

Programmazione e Architetture (Modulo B)

Lezione 5

Costruire un processore

Cosa abbiamo

E cosa ci manca

- Abbiamo circuiti logici in grado di
 - Effettuare operazioni (somma)
 - Selezionare uno tra molteplici input (multiplexer)
 - Dirigere un input in un dato output (demultiplexer) e decodificare un valore binario (decoder)
 - Salvare l'informazione (flip-flop)
- Possiamo ora combinarli per costruire una versione (molto semplificata) di un processore

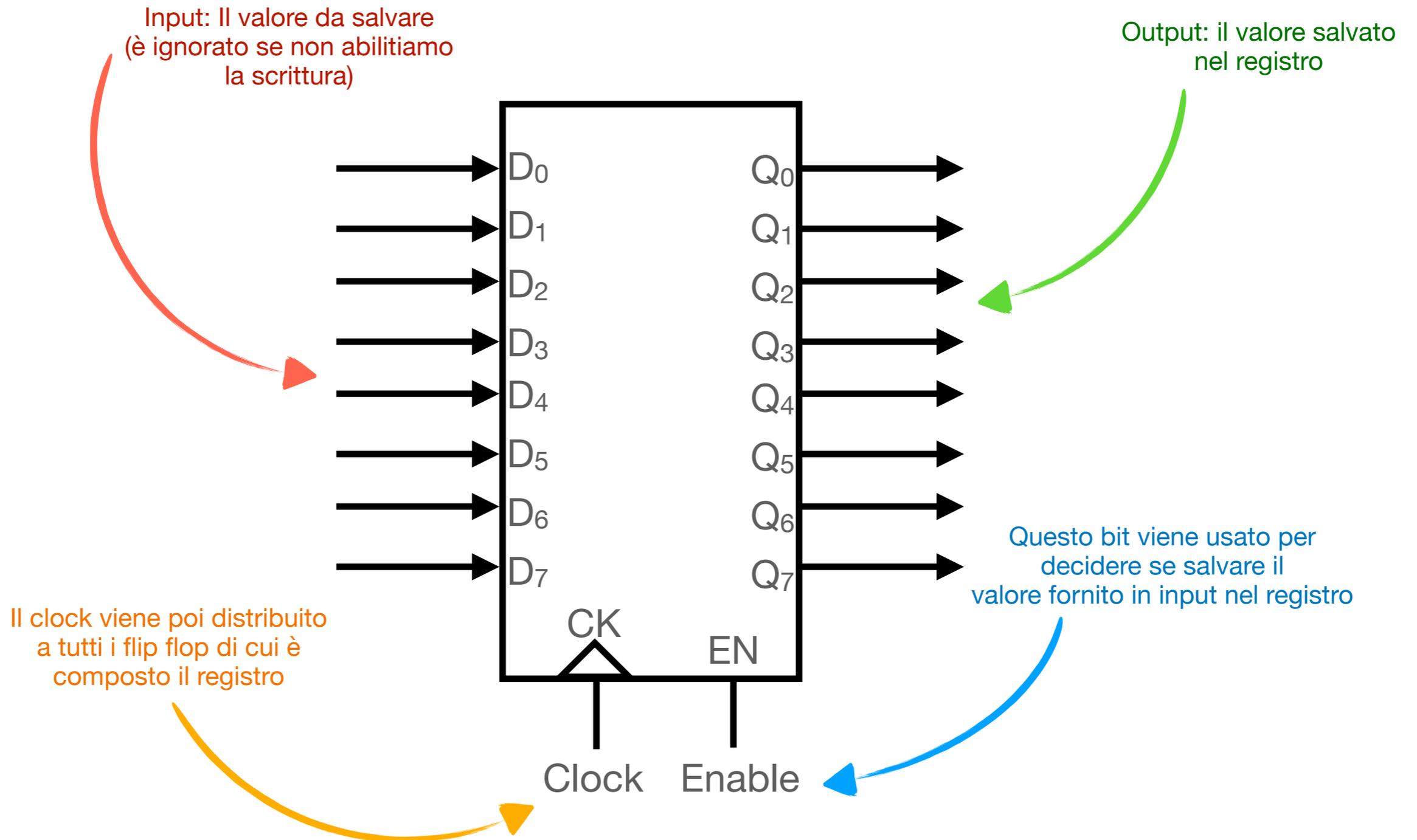
Registri

Salvare n-bit di informazione

- Possiamo combinare più flip flop per salvare non un solo bit ma un insieme di bit (e.g., 8, 16, 32, 64 bit)
- Generalmente avremo un solo input per il clock (poi diviso tra tutti i flip flop), un input per abilitare o disabilitare la scrittura del registro
- Ci servono poi tanti input e tanti output quanti sono i bit che vogliamo salvare
- Possiamo vedere il registro come un nuovo “blocco” da usare nella costruzione del nostro processore

