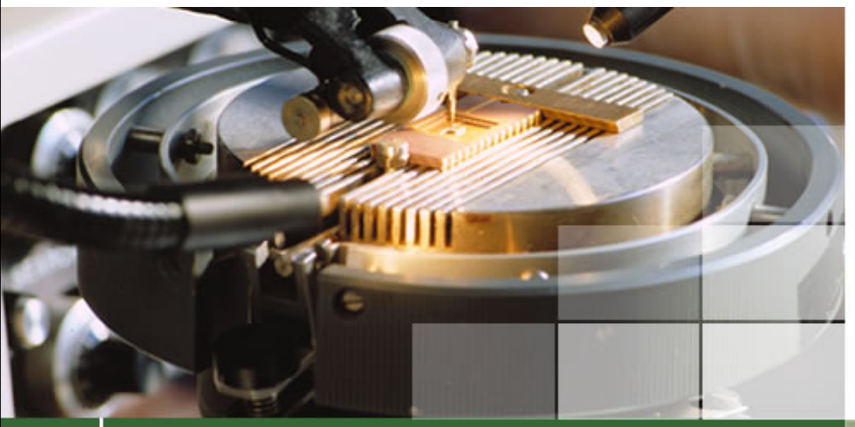


POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Terza esercitazione

Riccardo Cattaneo

Marco D. Santambrogio

POLITECNICO
DI MILANO

20/11/2013

Terza esercitazione: agenda

- (35’) Lezione su numeri in virgola fissa e mobile
- (5’) Pausa
- (30’) Es1: tabelle di verità
- (10’) Es2: conversioni B/D/E
- (10’) Es3: virgola fissa e mobile
- (10’) Es4: somma binaria algebrica CPL2
- (5’) Pausa
- (30’) Es5: l’anagrafica (strutture dati)
- (15’) Es6: ricerca anagrafica

Terza esercitazione: agenda

- (35’) Lezione su numeri in virgola fissa e mobile
- (5’) Pausa
- (30’) Es1: tabelle di verità
- (10’) Es2: conversioni B/D/E
- (10’) Es3: virgola fissa e mobile
- (10’) Es4: somma binaria algebrica CPL2
- (5’) Pausa
- (30’) Es5: l’anagrafica (strutture dati)
- (15’) Es6: ricerca anagrafica

Numeri in virgola fissa

- Fino a questo punto abbiamo assunto che
  - Un vettore di bit rappresentasse sempre un numero intero
  - Eventualmente con segno
- Tutte le considerazioni fatte fino ad ora e tutti i metodi esposti continuano a valere se si attribuisce ai vettori di bit il significato di numeri in *virgola fissa*

- Un sistema di numerazione in virgola fissa è quello in cui:
  - La posizione della virgola decimale è implicita
  - La posizione della virgola decimale uguale in tutti i numeri
- La posizione della virgola equivale alla interpretazione del valore intero moltiplicato per un fattore di scala

Numeri in virgola fissa: fattore di scala

- Si consideri ad esempio il vettore di $k+n$ bit (k bit per rappresentare la parte intera e n bit per rappresentare la parte frazionaria):

$$B = b_{k-1} \dots b_0, b_{-1} \dots B_{-n}$$

- Il suo valore è dato da

$$V(B) = b_{k-1}x2^{k-1} + \dots + \underbrace{b_0x2^0 + b_{-1}x2^{-1} + \dots + b_{-n}x2^{-n}}_{\text{parte frazionaria}}$$

- Il fattore di scala che consente di passare dalla rappresentazione intera a quella a virgola fissa è pari a

$$S_n = 2^{-n} = 1 / 2^n$$

- Detti V_I il valore intero e V_{VF} il valore in virgola fissa di B:

$$V_{VF}(B) = V_I(B) \times S_n = V_I(B) \times 2^{-n}$$

Esempio

- Si consideri il vettore binario:

  $B = 010.10110$

- Il suo valore in virgola fissa è:

  $V_{VF}(B) = 2^1 + 2^{-1} + 2^{-3} + 2^{-4} = 2 + {}^1/_2 + {}^1/_8 + {}^1/_{16}$

  $= {}^{43}/_{16} = 2.6875$

- Il fattore di scala da utilizzare per la conversione è:

  $S_5 = 2^{-5} = {}^1/_{32} = 0.03125$

- Il valore di *B*, considerandolo intero è:

  $V_I(B) = 2^6 + 2^4 + 2^2 + 2^1 = 64 + 16 + 4 + 2 = 86$

- Da cui, moltiplicando per il fattore di scala, si ha:

