Architetture degli Elaboratori

Luca

14, Novembre 2020

Indice

Conversione Binario - Ottale - Decimale - Esadecimale

Esercizi: 1) Convertire i seguenti numeri decimali in base 2: $371, 3224, 114.65625$

$\mathbf{371} = 2^8 + 115 \rightarrow 115 = 2^6 + 51 \rightarrow 51 = 2^5 + 19 \rightarrow 19 = 2^4 + 3$

$3 = 2^1 + 1 \rightarrow 1 = 2^0$

$2^8 + 2^6 + 2^5 + 2^4 + 2^1 + 2^0 = \mathbf{371}$

$\mathbf{371_{10}} = 101110011_2$ METODO 1

METODO 2

371	
185	1
92	1
46	0
23	0
11	1
5	1
2	1
1	0
0	1

↑ Dal basso verso l'Alto

$\mathbf{371_{10} = 101110011_2}$

3224

3224	
1612	0
806	0
403	0
201	1
100	1
50	0
25	0
12	1
6	0
3	0
1	1
0	1

$\mathbf{3224_{10} = 110010011000_2}$

114,65625

$$\begin{aligned}
114,65625 &= 2^6 + 50,65625 \\
50,65625 &= 2^5 + 18,65625 \\
18,65625 &= 2^4 + 2,65625 \\
2,65625 &= 2^1 + 0,65625 \\
0,65625 &= 2^{-1} + 0,15625 \\
0,15625 &= 2^{-3} + 0,03125 \\
0,03125 &= 2^{-5}
\end{aligned}$$

$$\mathbf{114,65625_{10} = 1110010,10101_2}$$

2) Convertire i seguenti numeri in base 8 e 16

$$\underbrace{11}_{2^1+2^0}\ \underbrace{100}_{4}\ \underbrace{110}_{6}\ \underbrace{100}_{2^2}\ \underbrace{110}_{2^2+2^1}$$

Raggruppo in gruppi di 3 (questo per la base 8) 8 è 2^3
$\mathbf{11100110100110_2 = 34646_8}$

$$\underbrace{11}_{2^1+2^0}\ \underbrace{1001}_{9}\ \underbrace{1010}_{10=A}\ \underbrace{0110}_{2^2+2^1}$$

Raggruppo in gruppi di 4 (questo per la base 16) 16 è 2^4
$\mathbf{11100110100110_2 = 39A6_{16}}$

3) Convertire i numeri esadecimali in binario

$\boxed{F}\boxed{A}\boxed{3}\boxed{1}\boxed{C}$ Una cifra può essere rappresentata da 4 cifre binarie

$$\begin{aligned}
15_{10} &= 2^3 + 2^2 + 2^1 + 2^0 = 1111_2 \\
A_{16} &= 10_{10} = 2^3 + 2^1 = 1010_2 \\
3_{16} &= 3_{10} = 2^1 + 2^0 = 0011_2 \\
1_{16} &= 1_{10} = 2^0 = 0001_2 \\
C_{16} &= 12_{10} = 2^3 + 2^2 = 1100_2
\end{aligned}$$

$\mathbf{FA31C_{16} = 11111010001100011100_2}$

4) Convertire da esadecimale a decimale

AAB_{16}

$$
\begin{aligned}
B \cdot 16^0 &= 11 \cdot 1 = 11 \\
A \cdot 16^1 &= 10 \cdot 16 = 160 \\
A \cdot 16^2 &= 10 \cdot 256 = 2560
\end{aligned}
$$

$12 + 160 + 2560 = 2731$

$\mathbf{AAB_{16} = 2731_{10}}$

Complemento a 2 - Eccesso 128 e 256 - Mascherature dei bits

5) Quanti valori sono codificabili con 2-4-7-10 bit?

N bit $\Longrightarrow 2^N$

6) Per codificare 1598 numeri quanti bit/byte sono necessari?

$2^N \geq 1598$
$2^{10} = 1024$
$2^{11} = 2048 > 1598$
$2 \text{ byte } = 16 \text{ bit}$
Il byte o lo si prende tutto o niente.
$2^{16} = 65536 > 1598$

7) Quante cifre binarie sono necessarie per codificare X numeri distinti ($log_2X = log_{10}X/log_{10}2$)?

8) Si convertono i seguenti numeri decimali in numeri binari in complemento a 2: $-56, -88, 243$ Si utilizzino 8 bit per rappresentare il risultato?

9) Si convertano i seguenti numeri decimali in numeri binari in eccesso 128 e 256. $-56, -37, -243$

10) Si pongano a 1 bit di posizione dspari del byte. 11010011

$$\begin{array}{cccccccc l} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & OR \\ \hline 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & \end{array}$$

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Poteva andare bene anche questa mascheratura:

$$\begin{array}{cccccccc l}
1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & \textit{OR} \\
\hline
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 &
\end{array}$$

11) Si pongano a 0 i primi 4 bit del byte. 11010011

$$\begin{array}{cccccccc l}
1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & \text{AND} \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 &
\end{array}$$

$$\begin{array}{cccccccc l}
1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \text{AND} \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 &
\end{array}$$

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

I primi 4 bit da sinistra o da destra? Qui l'abbiamo fatto in entrambi i modi.

12) Si verifichi che il terzo bit del byte 11001100 è 1.

$$\begin{array}{cccccccc l}
{\scriptstyle 7} & {\scriptstyle 6} & {\scriptstyle 5} & {\scriptstyle 4} & {\scriptstyle 3} & {\scriptstyle 2} & {\scriptstyle 1} & {\scriptstyle 0} & \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \text{AND} \\
\hline
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 &
\end{array}$$

Il terzo bit a sinistra sarebbe quello in posizione 2 (in rosso)

Controllo tra risultato e input

$$\begin{array}{cccccccc l}
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \textit{OR} \\
\hline
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 &
\end{array}$$

13) Si eseguano le seguenti somme in base 2:

$12_{10} + 7_{10} \quad 67_{10} + 38_{10} \quad 89_{10} + 147_{10}$

$$\begin{array}{l}
12 = 2^3 + 2^2 \\
+7 = 2^2 + 2^1 + 2^0 \\
19
\end{array}
\qquad
\begin{array}{ccccc l}
 & 1 & 1 & 0 & 0 & \\
 & & 1 & 1 & 1 & \qquad + \\
\hline
1 & 0 & 0 & 1 & 1 & = 2^4 + 2^1 + 2^0 = 16 + 2 + 1 = \mathbf{19}
\end{array}$$

$$67+38=105$$
$$67=2^6+2^1+2^0$$
$$38=2^5+2^2+2^1$$

$$\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ \hline 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array} + \quad = 2^6+2^5+2^3+2^0 = 64+32+8+1 = \mathbf{105}$$

$$89+147=236$$
$$89=2^6+2^4+2^3+2^0$$
$$147=2^7+2^4+2^2+2^0$$

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

$$=2^7+2^6+2^5+2^3+2^2=128+64+32+8+4=\mathbf{236}$$

14) Si eseguano le seguenti differenze in base 2 (complemento a 2, utilizzando 8 bit per rappresentare i numeri): $8_{10}-17_{10}$ $37_{10}-23_{10}$ $5_{10}-117_{10}$

- Complemento a due:
  - Il riporto dei bit più a sinistra viene ignorato.
  - 8 bit $[-128;+127]$
  - Se gli addendi sono di segno opposto non si può verificare un overflow.
  - L' overflow si verifica se il riporto generato è diverso dal riporto utilizzato. Insomma se i primi due bit a sinistra sono diversi, c'è un overflow.

$$8-17=-9 \Longrightarrow 8+(-17)$$
$$8=2^3=000001000$$
$$-17 \rightarrow 17=2^4+2^0=00010001$$

Ora invertiamo i bits del 17 e li sommiamo ad 1 (quelli in blu)

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

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

Quei due 0 in verde indicano che non c'è l'overflow, perchè i primi due bit a sinistra (del riporto) sono uguali e non sono diversi.

Quel uno rosso a sinistra (ovvero il primo bit a sinistra) indica che il segno deve essere un meno - . Mentre se fosse stato uno zero, il segno sarebbe stato positivo.