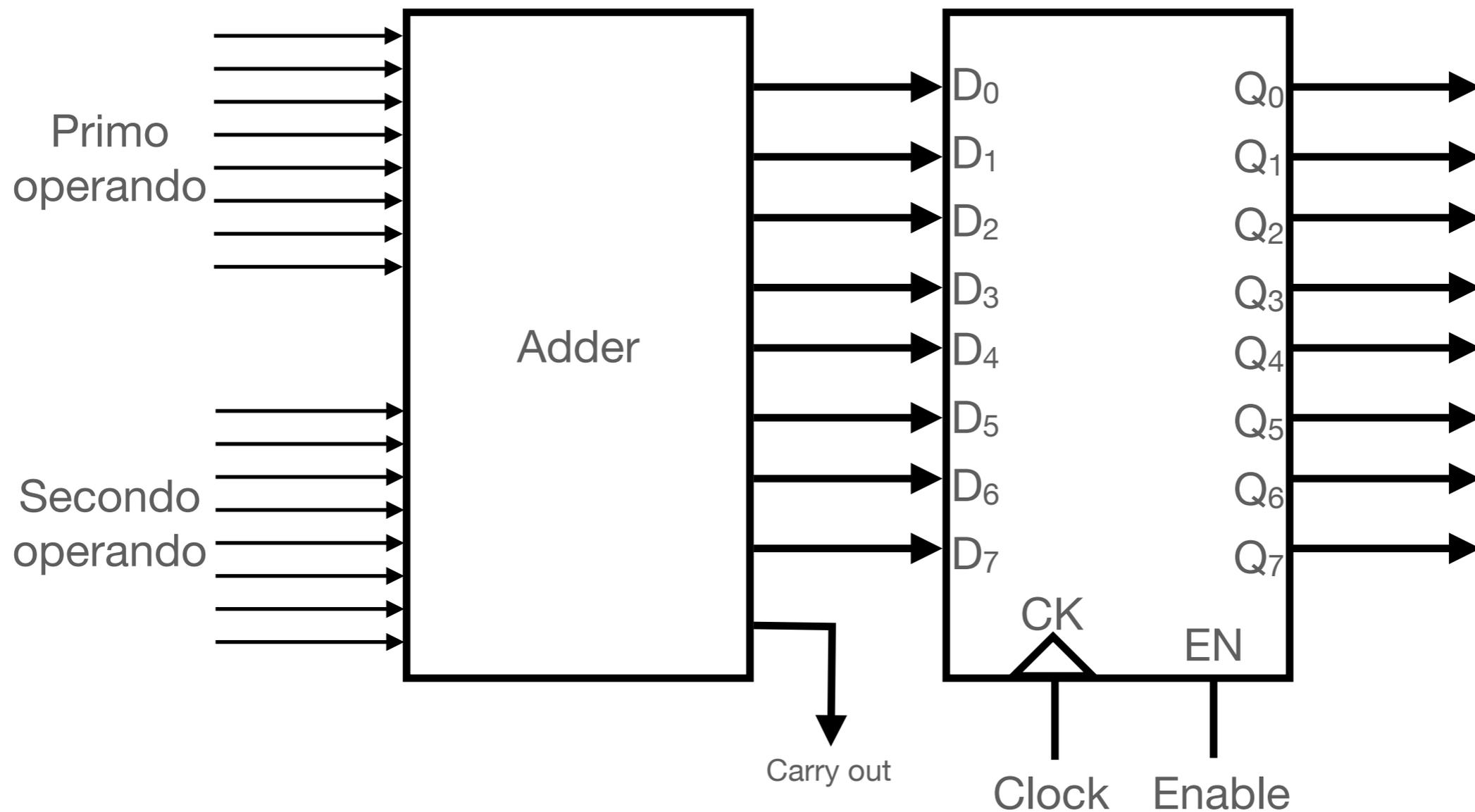
Un registro

Rappresentazione grafica



Registro: come possiamo usarlo?

Salvare il risultato della somma



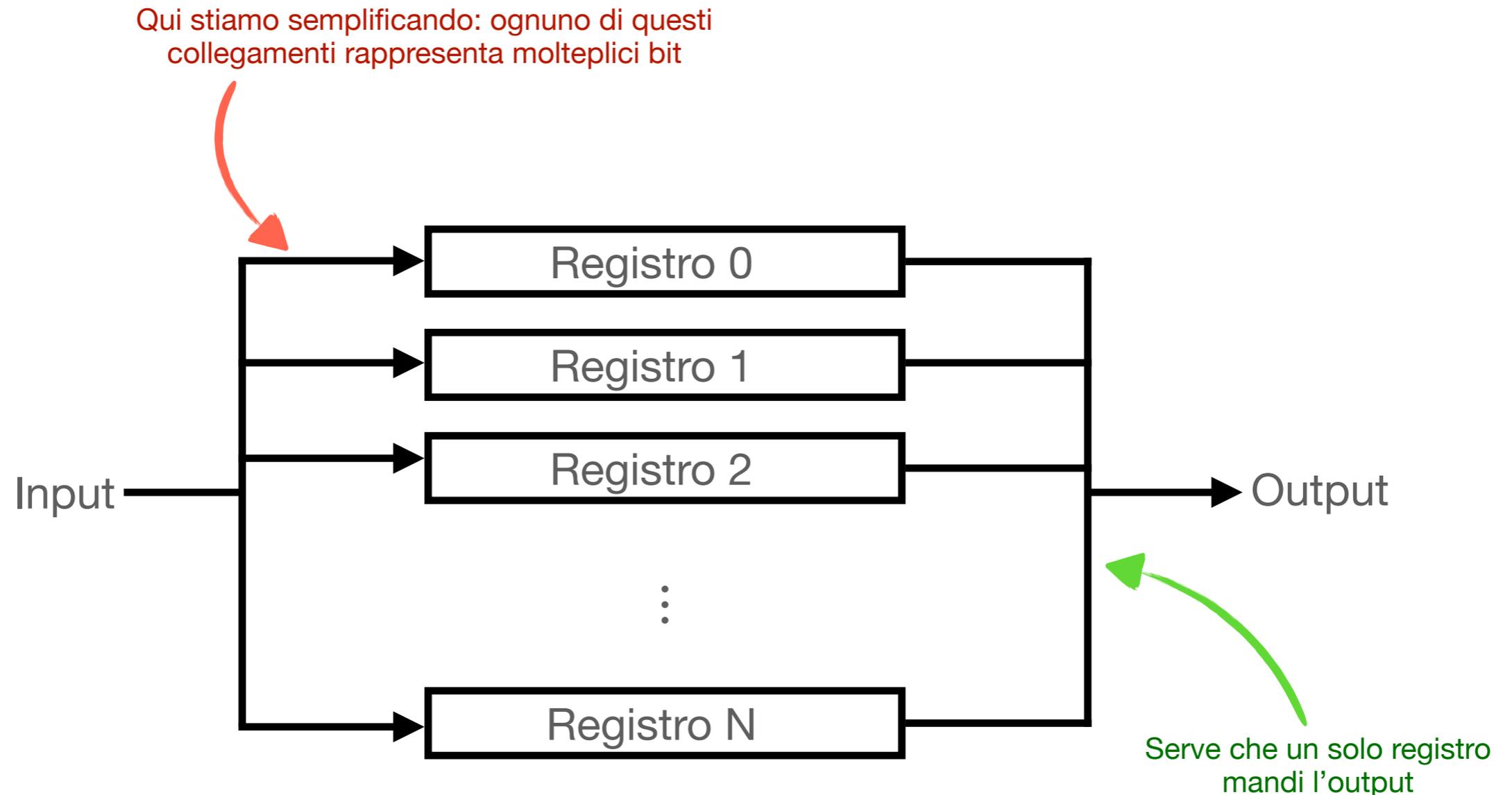
Registri multipli

E scegliere dove salvare

- Nello schema precedente salviamo sempre in un solo registro
- Potrebbe essere utile salvare il risultato in uno tra più registri
- Serve un modo per scegliere:
 - In che registro salvare
 - Da che registro leggere

