SISTEMI DIGITALI

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Lezioni

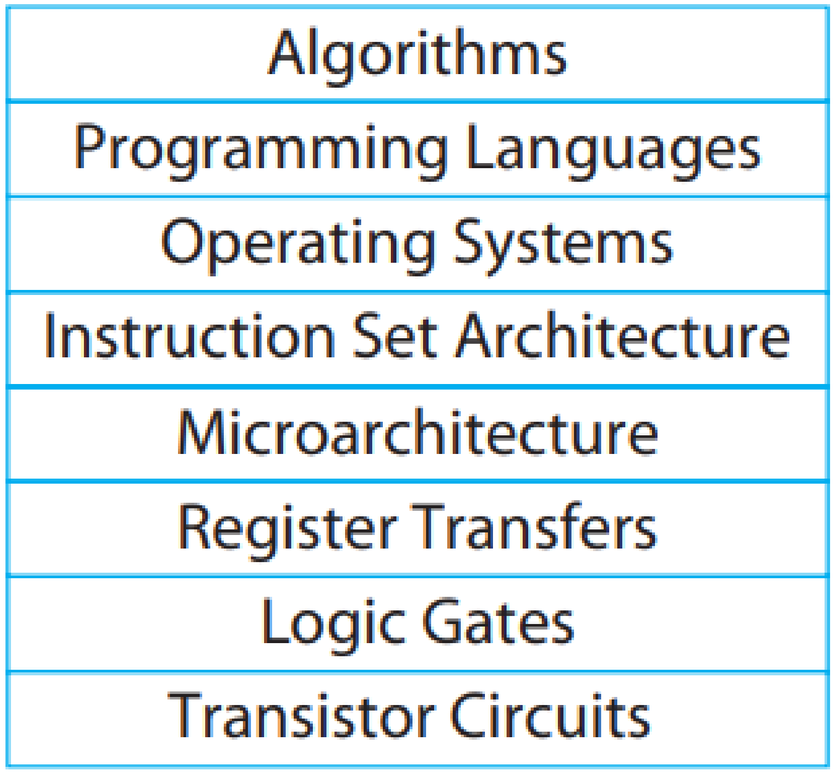
Come sempre:

- Martedì 1230
- Mercoledì 1230
- Giovedì 1230

- Lezioni «interattive»
  - Fate domande in aula / nel forum
  - In differita sandro.savino@dei.unipd.it
    sandro.savino@unipd.it

Prof. Savino
(al radiotelescopio di Arecibo)

Dove andremo



Argomenti delle prossime settimane (cap 8-12):

- Datapath e ALU
- Opcode e Control word
- L'uso delle memorie
- Input e Output
- Argomenti avanzati (es. pipeline)
- Esercizi

Argomenti degli ultimi mesi

Dove andremo

CPU e Datapath	Microoperazioni	ISA
Esercizi	Memorie	Cache
Esercizi	Memorie / Extra	IO
IO	Esercizi	Esercizi / Extra

Cap 8

Cap 7
Cap 12

Cap 9
Cap 10
Cap 11

Architettura di un elaboratore

Datapath e Arithmetic/Logic Unit (ALU)

Sandro Savino (sandro.savino@dei.unipd.it)
Department of Information Engineering, University of Padova

Argomenti:

- Programmazione in hardware/software
- Struttura di un elaboratore elementare
- Progettazione di una ALU

Materiale:

- Capitolo 8

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DOMANDE

- A cosa serve questo corso?
- A capire come funzionano / come costruire un sistema digitale

- A cosa serve un sistema digitale?
- A fare quello che vogliamo noi!

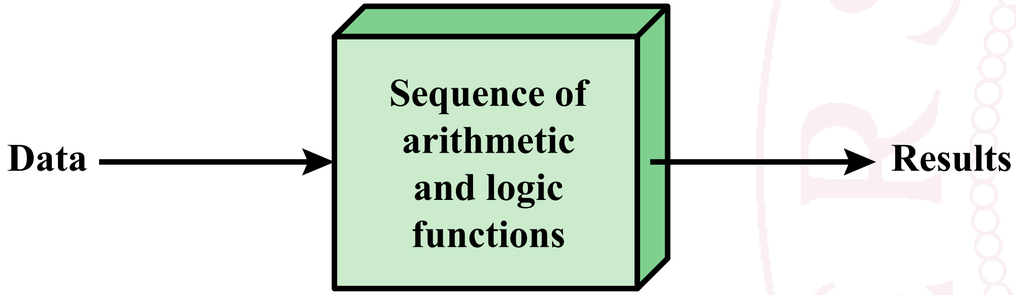
- Come si fa a convincere un sistema digitale a fare quello che vogliamo noi?
- Con la programmazione …

Programmazione in hardware e software

Come si implementa un programma?

Programmazione in hardware (hardwired programming)

- Esiste un insieme di componenti logiche base (and, or, multiplexer, ROM,…)
- Un programma è un opportuno circuito costituito da questi elementi
- I dati di input vengono processati dal circuito per produrre il risultato di output



Programmazione in hardware

PRO

- Permette di implementare un programma con area del circuito minima
- Esecuzione veloce
- Risparmio energetico

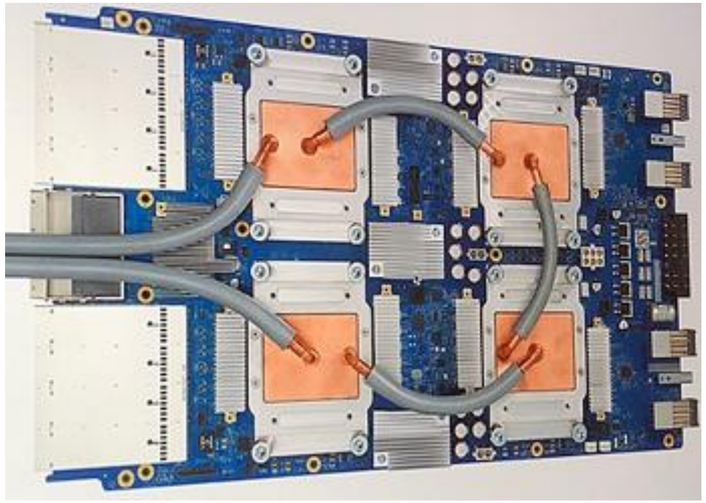
CONTRO

- Il programma non può essere modificato
- Costo di implementazione
- Approccio specifico e non generale

Pentium FDIV bug
https://it.wikipedia.org/wiki/Pentium_FDIV_bug

Esempi di programmazione in hardware

- Questa tecnica è attualmente molto usata per accelerare l’esecuzione di problemi di machine learning



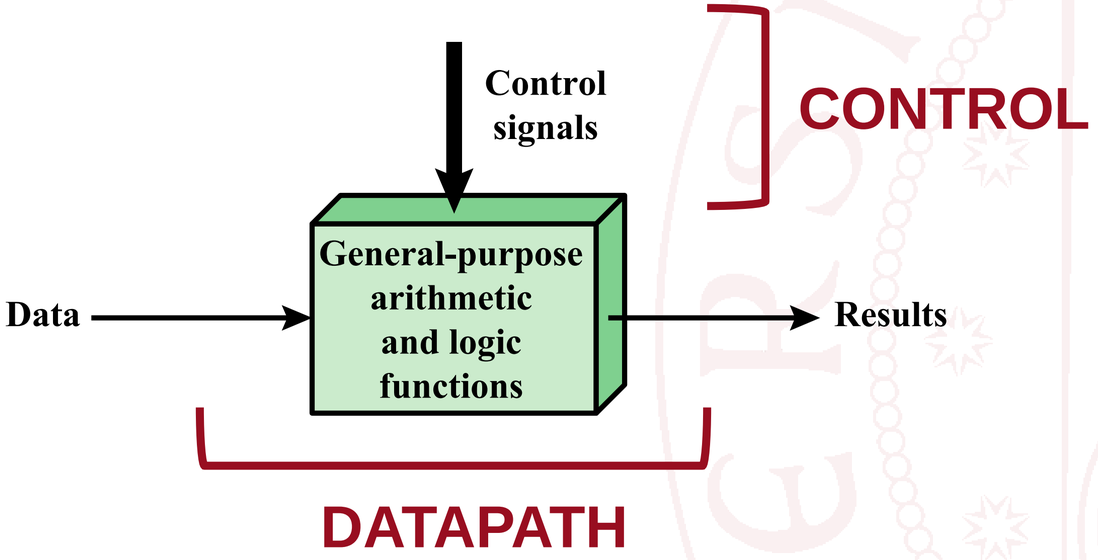
Google Tensor Processing Unit

Nvidia Quadro Volta

Programmazione in software

Programmazione in software:

- Una famiglia di funzioni logiche e aritmetiche è implementata in hardware
- Dei **segnali di controllo** indicano quali funzioni, e in che ordine, attivare sui dati di input
- Si possono implementare più funzioni variando i segnali di controllo (HW general purpose)



Programmazione in software

PRO

- Il programma può essere modificato
- Costo di implementazione di un programma ridotto
- Universalità

CONTRO

- Minor efficienza di calcolo
- Circuiti più estesi e costosi
- Maggior consumo energetico

Confronto

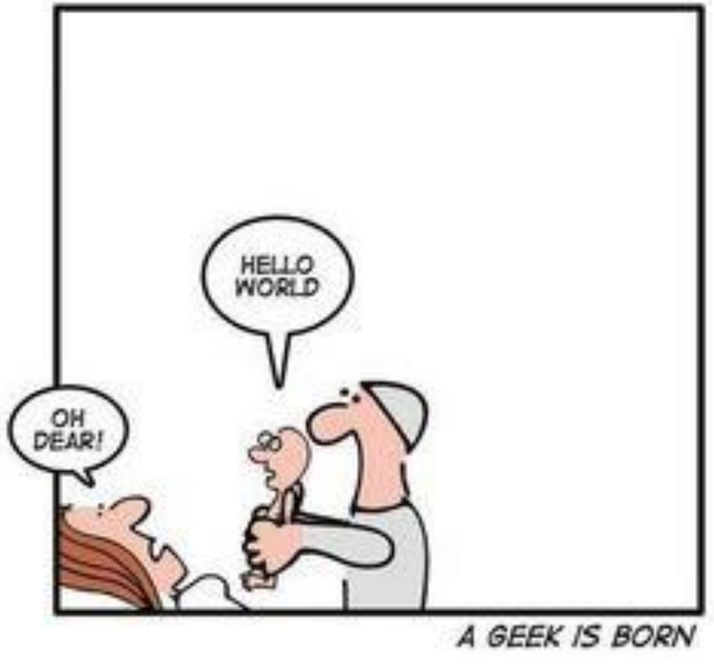
Programmazione in hardware

Programmazione in software

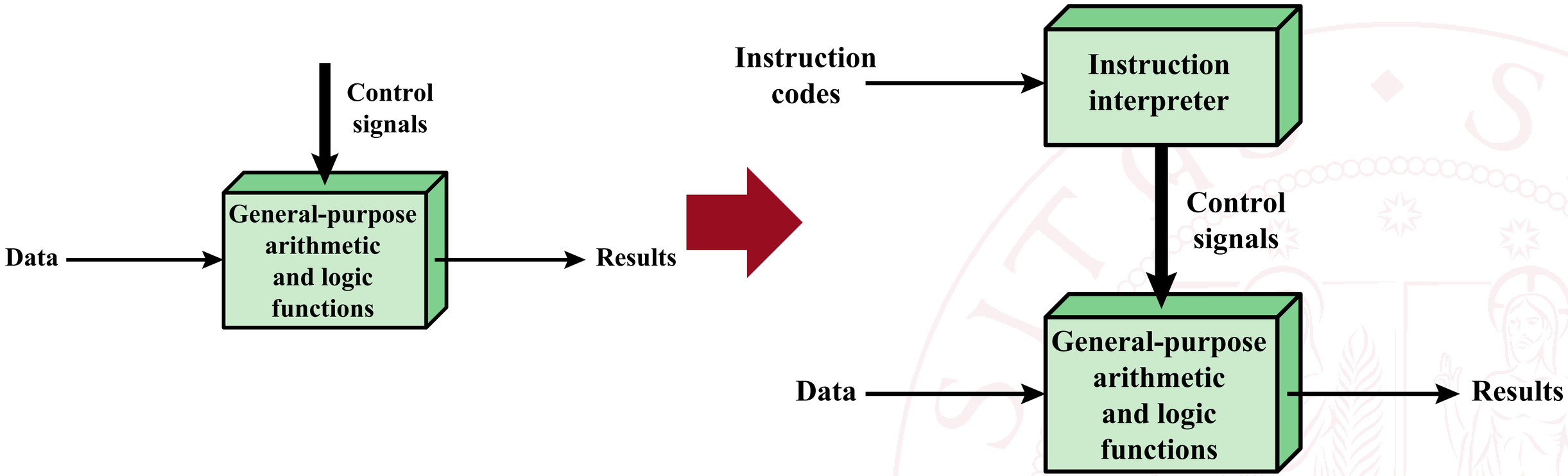
Struttura di un elaboratore

Come definire i segnali di controllo?