Ora prendiamo il risultato che abbiamo trovato quello colorato in azzurro e lo invertiamo (ovvero dove c'è zero mettiamo 1 e viceversa e poi lo sommiamo a 1.)

$$\begin{array}{cccccccccc} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & & \\ & & & & & & & 1 & & + \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & = 2^3 + 2^0 = 9 & \end{array}$$

Abbiamo ottenuto 9, ma visto che nell' operazione precedente, c'era quell'1 (colorato in rosso) come primo bit a sinistra; ciò significa che il valore del segno deve essere negativo, ovvero un meno - .
E quindi quel 9 positivo diventa, come risultato finale -9

Quindi, riepilogando i passaggi che abbiamo fatto:
Noi dovevamo fare 8 - 17
Abbiamo negato il 17. La negazione in complemento a due, richiede due passaggi:

- Sostituzione di tutti gli uno con degli zero e viceversa.
- Si aggiunge 1 al risultato.

Dopichè abbiamo fatto 8 + (-17) e abbiamo trovato il risultato, il primo bit a sinistra indicava il segno (positivo o negativo).
Abbiamo poi negato il risultato (invertendo i bit a zero e a 1 e sommando 1) e abbiamo trovato 9, che è diventato -9 perchè il primo bit a sinistra (dell'operazione 8 + (-17)) era un 1.

$37-23=37+(-23)=14$

$37=2^5+2^2+2^0$

$-23=23=2^4+2^2+2^1+2^0=00010111$

Ora **nego** il 23 per ottenere -23 (e quindi inverto i bit 0 e 1 e sommo 1.)

$$\begin{array}{cccccccccl} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 23 & \textbf{BITS INVERTITI} \\ & & & & & & & 1 & & \quad + \\ \hline 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & & 23 \ \textbf{NEGATO} \end{array}$$

Ora facciamo + 37 + (-23)

$$\begin{array}{ccccccccl} {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 1} & \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & (+37)+ \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & (-23) \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & \end{array}$$

Quel doppio 1 al riporto viene ignorato.
Quello 0 nel risultato indica che il risultato è di segno positivo.
$2^3+2^2+2^1=8+4+2=\mathbf{14}$

15) Quali numeri decimali sono codificabili in formato floating point utilizzando 5 cifre con segno per la mantissa e 3 cifre con segno per l'esponente?

$-0,9999\cdot10^{999}$ **underflow negativo** $+0,1\cdot10^{-999}$ **overflow positivo**

overflow negativo $-0,1\cdot10^{-999}$ 0 **underflow positivo** $+0,99999\cdot10^{999}$

16) Si calcoli l'errore di arrotondamento per la rappresentazione floating point decimale (3 cifre con segno per la mantissa e 2 cifre con segno per l'esponente) dei numeri: $1598,58978922 \quad -0,568282$

$v_1<v<v_2 \quad E=\min(|v-v_1|;|v-v_2|)$

$\underbrace{0,159\cdot10^4}_{1590}<1598<\underbrace{0,160\cdot10^4}_{1600}$

$|1598-1590|=8$

$|1598-1600|=2$

$\underbrace{0,589\cdot10^8}_{58900000}<58978922<\underbrace{0,590\cdot10^8}_{59000000}$

$\mathbf{E}=21078$

17) Si transformino in formato IEEE 754 in singola precisione i numeri: $1598, \quad -0,56640625$

$n = f \cdot 10^e$ (f = frazione o mantissa , e = esponente)

$1598_{10} = 11000111110_2$

- Ora normalizzo la mantissa, ovvero sposto la virgola al primo bit a sinistra.

$$1,1000111110 \cdot 2^{10}$$

- Esprimo l'esponente (il 10, esponente sopra il 2) in eccesso 127

$$10 + 127 = 137_{10} = 10001001_2$$

Rappresento i dati con il corretto numero di bit

$$\underbrace{0100}_{4}\ \underbrace{0100}_{4}\ \underbrace{1100}_{C}\ \underbrace{0111}_{7}\ \underbrace{1100}_{C}\ \underbrace{0000}_{0}\ \underbrace{0000}_{0}\ \underbrace{0000}_{0}$$

Il primo bit in giallo riguarda il segno (se 0 è positivo, se 1 è negativo.)

Questa è la codifica del numero 1598_{10} in IEEE 754, NON è il numero esadecimale per 1598.

$-0,56640625_{10} = 0,10010001_2$

Ora metto la virgola dopo il primo 1 a sinistra

$= 1,0010001 \cdot 2^{-1}$

Ora esprimo l'esponente in eccesso 127

$-1 + 127 = 126_{10} = 01111110_2$

$$\underbrace{1011}_{B}\ \underbrace{1111}_{F}\ \underbrace{0001}_{1}\ \underbrace{0001}_{1}\ \underbrace{0000}_{0}\ \underbrace{0000}_{0}\ \underbrace{0000}_{0}\ \underbrace{0000}_{0}$$

Aggiungere zeri se non si arriva a 23 bits (infatti gli ultimi zeri sono stati aggiunti).

18) Si calcoli il valore decimale corrispondente ai seguenti numeri in formato IEEE 754 in singola precisione.

$$\underbrace{0}_{\text{segno }+}\ \underbrace{10011001}_{153-127=26\text{ (esponente)}}\ \underbrace{10111011110000000000000}_{1,101110111\cdot 2^{26}=116260864\text{ (mantissa normalizzata)}}$$

Gli altri zeri nella mantissa non ci servono, quindi non li consideriamo, il numero sarebbe: $1,10111011110000000000000$, ma noi consideriamo solo fino all'ultimo 1 e quindi scriviamo: $\mathbf{1,101110111 \cdot 2^{26}}$

Esercizi Assembly

Usare le ultime 4 cifre (meno significative) della propria matricola
unsigned short int Mat (Matricola) $= 2602_{10} =$
$0A2A_{16}$ **(questa non è la mia matricola)**
1)

```
MOV AX, Mat
AND AX, 00ffh
NEG AX
MOV CX, – 4
SUB AX, CX
MOV Ris1, AX
```

1. MOV AX, Mat
   - $AX = 0A2A_{16}$ AX è un registro a 16 bit
   - $AX = 002A_{16} = 2 \cdot 16^1 + 10$ (che sarebbe A) $\cdot 16^0 = 42_{10}$
2. AND AX, 00ffh
   - h sta per hexadecimal (esadecimale)
   - $$\begin{array}{cccc} 0 & A & 2 & A \\ 0 & 0 & F & F \\ \hline 0 & 0 & 2 & A \end{array} \quad \text{AND}$$
   - Oppure possiamo scriverlo in base 2:

$$\begin{array}{cccccccccccccccc} 0&0&0&0&1&0&1&0&0&0&1&0&1&0&1&0 \\ 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ \hline 0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&0 \end{array} \quad \text{AND}$$

3. NEG AX

   - Ora neghiamo il numero invertendo i bits e sommando 1.

$$
\begin{array}{cccccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
 & & & & & & & & & & & & & & & 1 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0
\end{array}
$$

Ora suddividiamo i bits in gruppetti da 4 per ottenere il numero in esadecimale così otteniamo FFD6 ,ovvero il numero -42_{10}

4. MOV CX, -4

$$CX = -4$$

5. SUB AX, CX

$AX = -42_{10}$
$CX = -4_{10}$
$AX = -38_{10}$ (L'operazione che abbiamo fatto è stata $-42-(-4)$)
$AX = \text{FFDA}_{16}$

MOV Ris1, AX

$$\text{Ris1} = \text{AX} = \boxed{-38}$$

2)