  $V_{VF}(B) = V_I(B) \times S_5 = 86 \times 0.03125 = 2.6875$

Virgola fissa vs. virgola mobile

Intervallo di variazione di un numero binario di 32 bit

- Codifica intera

  $0 \le |V_I(B)| \le +2^{31} \approx 2.15 \times 10^9$

- Codifica a virgola fissa

  $+4.65 \times 10^{-10} \approx +2^{-31} \le |V_{VF}(B)| \le +1$

- A pari numero di bit disponibili
  - con la rappresentazione intera o in virgola fissa, i valori rappresentati sono distribuiti uniformemente nel campo di rappresentabilità
  - con la rappresentazione in virgola mobile, i valori rappresentati sono distribuiti non uniformemente nel campo di rappresentabilità
    - sono "più fitti" vicino allo 0 e "più radi" per valori assoluti grandi

- Nella rappresentazione in virgola mobile (floating point) la posizione della virgola è mobile ed è indicata dal valore di un fattore moltiplicativo

Errore di quantizzazione: virgola fissa vs. virgola mobile

Virgola fissa (con n bit per la parte frazionaria)

- $E_{Ass} = Val_{Vero} - Val_{Rappr}$ = costante
  con $(-1/2)2^{-n} < E_{Ass} < (+1/2)2^{-n}$
- $E_{Rel} = E_{Ass} / Val_{Vero}$
  (e cioè $E_{Rel}\, Val_{Vero}$ = costante)
- tanto più piccolo è il valore vero da rappresentare tanto maggiore è l'errore relativo che si commette nel rappresentarlo
- tanto più grande è il valore vero da rappresentare tanto minore è l'errore relativo che si commette nel rappresentarlo

Virgola mobile

- E_{Rel} = costante (= $2^{-\#bit\ della\ M}$)
- E_{Ass} = aumenta all' aumentare del valore valore vero da rappresentare

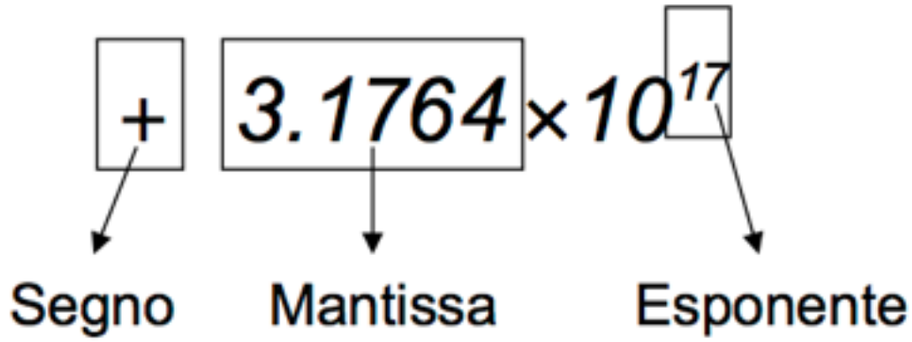
Esempio

- Numeri in virgola fissa
  - Dato 0.001 ed il suo successivo 0.002
    Errore percentuale:
    (0.002–0.001)/0.001*100 = 100%
  - Dato 100.001 ed il suo successivo 100.002
    Errore percentuale:
    (100.002–100.001)/100.001*100 = 0.001%

- Numeri in virgola mobile
  - Dato 0.128e–100 ed il suo successivo 0.129e–100
    Errore percentuale:
    ((0.129e–100–0.128e–100)/0.128e–100)*100 = 0.78125 %
  - Dato 0.128e+100 ed il suo successivo 0.129e+100
    Errore percentuale:
    ((0.129e+100–0.128e–+100)/0.128e+100)*100 = 0.78125 %

Numeri in virgola mobile

- Codifica in virgola mobile per i numeri in base 10
- Un numero in virgola mobile è composto da diverse parti:
- Si dice normalizzato un numero in cui $1 \leq M < 10$



- Facilmente estendibile al sistema di numerazione binario
- In un numero binario in virgola mobile e normalizzato
  - La prima cifra della mantissa è sempre 1 ($1 \leq M < 2$)
  - Tale cifra non viene rappresentata esplicitamente