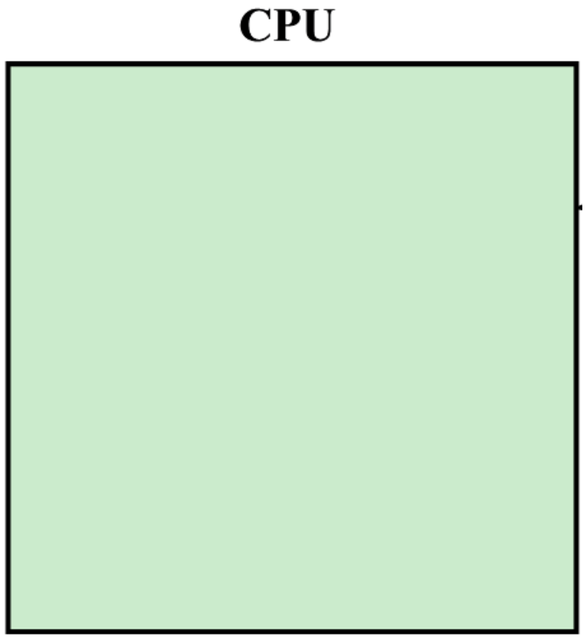
- Si assegna un **codice** univoco ad ogni possibile sequenza di segnali
- Si aggiunge una componente hardware per tradurre questi codici in segnali
- Ogni codice denota un'**istruzione** che determina in modo univoco i segnali da attivare
- Un programma è definito dalla sequenza di istruzioni/codici (**software**)



Un sistema per la programmazione in software



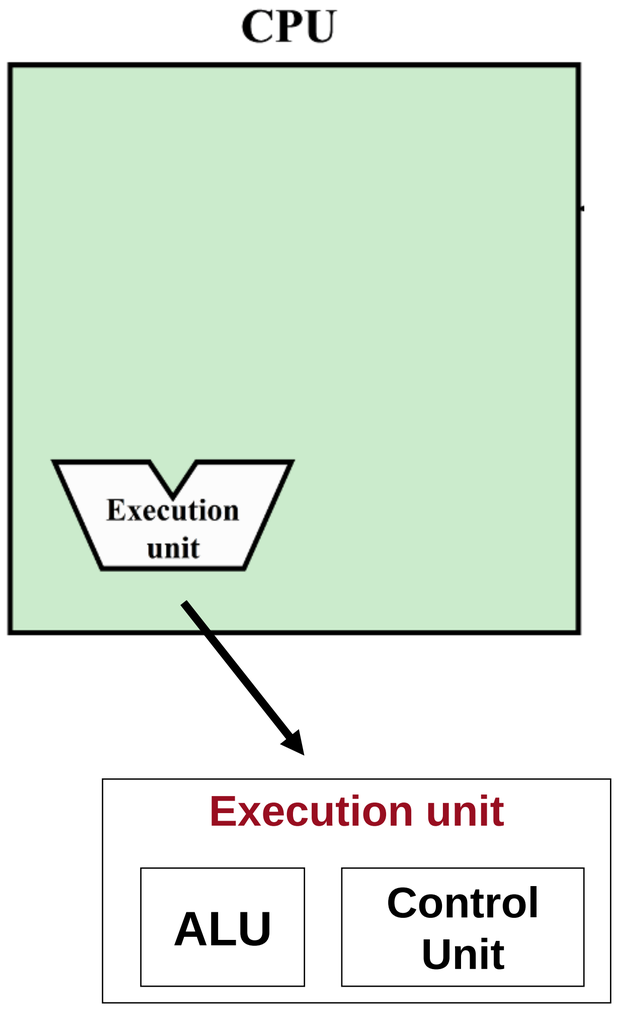
Un sistema per la programmazione in software: CPU



Serve un modulo per interpretare ed eseguire le istruzioni

Il modulo è chiamato
Central Processing Unit (CPU)

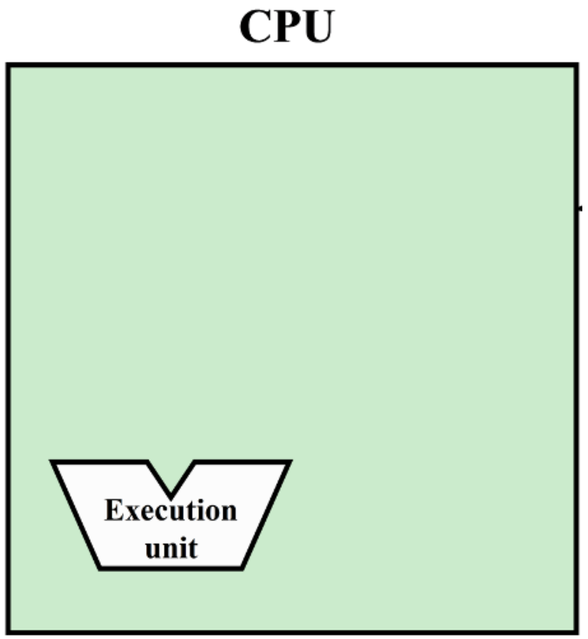
Un sistema per la programmazione in software: CPU

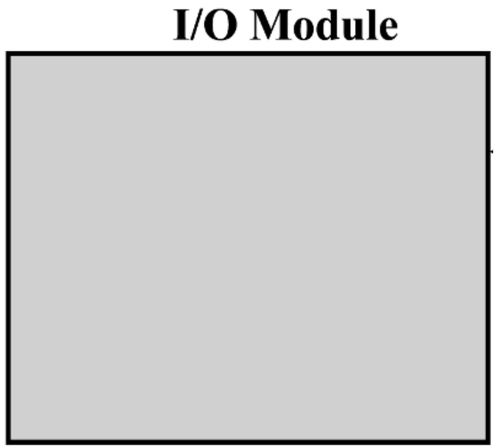


L’unità di esecuzione contiene delle unità per:

- Arithmetic and Logic Unit (ALU)
- Unità di controllo

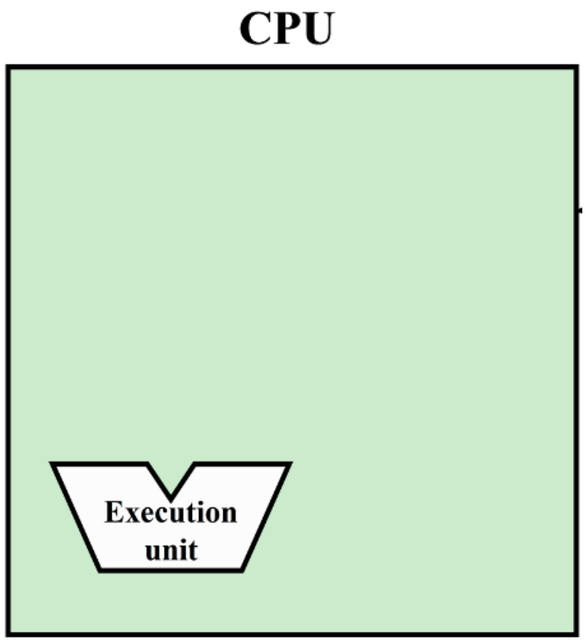
Un sistema per la programmazione in software: I/O

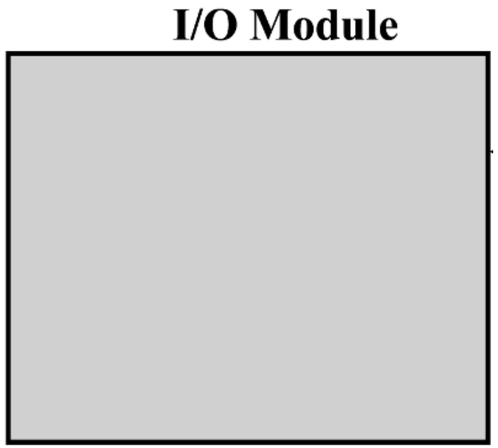


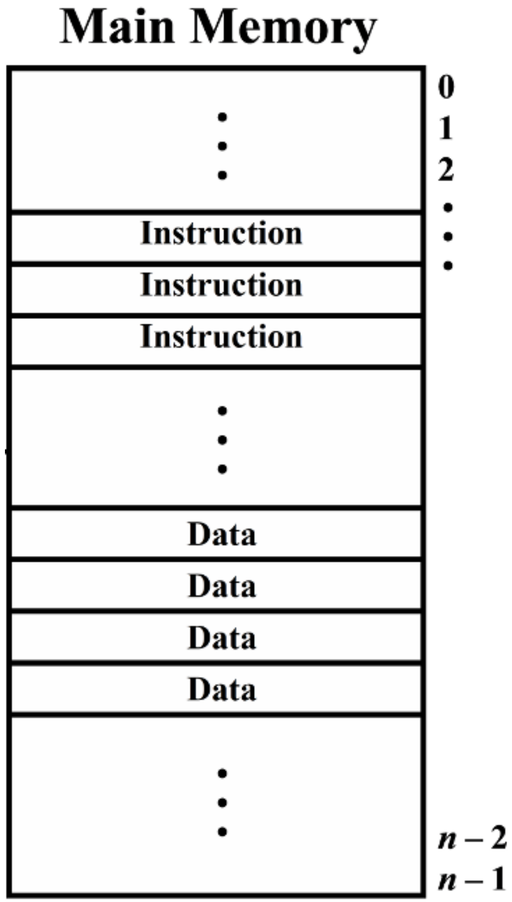


Serve un **modulo input/output** (I/O) per immettere l'input (dati e software) ed emettere l'output (risultato)

Un sistema per la programmazione in software: memoria





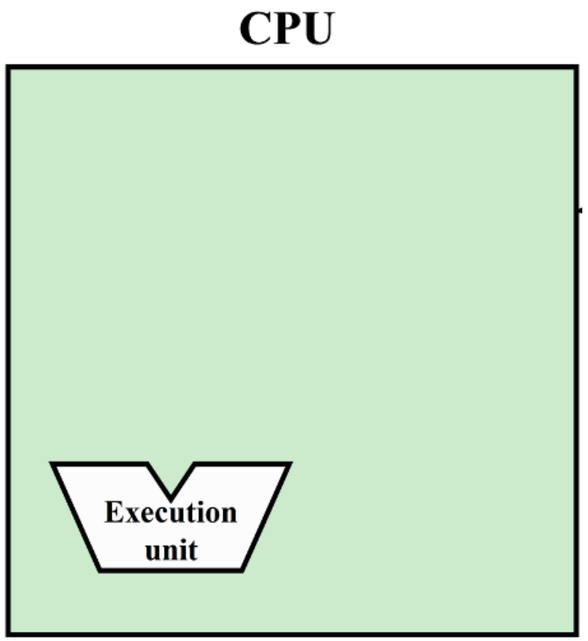


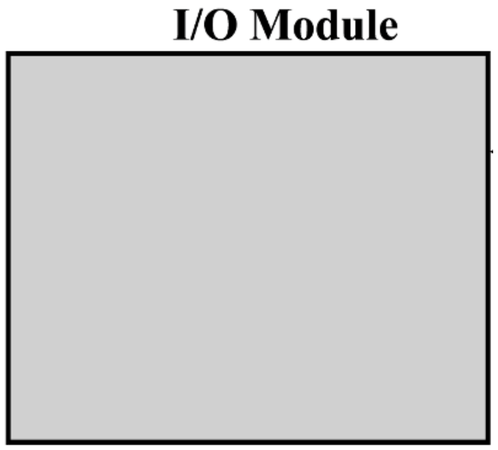
Serve una **memoria** per salvare:

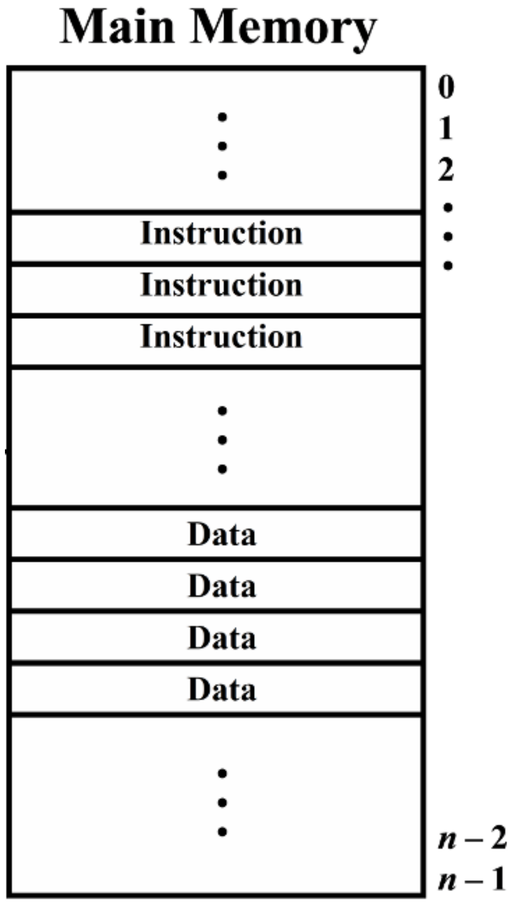
- Istruzioni
- Dati di input
- Dati temporanei