XOR EAX, EAX
MOV DX, Mat
MOV AX, Mat
SHL EDX, 16
OR EDX, EAX
BSWAP EDX
ROL EDX, 6
MOV Ris2, DX

1. XOR EAX, EAX
    - Azzera tutti i bits di EAX
2. MOV DX, Mat
    - DX $= 2602_{10} = 0A2A_{16}$
3. MOV AX, Mat
    - AX $= 2602_{10} = 0A2A_{16}$
4. SHL EDX, 16
    - SHL sta per "Shift Left"
      ciò significa che spostiamo 16 bits a sinistra, i bits che fuoriescono vengono persi.
    - EDX prima dello shift
      $????????????????\underbrace{0\,0\,0\,0}_{0}\underbrace{1\,0\,1\,0}_{A}\underbrace{0\,0\,1\,0}_{2}\underbrace{1\,0\,1\,0}_{A}$
    - EDX dopo lo shift
      $\underbrace{0\,0\,0\,0}_{0}\underbrace{1\,0\,1\,0}_{A}\underbrace{0\,0\,1\,0}_{2}\underbrace{1\,0\,1\,0}_{A}\underbrace{0\,0\,0\,0}_{0}\underbrace{0\,0\,0\,0}_{0}\underbrace{0\,0\,0\,0}_{0}\underbrace{0\,0\,0\,0}_{0}$
5. OR EDX, EAX
    - EDX
      $\underbrace{0\,0\,0\,0}_{0}\underbrace{1\,0\,1\,0}_{A}\underbrace{0\,0\,1\,0}_{2}\underbrace{1\,0\,1\,0}_{A}\underbrace{0\,0\,0\,0}_{0}\underbrace{0\,0\,0\,0}_{0}\underbrace{0\,0\,0\,0}_{0}\underbrace{0\,0\,0\,0}_{0}$
    - EAX
      $\underbrace{0\,0\,0\,0}_{0}\underbrace{0\,0\,0\,0}_{0}\underbrace{0\,0\,0\,0}_{0}\underbrace{0\,0\,0\,0}_{0}\underbrace{0\,0\,0\,0}_{0}\underbrace{1\,0\,1\,0}_{A}\underbrace{0\,0\,1\,0}_{2}\underbrace{1\,0\,1\,0}_{A}$
      EAX l'avevo xorato nella prima operazione, ciò significa che tutti i bits si sono azzerati e quindi non sono come quelli di EDX , un'incognita (quelli col punto interrogativo) ma sono a zero.
    - OR EDX, EAX
      $\underbrace{0\,0\,0\,0}_{0}\underbrace{1\,0\,1\,0}_{A}\underbrace{0\,0\,1\,0}_{2}\underbrace{1\,0\,1\,0}_{A}\underbrace{0\,0\,0\,0}_{0}\underbrace{1\,0\,1\,0}_{A}\underbrace{0\,0\,1\,0}_{2}\underbrace{1\,0\,1\,0}_{A}$
6. BSWAP EDX
    - BSWAP converte little-endian / big-endian
      $\underbrace{0\,A}_{b3}\underbrace{2\,A}_{b2}\underbrace{0\,A}_{b1}\underbrace{2\,A}_{b0}$ prima dello swap
      $\underbrace{2\,A}_{b0}\underbrace{0\,A}_{b1}\underbrace{2\,A}_{b2}\underbrace{0\,A}_{b3}$ dopo lo swap

7. ROL EDX, 6

    - La ROL che sta per "Rotate Left" serve per spostare i bits a sinistra, i bits che fuoriescono **non** vengono persi, tornano dall'altra parte.

$$\underbrace{0\,0\,1\,0}_{2}\,\underbrace{1\,0\,1\,0}_{A}\,\underbrace{0\,0\,0\,0}_{0}\,\underbrace{1\,0\,1\,0}_{A}\,\underbrace{0\,0\,1\,0}_{2}\,\underbrace{1\,0\,1\,0}_{A}\,\underbrace{0\,0\,0\,0}_{0}\,\underbrace{1\,0\,1\,0}_{A}\ \text{Prima della ROL}$$

$$\underbrace{1\,0\,0\,0}_{8}\,\underbrace{0\,0\,1\,0}_{2}\,\underbrace{1\,0\,0\,0}_{8}\,\underbrace{1\,0\,1\,0}_{A}\,\underbrace{1\,0\,0\,0}_{8}\,\underbrace{0\,0\,1\,0}_{2}\,\underbrace{1\,0\,0\,0}_{8}\,\underbrace{1\,0\,1\,0}_{A}\ \text{Dopo la ROL}$$

8. MOV Ris2, DX

    - DX = 828A

3)

```
MOV AX, Mat
LEA ESI, Vet
MOV [ESI+10], AX
SHL AX, 4
MOV [ESI+12], AX
MOV ECX, 4
L1: MOV AH, [ESI+ECX+9]
MOV [ESI+ECX+20], AH
LOOP L1
MOV AH, [ESI+24]
XOR AL, AL
MOV Ris3, AX
```

1. MOV AX, Mat

    - AX = 0A2A

2. LEA ESI, Vet

    - LEA serve per leggere l'indirizzo di memoria di una variabile/vettore
    - ESI = &Vet

3. MOV [ESI + 10], AX

    - AX = A2A0

4. SHL AX, 4
5. MOV [ESI + 12], AX
8. MOV [ESI + ECX + 20], AH
    - Shiftiamo di 4 bits che corrisponde ad una cifra esadecimale.

									2A	0A	A0	A2									2A	0A	A0	A2		

6. MOV ECX, 4
    - ECX = 4
7. L1: MOV AH, [ESI + ECX + 9]
9. LOOP L1
    - AX = A2A0
    - ECX = 3
    - AX = A0A0
    - ECX = 2
    - AX = A0A0
    - ECX = 1
    - AX = 2AA0
    - ECX = 0
10. MOV AH, [ESI + 24]
    - AX = A2A0
11. XOR AL, AL
    - AX = A200
12. MOV Ris3, AX
    - Ris3 = AX = $\boxed{A200}$

4)

MOV AX, Mat
AND AX, 5Eh
MOV BL, 0FDh
IDIV BL
MOV Ris4, AX

1. MOV AX, Mat
   - AX = 0A2A
2. AND AX, 5Eh
   -

0	0	0	0	1	0	1	0	0	0	1	0	1	0	1	0	AND
0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	

   - AX = $000A_{16} = 10_{10}$
3. MOV BL, 0FDh
   - BL = $FD_{16} = -3_{10}$
4. IDIV BL
   - AX = 01FD
5. MOV Ris4, AX
   - Ris4 = AX = 01FD

5)

MOV AX, Mat
XOR BL, BL
DEC BL
DEC BL
IMUL BL
MOV Ris5, AX

1. MOV AX, Mat
   - AX = 0A2A
2. XOR BL, BL
   - BL = 00
3. DEC BL
4. DEC BL
   - Il DEC è il contrario dell' INC, questo (DEC) decrementa di 1 il registro.
   - BL = -2
5. IMUL BL
   - AL = 2A = $0\,0\,1\,0\,1\,0\,1\,0 = 2^5 + 2^3 + 2^1 = 32 + 8 + 2 = 42$
   - $42 \cdot (-2) = -84$
   - $84 = 0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,0\,0$
     Ora invertiamo questi bits del numero 84 (i bits che stanno a 1 li mettiamo a 0 e viceversa) e sommiamo 1.

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

6. MOV Ris5, AX
   - Ris5 = AX = $\boxed{\text{FFAC}}$

6)

MOV AX, Mat
AND AL, 100b
JZ L2
OR AH, 1101010b
JNZ L3
L2: XOR AH, 1010100b
L3: MOV Ris6, AX

1. MOV AX, Mat
   - AX = 0A2A
2. AND AL, 100b

0	0	1	0	1	0	1	0	
0	0	0	0	0	1	0	0	AND
0	0	0	0	0	0	0	0	

   AX = 0A00
3. JZ L2
   - JZ sta per "Jump Zero", questo verifica se l'operazione precedente ha dato come risultato 0, se è così **salta** a L2.
   - L'operazione precedente si è conclusa con uno zero e quindi passiamo a L2 e saltiamo le operazione che c'erano in mezzo.

6. L2: XOR AH, 1010100b

0	0	0	0	1	0	1	0	
0	1	0	1	0	1	0	0	XOR
0	1	0	1	1	1	1	0	

   L'operazione di **XOR**:
   - se i due bit sono diversi il risultato è 1, altrimenti 0.

   AX = 5E00

4. L3: MOV Ris6, AX
   Ris6 = AX = $\boxed{5E00}$

7)

```
MOV DX, Mat
AND DX, 0FF0h
CMP DX, 2000
JB L4
AND DX, 00FFh
JMP L5
L4: AND DX, 0FF0h
L5: LEA ESI, Vet
MOV EDI, ESI
MOV [EDI], DX
INC EDI
NOT DX
MOV [EDI], DX
MOV AX, [ESI]
MOV Ris7, AX
```

1. MOV DX, Mat DX = 0A2A
2. AND DX, 0FF0h

0	A	2	A	
0	F	F	0	AND
0	A	2	0	

3. CMP DX, 2000
   - DX = 2592 > 2000
4. JB L4
   - **JB** sta per "Jump Below" se è minore, in questo caso, di 2000, ma non è così perchè è maggiore, quindi questo passaggio viene ignorato.