Numeri in virgola mobile – Valori rappresentabili

- IEEE standard: Numeri floating-point in singola precisione

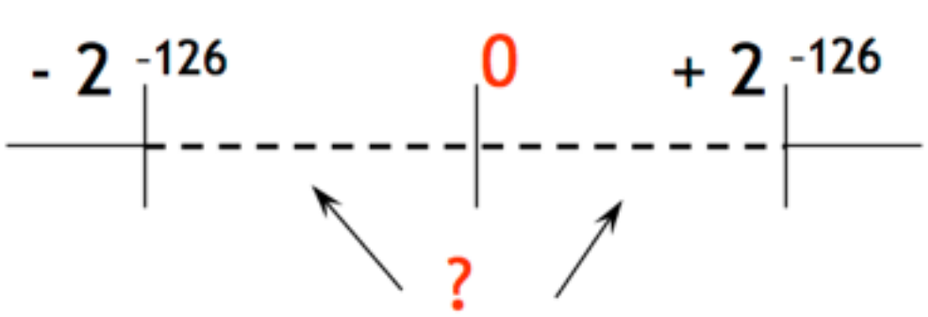
- L'esponente utilizza la codifica in *eccesso 127*, e cioè il valore effettivo dell'esponente è pari a (E–127)
  - *E = 0* e *M = 0* Rappresenta lo zero (pos/neg)
  - *E = 255* e *M = 0* Rappresenta infinito (pos/neg)
  - *E = 255* e *M !=0* *NotANumber*
  - 0<E<255 $(-1)^s \times 2^{(E-127)} \times (1,M)$ (127≤E≤254 esp.positivi 126≤E≤1esp.negativi)
  - *E = 0* e *M !=0* $(-1)^s \times 2^{-126} \times (0,M)$ non normalizzati
- Standard IEEE 32 bit: intervallo rappresentato $-1.M \times 10^{-38} \leq x \leq +1.M \times 10^{38}$
- La precisione consentita è di circa 7 cifre decimali

Numeri in virgola mobile: Valori rappresentabili

- Motivazione della rappresentazione non normalizzata
  - *E = 0* e *M !=0* $(-1)^s \times 2^{-126} \times (0,M)$ non normalizzati
- Il valore più piccolo rappresentabile normalizzato è

$$\pm 2^{1-127} \times 1,00\ldots00 = \pm 2^{-126}$$

- che espresso in virgola mobile da E=1 e M = 0



rappresentazione non normalizzata E=0 e M != 0

Interpretata nel modo seguente:

$$\text{Valore numerico} = \pm 2^{-126} \times 0,\ldots\ldots$$

Il più piccolo valore rappresentabile è

$$\pm 2^{-126} \times 0,00\ldots01 = \pm 2^{-126} \times 2^{-23} = \pm 2^{-149}$$

Operazioni in virgola mobile

- Le operazioni che si possono compiere su numeri in virgola mobile sono:
  - Somma
  - Sottrazione
  - Moltiplicazione
  - Divisione
  - Elevamento a potenza
  - Estrazione di radice

- Inoltre sono definite le operazioni di:
  - Normalizzazione
  - Troncamento

Operazioni in virgola mobile

- L'esecuzione di una operazione in virgola mobile può provocare una *eccezione*
- Una *eccezione* è il risultato di una operazione anomala, quale, ad esempio:
  - Divisione per zero
  - Estrazione della radice quadrata di un numero negativo

- Le eccezioni che vengono generate dalle unità aritmetiche in virgola mobile sono:
  - Operazione non valida
  - Divisione per zero
  - Overflow
  - Underflow

Op virgola mobile: *Normalizzazione*

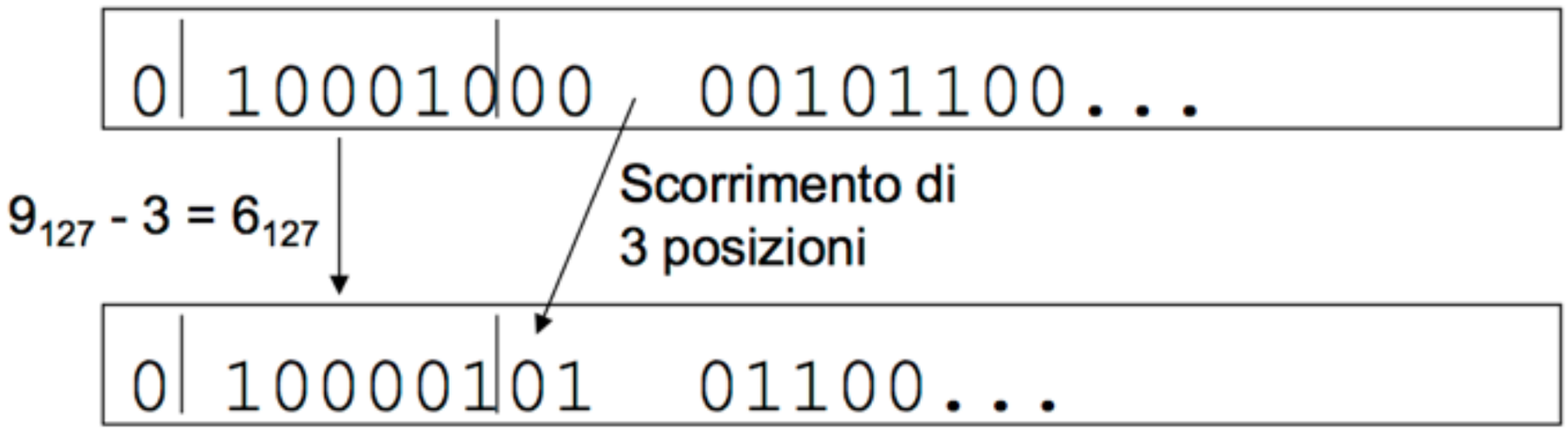
- Tutte le operazioni descritte nel seguito operano su numeri normalizzati (1 implicito prima della virgola)
- Se l'1 implicito manca, la normalizzazione di un numero con mantissa *M* ed esponente *n*, si esegue come segue:
  - Si fa scorrere verso sinistra la mantissa *M* fino al primo uno, compreso; sia *k* il numero di posizioni di tale scorrimento
  - Si sottrae *k* all'esponente *n*