La memoria permette di accedere a dati/istruzioni durante l'esecuzione

Un sistema per la programmazione in software: memoria (2)



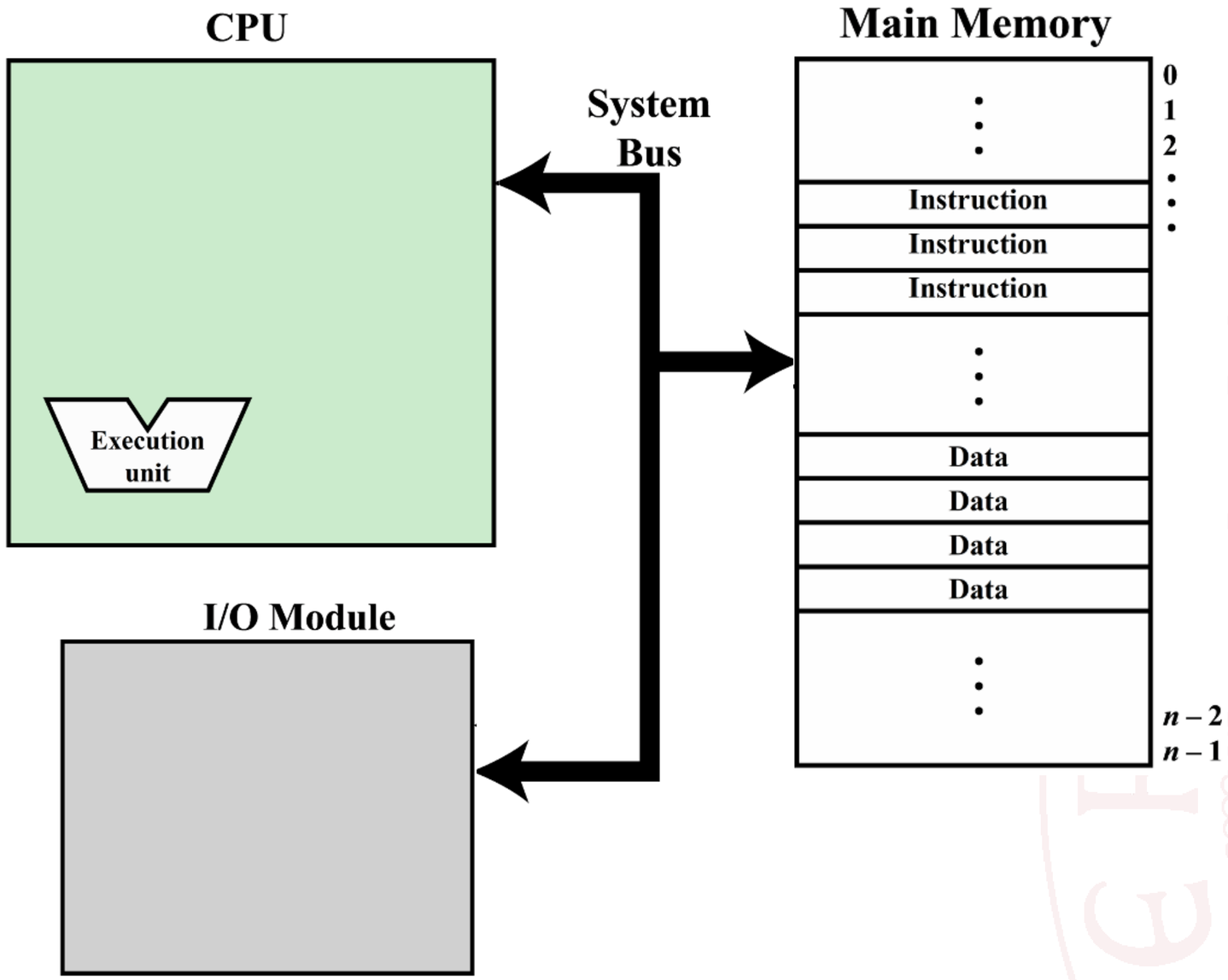




La memoria è divisa in **locazioni**

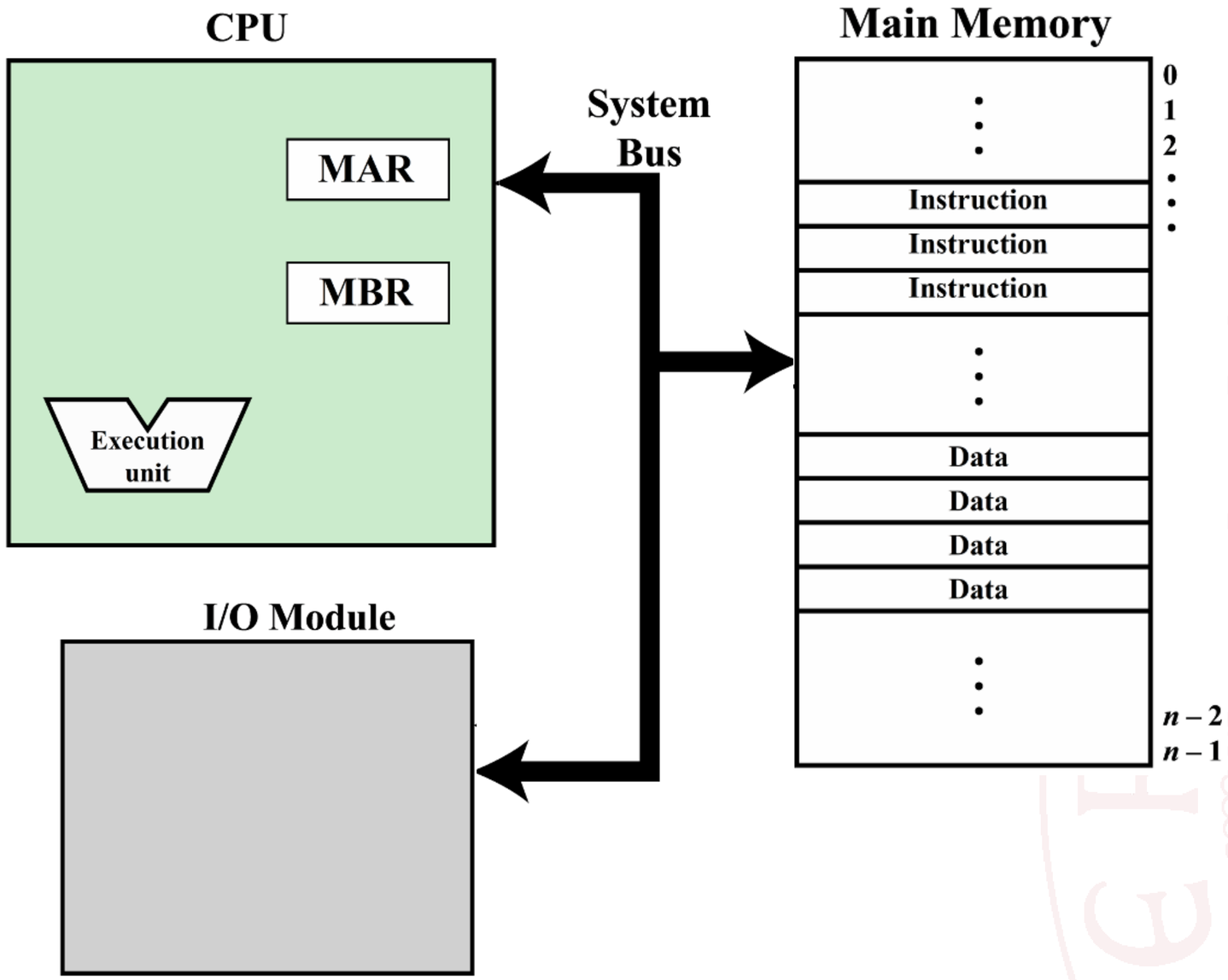
Ogni locazioni è accessibile tramite un **indirizzo**

Un sistema per la programmazione in software: bus



Serve un **bus** per connettere i vari moduli

CPU e memoria



La CPU legge o scrive dati/istruzioni in memoria tramite due registri:

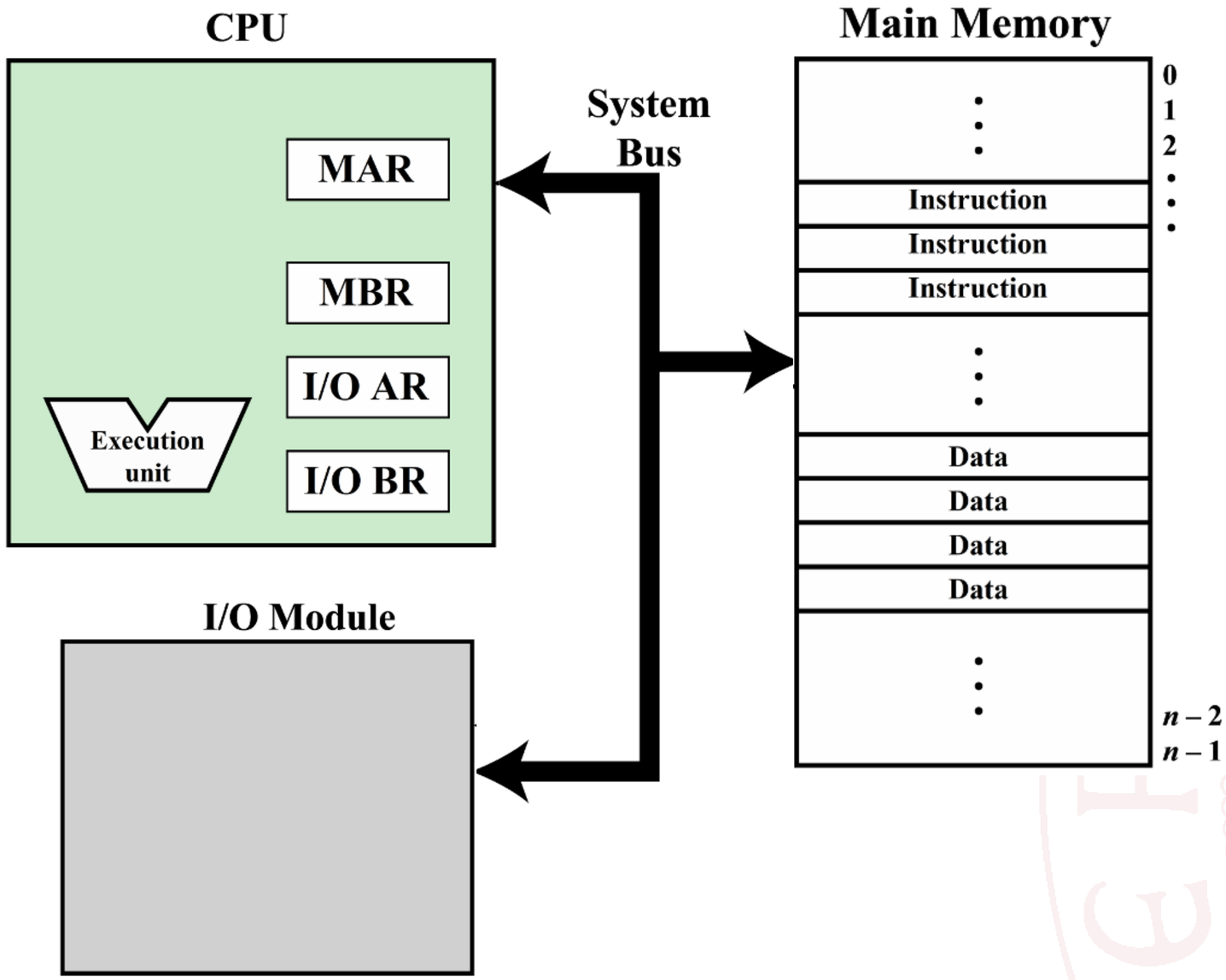
- **MAR**: memory address register

DOVE?

- **MBR**: memory buffer register

COSA?