Register file

Scegliere tra più registri

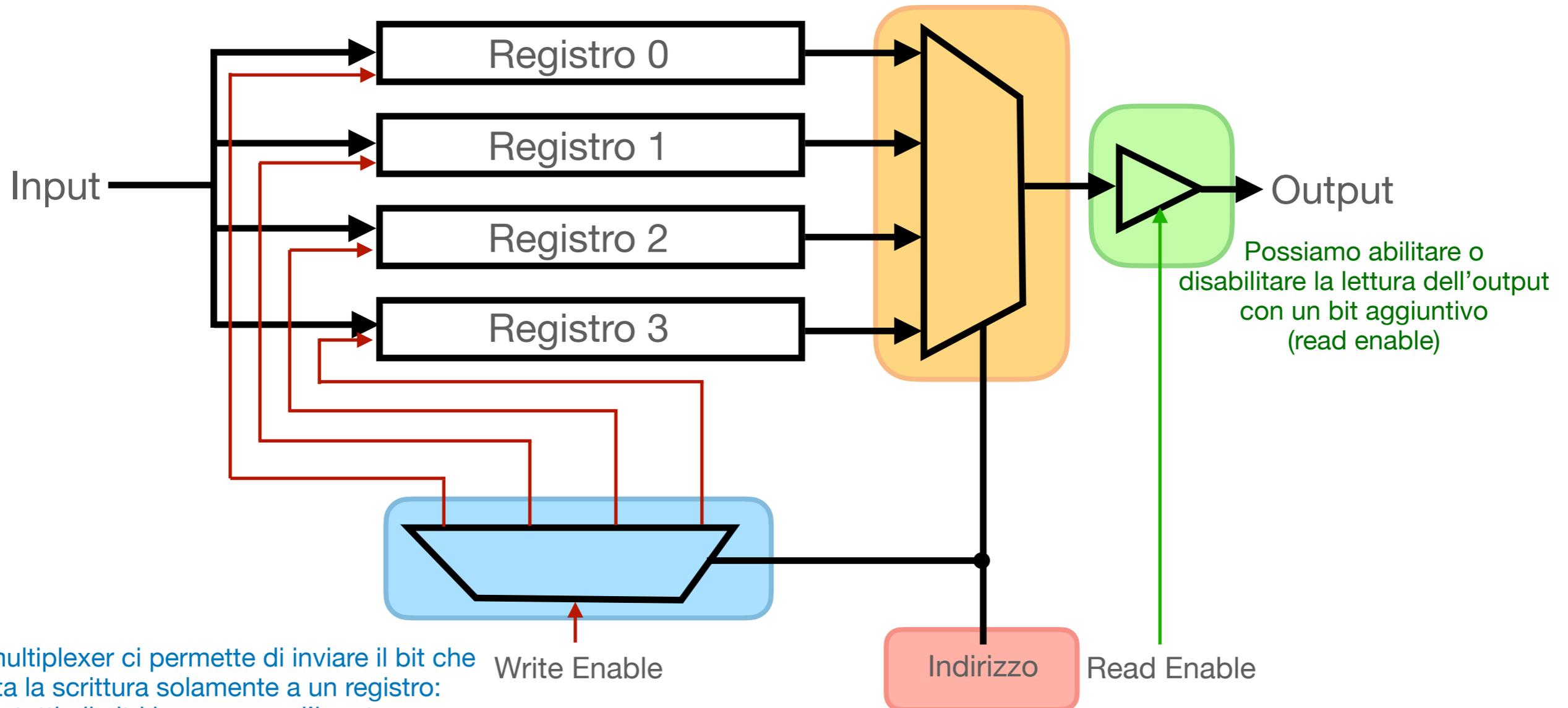


Idea: assegnamo un indirizzo (numero) a ciascun registro e quando vogliamo scrivere/leggere dobbiamo specificare anche in quale registro scrivere/leggere

Register file

Scegliere tra più registri

Un multiplexer ci permette di selezionare quale valore leggere tra quelli contenuti nei registri



Possiamo abilitare o disabilitare la lettura dell'output con un bit aggiuntivo (read enable)

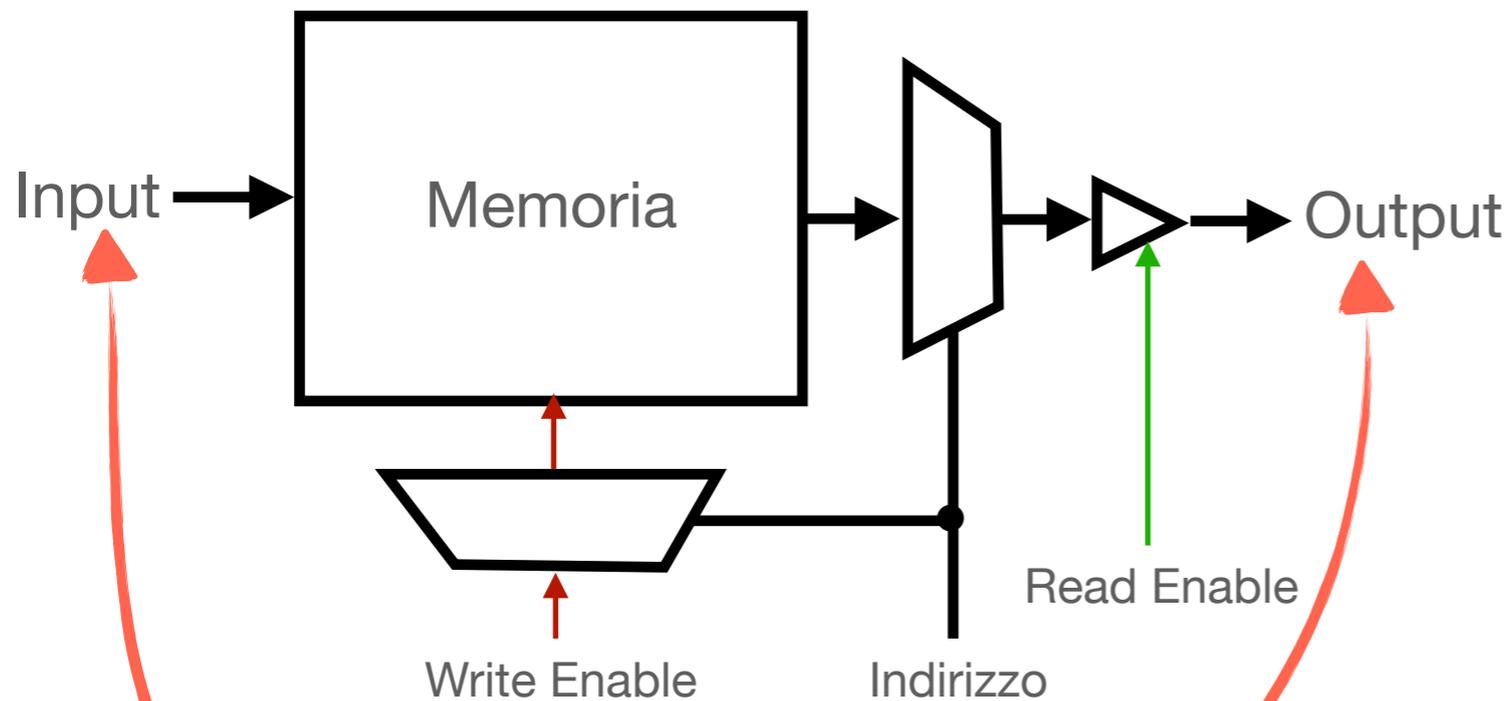
Un demultiplexer ci permette di inviare il bit che abilita la scrittura solamente a un registro: tutti gli altri ignoreranno l'input

La dimensione (numero di bit) dell'indirizzo dipenderà dal numero di registri che abbiamo a disposizione. In generale il logaritmo in base 2 del numero di registri