Ricorda:

Scorrimento a sx equivale a moltiplicazione

Scorrimento a dx equivale a divisione

- Ad esempio:



Op virgola mobile: *Somma e Sottrazione*

- La somma o sottrazione tra numeri in virgola mobile viene eseguita secondo i seguenti passi:
  - Si sceglie il numero con esponente minore
  - Si fa scorrere la sua mantissa a destra un numero di bit pari alla differenza dei due esponenti
  - Si assegna all'esponente del risultato il maggiore tra gli esponenti degli operandi
  - Si esegue l'operazione di somma (algebrica) tra le mantisse per determinare il valore ed il segno del risultato
  - Si normalizza il risultato così ottenuto
    - Non sempre quest'ultima operazione è necessaria
    - Attenzione!!! Il riporto si può propagare anche dopo la posizione della virgola

Op virgola mobile: *Moltiplicazione*

- La moltiplicazione tra numeri in virgola mobile viene eseguita secondo i seguenti passi:
  - Si sommano gli esponenti e si sottrae 127
  - Si calcola il risultato della moltiplicazione delle mantisse
  - Si determina il segno del risultato
  - Si normalizza il risultato così ottenuto
    - Non sempre quest'ultima operazione è necessaria

- La sottrazione di 127 dalla somma degli esponenti è necessaria in quanto sono rappresentati in eccesso 127

$$E_{a,127} = E_a + 127$$
$$E_{b,127} = E_b + 127$$
$$E_{axb,127} = E_{axb} + 127 = (E_a + 127) + (E_b + 127) - 127$$

Op virgola mobile: *Divisione*

- La divisione tra numeri in virgola mobile viene eseguita secondo i seguenti passi:
  - Si sottraggono gli esponenti e si somma 127
  - Si calcola il risultato della divisione delle mantisse
  - Si determina il segno del risultato
  - Si normalizza il risultato così ottenuto
    - Non sempre quest'ultima operazione è necessaria

- La somma di 127 alla differenza degli esponenti è necessaria in quanto sono rappresentati in eccesso 127

$$E_{a,127} = E_a + 127$$
$$E_{b,127} = E_b + 127$$
$$E_{a/b,127} = E_{a/b} + 127 = (E_a + 127) - (E_b + 127) + 127$$

Op virgola mobile: *Troncamento*

- Spesso accade di rappresentare i risultati intermedi di una operazione con una precisione maggiore di quella degli operandi e del risultato
- Al termine dell'operazione è necessario effettuare una operazione di *troncamento*
- Il troncamento serve a rimuovere un certo numero di bit per ottenere una rappresentazione approssimata del risultato
- Si consideri il valore numerico rappresentato dal vettore:

$$B = 0.b_{-1} \ldots b_{-(k-1)} b_{-k} b_{-(k+1)} \ldots b_{-n}$$

- Si voglia effettuare troncamento al bit *k*–esimo

Op virgola mobile: *troncamento*

- Chopping
  - Consiste nell'ignorare i bit dal *k*–esimo all'*n*–esimo
  - Questo metodo è *polarizzato* o *biased*
  - L'errore è sempre positivo e varia nell'intervallo:

    $0 < \varepsilon < +(2^{-k+1} - 2^{-n})$

- Rounding
  - Se il bit k–esimo vale 0, lasciare invariato il bit in posizione (k–1) e ignorare i bit dal *k*–esimo all'*n*–esimo
  - Se il bit k–esimo vale 1, sommare 1 in posizione (k–1) e ignorare i bit dal *k*–esimo all'*n*–esimo
  - Questo metodo è *simmetrico* o *unbiased*
  - L'errore è centrato sullo zero e vale:

    $-(2^{-k+1} - 2^{-n}) < \varepsilon < +(2^{-k+1} - 2^{-n})$

Terza esercitazione: agenda

- (35’) Lezione su numeri in virgola fissa e mobile
- (5’) Pausa
- (30’) Es1: tabelle di verità
- (10’) Es2: conversioni B/D/E
- (10’) Es3: virgola fissa e mobile
- (10’) Es4: somma binaria algebrica CPL2
- (5’) Pausa
- (30’) Es5: l’anagrafica (strutture dati)
- (15’) Es6: ricerca anagrafica

