

ARCHITETTURA DEGLI ELABORATORI

RAPPRESENTAZIONE DELL'INFORMAZIONE (ID 1)

SISTEMI NUMERICI

I calcolatori lavorano in **sistema binario**: 0 e 1.

Infatti servono due stati per sapere se c'è corrente elettrica: sì o no, 1 o 0.

Con il termine **bit** indichiamo l'unità di misura dell'informazione; può assumere valori di 0 e 1. 8 bit formano un byte.

Nei **sistemi posizionali** un numero è formato da:

- una singola cifra (d)
- la base del sistema (r)
- la parte intera (n)
- la parte frazionaria (m)
- il numero da rappresentare (N)

La cifra più a destra è il bit meno significativo (LSB), quello a sinistra il più significativo (MSB)

	<i>Sistema Decimale</i>	<i>sistema binario</i>
Regola	$N = d_{n-2} * 10^{n-1} + \dots + d^0 * 10^0 + d_{-1} * 10^{-1} + \dots + d_{-m} * 10^{-m}$	$N = d_{n-1} * 2^{n-1} + \dots + d_0 * 2^0 + d_{-1} * 2^{-1} + \dots + d_{-m} * 2^{-m}$
Esempio	$123,45 = 1 * 10^2 + 2 * 10^1 + 3 * 10^0 + 4 * 10^{-1} + 5 * 10^{-2}$	$101_2 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 5_{10}$

Con il sistema binario si può verificare il fenomeno di **overflow**, ovvero un numero che non si può rappresentare, perché richiede più bit.

Le operazioni di somma e sottrazione possono causare overflow.

<i>VALORI</i>	<i>SOMMA</i>	<i>SOTTRAZIONE</i>
0 0	0	0
0 1	1	1 con prestito dal bit superiore
1 0	1	1
1 1	0 con riporto 1	0

NUMERI CON SEGNO

Ci sono 4 metodi per rappresentare un numero *negativo*:

- Modulo e Segno (MS)
- Complemento a 1 (CA1)
- Complemento a 2 (CA2)
- Eccesso 128

<u>NOME</u>	<u>BIT</u>	<u>ESEMPIO</u>	<u>PROBLEMI</u>
<i>MS</i>	<p>7 Bit per rappresentare il numero.</p> <p>Il MSB per indicare il segno: 1 se negativo, 0 positivo.</p>	<p>$-4 = 1000\ 0100$</p> <p>$4 = 0000\ 0100$</p>	<p>1. Due rappresentazioni dello zero: 0000 e 1000.</p> <p>2. Spreco di un bit per il segno</p>
<i>CA1</i>	Si invertono gli 0 con gli 1 e viceversa.	<p>$9 = 0000\ 1001$</p> <p>$-9 = 1111\ 0110$</p>	Due rappresentazioni dello zero
<i>CA2</i>	<p>Numero positivo rimane invariato.</p> <p>Numero negativo si applica CA1 e si somma 1.</p>	<p>$127 = 0111\ 1111$</p> <p>$-127 = 1000\ 0001$</p>	-
<i>E128</i>	<p>Si somma al numero decimale 128 e si converte in binario.</p> <p>Oppure si complementa il MSB al CA2</p>	<p>$5 + 128 = 133_{10}$</p> <p>$-5 = 1000\ 0101_2$</p>	-

MODULO E SEGNO		
<u>OPERATORE</u>	<u>SEGNO UGUALI</u>	<u>SEGNO DIVERSO</u>
<i>SOMMA</i>	BIT DI SEGNO Lo <i>stesso</i> degli addendi BIT <i>Somma</i> bit a bit.	BIT DI SEGNO Quello dell'operando a <i>modulo maggiore</i> BIT <i>Differenza</i> bit a bit.
<i>SOTTRAZIONE</i>	BIT DI SEGNO Quello dell'operando a <i>modulo maggiore</i> BIT Il modulo della <i>differenza</i> dei due moduli	BIT DI SEGNO Quello del <i>minuendo</i> BIT <i>Somma</i> dei moduli dei due operandi

COMPLEMENTO A 2	
<u>OPERATORE</u>	<u>OPERAZIONE</u>
<i>SOMMA</i>	Si esegue la somma in tutti i bit segno compreso. Un riporto oltre al MSB viene <i>scartato</i> . Se i due operandi sono di segno concorde e il risultato è del segno opposto è overflow.
<i>SOTTRAZIONE</i>	Si utilizza la seguente formula: $A - B = A + CA_2(B).$ Se i due operandi sono di segno concorde e il risultato è del segno opposto è overflow.

NUMERI DECIMALI E CARATTERI

VIRGOLA FISSA	
<i>Bit</i>	1 bit segno $I < (n-1)$ = parte intera $D = n - (I+1)$ = parte decimale Decidiamo noi quanti bit riservare in base alle esigenze
<i>Conv 10 -> 2</i>	$5.25_{10} \rightarrow 101.01_2$ $0.25 * 2 = 0.5$ 0 rip 0.5 $0.5 * 2 = 1$ 1 rip 0
<i>Conv 2 -> 10</i>	$101.01 = 1 * 2^2 + 1 * 2^0 + 1 * 2^{-2} = 4 + 1 + 0.25 = 5.25$
<i>Problemi</i>	Rigidità della posizione assegnata alla virgola Poca precisione nel codificare i numeri.

VIRGOLA MOBILE	
<i>Bit</i>	1 bit segno S mantissa M esponente E $X = (-1)^S * M * B^E$ Posizione della virgola variabile.
<i>Esempio</i>	$1011.0110_2 \rightarrow 1.0110110_2 * 2^3$
<i>Precisione</i>	Singola Precisione: 1 bit Segno 8 bit Esponente 23 bit Mantissa Doppia precisione: 1 bit Segno 11 bit Esponente 52 bit Mantissa

<i>Conv 10 -> 2</i>	$7.5_{10} \rightarrow 111.1_2$ $7_{10} = 111_2$ $0.5_{10} = 1_2$
<i>Conv 2 -> 10</i>	<p>1 10000000 110000000000000000000000</p> <p>1 = segno = -</p> <p>10000000 = 128 - 127 (cost) = 1</p> <p>1 spostamento: 11.1000000000000000000000</p> <p>Ris:-3.5</p>
<i>Range e Precisione</i>	<p>Range: bit esponente</p> <p>Precisione. bit mantissa</p>

CARATTERI				
	<i>ASCII std</i>	<i>ASCII text</i>	<i>UNICODE</i>	<i>UTF-8</i>
<i>Rappresentazione di un carattere</i>	7 bit	8 bit	8-32 bit	
<i>Totale Simboli</i>	128	256	4,29 miliardi	
<i>Cosa Contiene</i>	26 + 26 lettere (a,A - z,Z) 10 cifre decimali segni di interpunzione caratteri di controllo	La tabella ASCII estesa varia in base alla zona geografica di utilizzo e al software utilizzato.	Codifica tutti i caratteri utilizzati nelle principali lingue del mondo Un carattere UNICODE è caratterizzato dal suo codice numerico, detto code point, solitamente rappresentato con 8 cifre esadecimali	Unione tra 8 bit e caratteri UNICODE

CIRCUITI DIGITALI (ID 2)

RETI COMBINATORIE PT. 1

I **circuiti logici** sono realizzati come circuiti integrati realizzati su chip di silicio (piastrina)




Porte (gate) e **fili** depositati su chip di silicio, inseriti in un package e collegati all'esterno con un certo insieme di pin.