Breve digressione sulla memoria

Memoria: un solo input e output

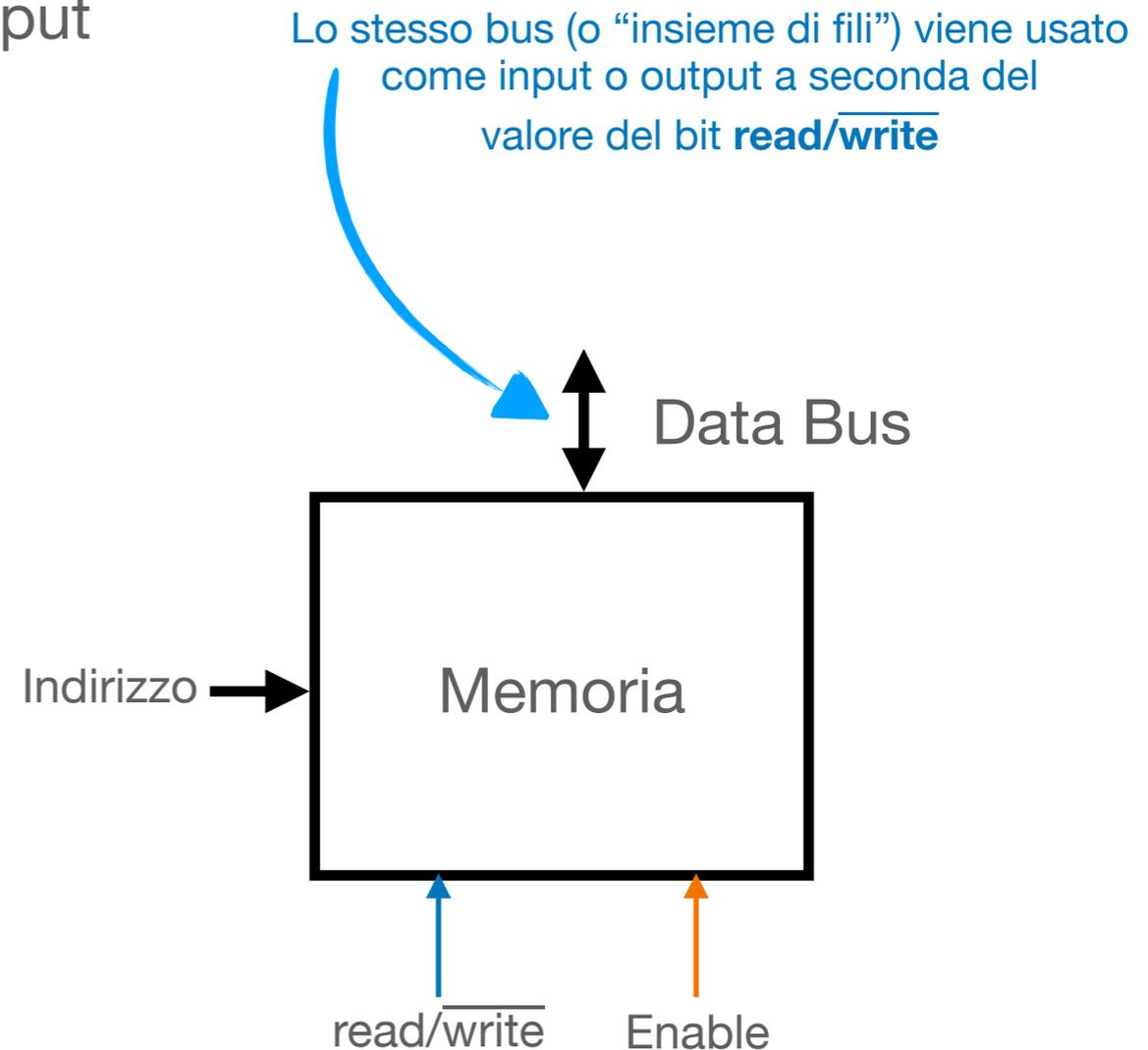
Data Bus



Nella nostra rappresentazione astratta non ci stiamo preoccupando di quanti "fili" servono...

Ma a chi costruisce il processore questo interessa!

Possiamo usare un solo set di "fili" alternativamente per l'input e per l'output



Random Access Memory

SRAM e DRAM

- La memoria visto fino a ora è ad accesso casuale, ovvero possiamo accedere alle locazioni di memoria in ogni ordine (ci basta fornire l'indirizzo)
- In termini di hardware (i.e., quando questa viene effettivamente realizzata) abbiamo due principali categorie:
 - Static RAM (SRAM). Estremamente veloce, ma richiede sei transistor per ogni bit (quindi più costosa e meno densa)
 - Dynamic RAM (DRAM). Più lenta, utilizza un transistor e un condensatore per ogni bit. Permette una densità maggiore e richiede un circuito di refresh per mantenere memorizzata l'informazione

