = 0$$

e stampi a monitor le radici, reali o complesse dell'equazione.

Soluzione

```
void reali(float a, float b, float d){
		printf("Le soluzioni sono:\n\t x1 = %f, x2 = %f\n", (-b + d)/(2*a), (-b - d)/(2*a));
}

void compl(float a, float b, float d){
		printf("Le soluzioni sono complesse e coniugate:\n\t p. reale = %f, p. imm = %f\n",(-b)/(2*a), d/(2*a));
}

int main(){
		float a,b,c, discr;
		scanf("%f%f%f",&a,&b,&c);
		printf("%+fx^2 %+fx %+f\n",a,b,c);
		discr = b*b - 4*a*c;
		if (discr >= 0)
				reali(a,b,sqrt(discr));
		else
				compl(a,b,sqrt(-discr));
}
```

Esercizio 4

Scrivere una funzione che riceve come argomento un numero intero m e stampa a video la scomposizione in fattori primi di m

Soluzione

```
void fattori(unsigned long int n){
        unsigned short int e;
        unsigned long int d = 2;
        while (n > 1){
                e = 0;
                while (n%d == 0){
                        n = n/d;
                        e++;
                }
                if (e)
                        printf("%lu %u\n",d, e);
                d += 1;
}
```

Esercizio 5

Scrivere un programma C che prende in ingresso un numero intero positivo n e stampa a video l'n-esimo numero della serie di fibonacci. (N.B. Il primo numero della serie non è associato ad n = 1 bensì ad n = 0, ovvero il primo numero della serie è lo 0-esimo)

Serie di fibonacci:
Sequenza di numeri in cui ogni termine, a parte i primi due, è la somma dei due che lo precedono.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

Il calcolo della serie fino alla posizione n-esima deve essere fatto attraverso una apposita funzione c che prende in ingresso l'intero n e restituisce il numero della serie di fibonacci in n-esima posizione.

Soluzione

```
unsigned long int ifib(unsigned short int n){
        int i;
        unsigned long int fn1,fn2, fn;
        if (n == 0)
                return 0;
        else if (n == 1)
                return 1;
        else{
                fn2 = 0;
                fn1 = 1;
                for (i = 2; i <= n; i++){
                        fn = fn1 + fn2;
                        fn2 = fn1;
                        fn1 = fn;
                }
                return fn;
        }
}
```