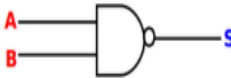
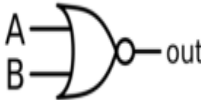

<i>Nome</i>	<i>Porte</i>
SSI (Small Scale Integrated)	1-10 porte
MSI (Medium Scale Integrated)	10-100 porte
LSI (Large Scale Integrated)	100-100.000 porte
VLSI (Very Large Scale Integrated)	> 100.000 porte

Nell'elettronica digitale sia gli ingressi che le uscite possono assumere solo i valori di **segnale alto** (1) con Volt >1 o **basso** (0) con volt <=1.

- Un **circuito combinatorio** è quel circuito il cui lo stato delle uscite dipende solo dalla funzione logica applicata allo **stato istantaneo** (cioè in un determinato istante di tempo) delle sue entrate.
- Un **circuito sequenziale** è quel circuito il cui lo stato delle uscite non dipende solo dalla funzione logica applicata ai suoi ingressi, ma anche sulla base di **valori pregressi collocati in memoria**.

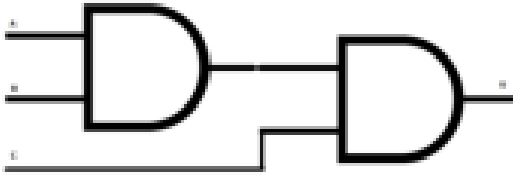
Una *porta logica* è un circuito elettronico che dati dei segnali 0 e 1 in input produce un segnale in output ottenuto effettuando una operazione booleana sugli ingressi.

Porte Logiche Fondamentali																		
<u>NOME</u>	<u>SIMBOLO</u>	<u>VALORI</u>	<u>SPIEGAZIONE</u>															
AND		<table><tr><th>A</th><th>B</th><th>$A \cdot B$</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	$A \cdot B$	0	0	0	0	1	0	1	0	0	1	1	1	1 se entrambi sono 1.
A	B	$A \cdot B$																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		<table><tr><th>A</th><th>B</th><th>$A + B$</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	$A + B$	0	0	0	0	1	1	1	0	1	1	1	1	1 se hai almeno un 1.
A	B	$A + B$																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		<table><tr><th>A</th><th>\overline{A}</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	\overline{A}	0	1	1	0	l'opposto dell'input.									
A	\overline{A}																	
0	1																	
1	0																	

Porte Logiche Derivate																							
<u>NOME</u>	<u>SIMBOLO</u>	<u>VALORI</u>	<u>SPIEGAZIONE</u>																				
NAND		<table><tr><th>A</th><th>B</th><th>$A \cdot B$</th><th>A NAND B</th></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	A	B	$A \cdot B$	A NAND B	0	0	0	1	0	1	0	1	1	0	0	1	1	1	1	0	L'opposto di AND
A	B	$A \cdot B$	A NAND B																				
0	0	0	1																				
0	1	0	1																				
1	0	0	1																				
1	1	1	0																				
NOR		<table><tr><th>A</th><th>B</th><th>$A + B$</th><th>A NOR B</th></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	A	B	$A + B$	A NOR B	0	0	0	1	0	1	1	0	1	0	1	0	1	1	1	0	L'opposto di OR
A	B	$A + B$	A NOR B																				
0	0	0	1																				
0	1	1	0																				
1	0	1	0																				
1	1	1	0																				
XOR		<table><tr><th>A</th><th>B</th><th>A XOR B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	A XOR B	0	0	0	0	1	1	1	0	1	1	1	0	1 se hai un solo 1.					
A	B	A XOR B																					
0	0	0																					
0	1	1																					
1	0	1																					
1	1	0																					

Ad eccezione della porta NOT, le altre porte logiche possono esistere anche ad *N ingressi* (2, 3, 4, N).

Esempio : Un AND a 3 ingressi



COMPONENTI ELETTRONICI																																																																																																						
NOME	SIGNIFICATO	SIMBOLO	VALORI																																																																																																			
DECODER	<p>Componente elettronico caratterizzato da:</p> <ul style="list-style-type: none">• n ingressi binari• 2^n uscite rappresentate in decimale.		<table><tr><th>A</th><th>B</th><th>C</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table>	A	B	C	0	1	2	3	4	5	6	7	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	1
A	B	C	0	1	2	3	4	5	6	7																																																																																												
0	0	0	1	0	0	0	0	0	0	0																																																																																												
0	0	1	0	1	0	0	0	0	0	0																																																																																												
0	1	0	0	0	1	0	0	0	0	0																																																																																												
0	1	1	0	0	0	1	0	0	0	0																																																																																												
1	0	0	0	0	0	0	1	0	0	0																																																																																												
1	0	1	0	0	0	0	0	1	0	0																																																																																												
1	1	0	0	0	0	0	0	0	1	0																																																																																												
1	1	1	0	0	0	0	0	0	0	1																																																																																												
MULTIPLEXOR	<p>Componente elettronico caratterizzato da:</p> <ul style="list-style-type: none">• 2n entrate principali• n entrate di controllo (selettore)• 1 uscita		$C = (A \cdot \bar{S}) + (B \cdot S)$																																																																																																			

Possiamo creare logiche a due livelli:

- **Somma di prodotti (PLA):** somma logica (OR) di prodotti (AND)
- **Prodotto di somme:** prodotto (E) di somme (OR)

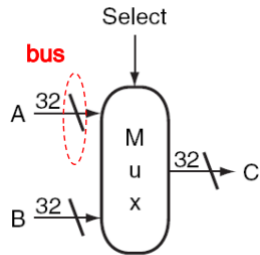
SOMMA DI PRODOTTI			
INPUT			OUTPUT
A	B	C	D
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

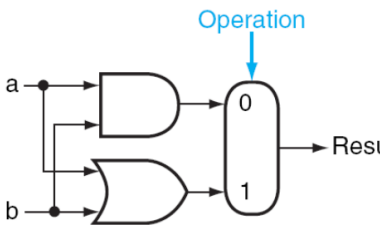
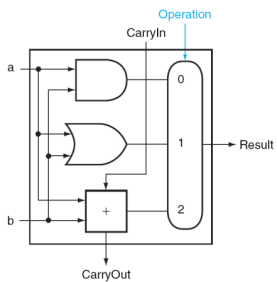
$$D = (\underline{A}BC)(\underline{A}\underline{B}\underline{C})(\underline{A}BC)(ABC)$$

(sottolineati i valori in NOT)

	ROM	PLA
<i>SPIEGAZIONE</i>	Circuito combinatorio in cui ad ogni ingresso corrisponde un'uscita	Somma di prodotti
<i>DECODIFICA</i>	Fully decoded	Partially decoded
<i>DIMENSIONI</i>	Più grandi di PLA	Meno grandi di ROM
<i>EFFICIENZA</i>	Meno efficienti di PLA	Più efficienti di ROM
<i>MODIFICABILITA'</i>	Possono implementare qualsiasi funzione logica con il numero di i/o specificato senza dover modificare la sua dimensione	Non possono
<i>SCRITTURA</i>	Possono essere solo lette e sono scritte una sola volta. Possono essere modificate solo se EROM.	