Unità aritmetico-logica

Una ALU

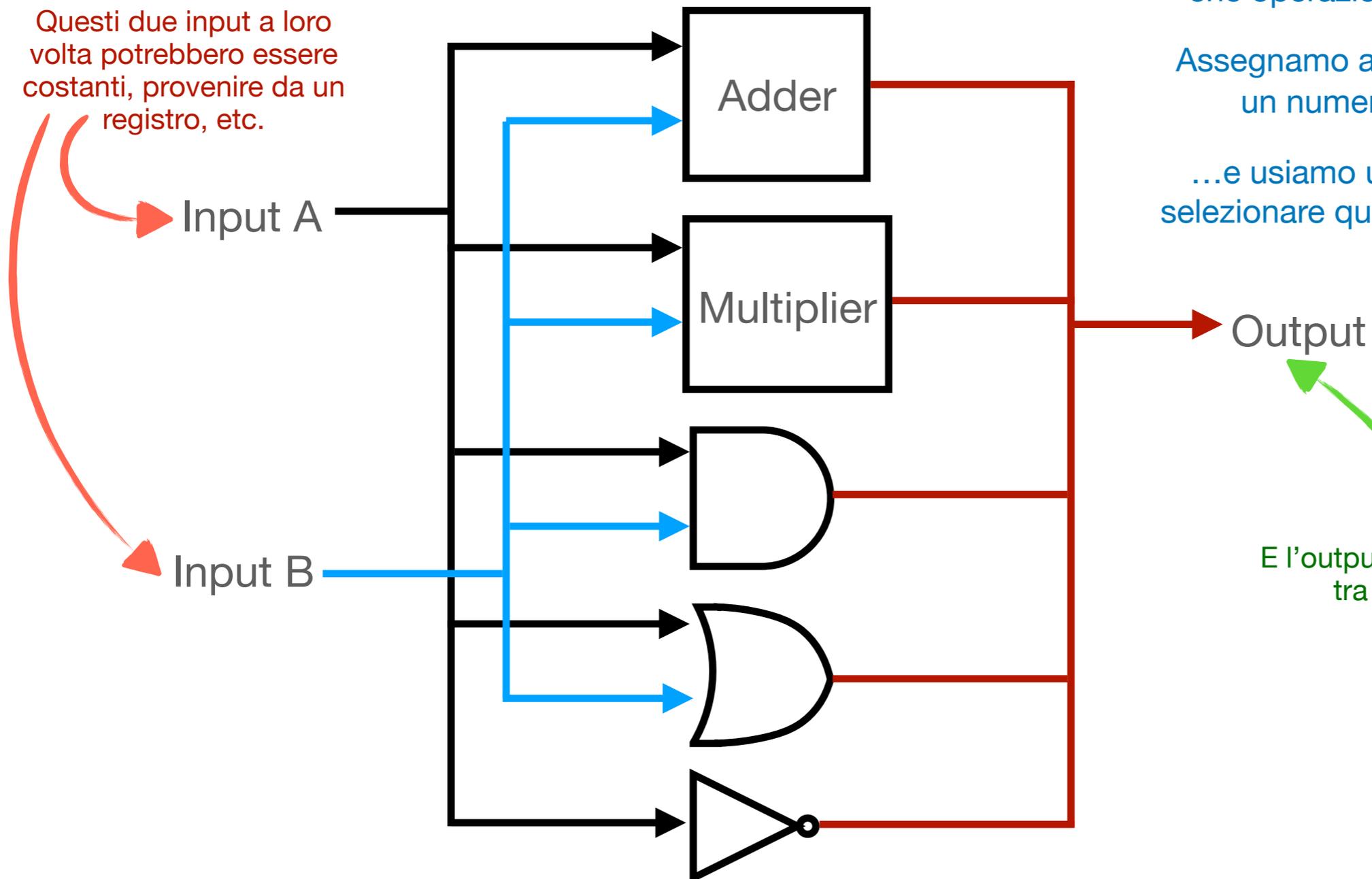
Non solo Adder

- Abbiamo visto come costruire un adder
- Possiamo costruire circuiti per molte altre operazioni
 - Semplici operazioni: negare tutti i bit in input, and/or bit a bit di due input, etc.
 - Operazioni più complesse: moltiplicazione, divisione, resto, etc.
- Dobbiamo però poter scegliere quale operazione svolgere
- Hint: abbiamo risolto un problema simile prima

Costruire una ALU

Selezionare tra più operazioni

Questi due input a loro volta potrebbero essere costanti, provenire da un registro, etc.



Come possiamo fare a selezionare che operazione far effettuare?

Assegnamo ad ogni operazione un numero (**opcode**)...

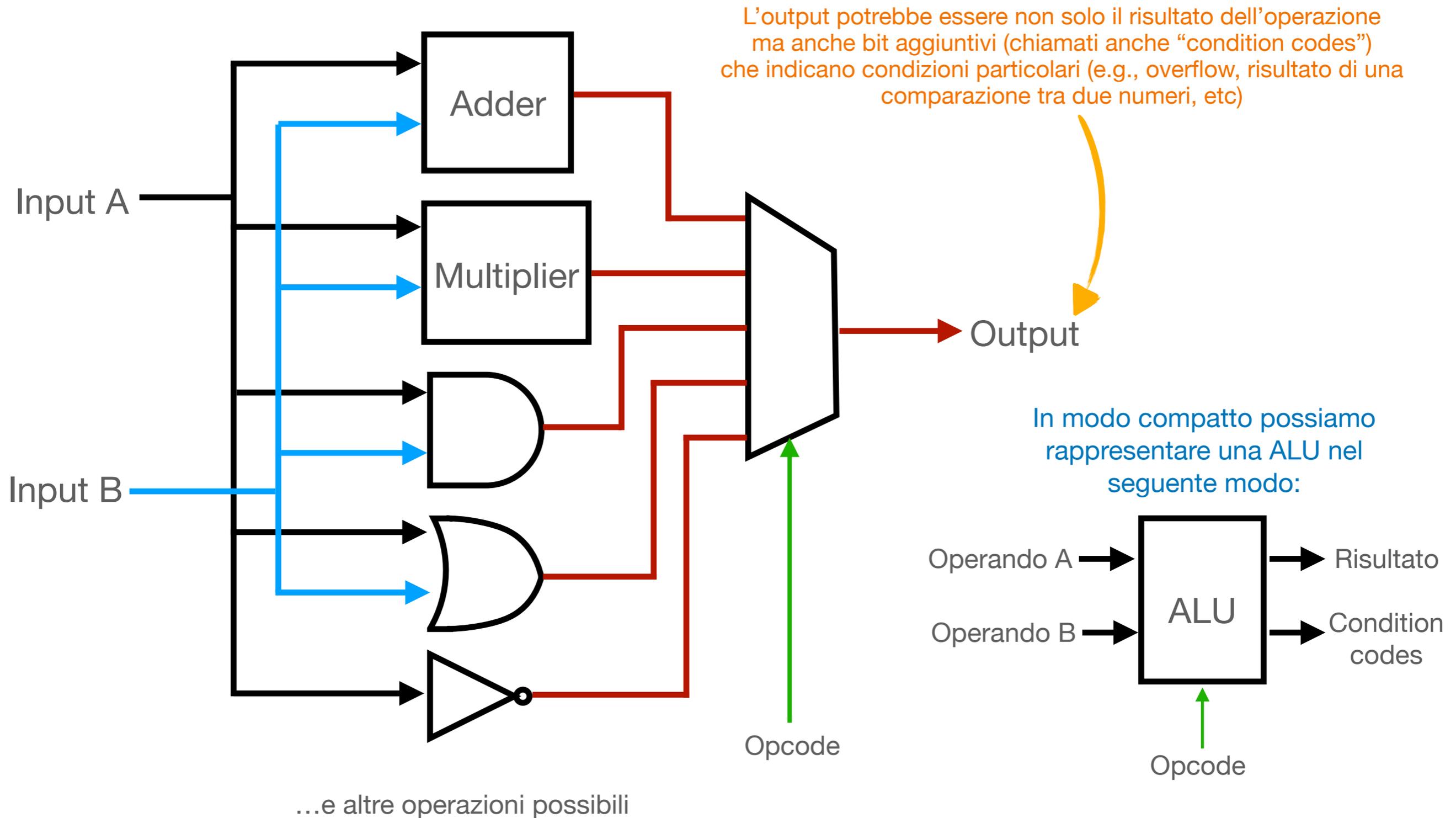
...e usiamo un multiplexer per selezionare quale output prendere