Es1: tabelle di verità

Alcuni interessanti operatori logici, loro tabelle di verità e leggi logiche

(necessario applicarle se si pretende di fare un discorso *logico-razionale*, anche e soprattutto nella vita quotidiana)
(pressoché necessario *NON* applicarle nel caso in cui si voglia *convincere* una persona - vedi arte? politica)

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A nor B
0	0	1
0	1	0
1	0	0
1	1	0

A	B	A nand B
0	0	1
0	1	1
1	0	1
1	1	0

A	B	A ⇒ B
0	0	1
0	1	1
1	0	0
1	1	1

A	B	(A ⇒ B) and (B ⇒ A) ≡ A⇔B
0	0	1
0	1	0
1	0	0
1	1	1

Es1: tabelle di verità

Alcuni interessanti operatori logici, loro tabelle di verità e leggi logiche

(necessario applicarle se si pretende di fare un discorso *logico-razionale*, anche e soprattutto nella vita quotidiana)
(pressoché necessario *NON* applicarle nel caso in cui si voglia *convincere* una persona - vedi arte? politica)

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A or B
0	0	
0	1	
1	0	
1	1	1

Implicazione logica: se … allora …
Interessante osservare che la frase *complessiva* é vera anche quando A ("la premessa") é **falsa**.
Questo significa, tra le varie cose, che se una premessa é falsa **si può dimostrare vera qualsiasi cosa**.
Fate quindi attenzione ad avere sempre le premesse (ad esempio, dei vostri discorsi) veri se non volete essere in grado di dimostrare tanto affermazioni vere quanto quelle false

A	B	A nand B
0	0	1
0	1	1
1	0	1
1	1	0

A	B	A ⇒ B
0	0	1
0	1	1
1	0	0
1	1	1

A	B	(A ⇒ B) and (B ⇒ A) ≡ A⇔B
0	0	1
0	1	0
1	0	0
1	1	1

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure un numero che non é primo e contemporaneamente pari

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure un numero che non é primo e contemporaneamente pari

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure un numero che non é primo e contemporaneamente pari

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure un numero che non é primo^{C} e contemporaneamente pari^{A}

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure un numero che non é primo e contemporaneamente pari

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure un numero che non é primo^{C} e contemporaneamente pari^{A}

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure un numero che non é primo e contemporaneamente pari

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure un numero che non é primo^{C} e contemporaneamente pari^{A}

Ipotesi: l'utente deve inserire un intero pari^{A} <u>e</u> divisibile per 9^{B} <u>oppure</u> un numero che <u>non é</u> primo^{C} <u>e contemporaneamente</u> pari^{A}

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure un numero che non é primo e contemporaneamente pari

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure un numero che non é primo^{C} e contemporaneamente pari^{A}

Ipotesi: l'utente deve inserire un intero pari^{A} <u>e</u> divisibile per 9^{B} <u>oppure</u> un numero che <u>non é</u> primo^{C} <u>e contemporaneamente</u> pari^{A}

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure un numero che non é primo e contemporaneamente pari

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure un numero che non é primo^{C} e contemporaneamente pari^{A}

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure un numero che non é primo^{C} e contemporaneamente pari^{A}

D=(A and B) or(not(C) and A)

Es1: tabelle di verità

D=(A and B) or(not(C) and A)

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

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure primo; e contemporaneamente uno che non sia: pari e divisibile per 9 oppure primo

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure primo; e contemporaneamente uno che non sia: pari e divisibile per 9 oppure primo

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure primo; e contemporaneamente uno che non sia: pari e divisibile per 9 oppure primo

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure primo^{C}; e contemporaneamente uno che non sia: pari^{A} e divisibile per 9^{B} oppure primo^{C}

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure primo; e contemporaneamente uno che non sia: pari e divisibile per 9 oppure primo

Ipotesi: l'utente deve inserire un intero $pari^A$ e $divisibile\ per\ 9^B$ oppure $primo^C$; e contemporaneamente uno che non sia: $pari^A$ e $divisibile\ per\ 9^B$ oppure $primo^C$

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure primo; e contemporaneamente uno che non sia: pari e divisibile per 9 oppure primo

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure primo^{C}; e contemporaneamente uno che non sia: pari^{A} e divisibile per 9^{B} oppure primo^{C}

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure primo^{C}; e contemporaneamente uno che non sia: pari^{A} e divisibile per 9^{B} oppure primo^{C}

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure primo; e contemporaneamente uno che non sia: pari e divisibile per 9 oppure primo

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure primo^{C}; e contemporaneamente uno che non sia: pari^{A} e divisibile per 9^{B} oppure primo^{C}

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure primo^{C}; e contemporaneamente uno che non sia: pari^{A} e divisibile per 9^{B} oppure primo^{C}

Es1: tabelle di verità

Ipotesi: l'utente deve inserire un intero pari e divisibile per 9 oppure primo; e contemporaneamente uno che non sia: pari e divisibile per 9 oppure primo