RETI COMBINATORIE PT. 2

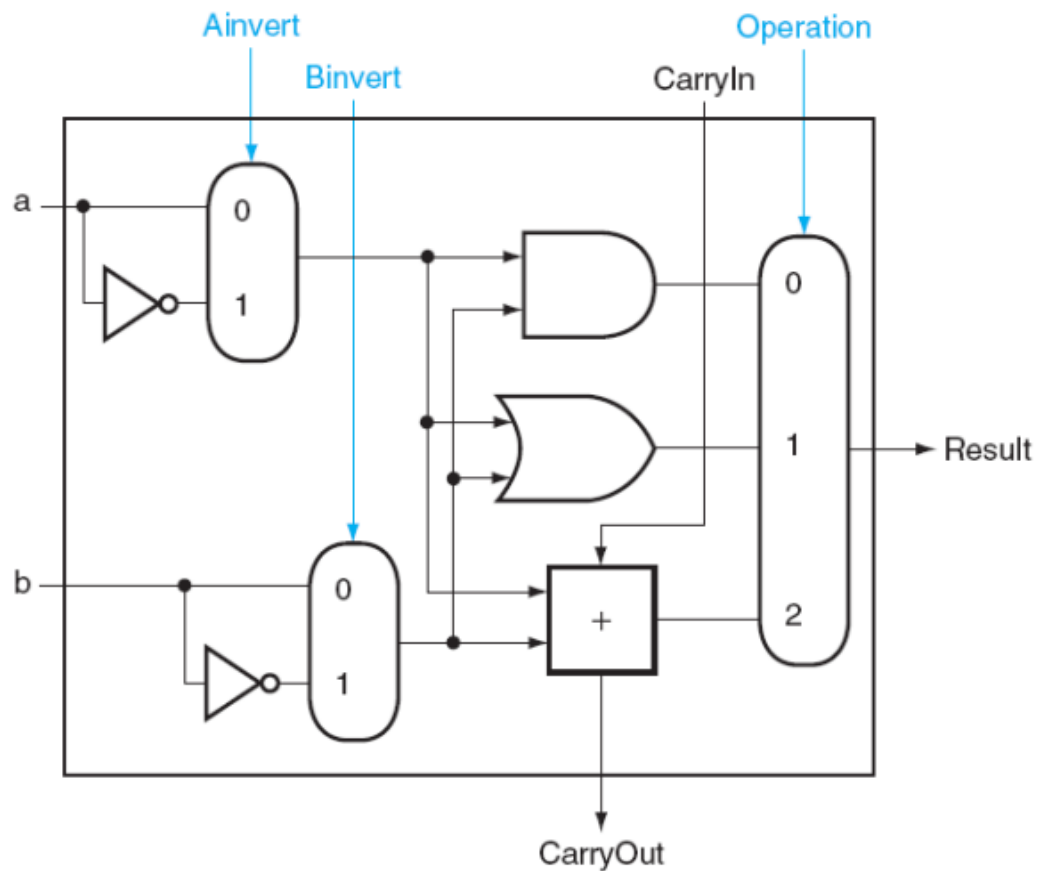
BUS	
DESCRIZIONE BUS	La maggior parte delle operazioni vengono svolte su 32 bit, mettendo in luce la necessità di creare array di elementi logici. Un bus è una collezione di linee di input che verranno trattate come un singolo segnale.
ESEMPIO BUS	 <p>The diagram shows a Multiplexer (MUX) block labeled 'Mux'. It has two 32-bit inputs, A and B, each labeled '32' with a diagonal slash. A 'Select' line is shown at the top. The output is a 32-bit line labeled 'C' with a diagonal slash. A red dashed circle labeled 'bus' encircles the input lines A and B.</p>

ALU		
<i>E' un insieme di circuiti combinatori che implementa:</i>		
Operazioni aritmetiche	Operazioni logiche	Operazioni aritmetiche e logiche: Somma + AND + OR
ADDIZIONE <pre> 0000000000000000 0001 1111 0110 1010 0000000000000000 0000 1011 1100 1011 + 0000000000000000 0000 0110 1001 1010 = 0000000000000000 0001 0010 0110 0101 </pre>	AND e OR  <p>The diagram shows two logic gates, AND and OR, with inputs 'a' and 'b'. Their outputs are connected to a 2-to-1 multiplexer. The multiplexer is controlled by an 'Operation' signal (0 for AND, 1 for OR). The output is labeled 'Res1'.</p>	 <p>The diagram shows a detailed ALU circuit. It includes AND, OR, and ADD gates. Inputs 'a' and 'b' are connected to the AND and OR gates. The output of the ADD gate is connected to a multiplexer. The multiplexer is controlled by an 'Operation' signal (0 for AND, 1 for OR, 2 for ADD). The output is labeled 'Result'. A 'CarryIn' input and a 'CarryOut' output are also shown.</p>



RETI COMBINATORIE PT. 3

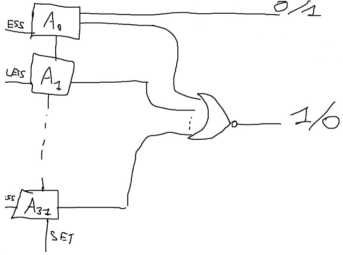
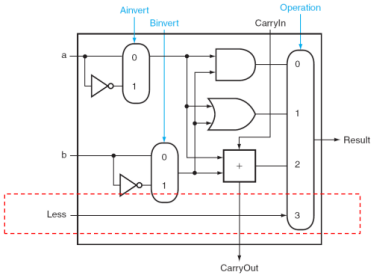
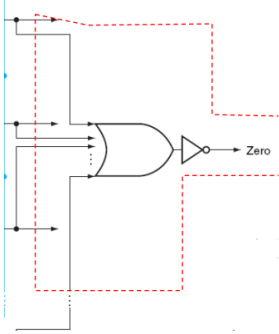
Prendiamo in considerazione questa ALU : Inverter, Inverter, AND, OR , Somma



Quadrato con + = Full Adder.

