NOTAZIONE IN VIRGOLA MOBILE

In questa notazione, un numero *n* viene rappresentato come una sequenza di bit che si divide in tre parti: segno, esponente e mantissa.
Il segno indica se il numero è positivo o negativo (sempre un bit: 0=positivo, 1=negativo)
L'esponente indica l'esponente di una potenza di 2. ($2^4 \rightarrow$ esp=4 rappresentato in binari)
La mantissa indica la parte frazionaria di un numero. (1,25 $\rightarrow$ mant=0,25 rappresentato in binari)
Esempio di una notazione in virgola mobile:
1 bit per il segno, 8 per l'esponente e 23 per la mantissa.
Quindi il numero 0 $\underbrace{10011001}$ 00010111011100101100111
E' in eccesso a 256, perciò si trova il numero totale e si sottrae 256, quindi 10011001=281 e esp=281-256=25

equivale a $+2^{25}$ *(1,00010111011100101100111) cioè $+2^{25}$ *($2^0+2^{-4}+2^{-6}+2^{-7}+2^{-8}+2^{-10}+2^{-11}+2^{-12}+2^{-15}+2^{-17}+2^{-18}+2^{-21}+2^{-22}+2^{-23}$). Per ogni 1 presente nella mantissa, si prende la potenza di 2 corrispondente, sapendo che la prima cifra dopo la virgola è 2^{-1} , la seconda è 2^{-2} e così via. Il numero è positivo perché segno=0.

PER PASSARE DALLA NOTAZIONE IN ECCESSO A QUELLA IN CP2, BASTA INVERTIRE IL BIT PIU' SIGNIFICATIVO.
Es. 1010 in ecc. = 0010 in CP2.
Es. 0010 in ecc. = 1010 in CP2 = -0110

Esercizio 1
Rappresentazione binaria in virgola mobile a 16 bit:
– 1 bit per il segno (0=positivo)
– 8 bit per l'esponente, in eccesso 128
– 7 bit per la parte frazionaria della mantissa normalizzata tra 1 e 2
• Calcolare gli estremi degli intervalli rappresentati, i numerali corrispondenti, e l'ordine di grandezza decimale.
• Rappresentare in tale notazione il numero *n* rappresentato in complemento a 2 dai tre byte FF5AB9.
• Calcolare l'errore relativo ed assoluto che si commette rappresentando il *n* nella notazione data.

• Si esclude il bit di segno:
- il numero più grande, che ha l'esponente con tutti 1 e la mantissa con tutti 1, quindi:
1111 1111 1111 111.
L'esponente 1111 1111 equivale a 255-128=127, si sottrae 128 perché in eccesso a 128.
La mantissa 1111 111 rappresenta il massimo valore della mantissa, che come sappiamo è compresa in [1, 2). Quindi vale circa 2.
Il numero più grande quindi è $2^{127} * 2 = 2^{128}$.
- il numero più piccolo, che ha l'esponente tutti 0 e la mantissa tutti 0, quindi:
0000 0000 0000 000
0000 0000 equivale a 0-128=-128 (si sottrae anche qui 128).
0000 000 è il valore minimo della mantissa, che ricordiamo compresa in [1 ,2). Quindi vale 1.
Il numero più piccolo quindi è $2^{-128}*1= 2^{-128}$.
Considerando il segno, l'intervalli ricoperti sono (-2^{128}, -2^{-128}] e [$+2^{-128}$, $+2^{128}$). Gli estremi sono intervalli aperti poiché la mantissa non è esattamente 2.
Ora bisogno portarlo in grandezza decimale. Sapendo che 2^{10} (1024) è circa 10^3 (1000) si usa il rapporto:
10:3=128:X X=128*3/10= circa 38. Quindi 2^{-128} e 2^{128} in grandezza decimale 10^{-38} e $10^{38.}$
• FF5AB9 in binari equivale a 1111 1111 0101 1010 1011 1001. Dato che il primo bit è 1 (quindi negativo) bisogna complementarlo a 2 e portarlo nella notazione descritta in alto.
Complementare a 2 significare invertire tutti i bit (0 con 1 e viceversa) e aggiungere 1 al risultato (in pratica si invertono tutti i bit prima dell'ultimo 1 nella sequenza).
Il risultato è 0000 0000 1010 0101 0100 0111. Il bit più significativo uguale a 1 (colorato in verde) corrisponde a 2^{15}.
Quindi l'esponente in eccesso a 128 è uguale a 15+128=143 (1000 1111) e la mantissa è tutto ciò che segue quell'1 quindi 010 0101 0100 0111, tuttavia secondo la notazione la mantissa è di 7 bit quindi prendo solo i primi 7 e il resto lo scarto. Il bit di segno sarà 1, perché FF5AB9 aveva 1 come primo bit.
FF5AB9 nella notazione descritta è 1 1000 1111 0100101.
•L'errore assoluto si calcola prendendo la parte che abbiamo scartato (colorata sopra di arancione) moltiplicandola per 2^{15} .
2^{-1} 2^{-2} … 2^{-9}
Prendiamo per ogni 1 nella parte scartata le potenze di 2 corrispondenti. 010 0101 0100 0111
($2^{-9} + 2^{-13} + 2^{-14} + 2^{-15}$) e la moltiplichiamo per 2^{15}.