5. AND DX, 00FFh

$$\begin{array}{cccc l} 0 & A & 2 & 0 & \\ 0 & 0 & F & F & \text{AND} \\ \hline 0 & 0 & 2 & 0 & \end{array}$$

6. JMP L5
   - **JMP** è un salto incodizionato (senza condizioni), salti e basta.

8. L5: LEA ESI, Vet
   - ESI = &Vet

9. MOV EDI, ESI
   - EDI = ESI = &Vet

10. MOV [EDI], DX

20	00																									

   MOV [EDI], DX LITTLE ENDIAN

11. INC EDI
   - EDI = &(Vet + 1)

12. NOT DX
   - Il **NOT** nega i valori (ovvero inverte i bits, quelli che sono a 0 diventano 1 e viceversa.) mentre il NEG = registro - 1.
     $DX = 0020_{16} = 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0$
     $\underbrace{1\,1\,1\,1}_{F}\,\underbrace{1\,1\,1\,1}_{F}\,\underbrace{1\,1\,0\,1}_{D}\,\underbrace{1\,1\,1\,1}_{F}$ **NEGATO**
   - DX = FFDF

13. MOV [EDI], DX

20	DF	FF																								

   EDI puntava al byte di posizione 1, perchè avevamo fatto l'incremento.

14. AX = DF20
   - EDI puntava alla cella del byte in posizione 0 (ovvero dove si trova 20)
   - Per via del **little endian** prendiamo prima il byte che sta dopo ed è per questo che AX equivale a:
   - AX = DF20

15. MOV Ris7, AX
   - Ris7 = AX = $\boxed{\text{DF20}}$

8)

MOV AX, Mat
MOV BL, 5
SHL AX, 1
JC L6
INC BL
L6: MUL BL
MOV Ris8, AX

1. MOV AX, Mat
   - AX = 0A2A
2. MOV BL, 5
   - BL = 5
3. SHL AX, 1
   $\cancel{0}\underbrace{0\,0\,0\,1}_{1}\underbrace{0\,1\,0\,0}_{4}\underbrace{0\,1\,0\,1}_{5}\underbrace{0\,1\,0\,0}_{4}$
   Il primo 0 va in CF (carry)
4. JC L6
   - **JC** sta per "Jump Carry" e riguarda il flag del Carry.
5. INC BL
   - BL = 5 + 1 = 6
6. MUL BL
   - AL = $54_{16} = 01010100_2 = 2^2 + 2^4 + 2^6 = 4 + 16 + 64 = 84_{10}$
     $84 \cdot 6 = 504 = 111111000_2$
     AX = $\underbrace{0\,0\,0\,0}_{0}\underbrace{0\,0\,0\,1}_{1}\underbrace{1\,1\,1\,1}_{F}\underbrace{1\,0\,0\,0}_{8}$
7. MOV Ris8, AX
   - Ris8 = AX = $\boxed{01F8}$

2) Vecchio Esame

MOV AX, Mat
LEA ESI, Vet
ADD ESI, 5
MOV ECX, 4
L1: MOV [ESI + ECX · 2], AX
LOOP L1
LEA ESI, Vet
ADD ESI, 9
MOV DX, [ESI]
SHR DX, 4
MOV Ris2, DX

1. MOV AX, Mat
   - AX = 0A2A
2. LEA ESI, Vet
   - ESI = &Vet
3. ADD ESI, 5
   - ESI = (&Vet) + 5
4. MOV ECX, 4
   - ECX = 4
5. L1: MOV [ESI + ECX · 2], AX
6. LOOP L1
   - ECX = 4 ... ECX = 3 ... ECX = 0 (a 0 ci fermiamo)
7. LEA ESI,Vet
   - ESI = &Vet

8. ADD ESI, 9

						2A	0A	2A	0A	2A	0A	2A	0A													

ESI = (&Vet) + 9

9. MOV DX, [ESI]

   - DX = 0A2A

10. SHR DX, 4

    - DX = 00A2

11. MOV Ris2, DX

    - Ris2 = DX = $\boxed{\text{00A2}}$

Algebra di Boole

1. Si scriva, utilizzando gli operatori booleani AND, OR, NOT, la funzione booleana che ritorna in uscita il valore 1 se sono veri un numero dispari dei tre input

A	B	C	f(a,b,c)	
0	0	0	0	
0	0	1	1	$\bar{A}\overline{B}C$
0	1	0	1	$\overline{A}B\overline{C}$
0	1	1	0	
1	0	0	1	$A\overline{B}\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

$f(A,B,C) = \bar{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$ SOLUZIONE

Il **+** è considerato un **OR**. La **moltiplicazione** è considerata un **AND**.

2. Si scriva, utilizzando gli operatori booleani AND, OR, NOT, la funzione booleana che ritorna in uscita il valore 1 se sono veri due dei quattro input

A	B	C	D	F	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	1	$\bar{A}\overline{B}CD$
0	1	0	0	0	
0	1	0	1	1	$\overline{A}B\overline{C}D$
0	1	1	0	1	$\overline{A}BC\overline{D}$
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	1	$A\bar{B}\bar{C}D$
1	0	1	0	1	$A\overline{B}C\overline{D}$
1	0	1	1	0	
1	1	0	0	1	$AB\bar{C}\bar{D}$
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	0	

$f(A,B,C,D) =$

$\bar{A}\overline{B}CD + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + A\bar{B}\bar{C}D + A\overline{B}C\overline{D} + AB\bar{C}\bar{D}$ SOLUZIONE

3. Si scriva, utilizzando gli operatori booleani AND, OR, NOT, la funzione booleana che riceve in ingresso un numero binario su 3 bit e ritorna in uscita il valore 1 se e solo se il numero che c'è in ingresso è maggiore o uguale a quattro.

3 variabili (A,B,C) e quindi 8 combinazioni

- Nella prima a destra (la C) ripetizioni 0 1 (in verticale)
- Nella centrale (la B) ripetizione: 0 0 1 1 (in verticale)
- In quella a sinistra (la A) ripetizioni 0 0 0 0 1 1 1 1

A	B	C	f(a,b,c)	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\bar{B}\bar{C}$
1	0	1	1	$A\overline{B}\,C$
1	1	0	1	$AB\overline{C}$
1	1	1	1	ABC

$f(A,B,C) = A\bar{B}\bar{C} + A\overline{B}C + AB\overline{C} + ABC$ SOLUZIONE

La soluzione può anche essere scritta:

$f(A,B,C) = A$

La colonna A è uguale alla colonna di F (non solo quei quattro 1, ma tutta la colonna dall'inizio, zeri compresi), se dovessi solo restituire A è come se restituissi F.

4. Dati gli operatori booleani AND, OR, NOT, scrivere l'espressione di una funzione booleana F avente come ingressi due numeri binari X e Y su 2 bit, che ritorni il valore 1 se X > Y.

x_1	x_0	y_1	y_0	$f(x,y)$	x_1x_0	y_1y_0	
0	0	0	0	0	0	0	
0	0	0	1	0	0	1	
0	0	1	0	0	0	2	
0	0	1	1	0	0	3	
0	1	0	0	1	1	0	$\overline{x_1}x_0\overline{y_1}\bar{y}_0$
0	1	0	1	0	1	1	
0	1	1	0	0	1	2	
0	1	1	1	0	1	3	
1	0	0	0	1	2	0	$x_1\overline{x_0}\bar{y}_1\overline{y_0}$
1	0	0	1	1	2	1	$x_1x_0\bar{y}_1\overline{y_0}$
1	0	1	0	0	2	2	
1	0	1	1	0	2	3	
1	1	0	0	1	3	0	$x_1x_0\bar{y}_1\overline{y_0}$
1	1	0	1	1	3	1	$x_1x_0\overline{y_1}y_0$
1	1	1	0	1	3	2	$x_1x_0y_1\overline{y_0}$
1	1	1	1	0	3	3	

$$f(x,y) = \overline{x_1}x_0\overline{y_1}\bar{y}_0 + x_1\overline{x_0}\bar{y}_1\overline{y_0} + x_1x_0\bar{y}_1\overline{y_0} + x_1x_0\bar{y}_1\overline{y_0} + x_1x_0\overline{y_1}y_0 + x_1x_0y_1\overline{y_0}$$

Le colonne: x_1x_0 e y_1y_0 sono in decimale considerando due bit.