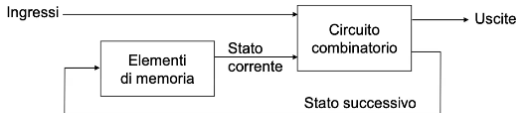
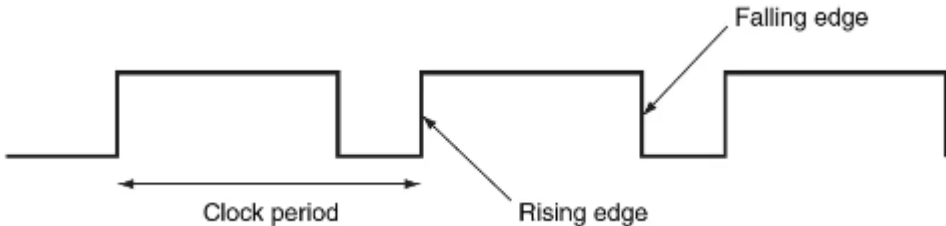
Multiplexer = coso a destra

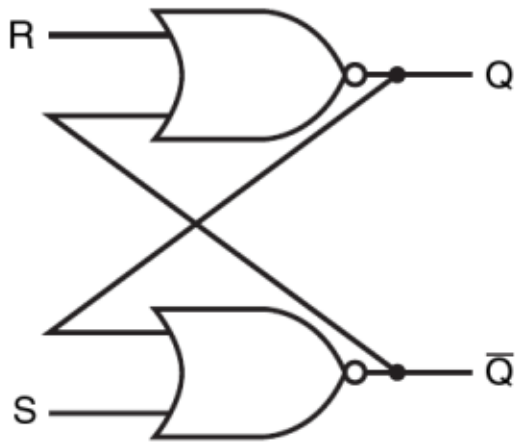
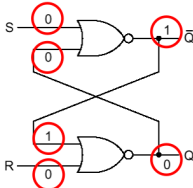
OPERAZIONI DI CONFRONTO		
	SLT set on less then	BEQ branch on equal
RISULTATO	<p>1 se $a < b$ 0 altrimenti</p> <p>Per eseguire questa istruzione si devono poter azzerare tutti i bit del bit-1 al bit-31 ed assegnare al bit-0 il valore il risultato.</p> <p>Bisogna fare una sottrazione $a < b$. se l'output è 0, sono uguali quindi ritorniamo 0 se l'output è > 0, $A > B$ quindi ritorniamo 0 se l'output è < 0, $A < B$ quindi</p>	<p>Per verificare l'uguaglianza di a e b: sottrazione.</p> <p>$a - b = 0 \leftarrow \rightarrow a = b$</p> <p>Sei fa un nor tra tutti i bit. Così se uno tra i bit da output 1 (quindi non sono uguali) con il nor sarà 0. Se invece i bit sono uguali l'output sarà 0 e con la negazione 1.</p>

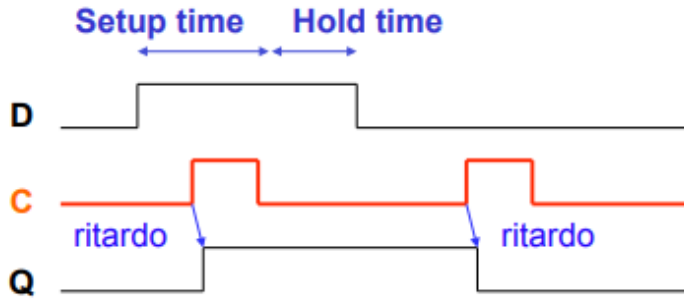
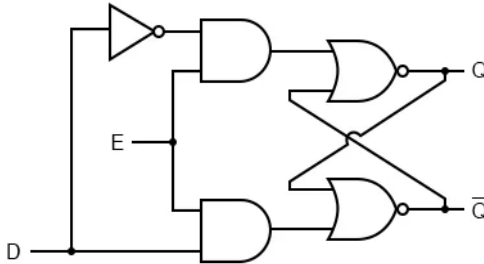
	<p>ritorniamo 1</p> <p>Affinché ciò sia possibile, ricordando che i contenuti dei registri sono in CA2, possiamo fare una set dal registro 31 al registro 0, dove l'output nel registro 0 sarà 0/1 in base ai contenuti</p>	
<p>SCHEMA ALU</p>		
<p>OVERFLOW</p>	<p>$(A-B) > 0$ bit 31 di $(A-B)=1$ $(A-B) < 0$ bit 31 di $(A-B)=0$</p>	

RETI SEQUENZIALI PT.1

<i>Combinatori</i>	<i>Sequenziali</i>
I circuiti combinatori calcolano funzioni che dipendono solo dai dati in input	I circuiti sequenziali calcolano funzioni che dipendono anche da uno stato, che dipende da informazioni memorizzate in elementi di memoria interni

CIRCUITI SEQUENZIALI	
elementi di memoria	reti combinatorie
memorizzano informazioni	elaborano informazione
	
<p>Il segnale di clock è fondamentale per le reti sequenziali che sono caratterizzate da uno stato.</p> <p>Determina il ritmo dei calcoli</p> <p>Un circuito diventa sincrono.</p> <p>Periodo T di clock</p> <p>Frequenza F ($1/T$) misurata in Hertz</p>	
<p>Un segnale (onda quadra) con un periodo predeterminato e costante</p>	
	

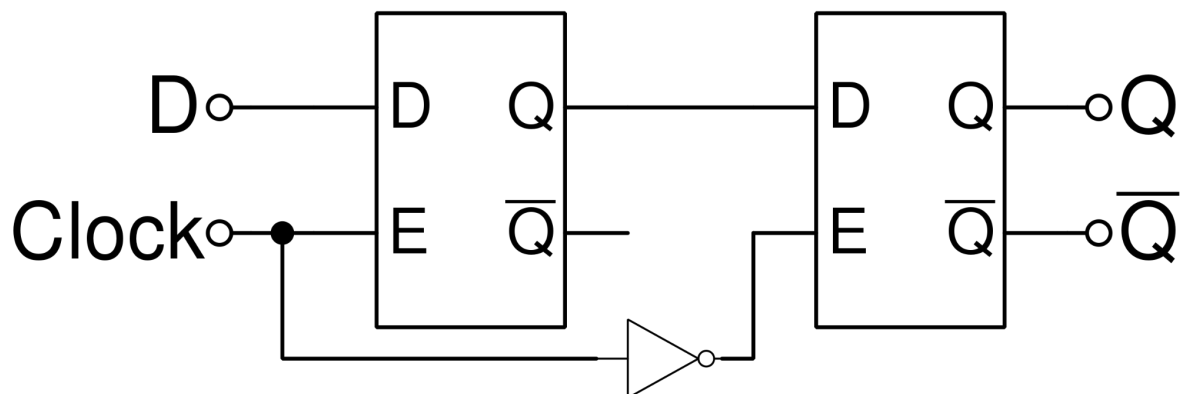
SR Latch																																			
Circuito,utilizzato come elemento di memoria, composto da 2 porte NOR concatenate. SR= Set e Reset																																			
Schema																																			
Tavola NOR	<table><tr><th>A</th><th>B</th><th>NOR</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	NOR	0	0	1	0	1	0	1	0	0	1	1	0																			
A	B	NOR																																	
0	0	1																																	
0	1	0																																	
1	0	0																																	
1	1	0																																	
Set Valori	<div><table><tr><th colspan="2">Input</th><th rowspan="2">Stato Interno Old Q</th><th colspan="2">Output</th></tr><tr><th>S</th><th>R</th><th>Q</th><th>\bar{Q}</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1/0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1/0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1/0</td><td>1</td><td>1</td></tr></table></div>	Input		Stato Interno Old Q	Output		S	R	Q	\bar{Q}	0	0	0	0	1	0	0	1	1	0	0	1	1/0	0	1	1	0	1/0	1	0	1	1	1/0	1	1
Input		Stato Interno Old Q	Output																																
S	R		Q	\bar{Q}																															
0	0	0	0	1																															
0	0	1	1	0																															
0	1	1/0	0	1																															
1	0	1/0	1	0																															
1	1	1/0	1	1																															
Stabilità	L'output richiede un certo tempo, che deve essere il più piccolo possibile e quindi evitare di memorizzare i valori intermedi. Soluzione: clock.																																		