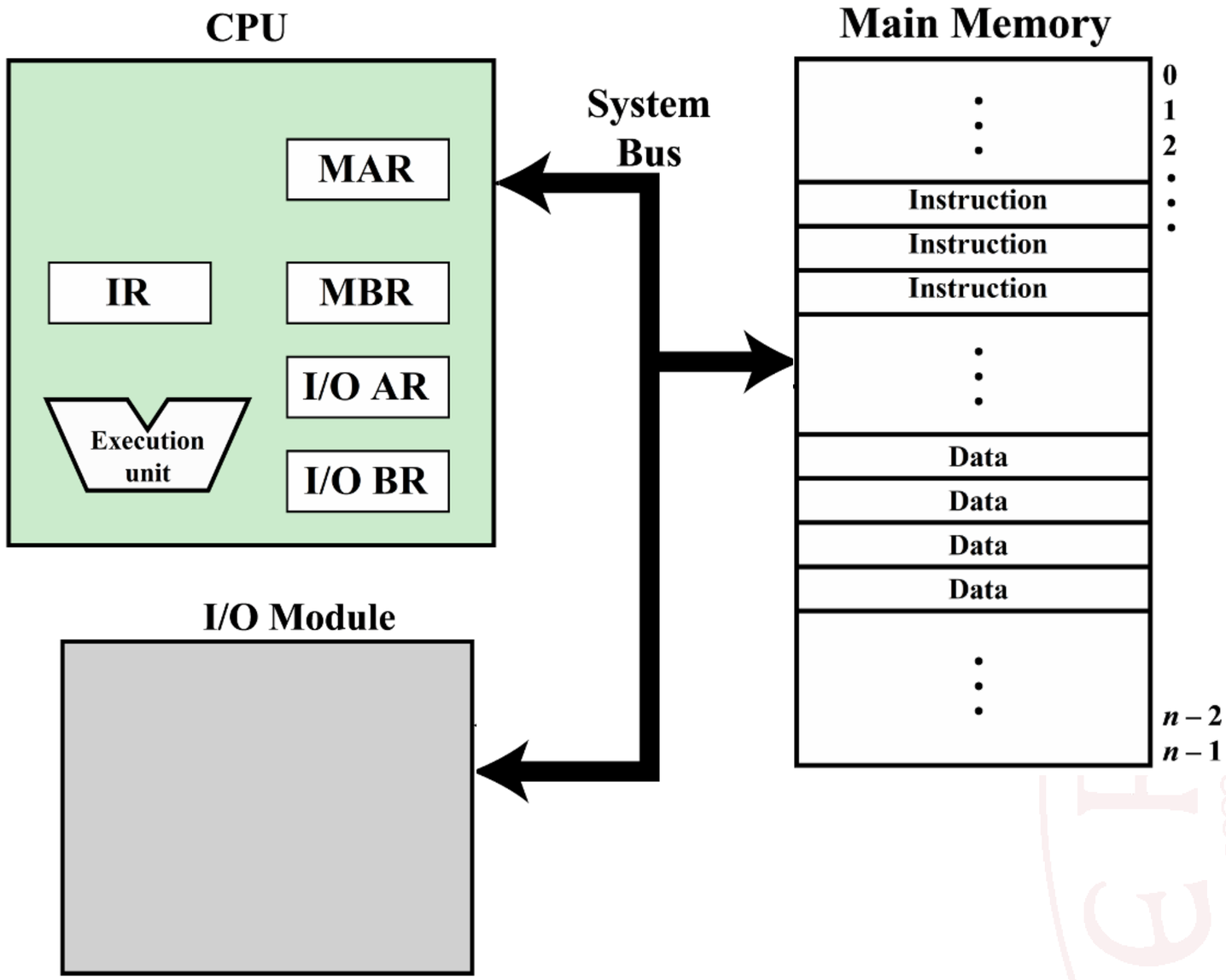
CPU e I/O



La CPU legge o scrive dati dall'I/O tramite due registri:

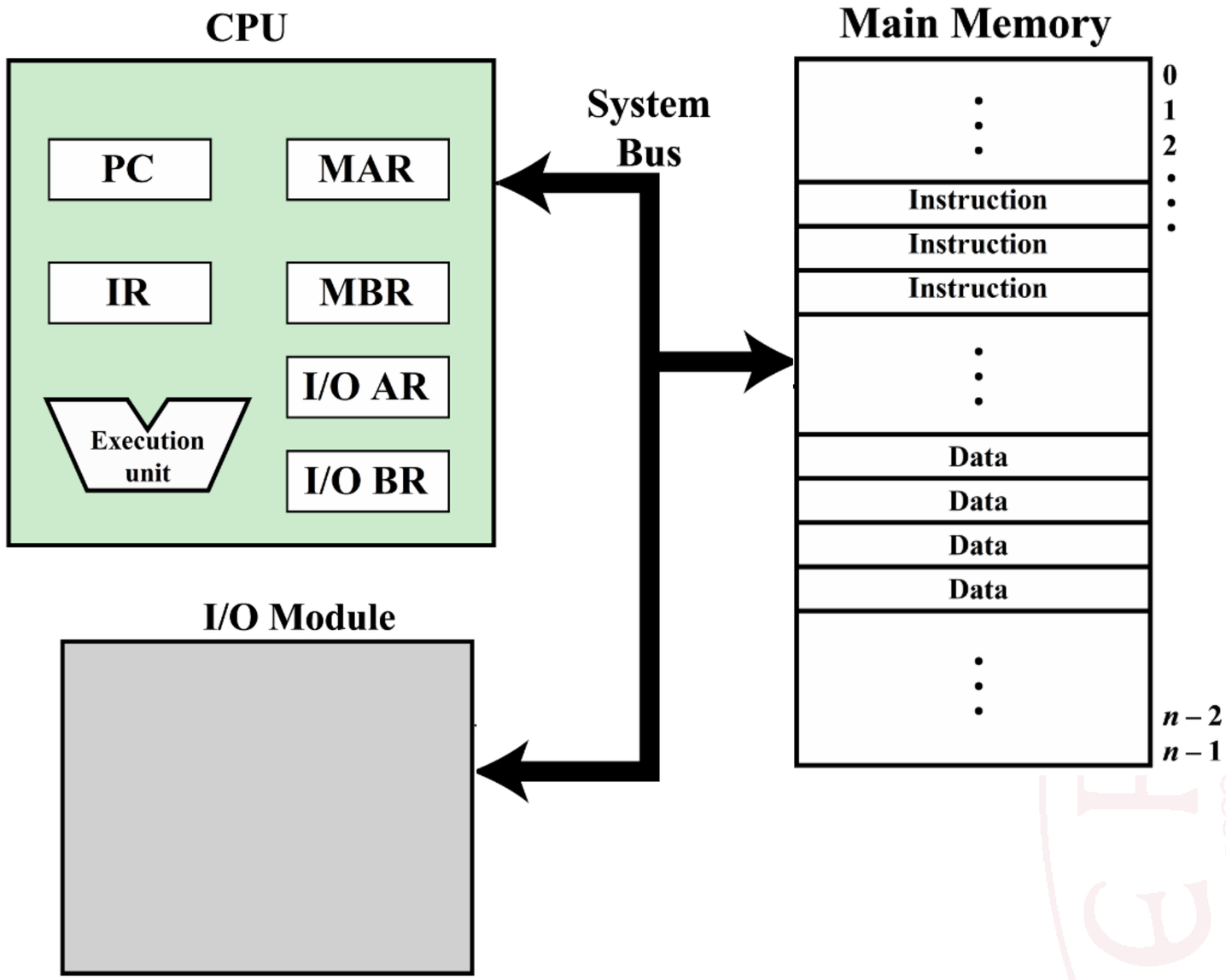
- **I/O AR**: I/O address register
- **I/O BR**: I/O buffer register

CPU e istruzioni



La CPU mantiene una copia locale dell'istruzione da eseguire nel registro **IR** (Instruction Register)

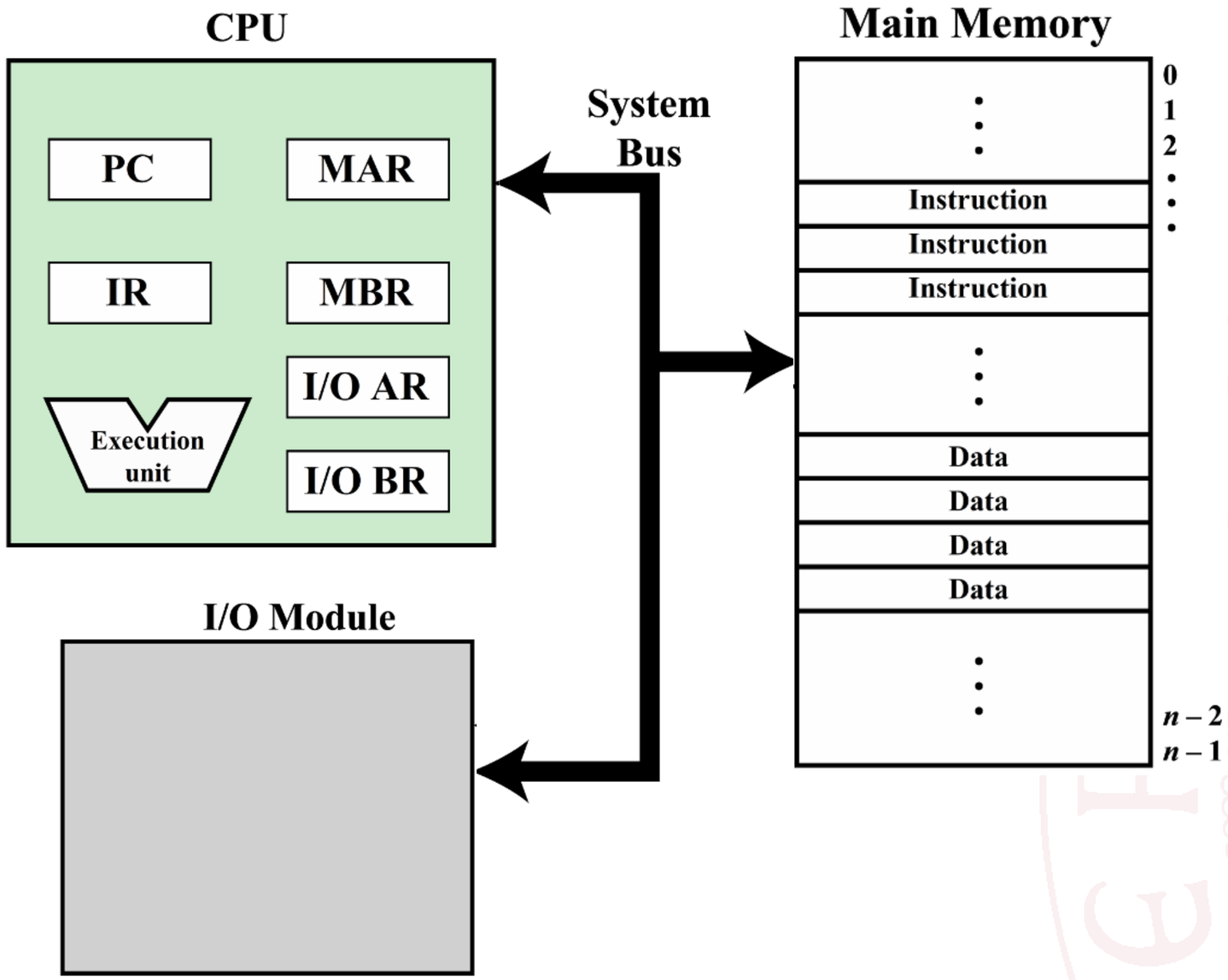
CPU e istruzioni (2)



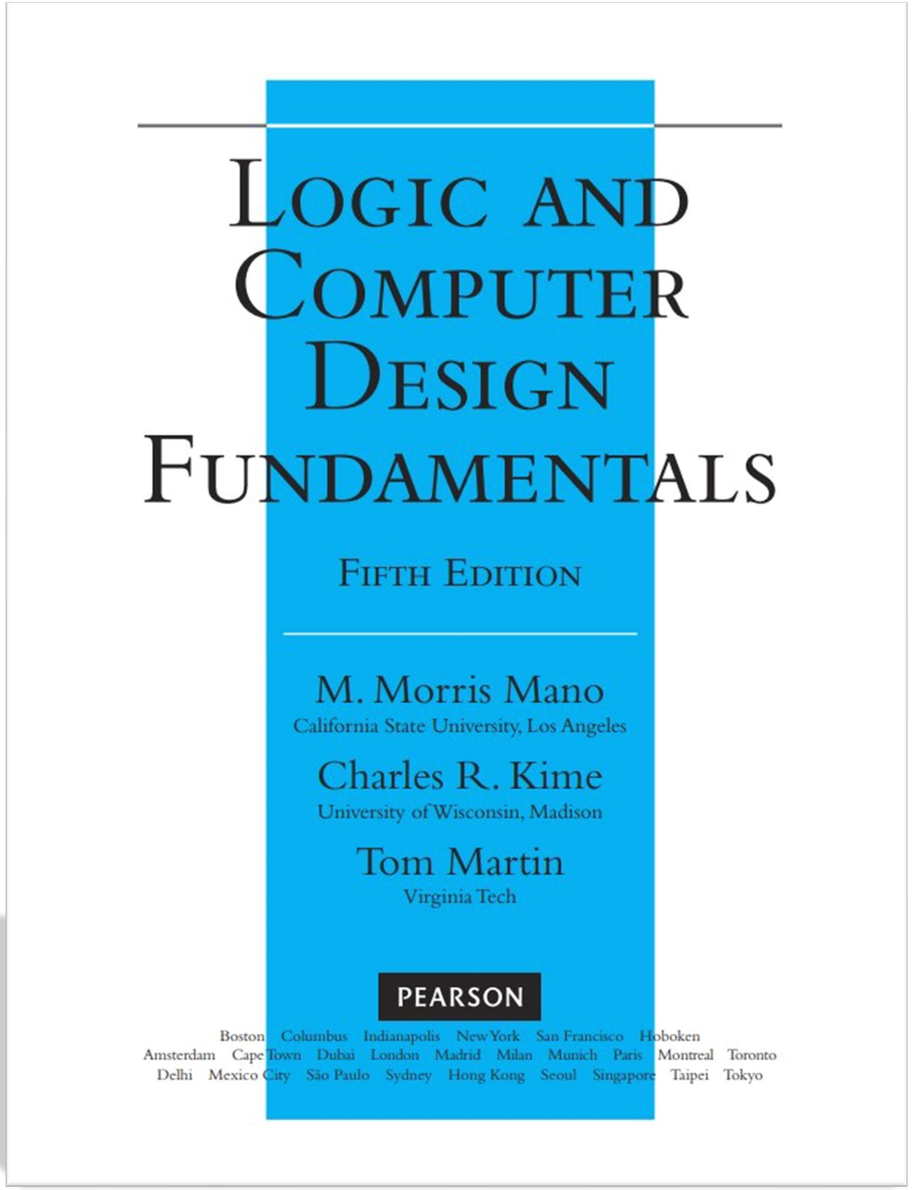
La CPU deve conoscere quale istruzione eseguire.