...e altre operazioni possibili

E l'output potrebbe finire in uno tra molteplici registri

Costruire una ALU

Selezionare tra più operazioni

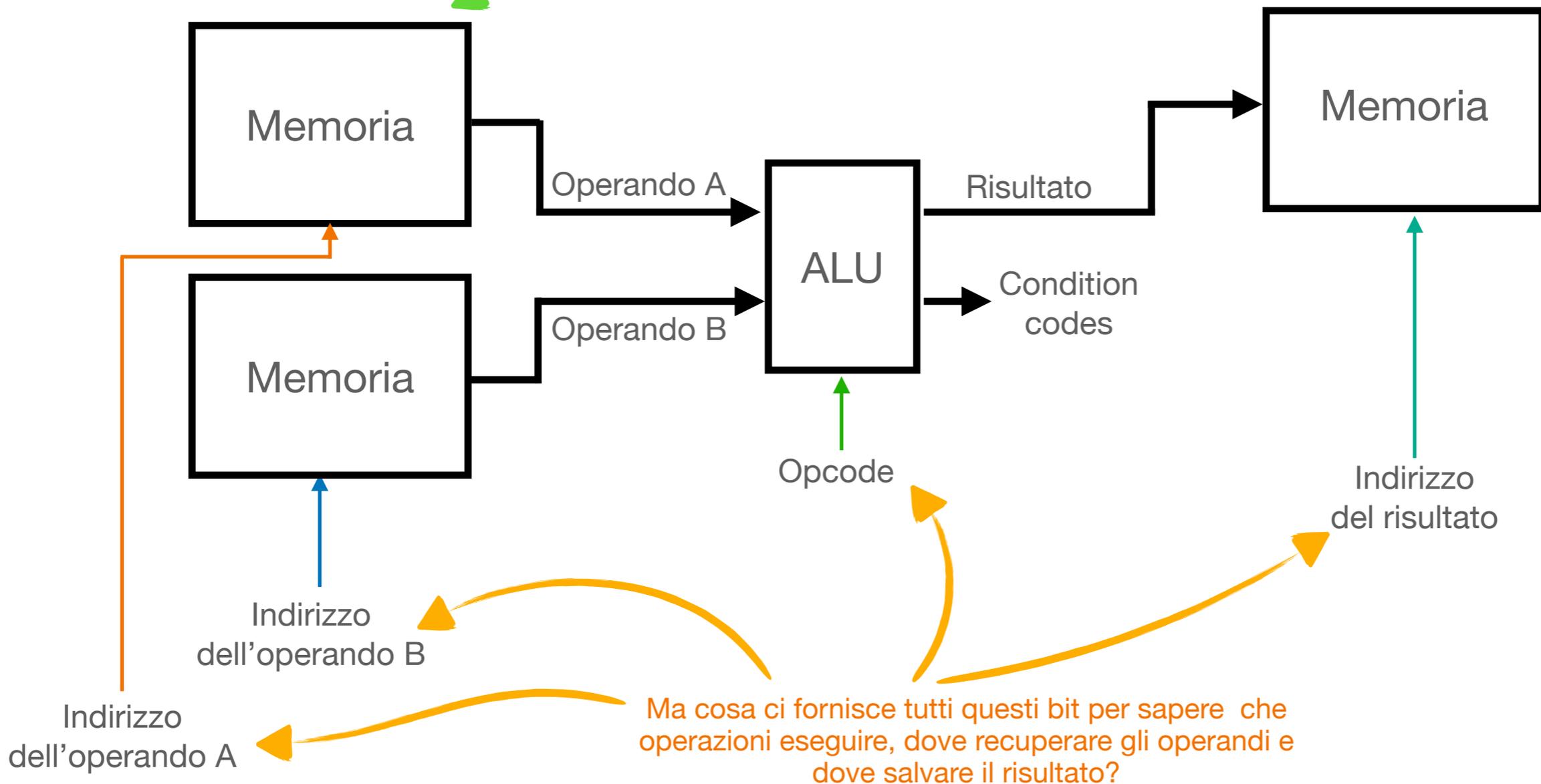


Istruzioni e Program counter

Proviamo a mettere assieme

Memoria e ALU

Nota: questa memoria è una unica solo rappresentata graficamente in tre parti



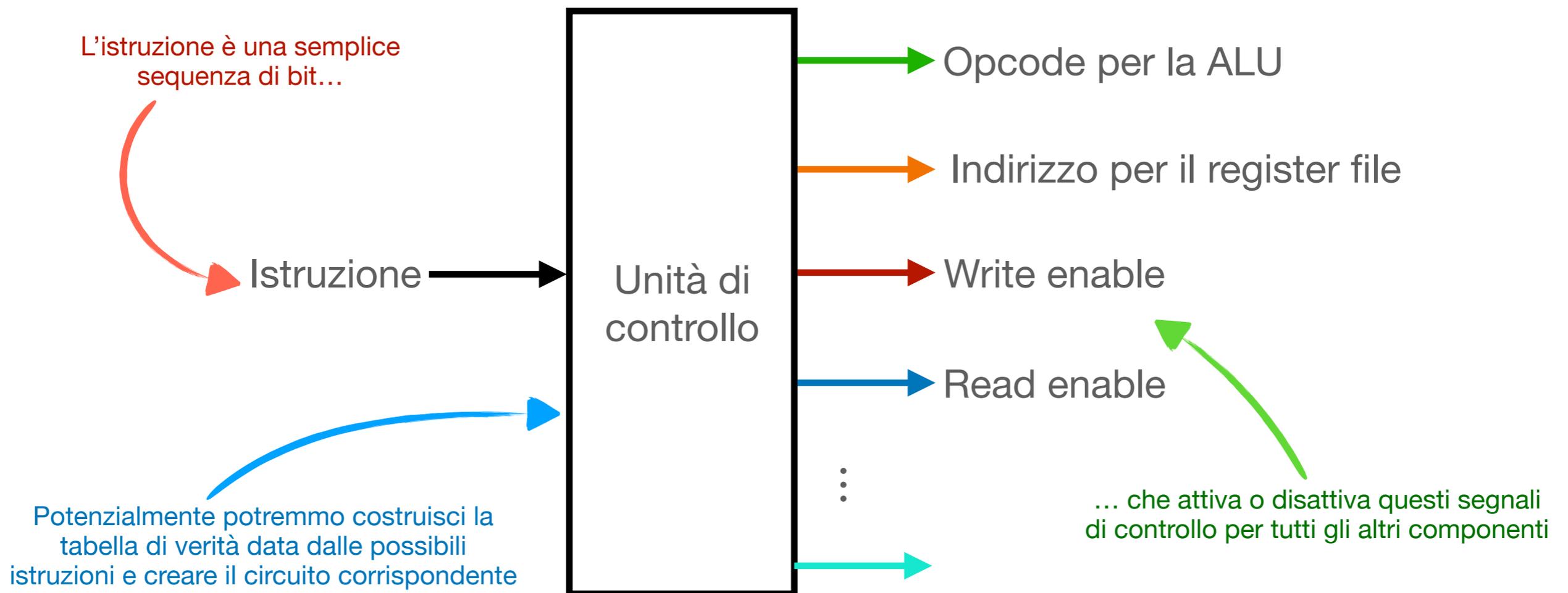
Istruzioni

E dare comandi al processore

- Abbiamo visto che più componenti hanno degli input aggiuntivi che dicono cosa fare:
 - Nel register file in che registro scrivere o leggere
 - Nella ALU che operazioni eseguire
- Questi sono input essenziali perché dicono ai vari componenti cosa fare
- Come vengono impostati questi input aggiuntivi?
- Una unità di controllo imposta questi bit a partire da una istruzione (a sua volta una sequenza di bit)