Tabella delle Proprietà delle Funzioni Booleane

Name	AND form	OR form
Identity law	1A = A	0 + A = A
Null law	0A = 0	1 + A = 1
Idempotent law	AA = A	A + A = A
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	AB = BA	A + B = B + A
Associative law	(AB)C = A(BC)	(A + B) + C = A + (B + C)
Distributive law	A + BC = (A + B)(A + C)	A(B + C) = AB + AC
Absorption law	A(A + B) = A	A + AB = A
De Morgan's law	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

5. Applicando i teoremi dell'algebra di Boole, verificare la seguente equivalenza tra espressioni

$$\overline{A}\,\overline{B}\,\overline{C}+B\,\overline{C}+A\left(B+\overline{BC}\right)\equiv A+\overline{C}$$

Cerchiamo di semplificare l'espressione

$$\underbrace{\overline{A}\overline{B}\overline{C}+B\overline{C}}_{\overline{C}(\overline{A}\overline{B}+B)\ \text{PROPRIETA' DISTRIBUTIVA}} + A(B+\underbrace{\overline{BC})}_{A(B+\overline{B}+\overline{C})\ \text{APPLICHIAMO DE MORGAN}}$$

$$\overline{C}(\underbrace{\overline{A}\overline{B}+B}_{\overline{C}((\overline{A}+B)\cdot(\overline{B}+B))\ \text{distr.}})+A(\underbrace{B+\overline{B}}_{A(1+\overline{C})\ \text{inverse law}}+\overline{C})$$

$$C((\overline{A}+B)\cdot(\underbrace{\overline{B}+B}_{\text{INVERSE LAW}}))+A(\underbrace{1+\overline{C}}_{\text{NULL LAW}})$$

$$\overline{C}(\underbrace{(\overline{A}+B)\cdot 1}_{\text{IDENTITY LAW}})+\underbrace{A\cdot 1}_{\text{IDENTITY LAW}}$$

$$\underbrace{\overline{C}(\overline{A}+B)}_{\text{DISTRIBUTIVE LAW}}+A$$

$$\underbrace{\overline{C}\overline{A}+\overline{C}B+A}_{\text{DISTRIBUTIVE LAW}}$$

$$(A+\overline{C})(\underbrace{A+\overline{A}}_{\text{INVERSE LAW}})+\overline{C}B$$

$$\underbrace{(A+\overline{C})\cdot 1}_{\text{IDENTITY LAW}}+\overline{C}B$$

$$(A+\overline{C})+\overline{C}B$$

$$A+\underbrace{\overline{C}+\overline{C}B}_{\text{ABSORPTION LAW}}$$

$$A+\overline{C}$$

6. Applicando i teoremi dell'algebra di Boole, semplificare le seguenti espressioni e disegnarne la tavola di verità

$$AB\overline{C}+AB+AC+C$$

$$\overline{A}\,\overline{B}C+A\overline{B}+\overline{A}\,\overline{B}+AB$$

$$A+AB+B+BC$$

$$\underbrace{\mathrm{AB\overline{C}+AB}}_{\text{ABSORPTION LAW}} + \underbrace{\mathrm{AC+C}}_{\text{ABSORPTION LAW}}$$

$$\mathrm{AB+C}$$

A	B	C	$\overset{x}{\mathrm{AB}}$	$\overset{y}{\mathrm{AC}}$	$\overset{z}{AB\overline{C}}$	$z+x+y+c$	$x+c$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1
0	1	0	0	0	0	0	0
0	1	1	0	0	0	1	1
1	0	0	0	0	0	0	0
1	0	1	0	1	0	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	0	1	1

Le due colonne cerchiate in rosso sono equivalenti.

ES.6 - N.2

$$\underbrace{\mathrm{\overline{A}\bar{B}}}_{\text{ABSORPTION LAW}} \quad \mathrm{C+A\overline{B}+} \quad \underbrace{\mathrm{\overline{A}\bar{B}}}_{\text{ABSORPTION LAW}} \quad \mathrm{+AB}$$

$$\mathrm{\overline{A}\bar{B}+} \underbrace{\mathrm{A\overline{B}+AB}}_{\text{DISTRIBUTIVE LAW}}$$

$$\mathrm{\overline{A}\bar{B}+A(} \underbrace{\mathrm{\overline{B}+B}}_{\text{INVERSE LAW}} \mathrm{)}$$

$$\mathrm{\overline{A}\bar{B}+} \underbrace{\mathrm{A\cdot 1}}_{\text{IDENTITY LAW}}$$

$$\underbrace{\mathrm{\overline{A}\bar{B}+\overline{A}}}_{\text{DISTRIBUTIVE LAW}}$$

$$\underbrace{\mathrm{(A+\overline{A})}}_{\text{INVERSE LAW}} \quad \mathrm{(A+\overline{B})}$$

$$\underbrace{\mathrm{1\cdot(A+\overline{B})}}_{\text{IDENTITY LAW}}$$

$$\boxed{\mathrm{A+\overline{B}}}$$

Ora non più semplificabile.

A	B	C	$\overset{x}{\bar{A}\bar{B}}$	$\overset{y}{XC}$	$\overset{z}{A\overline{B}}$	$\overset{t}{AB}$	$y+z+x+t$	$A+\overline{B}$
0	0	0	1	0	0	0	1	1
0	0	1	1	1	0	0	1	1
0	1	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
1	0	0	0	0	1	0	1	1
1	0	1	0	1	1	0	1	1
1	1	0	0	0	0	1	1	1
1	1	1	0	0	0	1	1	1

Le due colonne cerchiate in rosso sono equivalenti.

7. Si valutino le espressioni booleani precedenti assumendo A=1

8. Una cassaforte ha quattro lucchetti, x, y, v, w, che devono essere tutti aperti affinché la cassaforte possa essere aperta. Le chiavi sono distribuite tra 3 persone, A, B, C, come segue: A possiede le chiavi v e y ; B possiede le chiavi v e x ; C possiede le chiavi w e y. Siano le variabili A, B e C uguali a 1 se la persona corrispondente è presente, altrimenti uguali a 0. Costruire la tavola della verità della funzione f(A,B,C) che è uguale ad 1 se e solo se la cassaforte può essere aperta, ed esprimere f in forma algebrica.

A	B	C	f	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

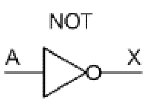
$$f(A,B,C) = \underbrace{\overline{A}BC + ABC}_{\text{DISTRIBUTIVE LAW}}$$

$$BC \underbrace{(\overline{A} + A)}_{\text{INVERSE LAW}}$$

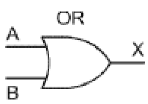
$$\underbrace{BC \cdot 1}_{\text{IDENTITY LAW}}$$

$$\boxed{BC} \quad \text{SOLUZIONE}$$

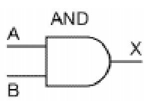
NOT - OR - AND | LOGIC GATES



A	X
0	1
1	0



A	B	X
0	0	0
0	1	1
1	0	1
1	1	1



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

9. Una cisterna è dotata di 4 pompe: P_A e P_B riempiono la cisterna, P_C e P_D la svuotano. Sapendo che la capacità delle pompe è rispettivamente di 7, 5, 8 e 4 litri/secondo si indichi (tramite tabella di verità) per quali configurazioni di pompe accese la vasca si riempirà. Si derivi e si semplifichi quindi la corrispondente espressione booleana e si disegni il relativo circuito digitale.