Il registro **PC** (program counter) mantiene l'indirizzo in memoria della **prossima** istruzione da eseguire

Architettura di von Neumann



- Dati e istruzioni condividono la stessa memoria
  - Concetto di **stored-program**, introdotto da von Neumann
  - Precedentemente il programma era salvato in hardware o su altri dispositivi
- Le locazioni di memoria possono contenere dati o istruzioni
- Un programma in memoria può essere modificato



Una possibile realizzazione

"morris"

The Arithmetic/Logic Unit (ALU)

- La ALU è una rete combinatoria in grado di realizzare micro-operazioni aritmetiche e logiche

input A e B (n bits)

Date k line di selezione
è possibile selezionare 2^k
operazioni

Selezione fra modalità
aritmetica o logica

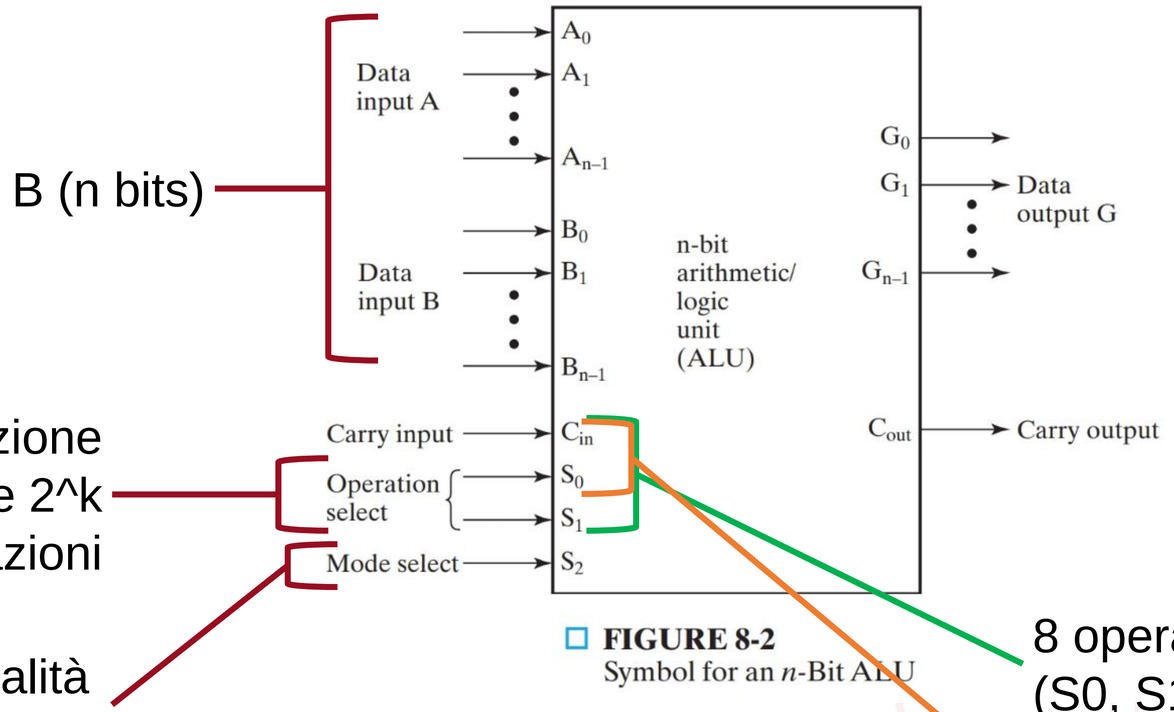


FIGURE 8-2
Symbol for an *n*-Bit ALU

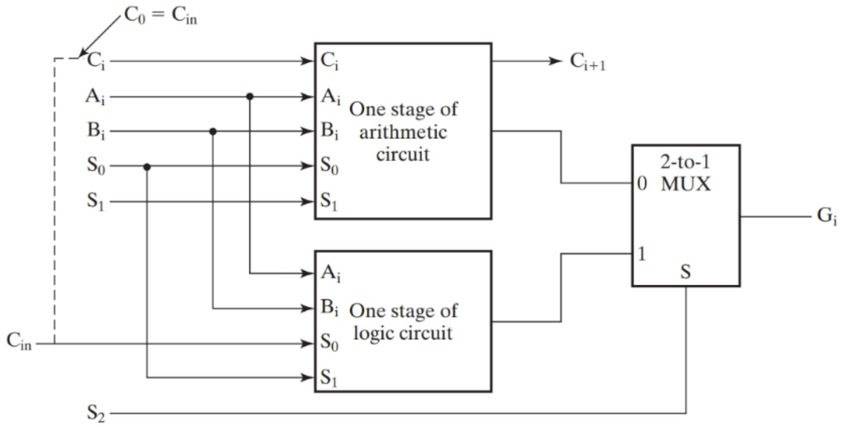


FIGURE 8-7
One Stage of ALU

8 operazioni aritmetiche
(S0, S1, Cin e S2=0)

4 operazioni logiche
(S0, Cin e S2=1)

ALU: Circuito aritmetico

- Addizionatore parallelo con full-adder in cascata
- Controllando gli input dell'addizionatore è possibile ottenere diverse operazioni aritmetiche

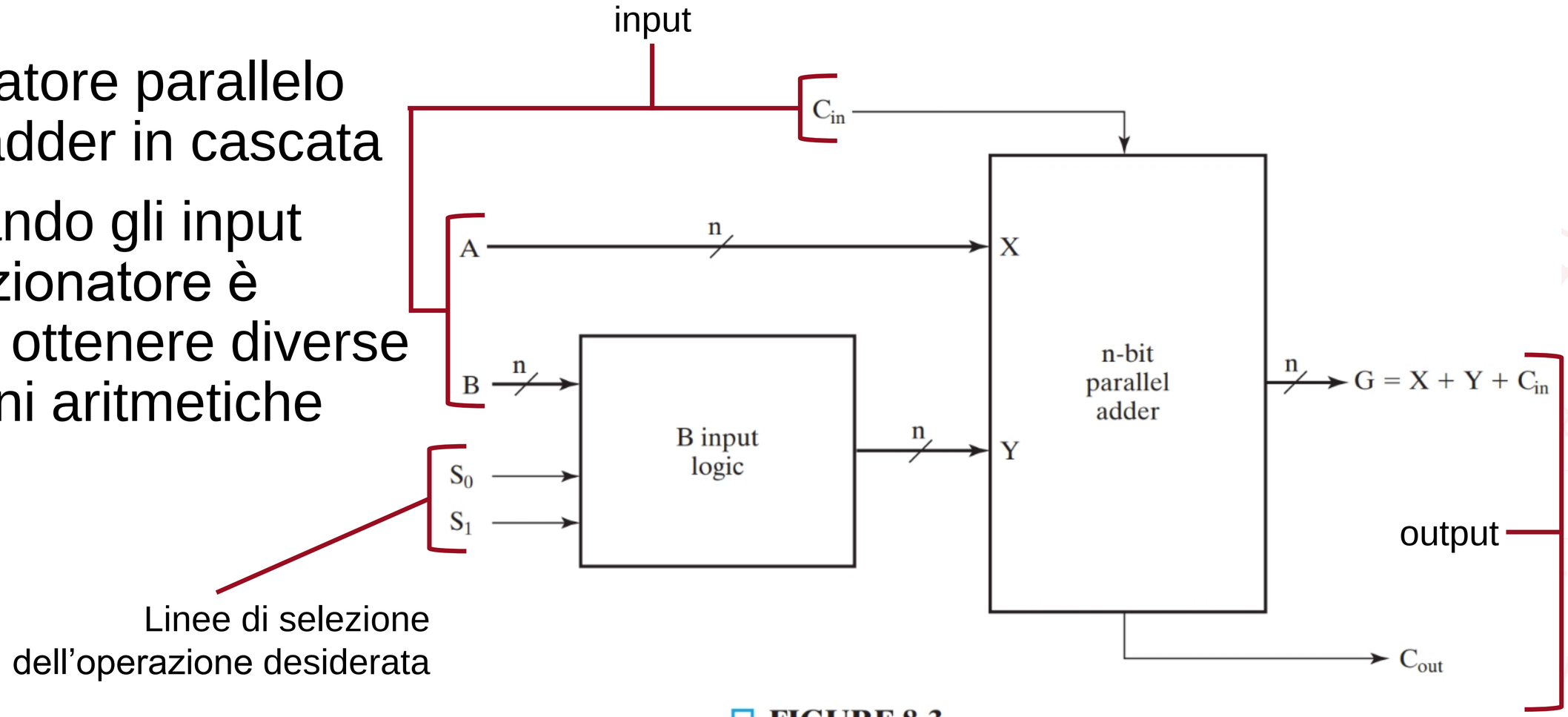


FIGURE 8-3
Block Diagram of an Arithmetic Circuit

ALU: Circuito aritmetico (2)

- Addizionatore parallelo con full-adder in cascata
- Controllando gli input dell'addizionatore è possibile ottenere diverse operazioni aritmetiche