Errore assoluto = $2^{15}*(2^{-9} + 2^{-13} + 2^{-14} + 2^{-15}) = 2^6+2^2+2^1+2^0$ = 64+4+2+1 = 71.
Per calcolare l'errore relativo, basta calcolare la distanza tra il bit più significativo non rappresentato (2^{-9}) e il bit a sinistra della virgola $\underbrace{1,010\ 0101\ 0}_{\text{9 posizioni}}100\ 0111$. Errore relativo = 2^{-9}

Esercizio 2
Rappresentazione binaria in virgola mobile a 16 bit:
– 1 bit per il segno (0=positivo)
– 8 bit per l'esponente, in eccesso 128
– 7 bit per la parte frazionaria della mantissa normalizzata tra 1 e 2
• Dato il numero razionale *m* rappresentato in tale notazione dai due byte 41A5, calcolare l'intero *n* che approssima *m* per difetto, e rappresentarlo in complemento a 2 con 16 bit.

• 41A5 in binario è 0100 0001 1010 0101. L'esponente (in rosso) è uguale in decimale a 131 a cui si sottrae l'eccesso (131-128=3). Moltiplicando allora 2^3 per la mantissa 1,010 0101 otteniamo:
2^3*(1,010 0101)=1010,0101 (basta spostare la virgola di 3 posti a destra).
1010,0101 è il numero *m*. Per calcolare l'intero *n*, arrotondiamo per difetto e eliminiamo la parte frazionale.
1010 è il numero *n*. Con 16 bit equivale a 0000 0000 0000 1010.

Esercizio 3
Rappresentazione binaria in virgola mobile a 16 bit:
– 1 bit per il segno (0=positivo)
– *e* bit per l'esponente, in eccesso 2^{e-1}
– (15 – *e*) bit per la parte frazionaria della mantissa normalizzata tra 1 e 2
• Calcolare il valore minimo di bit per l'esponente che consenta di rappresentare il numero *n* rappresentato in compl. a 2 dai tre byte FF5AB9.

• FF5AB9 corrisponde in binario a 1111 1111 0101 1010 1011 1001. Il primo bit è 1 quindi complementiamo a 2.
Si ottiene dunque 0000 0000 1010 0101 0100 0111.
Il bit più significativo è il 15esimo, che corrisponde a 2^{15} quindi l'esponente senza eccesso è 15. Bastano 4 bit per rappresentare 15, ai quali va aggiunto 1 bit per l'eccesso, in totale fanno 5 bit per l'esponente.
I restanti 15-5=10 bit vanno a comporre la mantissa.
Si considerano allora solo i 10 bit successivi al bit più significativo. 1,010 0101 0100 0111
L'errore assoluto è dato da $2^{15}*(2^{-13} + 2^{-14} + 2^{-15}) = 2^2 + 2^1 + 2^0$ = 4 + 2 +1 = 7.
L'errore relativo è dato dal bit più significativo della parte non rappresentata cioè 2^{-13}.

Esercizio 4
Rappresentazione binaria in virgola mobile a 16 bit:
– 1 bit per il segno (0=positivo)
– 7 bit per l'esponente, in eccesso 64
– 8 bit per la parte frazionaria della mantissa normalizzata tra 1 e 2
• Dati *m* e *n* rappresentati in tale notazione dalle stringhe esadecimali FA53 e F9F2: calcolare la somma di *m* e *n* e fornire la stringa esadecimale che la rappresenta nella notazione suddetta.