A	B	C	D	f	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	1	$\overline{A}B\overline{C}\overline{D}$
0	1	0	1	1	$\overline{A}B\overline{C}D$
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	1	$A\overline{B}\overline{C}\overline{D}$
1	0	0	1	1	$A\overline{B}\overline{C}D$
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	1	$AB\overline{C}\overline{D}$
1	1	0	1	1	$AB\overline{C}D$
1	1	1	0	1	$ABC\overline{D}$
1	1	1	1	0	

$$f(A,B,C,D) = \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D + AB\overline{C}\overline{D} + AB\overline{C}D + ABC\overline{D}$$

$$\underbrace{\overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D} + \underbrace{A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D} + \underbrace{AB\overline{C}\overline{D} + AB\overline{C}D} + ABC\overline{D} +$$

$$\overline{A}B\overline{C}(\underbrace{\overline{D}+D}_{\text{INVERSE LAW}}) + A\overline{B}\overline{C}(\underbrace{\overline{D}+D}_{\text{INVERSE LAW}}) + AB\overline{C}(\underbrace{\overline{D}+D}_{\text{INVERSE LAW}})$$

$$\Rightarrow \overline{A}B\overline{C} + A\overline{B}\overline{C} + \underbrace{AB\overline{C} + ABC\overline{D}} + AB(\underbrace{\overline{C} + C\overline{D}}_{\overline{C}+\overline{D}})$$

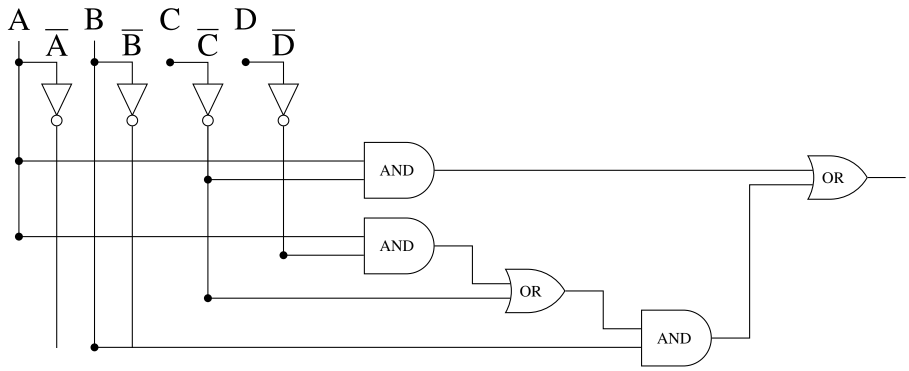
$$\underbrace{\overline{A}B\overline{C}} + A\overline{B}\overline{C} + \underbrace{AB\overline{C}} + AB\overline{D}$$

$$B\overline{C}(\underbrace{\overline{A}+A}_{1\,\text{INVERSE LAW}})$$

$$\underbrace{B\overline{C} + A\overline{B}\overline{C}} + AB\overline{D} + \overline{C}(\underbrace{A + A\overline{B}}_{B+A})$$

$$A\overline{C} + \underbrace{B\overline{C} + AB\overline{D}}_{B(\overline{C}+A\overline{D})}$$

$$\boxed{A\overline{C} + B(\overline{C} + A\overline{D})} \quad \text{SOLUZIONE}$$



10. Data la seguente espressione:

$$C\big(AC(\bar{A}+\bar{C})\big)B+(A\bar{B}+\bar{A}B)\bar{B}+\big(\overline{A\bar{B}}\big)(A+\bar{A}\bar{B})B$$

si ricavi un'espressione booleana semplificata equivalente.

$$C(\underbrace{AC(\bar{A}+\bar{C})}_{\underbrace{AC\bar{A}}_{0}+\underbrace{AC\bar{C}}_{0}})B+\underbrace{(A\bar{B}+\bar{A}B)\bar{B}}_{\underbrace{A\bar{B}\bar{B}}_{A\bar{B}}+\underbrace{\bar{A}B\bar{B}}_{0}}+\underbrace{(\overline{A\bar{B}})}_{\bar{A}+B}+(\underbrace{A+\bar{A}\bar{B}}_{A+\bar{B}})B$$

$$\Rightarrow$$

$$0\ \text{(Se fossero stati in AND avrebbe azzerato tutto)}+A\bar{B}+0+(\bar{A}+B)\underbrace{(A+\bar{B})B}_{AB+\underbrace{\bar{B}B}_{0}}$$

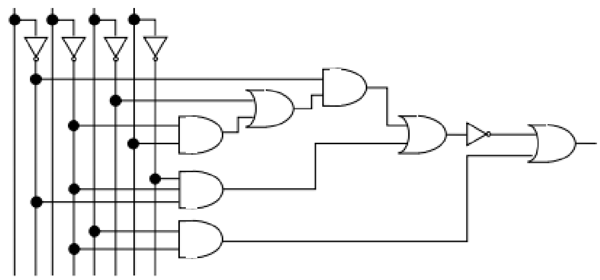
$$A\bar{B}+\underbrace{(\bar{A}+B)AB}_{\underbrace{\bar{A}AB}_{0}+\underbrace{BAB}_{AB}}$$

$$\underbrace{A\bar{B}+AB}_{A\underbrace{(\bar{B}+B)}_{1}}$$

$\boxed{A}$ SOLUZIONE

11. Dato il seguente circuito si scriva l'espressione booleana equivalente, si ricavi l'espressione booleana semplificata e si disegni il relativo circuito digitale.

$A\ \bar{A}\ B\ \bar{B}\ C\ \bar{C}\ D\ \bar{D}$



$$\overline{(\underbrace{(\overline{B}D+\overline{C})\overline{A}}_{\overline{A}\overline{B}D+\overline{A}\overline{C}}+\overline{A}\overline{B}\overline{D})}+\overline{B}C$$

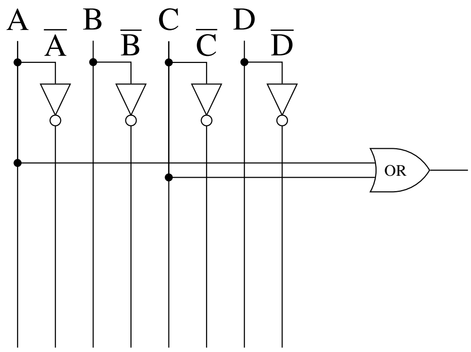
$$\overline{\overline{A}\overline{C}+\underbrace{\overline{A}\overline{B}D+\overline{A}\overline{B}\overline{D}}_{\overline{A}\overline{B}(\underbrace{D+\overline{D}}_{1})}}+BC$$

$$\overline{\underbrace{\overline{A}\overline{C}+\overline{A}\overline{B}}_{\text{DE MORGAN}}}+BC$$

$$\underbrace{(A+C)(A+B)}_{A+BC}+BC$$

$$A+\underbrace{BC+\overline{B}C}_{C(\underbrace{B+\overline{B}}_{1})}$$

$$\boxed{A+C} \quad \text{SOLUZIONE}$$



12. Data la seguente espressione:

$$(A+B)(B+A)[(A+C)+(C+A)(A+C)]+\big((D\oplus\bar{A})\oplus\bar{A}\big)\big(\bar{D}(E+\bar{D})\big)+(\bar{B}(\bar{B}+\bar{E})+\bar{B}D)$$

Si ricavi un'espressione booleana semplificata equivalente e si disegni il relativo circuito digitale.

$$\underbrace{(A+B)(B+A)}_{A+B}[(A+C)+\underbrace{(C+A)(A+C)}_{A+C}]+((D\overset{\text{XOR}}{\oplus}\overline{A})\oplus\overline{A})(\underbrace{\overline{D}(E+\overline{D})}_{\overline{D}\ \text{ABSORPTION LAW}})+(\underbrace{\overline{B}(\overline{B}+\overline{E})}_{\overline{B}}+\overline{B}D)$$

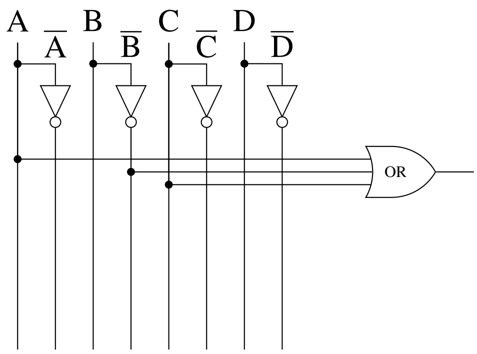
$$(A+B)[\underbrace{(A+C)+(A+C)}_{A+C\ \text{INVERSE LAW}}]+((D\oplus\overline{A})\oplus\overline{A})\overline{D}+(\underbrace{\overline{B}+\overline{B}D}_{\overline{B}\ \text{ABSORPTION LAW}})$$

$$\underbrace{(A+B)(A+C)}_{A+BC\ \text{DISTRIBUTIVE LAW}}+(\underbrace{(D\oplus\overline{A})\oplus\overline{A}}_{D})\overline{D}+\overline{B}$$

$$A + BC + \underbrace{D\overline{D}}_{0} + B$$

$$\Rightarrow A + \underbrace{BC + \overline{B}}_{\overline{B}+C}$$

$\boxed{A + \overline{B} + C}$ SOLUZIONE



13. Siano A = (a_1 a_0) e B = (b_1 b_0) due numeri binari (con a_1 e b_1 bit più significativi); si consideri la funzione booleana F(A,B) che è vera se $A \oplus B$ è un numero primo (si ricordi che 0 e 1 non sono numeri primi). Si scriva la tabella di verità di F, si determini e si semplifichi la corrispondente espressione booleana e si disegni il relativo circuito digitale.