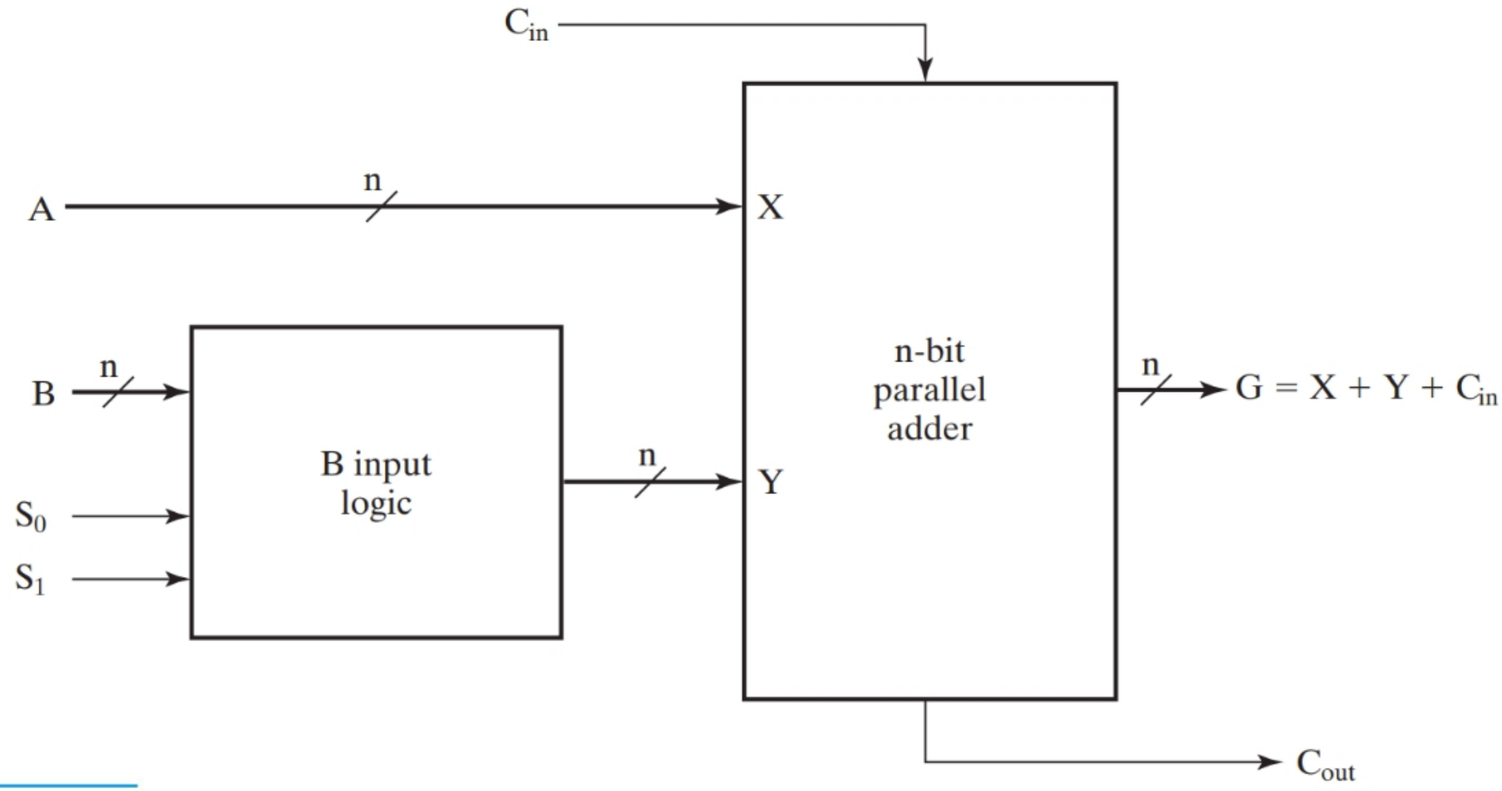


FIGURE 8-3
Block Diagram of an Arithmetic Circuit

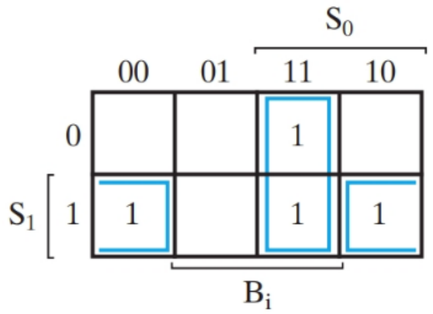
TABLE 8-1
Function Table for Arithmetic Circuit

Select		Input	$G = (A + Y + C_{in})$	
S_1	S_0	Y	$C_{in} = 0$	$C_{in} = 1$
0	0	all 0s	$G = A$ (transfer)	$G = A + 1$ (increment)
0	1	B	$G = A + B$ (add)	$G = A + B + 1$
1	0	$\overline{B}$	$G = A + \overline{B}$	$G = A + \overline{B} + 1$ (subtract)
1	1	all 1s	$G = A - 1$ (decrement)	$G = A$ (transfer)

ALU: Circuito aritmetico (3)

Inputs			Output	
S_1	S_0	B_i	Y_i	
0	0	0	0	$Y_i = 0$
0	0	1	0	
0	1	0	0	$Y_i = B_i$
0	1	1	1	
1	0	0	1	$Y_i = \overline{B}_i$
1	0	1	0	
1	1	0	1	$Y_i = 1$
1	1	1	1	

(a) Truth table



(b) Map simplification:
$Y_i = B_i S_0 + \overline{B}_i S_1$

FIGURE 8-4
B Input Logic for One Stage of Arithmetic Circuit

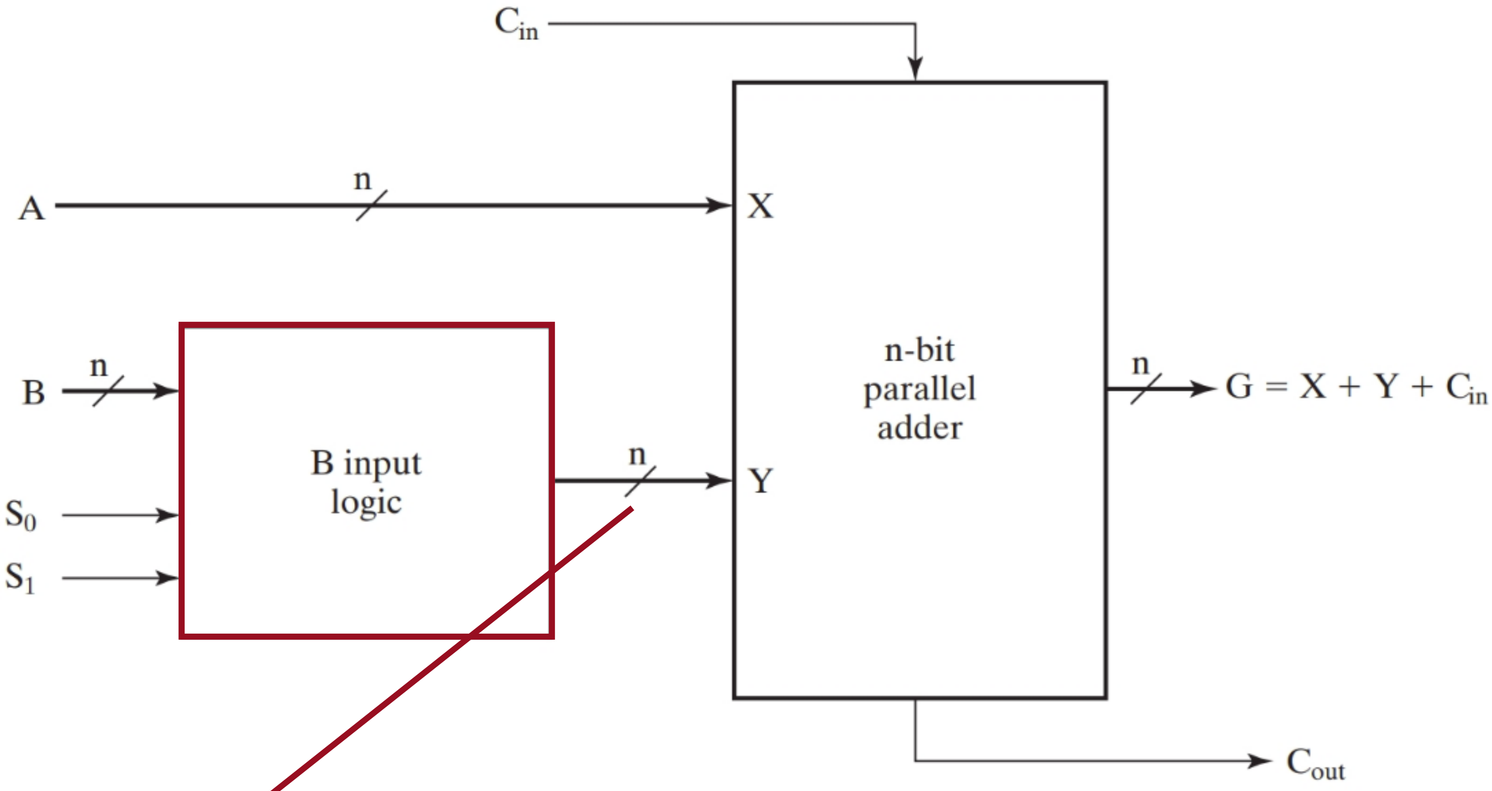


FIGURE 8-3
Block Diagram of an Arithmetic Circuit

$$Y_i = B_i S_0 + \overline{B_i} S_1$$

ALU: Circuito aritmetico (4)

Input Logic
(input: B, S0, S1
output: Y)

$$Y_i = B_i S_0 + \overline{B_i} S_1$$

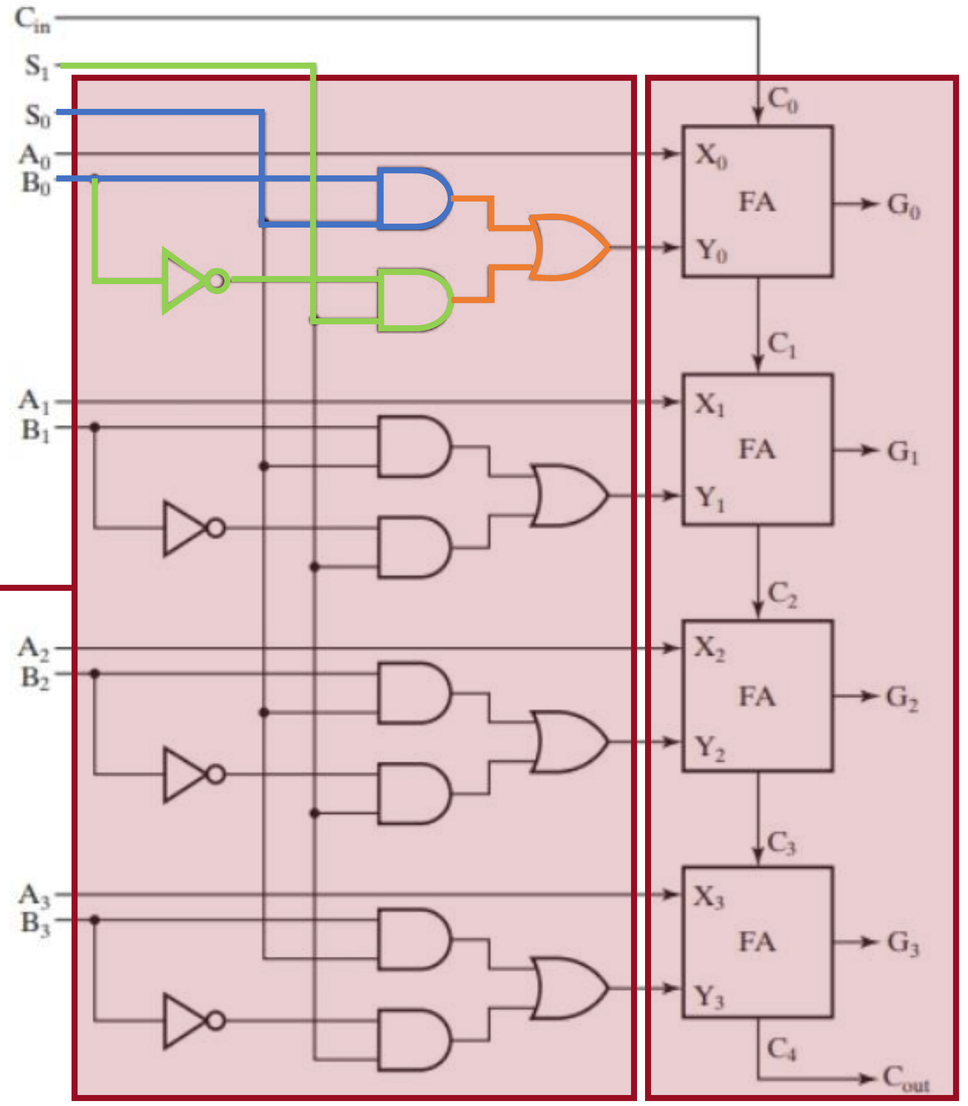
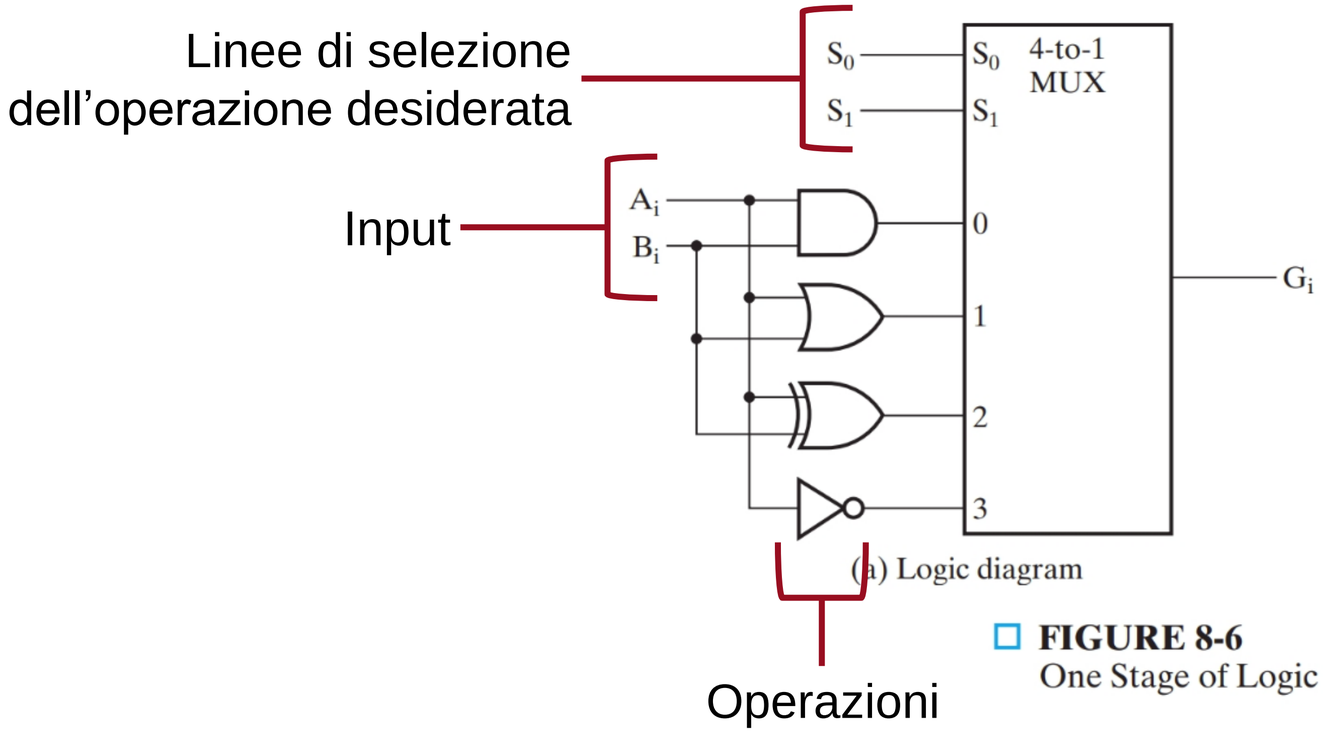