D Latch																					
Il D-Latch è un circuito nel quale viene eliminata la condizione di indeterminazione tipica del latch SR. Per fare questo l'ingresso S viene portato all'esterno sotto il nome di D, mentre l'ingresso R non è accessibile all'esterno e riceve il segnale di D negato. Dunque S e R																					
ASSERTED	D=1 setting (S=1 e R=0) D=0 resetting (S=0 R=1)																				
DEASSERTED	Quando il clock =0																				
D= Delay	deve essere stabile quando C (clock) diventa asserted deve rimanere stabile per il Setup time (livello alto di c) deve rimanere stabile per altro tempo per evitare malfunzionamenti (Hold Time)																				
SCHEMA																					
SCHEMA	 <table data-bbox="1144 1503 1422 1682"><tr><th>E</th><th>D</th><th>Q</th><th>\bar{Q}</th></tr><tr><td>0</td><td>0</td><td>latch</td><td>latch</td></tr><tr><td>0</td><td>1</td><td>latch</td><td>latch</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	E	D	Q	\bar{Q}	0	0	latch	latch	0	1	latch	latch	1	0	0	1	1	1	1	0
E	D	Q	\bar{Q}																		
0	0	latch	latch																		
0	1	latch	latch																		
1	0	0	1																		
1	1	1	0																		
GLITCH	Gli output possono temporaneamente cambiare da valori corretti a valori errati, e ancora a valori corretti																				

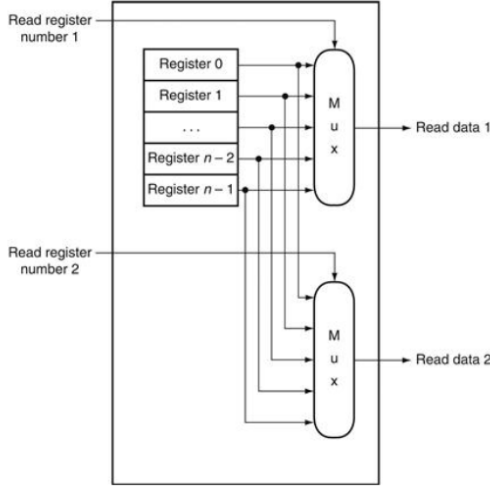
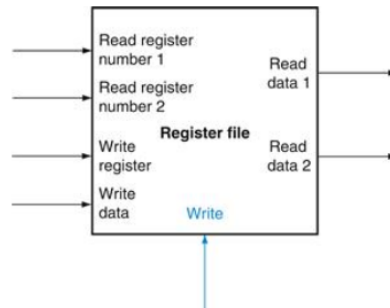
<i>level triggered</i>	<i>edge-triggered</i>
<p>Avviene sul livello alto o basso del clock. Non va bene :’(</p> <p>(Sr Latch, D Latch)</p>	<p>Avviene sul fronte di salita o di discesa del clock</p> <p>La memorizzazione avviene istantaneamente l’eventuale segnale di ritorno “sporco” non fa in tempo ad arrivare a causa dell’istantaneità della memorizzazione</p> <p>(2 D Latch -> D Flip Flop)</p>

Il *D Flip-flop* può essere utilizzato come input e output durante lo stesso ciclo di clock

Realizzato prendendo in serie 2 D-latch: il primo viene detto *master* e il secondo *slave*



RETI SEQUENZIALI PT.2

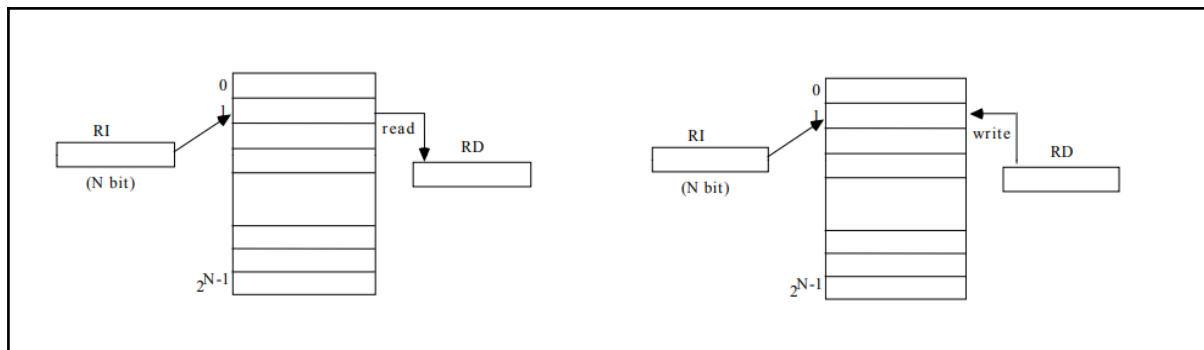
Register File	
<p>Struttura principale del datapath che consiste da un insieme di registri che possono essere letti e scritti fornendo il numero del registro da utilizzare.</p>	
<p>Dato che leggere un registro non cambia nessuno stato basta fornire in ingresso un indirizzo e si avrà come output l'informazione contenuta nel registro.</p> <p>Per scrivere un registro si avrà bisogno di 3 input:</p> <ol style="list-style-type: none"> 1. il numero del registro 2. l'informazione da scrivere 3. un segnale di clock che controlli l'operazione di scrittura <p>Read Register 1-2 : Due bus da 5 bit (Indirizzi da leggere)</p> <p>Write Register 5 bit: dice dove scrivere</p> <p>Write Data 32 bit contiene le informazioni</p> <p>Write può essere 0: non scrivi e 1: scrivi. Controllo</p> <p>Read Data (32 bit): Output</p>	 <p style="text-align: center;">un Register File con due porte di lettura e una di scrittura ha 5 input e 2 output</p>

MEMORIE

<p>Oltre alle piccole memorie implementate per mezzo di registri e file di registro, esistono altri tipi di memorie che possiamo distinguere in base a diversi parametri:</p>	<ol style="list-style-type: none"> 1. Dimensione: quantità di dati memorizzabili 2. Velocità: l'intervallo di tempo tra la richiesta del dato e il momento in cui è disponibile 3. Consumo: potenza assorbita 4. Costo: costo per bit <p>Le memorie più piccole e veloci sono poste ai livelli alti, le più ampie e lente ai bassi.</p>
<p>The diagram illustrates the memory hierarchy as a pyramid. The left side represents Capacity in bytes, and the right side represents Access time in seconds. An upward arrow on the far right indicates increasing Costo. The levels from top to bottom are: Registers (Capacity: 2^4, Access time: $5 \cdot 10^{-9}$), Cache (Capacity: 2^{19}, Access time: $20 \cdot 10^{-9}$), Main memory (Capacity: 2^{30}, Access time: $80 \cdot 10^{-9}$), Magnetic disk (Capacity: 2^{40}, Access time: $10 \cdot 10^{-3}$), and Tape/Optical disk (Capacity: $>2^{40}$, Access time: $150 \cdot 10^{-3}$).</p>	
<p>Per memorizzare dati strutturati e codice di programma, abbiamo bisogno di memorie più grandi: RAM - Random Access Memory</p>	