a_1	a_0	b_1	b_0	A ⊕ B	F	
0	0	0	0	00	0	
0	0	0	1	01	0	
0	0	1	0	10	1	$\mathbf{\overline{a_1}\bar{a}_0 b_1 \overline{b_0}}$
0	0	1	1	11	1	$\mathbf{\overline{a_1}\bar{a}_0 b_1 b_0}$
0	1	0	0	01	0	
0	1	0	1	00	0	
0	1	1	0	11	1	$\mathbf{\overline{a_1} a_0 b_1 \overline{b_0}}$
0	1	1	1	10	1	$\mathbf{\overline{a_1} a_0 b_1 b_0}$
1	0	0	0	10	1	$\mathbf{a_1 \overline{a_0} \bar{b}_1 b_0}$
1	0	0	1	11	1	$\mathbf{a_1 \overline{a_0} \bar{b}_1 b_0}$
1	0	1	0	00	0	
1	0	1	1	01	0	
1	1	0	0	11	1	$\mathbf{a_1 a_0 \overline{b_1} \bar{b}_0}$
1	1	0	1	10	1	$\mathbf{a_1 a_0 \overline{b_1} b_0}$
1	1	1	0	01	0	
1	1	1	1	00	0	

$$F(a,b,c) =$$

$$\underbrace{\overline{a_1}\bar{a}_0 b_1 \overline{b_0} + \overline{a_1}\bar{a}_0 b_1 b_0 + \overline{a_1} a_0 b_1 \overline{b_0} + \overline{a_1} a_0 b_1 b_0}_{\boxed{\overline{a_1} b_1 + a_1 \overline{b_1}}} + \underbrace{a_1 \overline{a_0} \bar{b}_1 b_0 + a_1 \overline{a_0} \bar{b}_1 b_0 + a_1 a_0 \overline{b_1} \bar{b}_0 + a_1 a_0 \overline{b_1} b_0}$$

SOLUZIONE

$$a_1 \oplus b_1$$

Architetture a Confronto

Prestazioni dei Calcolatori

$$T_{esecuzione} = T_{clock} \cdot \left(\sum_{i=0}^{n} N_i \cdot CPI_i \right)$$

(a) Tempo di Esecuzione

$$MIPS = \frac{Frequenza\ del\ clock}{10^6 \cdot CPI}$$

(b) Milion Instruction Per Second

$$Prestazione\ P = \frac{1}{T_{esecuzione}}$$

(a) Prestazione

$$T_{es.finale} = \frac{p \cdot T_{es.iniziale}}{a} + (1 - p) \cdot T_{es.iniziale}$$

(b) Legge di Amdhal

1) Siano x e y le ultime due cifre della propria matricola in base 10 (esempio: matricola 3465 → x=5, y=6). Si consideri una CPU in grado di eseguire un programma P in 100ns. In una versione più evoluta della stessa CPU sono state accelerate alcune istruzioni di un fattore *α*: tale CPU è in grado di eseguire P in (80+y) ns. Sapendo che la percentuale di istruzioni accelerate rispetto al totale di quelle contenute nel programma P è [(x+6)*5]%, si calcoli il fattore di accelerazione *α*.

Esempio: matricola 3465 $\rightarrow x = 5, y = 6$

Matricola (non la mia) = 2602

$$x = 2$$
$$y = 0$$
$$\mathrm{T}_i \text{ (Tempo iniziale) } = 100 \text{ ns (nanosecondi)}$$
$$\mathrm{T}_f \text{ (Tempo finale) } = 80 + y = 80 + 0 = 80 \text{ ns}$$
$$P = [(x+6)\cdot 5]\% \rightarrow [(2+6)\cdot 5]\% = 40\%$$
$$\alpha = ?$$

Legge di Amdhal a pag.37

$$\mathrm{T}_f = \frac{\mathrm{P}\cdot \mathrm{T}_i}{\alpha} + (1-p)\cdot \mathrm{T}_i$$
$$80 = \frac{0.04\cdot 100}{\alpha} + 0,6\cdot 100$$
$$\rightarrow 80 = \frac{40}{\alpha} + 60 \rightarrow 80 - 60 = \frac{40}{\alpha} \rightarrow 20\cdot\alpha = 40 \rightarrow \boxed{\alpha = 2}$$

IL DOPPIO PIU' VELOCE

2) Siano y e x le ultime due cifre della propria matricola in base 10 (esempio: matricola 3465 → x=5, y=6). Un programma P viene eseguito in 13+y secondi su una CPU A, dotata di una frequenza di clock di (3+x)*15 MHz. È stata progettata una nuova CPU B in grado di operare a una frequenza maggiore: B esegue P in soli 5+y secondi. Sapendo che, al fine di consentire l'aumento della frequenza, si è dovuto aumentare il numero di clock per istruzione (CPI) in media del 25% (ad esempio, un'istruzione che nella CPU A richiedeva 4 CPI, nella CPU B ne richiede 5), si stimi la frequenza a cui opera la CPU B.

Matricola = 2602 $(x = 2, y = 0)$

$$\mathrm{T_A} = 13 \text{ s (secondi)}$$
$$\mathrm{F_A} = 75 \text{ MHz (MegaHertz)}$$
$$\mathrm{T_B} = 5 \text{ s}$$
$$\mathrm{CPI_B} \text{ (Clock per Instruction)} = 1,25\cdot \mathrm{CPI_A}$$
$$\mathrm{F_B} = ?$$

Prestazioni dei Calcolatori a pag.37

$$\mathrm{T_{ES}} = \mathrm{T_{clock}} \cdot \underbrace{\sum_{i=0}^{n} \mathrm{N}_i \cdot \mathrm{CPI}_i}_{\mathrm{CPI_{TOT}}}$$

- $\mathrm{T_{clock}}$ periodo di clock della macchina
- $\mathrm{CPI_i}$ numero di clock per istruzione di tipo i numero di clock occorrenti affinchè avvenga l'esecuzione dell'istruzione
- $\mathrm{N_i}$ numero di istruzioni di tipo i (somme, sottrazioni, salti, ecc..).

$$T_{ES} \simeq T_{clock} \cdot N_{TOT} \cdot \widetilde{CPI}$$

Quella tilde sopra il CPI significa "Mediamente per ogni istruzione"

$$T_{clock} = \frac{1}{F}$$

$$\begin{cases} 13 = \frac{1}{75 \cdot 10^6} \cdot N_{TOT} \cdot \widetilde{CPI} \\ 5 = \frac{1}{F_B} \cdot N_{TOT} \cdot \widetilde{CPI_B} \end{cases}$$

N_{TOT} c'è due volte perchè è lo stesso programma e quindi stesse istruzioni

$$\begin{cases} 13 \cdot 75 \cdot 10^6 = N_{TOT} \cdot \widetilde{CPI_A} \\ 5 \cdot F_B = N_{TOT} \cdot 1,25 \cdot \widetilde{CPI_A} \end{cases} \qquad \begin{cases} 975 \cdot 10^6 = N_{TOT} \cdot \widetilde{CPI_A} \\ F_B = \frac{1,25 \cdot \cancel{975} \cdot 10^6}{\cancel{5}} \end{cases}$$

$$F_B = 243,75 \cdot 10^6 \text{ Hz} = 243,75 \text{ Mhz}$$

3) Siano y e x le ultime due cifre della propria matricola in base 10 (esempio: matricola 3465 → x=5, y=6). Si consideri un calcolatore dotato di una cache a 2 livelli. Il tempo di accesso a una parola è di 7ns per la cache di primo livello, 13ns per la cache di secondo livello. Durante l'esecuzione di un programma, 10+(x*y) parole sono lette dalla RAM, 6+x dalla cache di primo livello e 1+x+y parole sono lette dalla cache di secondo livello. Sapendo che il tempo totale di lettura delle parole è di 3750ns, determinare il tempo di lettura di una singola parola dalla RAM.