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure primo^{C}; e contemporaneamente uno che non sia: pari^{A} e divisibile per 9^{B} oppure primo^{C}

Ipotesi: l'utente deve inserire un intero pari^{A} e divisibile per 9^{B} oppure primo^{C}; e contemporaneamente uno che non sia: pari^{A} e divisibile per 9^{B} oppure primo^{C}

$$D=[(A \text{ and } B) \text{ or } C] \text{ and } \{\text{not}[(A \text{ and } B) \text{ or } C]\}$$

Es1: tabelle di verità

D=[(A and B) or C] and {not[(A and B) or C]}

A	B	C	A and B	(A and B) or C	not[(A and B) or C]	[A and B) or C] and {not[(A and B) or C]}
0	0	0	0	0	1	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	1	0	0
1	0	0	0	0	1	0
1	0	1	0	1	0	0
1	1	0	1	1	0	0
1	1	1	1	1	0	0

Es1: tabelle di verità

D=[(A and B) or C] and {not[(A and B) or C]}

A	B	C	A and B	(A and B) or C	not[(A and B) or C]	[A and B) or C] and {not[(A and B) or C]}
0	0	0	0	0	1	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	1	0	0
1	0	0	0	0	1	0
1	0	1	0	1	0	0
1	1	0	1	1	0	0
1	1	1	1	1	0	0

Morale: leggete il testo prima di lanciarvi sui calcoli

Terza esercitazione: agenda

- (35’) Lezione su numeri in virgola fissa e mobile
- (5’) Pausa
- (30’) Es1: tabelle di verità
- (10’) Es2: conversioni B/D/E
- (10’) Es3: virgola fissa e mobile
- (10’) Es4: somma binaria algebrica CPL2
- (5’) Pausa
- (30’) Es5: l’anagrafica (strutture dati)
- (15’) Es6: ricerca anagrafica

Es2: Conversioni B/D/E

Convertire 41_{10} e 18_{10} in binario e farne la somma, quindi verificare che il risultato ottenuto sia coerente con il risultato ottenuto in codifica decimale

Es2: Conversioni B/D/E

Convertire 41_{10} e 18_{10} in binario e farne la somma, quindi verificare che il risultato ottenuto sia coerente con il risultato ottenuto in codifica decimale

41:

32	16	8	4	2	1
1	0	1	0	0	1

32 + 8 + 1 = 41

18:

16	8	4	2	1
1	0	0	1	0

16 + 2 = 18

Ossia, si sommano progressivamente le potenze del 2 fino a raggiungere il valore desiderato

(Si può vedere come una sottrazione della potenza del 2 dal numero originario 1) solo se la sottrazione non rende il risultato negativo 2) ripetuta fino al raggiungimento dello 0)

Es2: Conversioni B/D/E

Convertire 41_{10} e 18_{10} in binario e farne la somma, quindi verificare che il risultato ottenuto sia coerente con il risultato ottenuto in codifica decimale

Es2: Conversioni B/D/E

Convertire 41_{10} e 18_{10} in binario e farne la somma, quindi verificare che il risultato ottenuto sia coerente con il risultato ottenuto in codifica decimale

$$\begin{array}{cccccc} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 1 & 1 & 1 & 0 & 1 & 1 \end{array} +$$

Ossia…

$32+16+8+0+2+1=59_{10}$ 😄

Es2: Conversioni B/D/E

Convertire 43_{10} e 18_{10} in binario e farne la somma, quindi verificare che il risultato ottenuto sia coerente con il risultato ottenuto in codifica decimale

Es2: Conversioni B/D/E

Convertire 43_{10} e 18_{10} in binario e farne la somma, quindi verificare che il risultato ottenuto sia coerente con il risultato ottenuto in codifica decimale

$$\begin{array}{cccccc} 1 & 0 & 1 & 0^{1} & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 & 0 & 1 \end{array} +$$

Il riporto si tratta analogamente alla somma tradizionale

$32+16+8+4+0+1=61_{10}$ 😄

Es2: Conversioni B/D/E

Convertire 43_{10} e 18_{10} in binario e farne la somma, quindi verificare che il risultato ottenuto sia coerente con il risultato ottenuto in codifica decimale

Se capita qui?
Overflow
Perché é importante? Perché nei circuiti digitali implica impossibilità di rappresentazione “corretta” del risultato (in realtà se si assume algebra *modulare*, il risultato é corretto)

$$\begin{array}{cccccccc} 1 & 0 & 1 & 0^{1} & 1 & 1 & + \\ 0 & 1 & 0 & 0 & 1 & 0 & \\ \hline 1 & 1 & 1 & 1 & 0 & 1 & \\ \hline \end{array}$$

Il riporto si tratta analogamente alla somma tradizionale

$32+16+8+4+0+1=61_{10}$ 😄

Es2: Conversioni B/D/E

Convertire 41_{10} in formato esadecimale

0 1 2 3 4 5 6 7 8 9 A B C D E F

La rappresentazione binaria era 101001_2

Un modo veloce per effettuare la conversione da binario a esadecimale é considerare **gruppi di 4 bit** per effettuare la conversione.