RAM	
Operazioni	Lettura e Scrittura
Indirizzamento	Attività con cui l'elaboratore seleziona una particolare cella di memoria
Tipi	SRAM e DRAM

<i>Lettura</i>	<i>Scrittura</i>
Il contenuto della cella di memoria indirizzata dal Registro indirizzi è copiato nel Registro Dati	Il contenuto del Registro Dati è copiato nella memoria indirizzata dal Registro Indirizzi



<i>SRAM (Static)</i>	<i>DRAM (Dynamic)</i>
Vengono usati i latch tempi di accesso 0,5 -2,5 ns Basso consumo	Memorizzazione bit tramite condensatore e quindi necessario rinfrescare il contenuto a intervalli di tempo tempi di accesso 50-70 ns
	La Dram è meno costosa perchè ha un solo transistor per bit, e un condensatore

<i>BLOCCO INDIRIZZI???</i>	
Per diminuire la complessità dei decoder è opportuno suddividere gli indirizzi in 2 blocchi	
parte alta per accedere una riga	parte bassa per accedere una specifica colonna
	Le celle consecutive hanno indirizzi che differiscono solo per la parte bassa dell'indirizzo

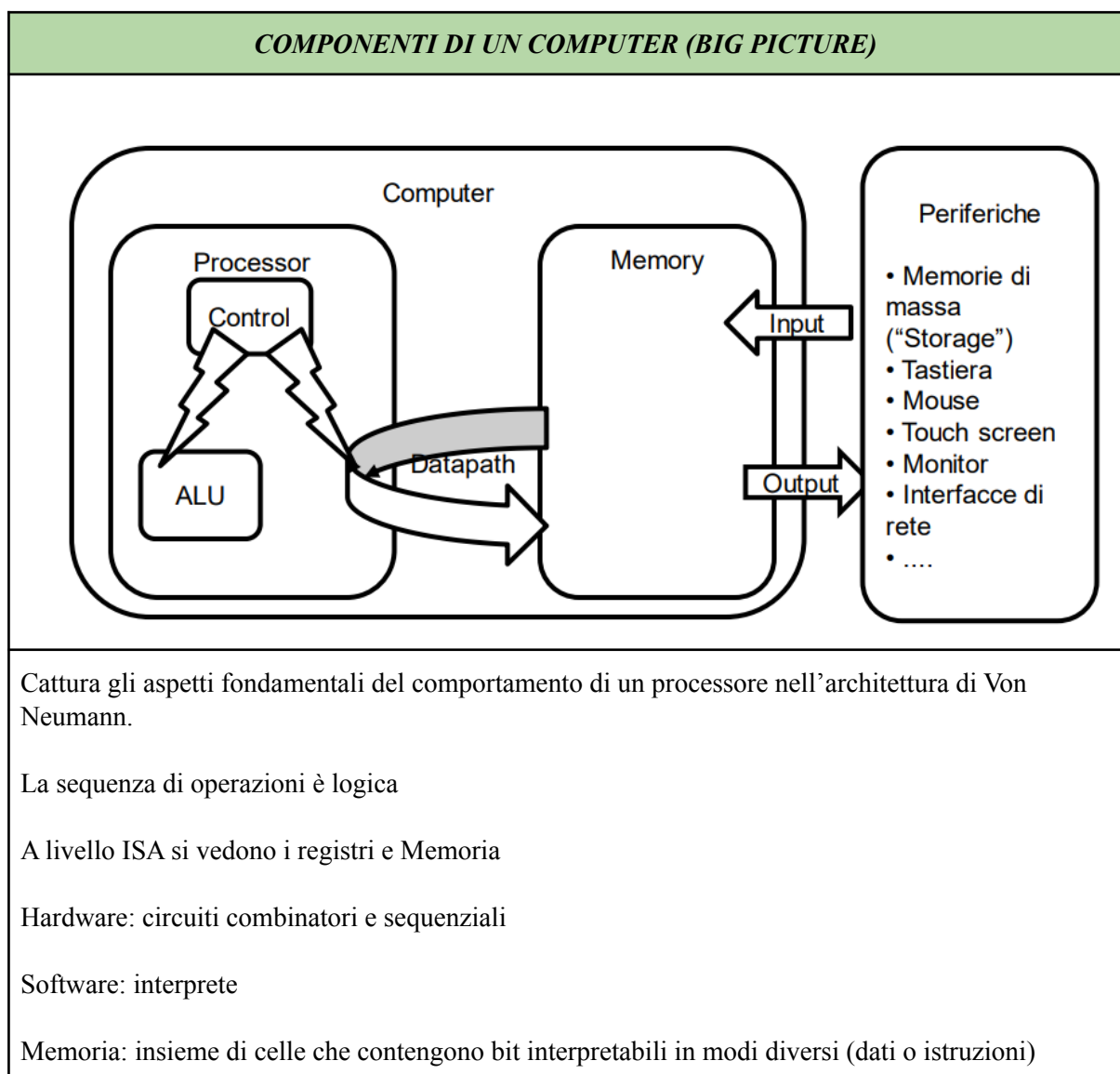
<i>SRAM SRAM</i>	
Le Synchronous SRAM e DRAM (SRAM e SDRAM) permettono di aumentare la banda di trasferimento della memoria sfruttando questa proprietà	
Memorie sincrone con segnale di clock	
E' possibile specificare che vogliamo trasferire dalla memoria una una sequenza di celle consecutive (burst)	
Ogni burst è specificato da un indirizzo di partenza, e da una lunghezza. Le celle del burst sono contenute all'interno di una stessa Riga, selezionata una volta per tutte tramite decoder	
La memoria fornisce una delle celle del burst a ogni ciclo di clock ⇒ Migliora banda di trasferimento (numero di trasf. al sec)	
Non è necessario ripresentare l'indirizzo per ottenere ogni cella del burst Costo del decoder pagato una sola volta all'inizio	

<i>Finite State Machine (FSM)</i>	
usate per descrivere i circuiti sequenziali Composte da un set di stati e 2 funzioni	
<i>Next state function</i>	<i>Output function</i>
determina lo stato successivo partendo dallo stato corrente e dai valori in ingresso	produce un insieme di risultati partendo dallo stato corrente e dai valori in ingresso
<i>Moore</i>	<i>Mealy</i>
output dipende solo dallo stato corrente	se dipende dallo stato corrente e dagli input