FIGURE 8-5
Logic Diagram of a 4-Bit Arithmetic Circuit

Parallel adder
(input: A, Cin, Y
output: G, Cout)

ALU: Circuito logico

- Manipola gli operandi permettendo operazioni bitwise
- Quattro operazioni comuni: AND, OR, XOR e NOT
- Altre operazioni più complesse possono essere implementate a partire da queste



(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$G = A \wedge B$	AND
0	1	$G = A \vee B$	OR
1	0	$G = A \oplus B$	XOR
1	1	$G = \overline{A}$	NOT

(b) Function table

FIGURE 8-6
One Stage of Logic Circuit

ALU: Arithmetic/Logic Unit

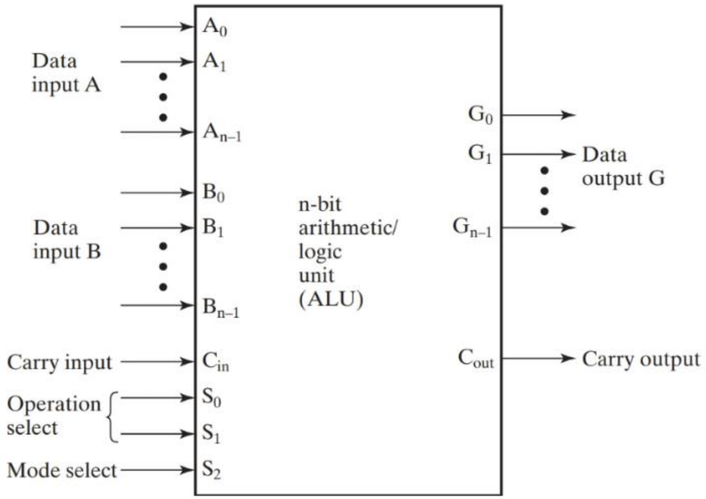


FIGURE 8-2
Symbol for an *n*-Bit ALU

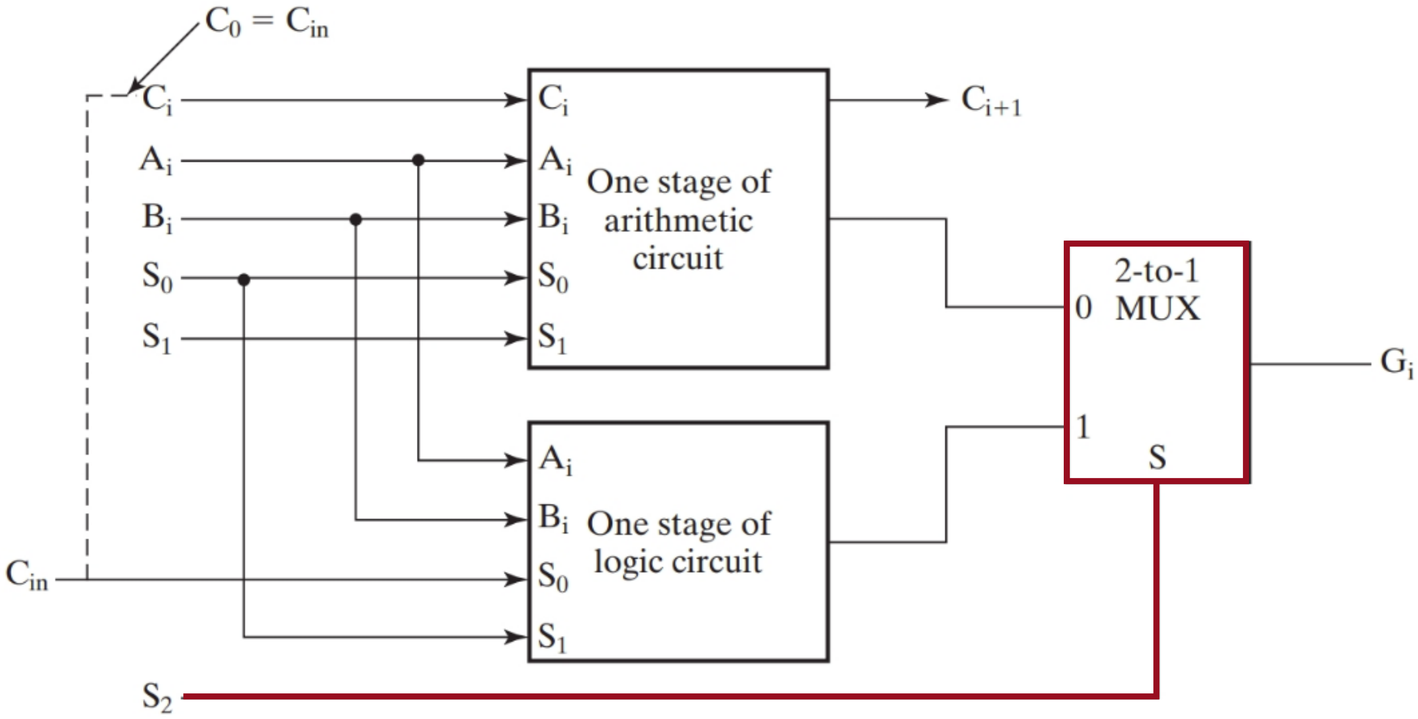


FIGURE 8-7
One Stage of ALU

Selezione circuito aritmetico o logico (S2)

ALU: Arithmetic/Logic Unit

TABLE 8-2
Function Table for ALU

Operation Select					
S_2	S_1	S_0	C_{in}	Operation	Function
0	0	0	0	$G = A$	Transfer A
0	0	0	1	$G = A + 1$	Increment A
0	0	1	0	$G = A + B$	Addition
0	0	1	1	$G = A + B + 1$	Add with carry input of 1
0	1	0	0	$G = A + \overline{B}$	A plus 1s complement of B
0	1	0	1	$G = A + \overline{B} + 1$	Subtraction
0	1	1	0	$G = A - 1$	Decrement A
0	1	1	1	$G = A$	Transfer A
1	X	0	0	$G = A \wedge B$	AND
1	X	0	1	$G = A \vee B$	OR
1	X	1	0	$G = A \oplus B$	XOR
1	X	1	1	$G = \overline{A}$	NOT (1s complement)

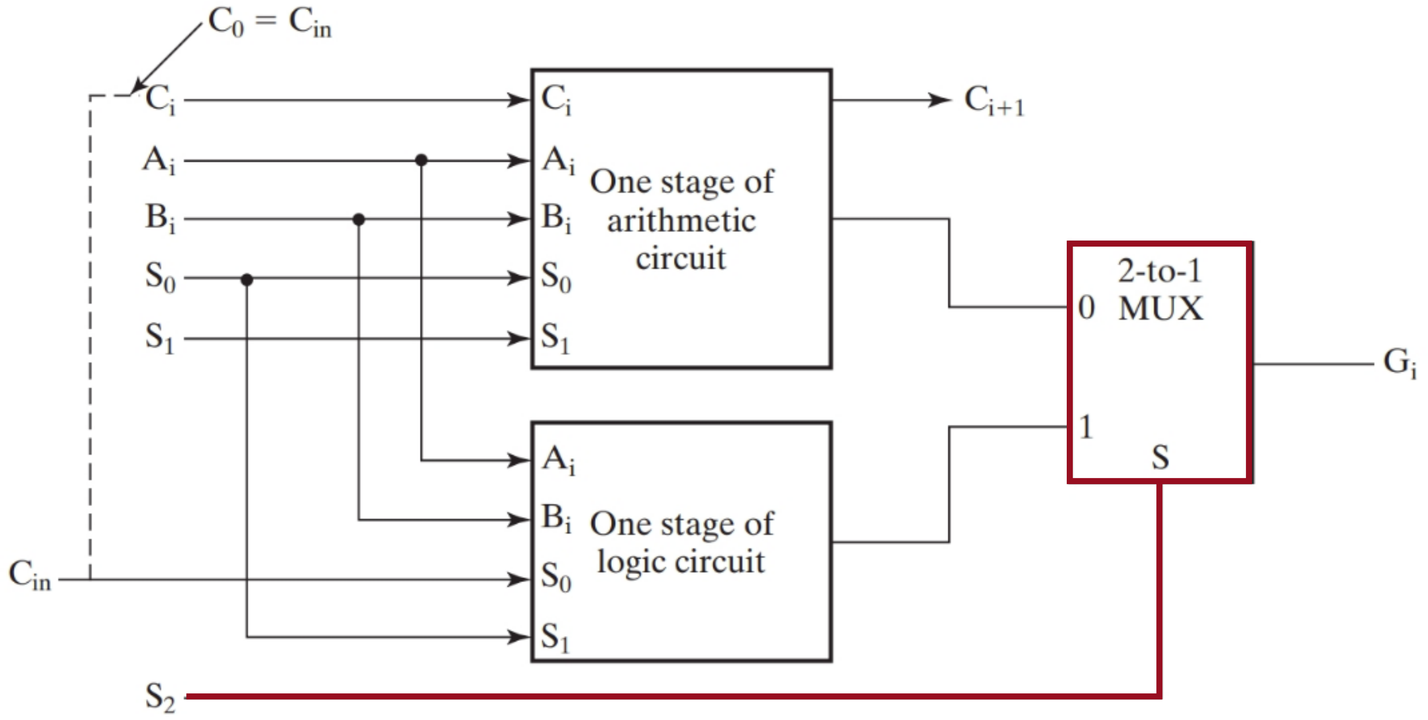


FIGURE 8-7
One Stage of ALU

Shifter

- È spesso utile poter shiftare i dati di n bits
- Per questo motivo, accanto all'ALU spesso viene progettato un Barrel Shifter, in grado di ruotare l'input a sinistra di un dato numero di bit in un singolo ciclo di clock