Unità di controllo

Versione semplificata



Cosa è il program counter

E a cosa serve

- Il program counter mantiene **l'indirizzo** in memoria dell'istruzione da eseguire
- Esso è quindi un registro con due funzionalità particolari:
 - Se non viene modificato passiamo all'istruzione successiva nella sequenza (i.e., il valore del program counter viene incrementato)
 - Se viene modificato cambierà la prossima istruzione che verrà eseguita. Questa operazione permette di implementare costrutti di selezione (come *if*) e costrutti iterativi (come *for* e *while*)

Un semplice processore

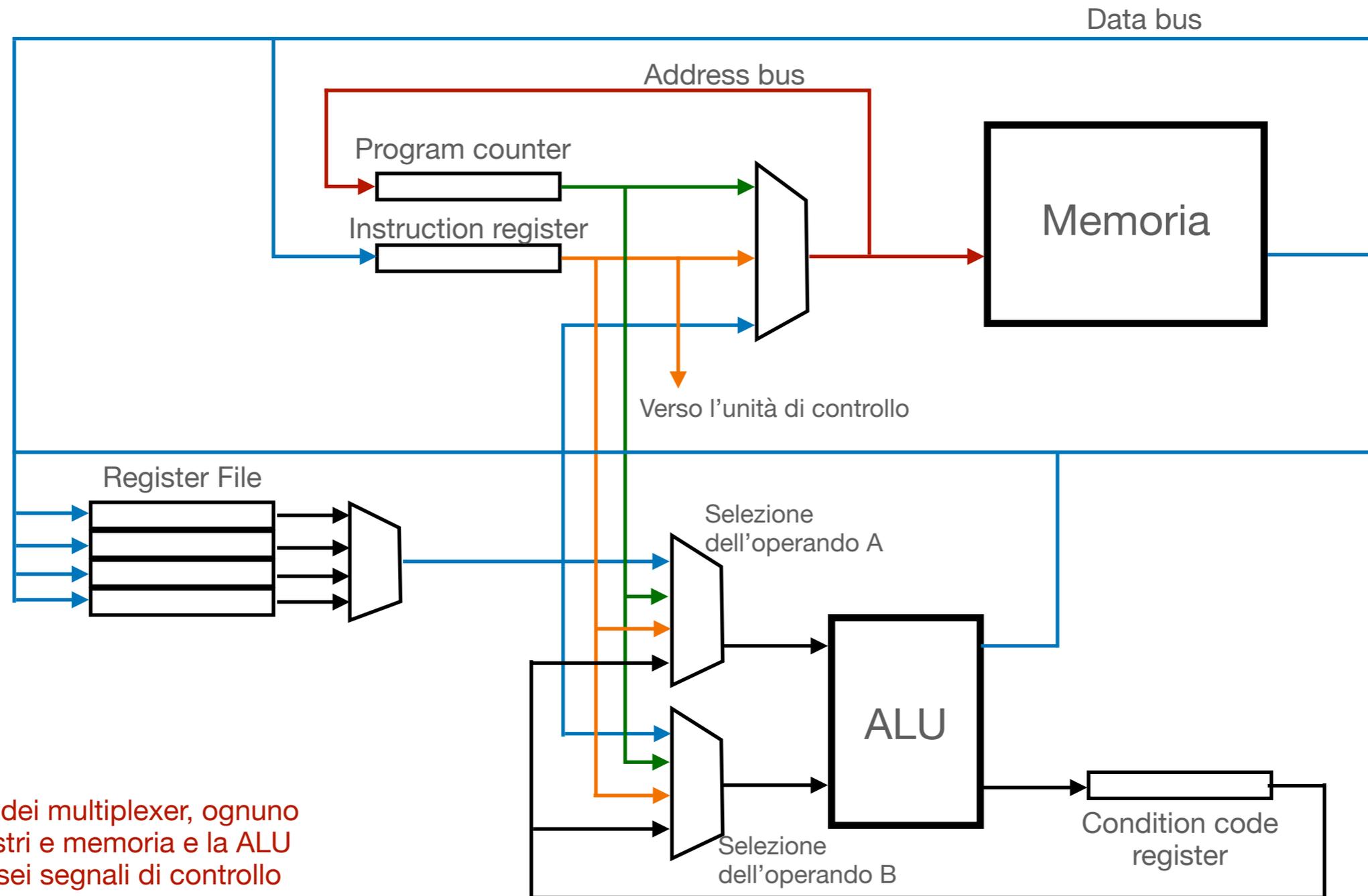
Un semplice processore

Ingredienti

- Servirà una ALU
 - Dobbiamo scegliere per ogni operando da dove deve essere preso l'input
- Un register file, con la possibilità di leggere e scrivere su ogni registro
- Una memoria (da cui dovremo reperire sia dati che istruzioni)
- Un program counter
- Un instruction register per salvarsi l'istruzioni da eseguire dopo averla letta dalla memoria

Struttura di un processore

Versione semplificata



Ognuno dei multiplexer, ognuno dei registri e memoria e la ALU hanno sei segnali di controllo

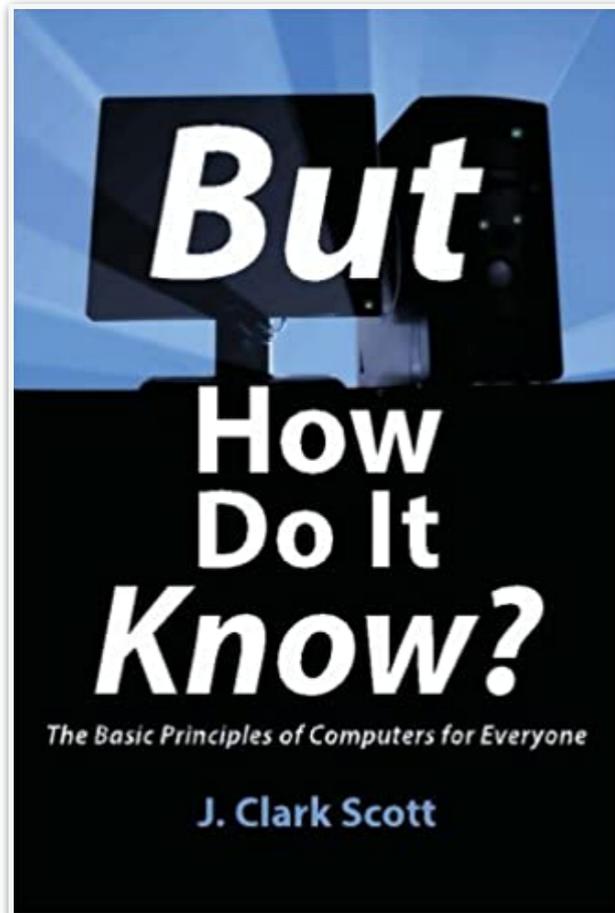
Quanto stiamo semplificando?