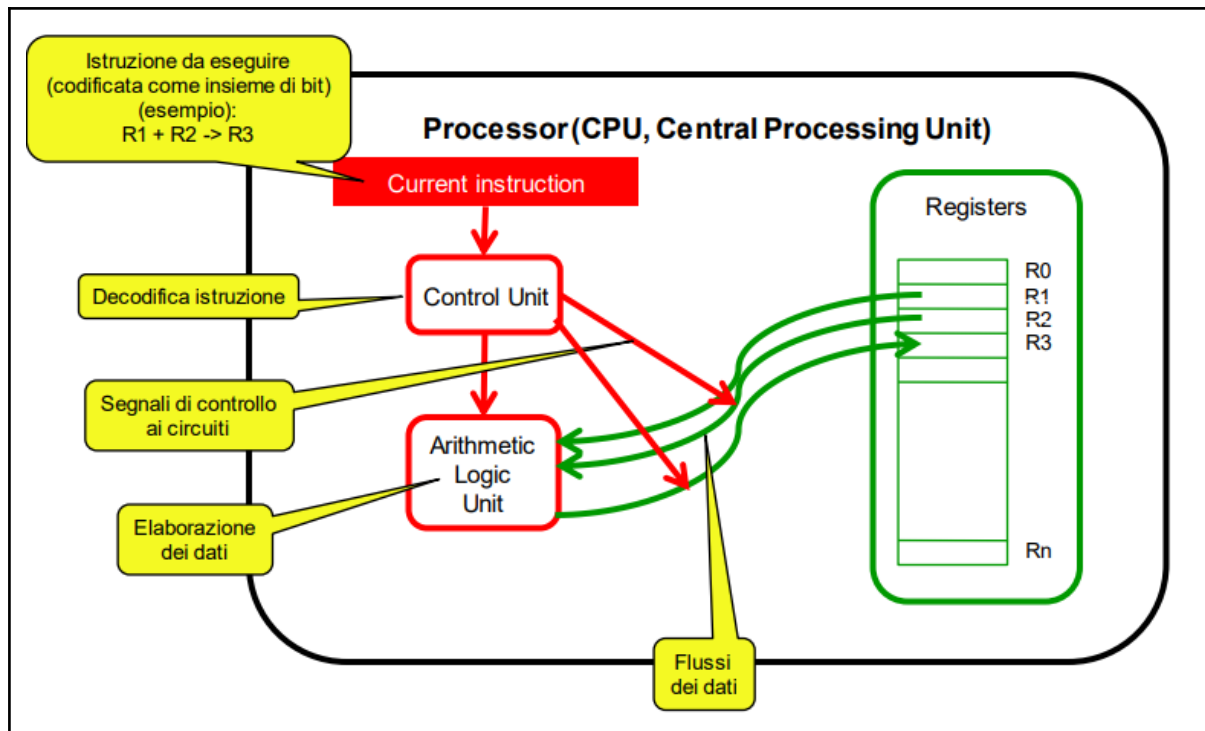
SOFTWARE - ISA (ID 3)

INSTRUCTION SET ARCHITECTURE PT 1

OUTLINE	
Cosa fa?	ISA (Instruction Set Architecture)
Come si programma	Assembly
Come è fatto?	Circuiti e DataPath



PROCESSORE (CPU, CENTRAL PROCESSING UNIT)

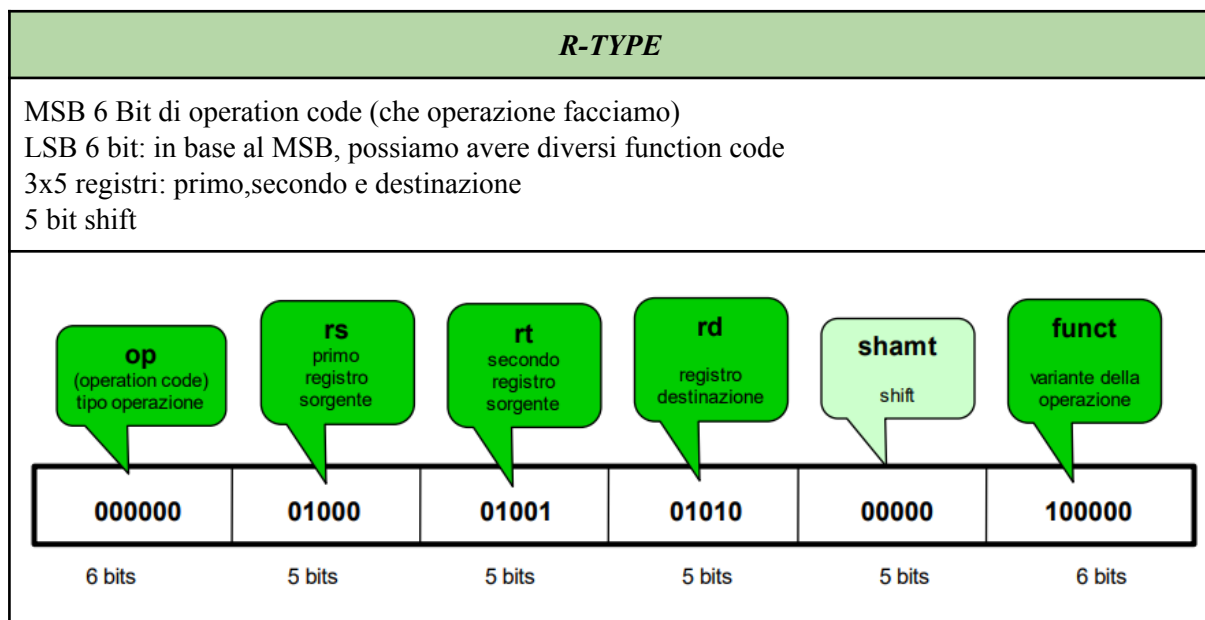


MEMORIA	
CPU	Dentro la CPU i registri sono pochi, veloci e costosi.
Memoria	La Memoria è grande (Gigabyte) , relativamente lenta e mediamente costosa
Memoria di massa	La Memoria di massa è una periferica molto grande (Terabyte) lenta, poco costosa e persistente