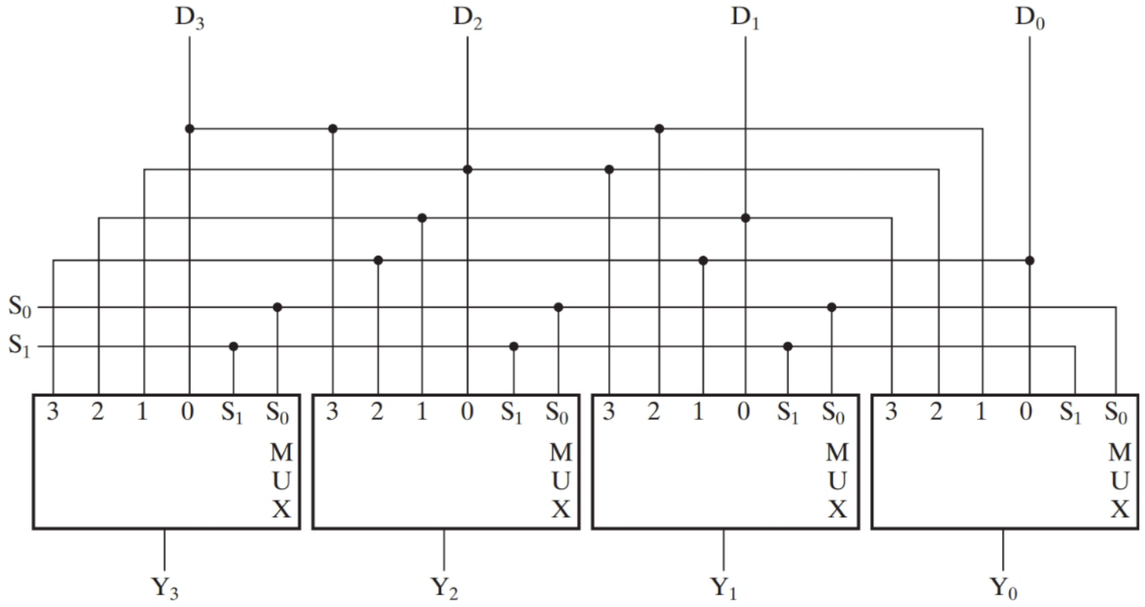
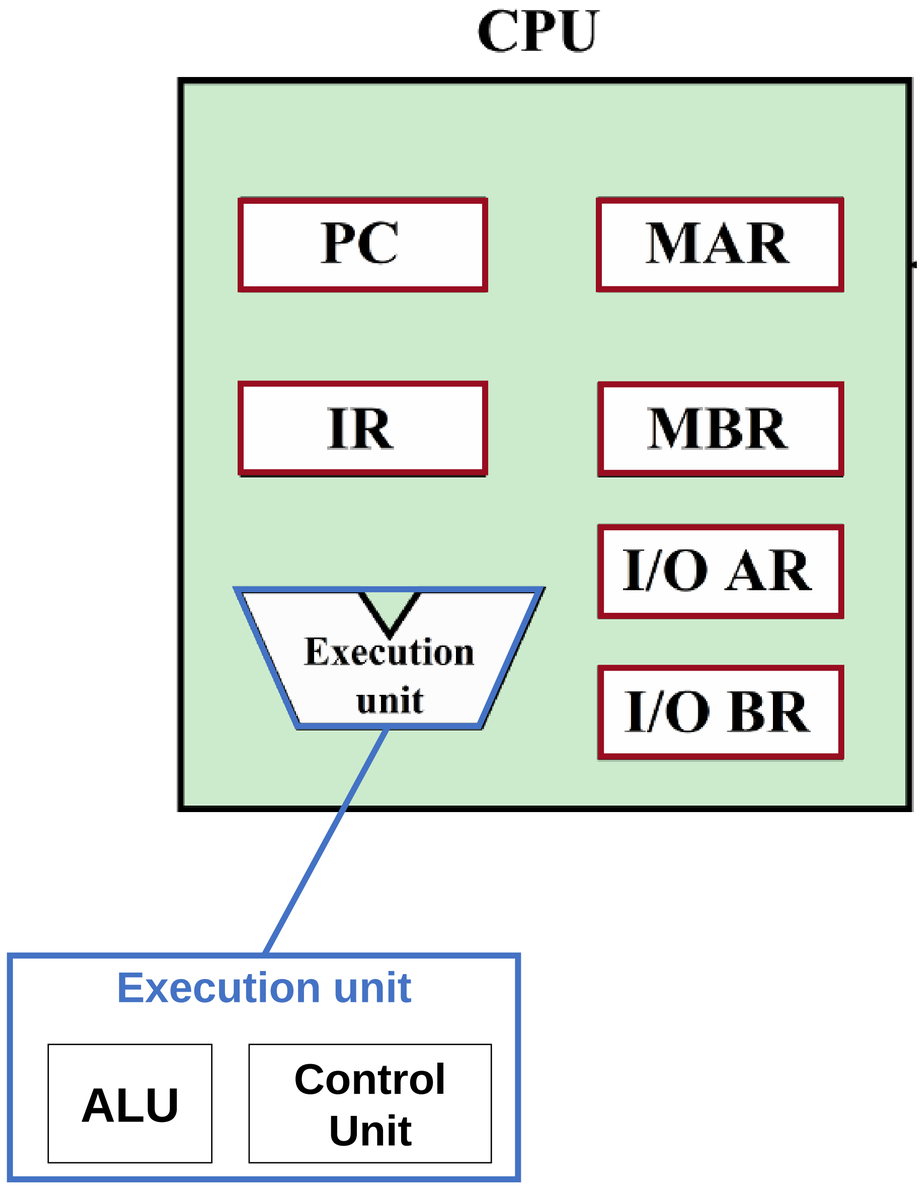


FIGURE 8-9
4-Bit Barrel Shifter

TABLE 8-3
Function Table for 4-Bit Barrel Shifter

Select		Output				
S_1	S_0	Y_3	Y_2	Y_1	Y_0	Operation
0	0	D_3	D_2	D_1	D_0	No rotation
0	1	D_2	D_1	D_0	D_3	Rotate one position
1	0	D_1	D_0	D_3	D_2	Rotate two positions
1	1	D_0	D_3	D_2	D_1	Rotate three positions

Registri



I registri sono elementi che permettono immagazzinamento di dati all'interno della CPU

L'accesso in lettura/scrittura ai registri è molto più veloce rispetto all'accesso su memoria

Esistono altri registri generici per l'immagazzinamento di dati (ad esempio: R0-R7)

Esempio di registri in un datapath generico

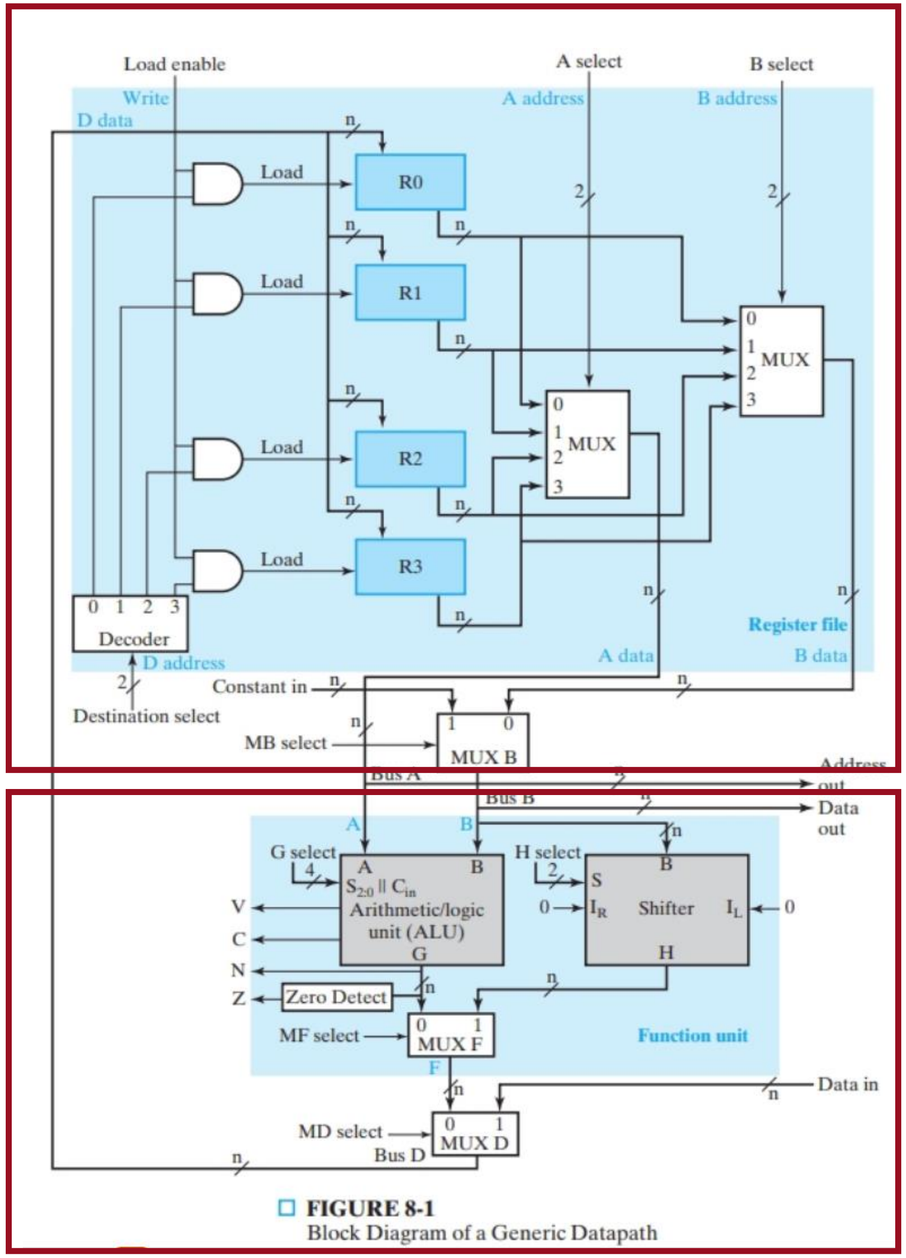


FIGURE 8-1
Block Diagram of a Generic Datapath

Register file

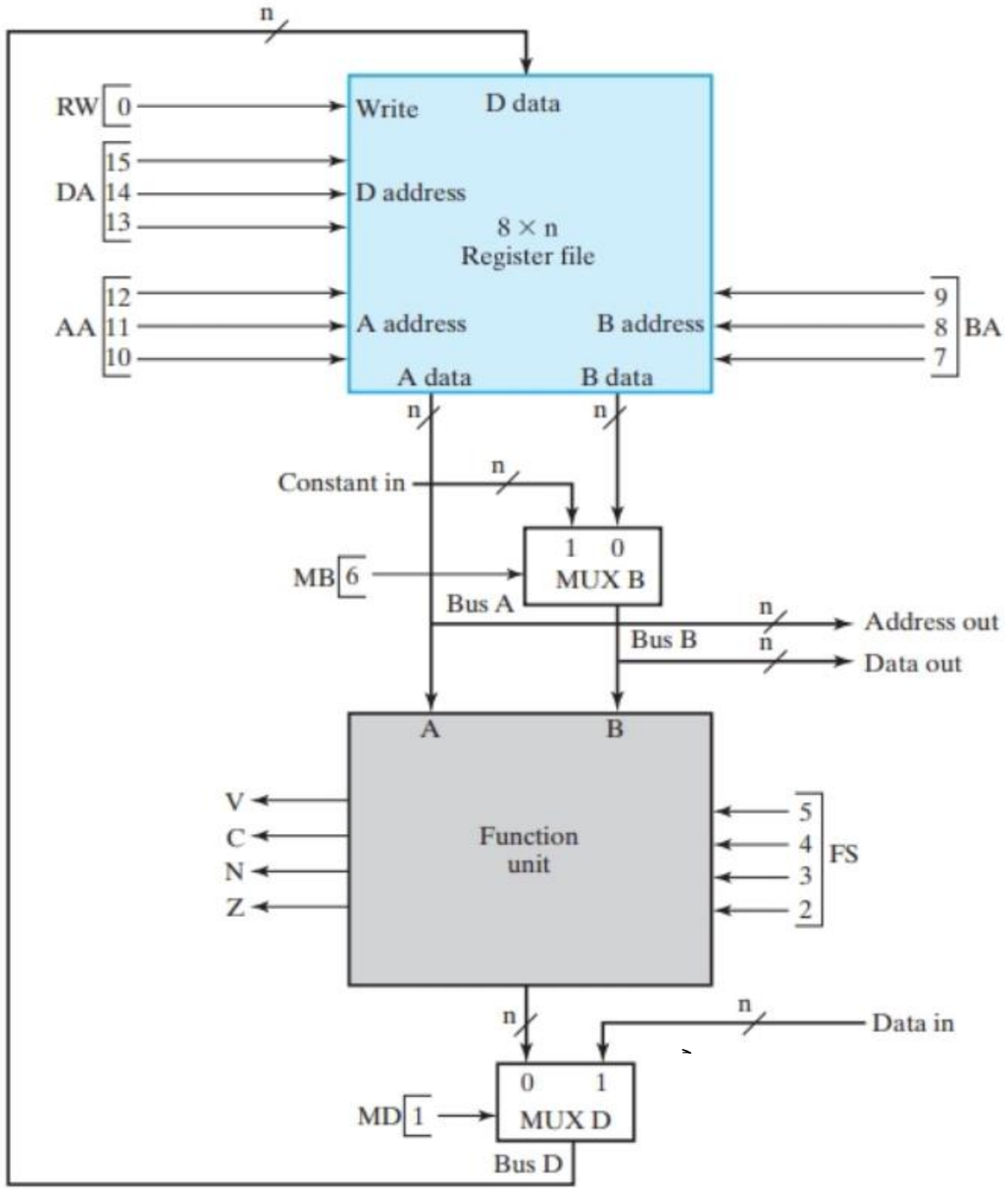
- 4 registri da n bits (R0-R3)
- 2 MUX che formano gli input A e B dell'ALU e dello shifter
- *A select* e *B select* selezionano gli input dai registri R0-R3

Function unit

Organizzazione e controllo del datapath

- Di solito esistono più di 4 registri in un datapath e vengono organizzati in un *register file*
- La dimensione del register file è $2^m \times n$, dove m è il numero di linee di selezione dei registri e n il numero di bits contenuti in ogni registro
- La selezione di input, operazioni, output è eseguita tramite una *control word* che le identifica in maniera univoca
- Le operazioni vengono eseguite in un ciclo di clock

….lo vedremo la prossima volta!



(a) Block diagram