Hint: moltissimo

- Nelle architetture moderne i registri (decine) sono di 64 bit...
- ...ma c'è hardware speciale per eseguire operazioni su gruppi di bit più grandi, quindi anche registri appositi per operazioni su 256 o 512 bit alla volta
- Abbiamo ignorato tutta la gestione dei numeri floating point
- Funzionalità aggiuntive possono essere integrate nella CPU (e.g., per la crittografia)
- Possono esserci più unità (adder, multiplier, etc) per eseguire più operazioni in contemporanea
- ...

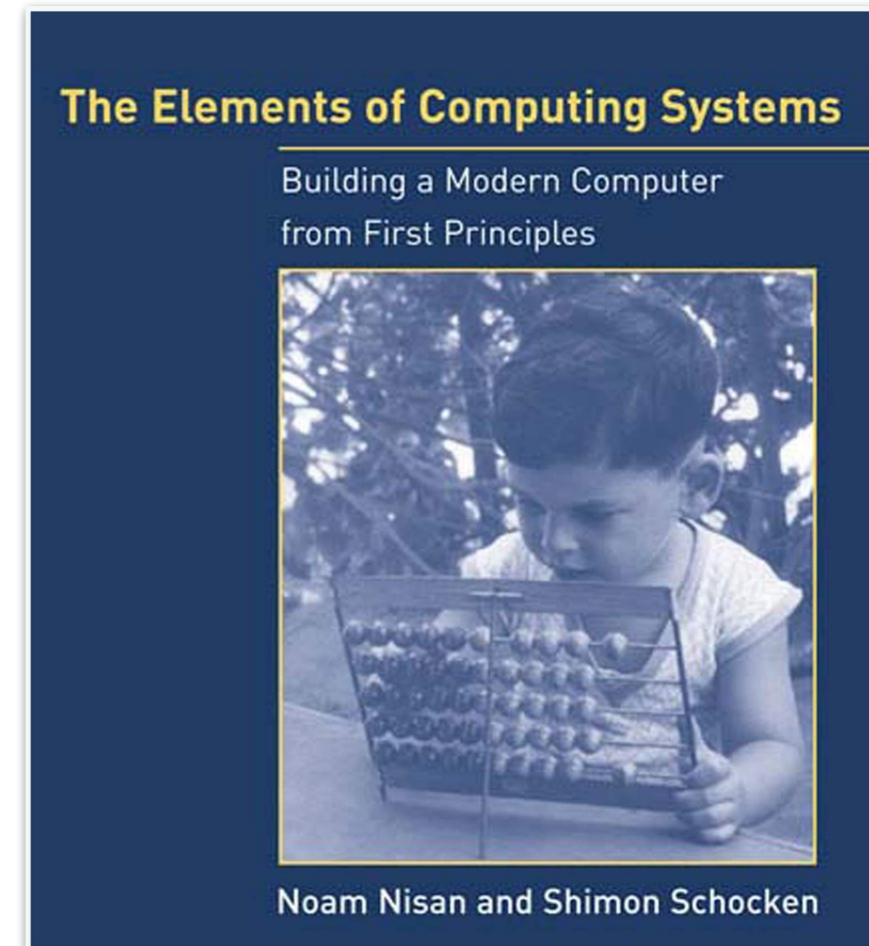
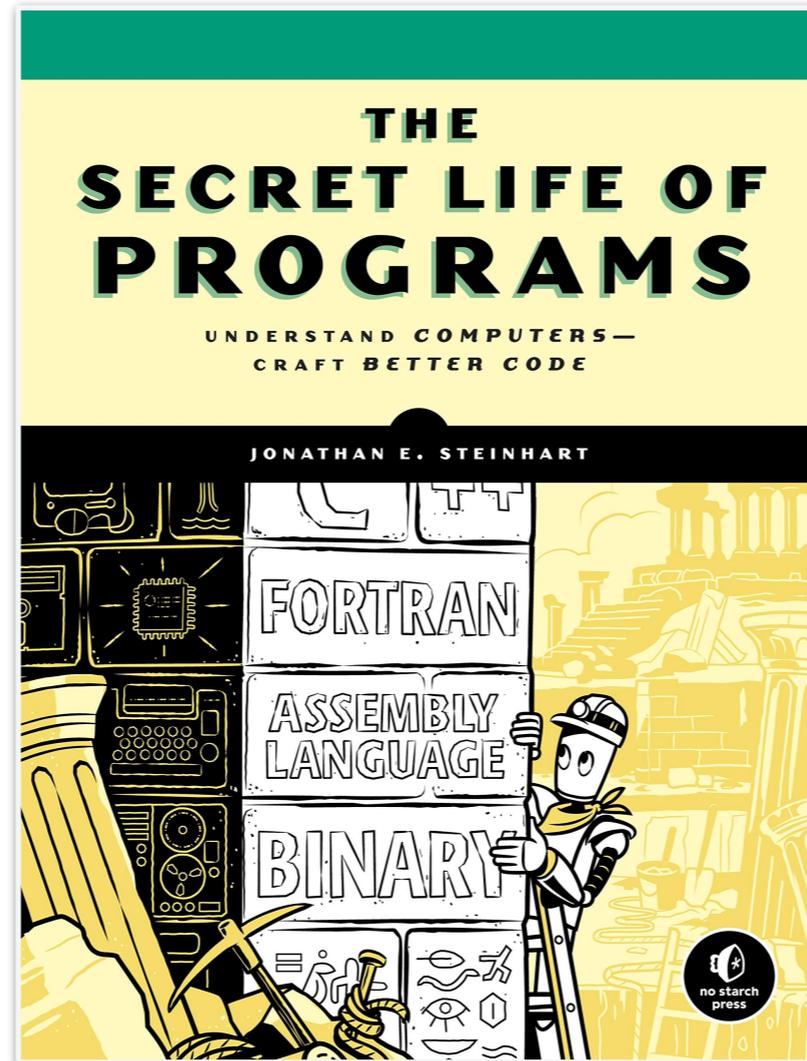
Materiale di lettura

Approfondimento e studio per interesse personale



Mostra come costruire un (semplice) computer a partire da zero.

Prova a rispondere alla domanda di cosa succeda realmente quando eseguiamo un programma. A partire dal processore per poi passare alla parte software



Libro di testo del famoso corso denominato “Nand2Tetris”, il cui scopo è la costruzione di un intero computer, il corrispondente compilatore fino ad arrivare a programmi applicativi