41_{10} => 101001_2 => $10_2/1001_2$ => $2_{16}/9_{16}$ => 29_{16}

Es2: Conversioni B/D/E

Convertire 191_{10} in formato esadecimale

0 1 2 3 4 5 6 7 8 9 A B C D E F

La rappresentazione binaria é 10111111_2

Un modo veloce per effettuare la conversione da binario a esadecimale é considerare **gruppi di 4 bit** per effettuare la conversione.

$191_{10} => 10111111_2 => 1011_2/1111_2 => B_{16}/F_{16} => BF_{16}$

Es2: Conversioni B/D/E

Convertire BF_{16} in formato decimale

0 1 2 3 4 5 6 7 8 9

$BF_{16} => B_{16}/F_{16} => 11_{10}*16^{1+}\ 15_{10}*16^{0} = 176_{10}+15_{10}=191_{10}$

A_{16} vale 10_{10}
B_{16} vale 11_{10}
C_{16} vale 12_{10}
etc…

Terza esercitazione: agenda

- (35’) Lezione su numeri in virgola fissa e mobile
- (5’) Pausa
- (30’) Es1: tabelle di verità
- (10’) Es2: conversioni B/D/E
- (10’) Es3: virgola fissa e mobile
- (10’) Es4: somma binaria algebrica CPL2
- (5’) Pausa
- (30’) Es5: l’anagrafica (strutture dati)
- (15’) Es6: ricerca anagrafica

Es3: Virgola fissa (dec2bin)

Convertire in binario : 0,125

				riporto
0,125	· 2	=	0,25	0
0,25	· 2	=	0,50	0
0,50	· 2	=	1,00	1
0,00	· 2	=	0,00	FINE

$(0,125)_{10} = (0,001)_2$

Convertire in binario : 0,375

				riporto
0,375	· 2	=	0,75	0
0,75	· 2	=	1,5	1
0,5	· 2	=	1,0	1
0,0	· 2	=	0	FINE

$(0,125)_{10} = (0,011)_2$

(credits: Ing. A.A. Nacci)

Es3: Virgola fissa (dec2bin)

Esercizio 10

Convertire 17,55 in binario (8 cifre decimali, troncamento

$(17)_{10} \rightarrow (10001)_2$

$(0,55)_{10} \rightarrow ?$

		riporto
$0,55 \cdot 2 = 1,10$		1
$0,10 \cdot 2 = 0,20$		0
$0,20 \cdot 2 = 0,40$		0
$0,40 \cdot 2 = 0,80$		0
$0,80 \cdot 2 = 1,60$		1
$0,60 \cdot 2 = 1,20$		1
$0,20 \cdot 2 = 0,40$		0
$0,40 \cdot 2 = 0,80$		0

→ Troncamento

Quindi $(14,55)_{10} \rightarrow (10001, 10001100)_2$

(credits: Ing. A.A. Nacci)

Es3: Virgola fissa (bin2dec)

Convertire il decimale: $11,11$

Parte intera : $I = (11)_2 = (1 \cdot 2 + 1)_{10} = (3)_{10}$

Parte decimale : $D = (11)_2 = 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = \frac{1}{2} + \frac{1}{4} = 0{,}75$

$$(11,11)_2 = (3{,}75)_{10}$$

(credits: Ing. A.A. Nacci)

Es3: Virgola mobile (bin2dec)

Esercizio 13 : codificare secondo lo standard IEEE a precisione singola il seguente numero decimale

$X = 0{,}875 \rightarrow 0{,}111 = 1{,}11 \cdot 2^{(-1)} = m$

S = 0 (+) 1 bit

E = (-1) = m + K = -1 + 127 = 126 8 bit

M = 1100 0000 0000 0000 0000 000 23 bit

(credits: Ing. A.A. Nacci)

Es3: Virgola mobile (dec2bin)

Esercizio 14 : convertire in base 10 il seguente numero espresso secondo la codifica floating point

$S = 0$

$M = 10010011$ 0000000 0000000

$E = 10000100$

Troviamo prima questo valore utilizzando l'espressione.