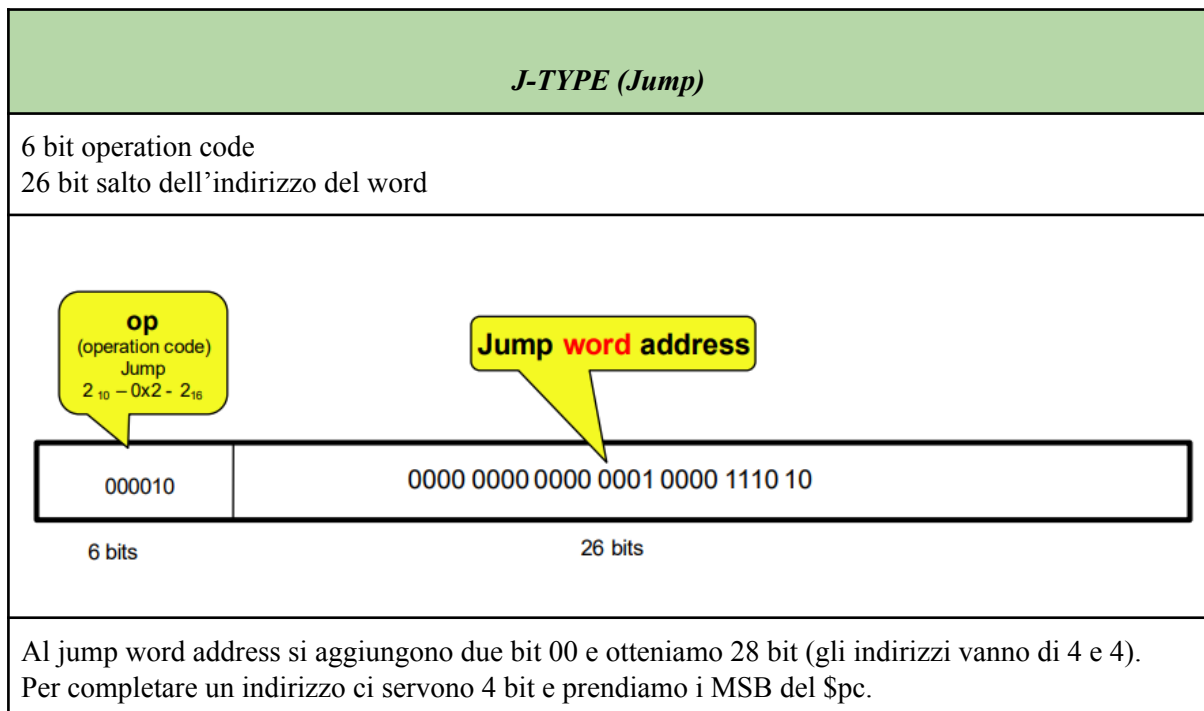
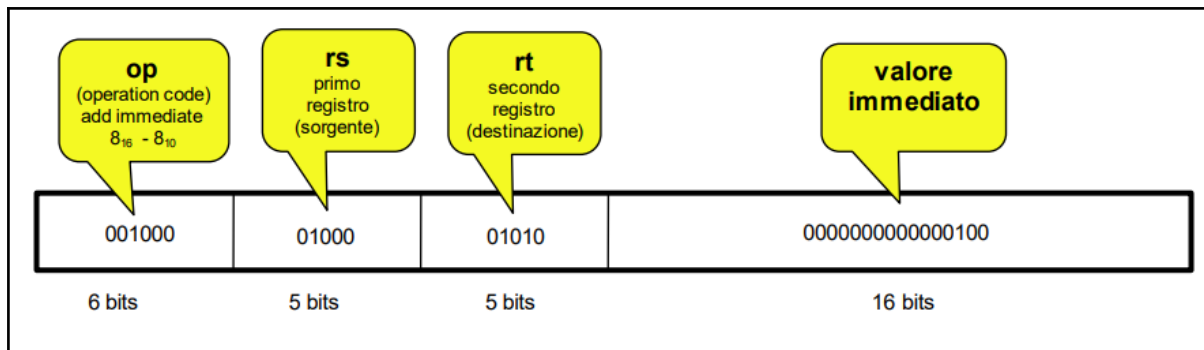
FILOSOFIE DI PROGETTO CPU	
<i>RISC</i>	<p>Poche istruzioni semplice Circuito semplice Esecuzione veloce della singola istruzione Occorrono più istruzioni per cose semplici</p> <p>Es: MIPS; ARM</p>
<i>CISC</i>	<p>Istruzioni complesse Circuito complicato Esecuzione lenta della singola istruzione Occorrono meno istruzione</p>

	Es: Intel x86
--	---------------

MIPS	
<i>Registri</i>	32 registri di 32 bit
<i>Istruzioni</i>	32 bit
<i>Dati</i>	Manipolazione solo su registri. Trasferimento tra memoria e registri.
<i>Formati Istruzione</i>	R-Type I-Type J-Type



I-TYPE (add immediate)
MSB 6 Bit operation code, 5 bit primo registro 5 bit registro destinazione 16 bit costante



ASSEMBLY

Registri	
<i>Registri di sistema</i>	<i>Registri utilizzabili</i>
\$zero rappresenta il valore zero \$ra registro del valore di ritorno \$pc: Program Counter, contiene l'indirizzo della prossima istruzione da eseguire.	\$t0 - \$t9 per variabili temporanee (Temporary) \$s0 - \$s7 per variabili da salvare (saved) \$a0 - \$a3 parametri (Arguments)
Direttive	
.text indica che tutto quello che viene dopo verrà salvato nello user text segment. Tipicamente sono le istruzioni .data tutto quello che viene dopo verrà salvato nel data segment, tipicamente i dati salvati in memoria	
Strutture dati	
<i>Array</i>	<i>Stringhe</i>
.word w1, ..., wn collezione di valori di 32 bit .half h1, ..., hn collezione di valori di 16 bit .byte b1, ..., bn collezione di valori di 8 bit	.ascii "str" salva la stringa in caratteri ascii .asciiz "str" salva la stringa in caratteri ascii, e aggiunge il carattere di terminazione '\0'.

Operazioni Matematiche		
<i>ISTRUZIONE</i>	<i>ESEMPIO</i>	<i>COMMENTI</i>
add	add \$1,\$2,\$3	
subtract	sub \$1,\$2,\$3	
add immediate	addi \$1,\$2,100	somma immediatamente un numero costante
multiply (without overflow)	mul \$1,\$2,\$3	il risultato `e in 32 bit
multiply	mult \$2,\$3	il risultato `e diviso in 32 e 32 bit, rispettivamente in hi e low

divide	div \$2,\$3	quoziente nel registro low, resto nel registro hi
--------	-------------	--

<i>Operazioni Logiche</i>		
<i>ISTRUZIONE</i>	<i>ESEMPIO</i>	<i>COMMENTI</i>
and	and \$1,\$2,\$3	and tra i singoli bit
or	or \$1,\$2,\$3	or tra i singoli bit
shift left logical	sll \$1,\$2,10	shifta a sinistra di un numero costante di bit
shift right logical	mul \$1,\$2,\$3	shifta a sinistra di un numero costante di bit

<i>Operazioni di Trasferimento</i>		
I dati sono in due posizioni diverse: <ul style="list-style-type: none"> • Memoria che è l'area che accoglie i dati inseriti, nel codice, nella sezione .data • Registri che vengono usati e manipolati nel codice 		
<i>ISTRUZIONE</i>	<i>ESEMPIO</i>	<i>COMMENTI</i>
load word	lw \$1,0(\$2)	copia dalla memoria al registro
store word	sw \$1,0(\$2)	copia dal registro alla memoria
load address	la \$1,label	carica l'indirizzo di una label nel registro
load immediate	li \$1,100	carica il valore nel registro
move	move \$1,\$2	copia da registro a registro

<i>Rami Condizionali</i>

<i>ISTRUZIONE</i>	<i>ESEMPIO</i>	<i>COMMENTI</i>
beq	beq \$1,\$2,label	==
bne	bne \$1,\$2,label	!=
bgt	bgt \$1,\$2,label	>
bge	bge \$1,\$2,label	>=
blt	blt \$1,\$2,label	<
ble	ble \$1,\$2,label	<=
slt	slt \$1,\$2,\$3	if(\$2!= \$3)\$1=1; else \$1=0
slti	slti \$1,\$2,100	if(\$2!=100)\$1=1; else \$1=0
I cicli vanno costruiti attraverso l'uso dei jump. Un Jump serve per saltare ad un registro o ad una label, si usa nella forma j label		

CATENA PROGRAMMATIVA

<i>CATENA PROGRAMMATIVA</i>	
Compilatore	Programma Passato al compilatore che traduce da alto livello a linguaggio macchina oppure ad assembly
Assembler	Codice passato all'assembler, traduce in istruzione binario (file object)
Linker	Passato al Linker, collega con altri file obj/lib (file exe)
Loader	Loader carica programma su pc e può essere eseguito