m = FA53 = 1111 1010 0101 0011. L'esponente è 122-64=58.
n = F9F2 = 1111 1001 1111 0010. L'esponente è 121-64=57.
Per sommare i due numeri bisogna averli con lo stesso esponente. La differenza tra gli esponenti è di 1, perciò moltiplico il numero con esponente più basso di 2^1 e divido la sua mantissa per 2^1. Ovvero:
Partendo da 2^{57}*(1,1111 0010) faccio ($2^{57}*2_1$)*((1,1111 0010)/2^1)=2^{58}*(0,1111 1001 0). Prendo solo i primi 8 bit della mantissa, l'ultima cifra non la considero. n = 1111 1010 1111 1001
A questo punto facciamo l'addizione sommando le mantisse:
1,0101 0011 +
0,1111 1001 =
10,0100 1100 = 2^1*(1,0010 0110 0). Anche qui considero solo i primi 8 bit dopo la virgola, e l'ultima cifra la escludo.
Allora m+n= $2^{58}*2^1$*(1,0010 0110) = 2^{59}*(1,0010 0110). Sto sommando due numeri negativi, quindi il bit di segno sarà 1.
Perciò m+n= 1111 1011 0010 0110 che in esadecimale è FB26.

Esercizio 5

Si consideri una notazione binaria in virgola mobile a 8 bit di cui (nell'ordine da sinistra a destra) si usa 1 bit per il segno (0=positivo), k bit per l'esponente, che è rappresentato in eccesso a 2^{k-1}, ed i rimanenti bit per la parte frazionaria della mantissa, di cui si rappresenta solo la parte frazionaria. Sia X il numero meno significativo non nullo del vostro numero di matricola.

A. **Individuare il valore di k che consente di rappresentare tutti i numeri compresi tra -100 e 100 (espressi in notazione decimale) con la massima precisione possibile e indicare, per la notazione individuata, l'intervallo di rappresentazione tenendo conto del fatto che le configurazioni dell'esponente composte da tutti 0 e da tutti 1 sono riservate;**
B. **Rappresentare, nella notazione definita al punto A, i numeri decimali 0, -65 e 7,5 indicando gli eventuali errori di rappresentazione commessi;**
C. **Indicare quale numeri decimali rappresentano, nella notazione definita al punto A, le stringhe esadecimali CX e 4X;**
D. **Individuare il numero e di bit dell'esponente e il numero m di bit della mantissa di una notazione in virgola mobile a 16 bit che sia in grado di rappresentare tutti i numeri rappresentabili nella definita al punto A e che abbia l'intervallo di rappresentazione più grande possibile**

•Per rappresentare l'intervallo [-100, 100], serve una potenza di 2 maggiore di 100. In questo caso va bene 2^7 = 128.
Attenzione: non servono 7 bit, ma serve un numero di bit che possa rappresentare 7. Bastano 3 bit per rappresentare 7 (111) più 1 bit per l'eccesso, quindi 4 bit.
Quindi la sequenza 1111 basterebbe per indicare 7 con l'eccesso a 8, <u>tuttavia</u> dato che la codifica tutti 1 è riservata, dobbiamo aggiungere 1 ulteriore bit, in modo che 7 con l'eccesso a 16 sia rappresentabile (10111).
Dunque k=5 e mantissa= 7-5=2.
Per calcolare l'intervallo di rappresentazione, ricordiamo che le codifiche con l'esponente tutti 0 e tutti 1 sono riservate.
Il numero MAX = ± 11110 11 = $\pm 2^{(30-16)} * (1{,}11) = \pm 2^{14} * (2^0+2^{-1}+2^{-2}) = \pm 2^{14} * 1{,}75$
Il numero MIN = ± 00001 00 = $\pm 2^{(1-16)} * (1{,}00) = \pm 2^{-15} * 1 = \pm 2^{-15}$
• Il numero 0 è rappresentato dalla codifica riservata con tutti 0, quindi in questa notazione è 0 00000 00.
Gli altri numeri vanno presi come somma di potenze di 2. Poi estraggo la potenza con esponente più alto, e ciò che rimane tra parentesi è la mantissa (di cui considero solo i 2 bit dopo la virgola).
Quindi $-65 = -(64+1) = -(2^6+2^0) = -2^6 * (2^0+2^{-6}) = -2^6 * (1{,}000001)$ = 1 10110 00. Non è possibile rappresentare 2^{-6} in questa notazione perciò l'errore assoluto è $2^6 * 2^{-6} = 1$.
Mentre $7{,}5 = (4+2+1+0{,}5) = (2^2+2^1+2^0+2^{-1}) = 2^2 * (2^0+2^{-1}+2^{-2}+2^{-3}) = 2^2 * (1{,}111)$ = 0 10010 11. Non è possibile rappresentare 2^{-3} in questa notazione perciò l'errore assoluto è $2^2 * 2^{-3} = 2^{-1} = 0{,}5$.
•Con X=5, abbiamo C5 e 45.
C5 = 1100 0101 = $-2^1 * (1{,}01) = -2^1 * (2^0 + 2^{-2}) = -2 * (1+0{,}25) = -2{,}5$
45 = 0100 0101 = $2^1 * (1{,}01) = 2^1 * (2^0 + 2^{-2}) = 2 * (1+0{,}25) = 2{,}5$
•Per rappresentare la parte razionale rappresentabile dalla notazione del primo punto, la notazione a 16 bit deve avere come minimo la mantissa con stesso numero di bit. Dato che vogliamo rappresentare l'intervallo più grande possibile, prendiamo il minimo della mantissa necessaria (2 bit) e il resto (16-1-2=13 bit) li assegniamo all'esponente.