$$\text{MATR } = 2602\ (x = 2, y = 0)$$
$$T_{C1} = 7 \text{ ns}$$
$$T_{C2} = 13 \text{ ns}$$

$$\left.\begin{array}{l} N_{RAM} = 10 + (x \cdot y) \\ N_{C1} = 8 \\ N_{C2} = 3 \end{array}\right\} 21\ (N_{TOT}) \text{ PAROLE}$$

$$T_{TOT} = 3750 \text{ ns}$$
$$T_{RAM} = ?$$
$$T_{TOT} = (N_{TOT} \cdot T_{C1}) + [(N_{TOT} - N_{C1}) \cdot T_{C2}] + \cdots + [(N_{TOT} - N_{C1} - N_{C2}) \cdot T_{RAM}]$$

Parole da cercare (N_{TOT}) , nella cache L1 (T_{C1}) , Parole da cercare meno quelle che ho già trovato nella cache 1 $(N_{TOT} - N_{C1})$

$$3750 = 21 \cdot 7 + (21 - 8) \cdot 13 + (21 - 8 - 3) \cdot T_{RAM}$$
$$3750 = 147 + 169 + 10 \cdot T_{RAM}$$
$$\frac{3750 - 147 - 169}{10} = T_{RAM} \Rightarrow T_{RAM} \simeq \boxed{343,3 \text{ ns}}$$

Esercizi non necessariamente realistici.

4) Siano x e y le ultime due cifre della propria matricola in base 10 (esempio: matricola 3465 → x=5, y=6). Si consideri una CPU in grado di eseguire un programma P in 180ns. In una versione più evoluta della stessa CPU sono state accelerate alcune istruzioni di un fattore *α*: tale CPU è in grado di eseguire P in (120+y) ns. Sapendo che il fattore di accelerazione *α* è pari a x+2, si calcoli la percentuale di istruzioni accelerate rispetto al totale di quelle contenute nel programma P.

$$\mathrm{MATR} = 2602\ (x = 2, y = 0)$$
$$\mathrm{T_i} = 180\ \mathrm{ns}$$
$$\mathrm{T_F} = 120\ \mathrm{ns}$$
$$\alpha = 4$$
$$120 = \frac{\mathrm{P}\cdot\cancel{180}}{\cancel{4}} + (1-\mathrm{P})\cdot 180 \Rightarrow 120 = \mathrm{P}\cdot 45 + (1-\mathrm{P})\cdot 180$$
$$120 = \mathrm{P}\cdot 45 + 180 - \mathrm{P}\cdot 180$$
$$120 = \mathrm{P}(45-180) + 180$$
$$\frac{120-180}{45-180} = \mathrm{P} \Rightarrow \mathrm{P} = \frac{-60}{-135} \simeq \boxed{44,4\%}$$

Andava bene anche tenere la frazione semplificata ($\frac{\cancel{-60}}{\cancel{-135}} = \frac{\cancel{12}}{\cancel{22}} = \frac{4}{9}$) al posto di scriverlo in percentuale.

5) Siano y e x le ultime due cifre della propria matricola in base 10 (esempio: matricola 3465 → x=5, y=6). Si consideri una CPU dotata di una cache di 1° livello con tempo di accesso pari a 7+y nanosecondi e una di 2° livello con tempo di accesso pari a 12+y nanosecondi. Durante l'esecuzione di un programma, la CPU ha impiegato 6,58 microsecondi per leggere dalla memoria 37+(2*x) parole; sapendo che, all'inizio dell'esecuzione del programma, y+3 di tali parole erano già contenute nella cache di 1° livello e y+7 in quella di 2° livello e che durante l'esecuzione non è mai stato necessario rimpiazzare parole nelle cache, si calcoli il tempo di accesso a una parola della RAM, esprimendo il risultato in nanosecondi.

Non è mai stato necessario rimpiazzare parole nella cache = cache non variano.

$$\mathrm{MATR} = 2602\ (x = 2, y = 0)$$
$$\mathrm{T_{C1}} = 7\ \mathrm{ns}$$
$$\mathrm{T_{C2}} = 12\ \mathrm{ns}$$
$$\mathrm{T_{TOT}} = 6,58\ \overset{10^{-6}}{\mathrm{ms}\ (\mathrm{millisecondi})} = 6580\ \overset{10^{-9}}{\mathrm{ns}\ (\mathrm{nanosecondi})}$$
$$\mathrm{N_{TOT}} = 41$$
$$\mathrm{N_{C1}} = 3$$
$$\mathrm{N_{C2}} = 7$$
$$\mathrm{T_{RAM}} = ?$$
$$6580 = 41\cdot 7 + (41-3)\cdot 12 + (41-3-7)\cdot \mathrm{T_{RAM}}$$
$$6580 = 287 + 38\cdot 12 + 31\cdot \mathrm{T_{RAM}}$$
$$6580 = 287 + 456 + 31\cdot \mathrm{T_{RAM}}$$
$$\frac{6580-287-456}{31} = \mathrm{T_{RAM}}$$
$$\mathrm{T_{RAM}} = \frac{5837}{31} \simeq \boxed{188,3\ \mathrm{ns}}$$

6) Siano y e x le ultime due cifre della propria matricola in base 10 (esempio: matricola 3465 → x=5, y=6). Un programma P viene eseguito in 15+y secondi su una CPU A, dotata di una frequenza di clock di (5+x)*18 MHz. È stata progettata una nuova CPU B in grado di operare a una frequenza maggiore: B esegue P in soli 7+y secondi. Sapendo che, al fine di consentire l'aumento della frequenza, si è dovuto aumentare il numero di clock per istruzione (CPI) in media del 30+x%, si stimi la frequenza a cui opera la CPU B.

$$\begin{gathered}\mathrm{MATR} = 2602\ (x = 2, y = 0)\\ \mathrm{T_A} = 15\ \mathrm{s}\\ \mathrm{F_A} = 126\ \mathrm{Mhz}\\ \mathrm{T_B} = 7 + y\ \mathrm{secondi}\\ \mathrm{CPI_B} = 1{,}32 \cdot \mathrm{CPI_A}\\ \mathrm{F_B} = ?\end{gathered}$$

$$\begin{cases} 15 = \frac{1}{126 \cdot 10^6} \cdot \mathrm{N_{TOT}} \cdot \widetilde{\mathrm{CPI_A}} \\ 7 = \frac{1}{\mathrm{F_B}} \cdot \mathrm{N_{TOT}} \cdot \widetilde{\mathrm{CPI_B}} \end{cases} \qquad \begin{cases} 15 \cdot 126 \cdot 10^6 = \mathrm{N_{TOT}} \cdot \widetilde{\mathrm{CPI_A}} \\ 7 \cdot \mathrm{F_B} = \mathrm{N_{TOT}} \cdot 1{,}32 \cdot \widetilde{\mathrm{CPI_A}} \end{cases}$$

$$\begin{cases} 1890 \cdot 10^6 = \mathrm{N_{TOT}} \cdot \widetilde{\mathrm{CPI_A}} \\ \mathrm{F_B} = \frac{1890 \cdot 10^6 \cdot 1{,}32}{7} \end{cases}$$

$$\mathrm{F_B} \simeq 356{,}4 \cdot 10^6\ \mathrm{hz} = \boxed{356{,}4\ \mathrm{Mhz}}$$

Conclusioni

13, Dicembre 2020

In questo documento/articolo ho raccolto tutti o quasi gli esercizi svolti durante il corso di "Architetture degli Elaboratori" tranne alcuni che non ho scritto (come la Prova di Autovalutazione).
Al momento ho concluso questo documento/articolo.
Ma in futuro, potreì aggiungere degli esercizi, in fondo, su tutti gli argomenti e poi le soluzioni a questi.
Inoltre potreì anche mostrare i miei esercizi in Assembly che ho consegnato.

Firma: Luca