$E = (10000100)_{2} = 132 = m + K \Rightarrow \quad m = 132 - 127 = 5$

Poi sappiamo che il numero sarà quindi:

$+ 1,\underbrace{10010011}_{5} \cdot 2^{5} \longrightarrow \underbrace{110010}_{\downarrow 50},\underbrace{011}_{\downarrow}$

$0 \cdot 2^{-1} + 2^{-2} + 2^{-3} = 0,375$

Quindi il numero è: $\boxed{X = 50,375}$

(credits: Ing. A.A. Nacci)

Terza esercitazione: agenda

- (35’) Lezione su numeri in virgola fissa e mobile
- (5’) Pausa
- (30’) Es1: tabelle di verità
- (10’) Es2: conversioni B/D/E
- (10’) Es3: virgola fissa e mobile
- (10’) Es4: somma binaria algebrica CPL2
- (5’) Pausa
- (30’) Es5: l’anagrafica (strutture dati)
- (15’) Es6: ricerca anagrafica

Es4: Somma binaria algebrica CPL2

Convertire 5_{10} e -13_{10} in binario e farne la somma, quindi verificare che il risultato ottenuto sia coerente con il risultato ottenuto in codifica decimale

Segno +

$(5)_{10}=101_2=0101_{2\text{ con segno }+}$ => fine

$(13)_{10}=1101_2=>01101_{2\text{ con segno }+}=>10010_{CPL1}=>10011_{CPL2}$ (ossia, -13)

Introduzione del segno +, che a valle della CPL2 diventa -

Segno +, che a valle della CPL2 diventa -

Ricordatevi che a valle della CPL2, il numero cambia di segno.

Es4: Somma binaria algebrica CPL2

Convertire 5_{10} e -13_{10} in binario e farne la somma, quindi verificare che il risultato ottenuto sia coerente con il risultato ottenuto in codifica decimale

Es4: Somma binaria algebrica CPL2

Convertire 5_{10} e -13_{10} in binario e farne la somma, quindi verificare che il risultato ottenuto sia coerente con il risultato ottenuto in codifica decimale

$$\begin{array}{ccccc} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 & 0 \end{array} +$$

Quanti bit considero? C'é overflow? Se si, quanti bit *non* considero?

Es4: Somma binaria algebrica CPL2

Quanti bit considero?

5_{10} - 3_{10} => parto con 3 bit, arrivo con 2 bit (5_{10}-3_{10}=2_{10} si rappresenta con 2 bit, ossia 10_2)
No overflow!

5_{10} + 5_{10} => parto con 3 bit, arrivo con 4 bit (5_{10}+5_{10}=10_{10}, si rappresenta con 4 bit ossia 1010_2)
Si overflow!

5_{10}-13_{10}=-8_{10} quindi solo 4 bit (non si considera il segno nel calcolo, 8_{10} é 1004bit0_2 ossia 4bit)

Es4: Somma binaria algebrica CPL2

Convertire 5_{10} e -13_{10} in binario e farne la somma, quindi verificare che il risultato ottenuto sia coerente con il risultato ottenuto in codifica decimale

Es4: Somma binaria algebrica CPL2

Convertire 5_{10} e -13_{10} in binario e farne la somma, quindi verificare che il risultato ottenuto sia coerente con il risultato ottenuto in codifica decimale

$$\begin{array}{ccccc} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ \hline 1_{\text{cade}} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \end{array} +$$

Il maggiore dei due numeri aveva il segno -, per cui il risultato é **negativo**; inoltre, 1000_2 vale 8_{10}, per cui la somma (algebrica) vale -8_{10}

Terza esercitazione: agenda

- (35’) Lezione su numeri in virgola fissa e mobile
- (5’) Pausa
- (30’) Es1: tabelle di verità
- (10’) Es2: conversioni B/D/E
- (10’) Es3: virgola fissa e mobile
- (10’) Es4: somma binaria algebrica CPL2
- (5’) Pausa
- (30’) Es5: l’anagrafica (strutture dati)
- (15’) Es6: ricerca anagrafica

Es5: l'anagrafica (strutture dati)

Si ipotizzi di dover gestire l'anagrafica del proprio Comune.
Si definisca un tipo di dato opportuno per rappresentare una *persona*. Il tipo di dato dovrà avere un campo **nome**, **cognome**, **data di nascita** e **codice fiscale**. La data di nascita, inoltre, é a sua volta un tipo di dato che contiene un campo **giorno**, **mese** ed **anno**.
Si scriva quindi un frammento di codice che chieda all'utente di inserire i dati di un individuo e riempia una struttura dati di tipo *persona*.

Terza esercitazione: agenda

- (35’) Lezione su numeri in virgola fissa e mobile
- (5’) Pausa
- (30’) Es1: tabelle di verità
- (10’) Es2: conversioni B/D/E
- (10’) Es3: virgola fissa e mobile
- (10’) Es4: somma binaria algebrica CPL2
- (5’) Pausa
- (30’) Es5: l’anagrafica (strutture dati)
- (15’) Es6: ricerca anagrafica

Es6: ricerca nell'anagrafica

Si ipotizzi di dover gestire l'anagrafica del proprio Comune.
Effettuare una ricerca all'interno di un array di persone (ipotizzato già caricato in precedenza) dell'individuo con un dato codice fiscale (ipotizziamo che sia stato richiesto all'utente, non é necessario effettuare un controllo sulla validità del CF inserito).