Esercizio 6

Si consideri un sistema di rappresentazione binaria in virgola mobile a n bit, di cui si usano (nell'ordine da sinistra a destra): 1 bit per il segno (0 = positivo), e bit per l'esponente, che è rappresentato in eccesso a 2^{e-1}, e n−e−1 bit per la parte frazionaria della mantissa che è normalizzata tra 1 e 2. Sia X la cifra meno significativa non nulla del vostro numero di matricola.
• Indicare l'ordine di grandezza decimale dei seguenti numeri: a espresso in eccesso a 2^{+19} dalla stringa esadecimale 973AX, b espresso in complemento a due dalla stringa esadecimale 2D4FX, e c espresso in complemento a uno dalla stringa esadecimale FFC4X;
• Calcolare il valore minimo di n che consente di rappresentare tutti i numeri a, b e c al punto A nella notazione in virgola mobile suddetta senza commettere errori di rappresentazione;
• Calcolare il numero d = a − c e rappresentarlo nel sistema di rappresentazione individuato al punto B, indicando l'errore assoluto che si commette;

• Quale è l'ordine di grandezza binario del numero più piccolo rappresentabile con numerali denormalizzati con il sistema individuato?

• Prendiamo X = 5.
973A5 in binari è 1001 0111 0011 1010 0101. In eccesso a 2^{19} il numero *a* è 0001 0111 0011 1010 0101.
Quindi 2^{16} * (1, 0111 0011 1010 0101). L'ordine di grandezza decimale è 10^4. Si ricava da $2^{10} : 10^3 = 2^{16} : 10^x$.
2D4F5 in binari è 0010 1101 0100 1111 0101. In CP2 il numero *b* resta 0010 1101 0100 1111 0101.
Quindi 2^{17} * (1, 0 1101 0100 1111 0101). L'ordine di grandezza decimale è 10^5. Si ricava da $2^{10} : 10^3 = 2^{17} : 10^x$
FFC45 in binari è 1111 1111 1100 0100 0101. In CP1 il numero *c* diventa –(0000 0000 0011 1011 1010).
Quindi -2^9 * (1, 1 1011 1010). L'ordine di grandezza decimale è 10^2. Si ricava da $2^{10} : 10^3 = 2^9 : 10^x$.
•Per rappresentare *a, b,* e *c* senza errori, ho bisogno di un numero di bit per la mantissa pari a quello del numero con mantissa più estesa (nel nostro caso *b,* che ha 17 bit di mantissa). L'esponente avrà un numero di bit almeno sufficiente per rappresentare in eccesso l'esponente della potenza di 2 più alta (nel nostro caso sempre *b*, che ha esponente=17). Per rappresentare 17, servono 5 bit, più 1 per l'eccesso, quindi 6 bit.
Quindi n=1+6+17=24 bit.
•Il numero d = a – c equivale a fare l'operazione 0001 0111 0011 1010 0101 – (–(0000 0000 0011 1011 1010)) = 0001 0111 0011 1010 0101 + 0000 0000 0011 1011 1010. Il risultato della somma è:

0001 0111 0011 1010 0101 +
<u>0000 0000 0011 1011 1010 =</u>
0001 0111 0111 0101 1111 = 2^{16} * (1, 0111 0111 0101 1111)
che nella notazione definita prima è 0 110000 0111 0111 0101 11110. (E' stato aggiunto 0 alla fine della mantissa)
•L'ordine di grandezza binario del numero più piccolo denormalizzato è:
dato 000000 0000 0000 0000 0000 1 = 2^{-32} * (1, 0000 0000 0000 0000 1) = $2^{-32} * 2^{-17} = 2^{-49}$