5 - Memoria: registri e RAM

REGISTRI

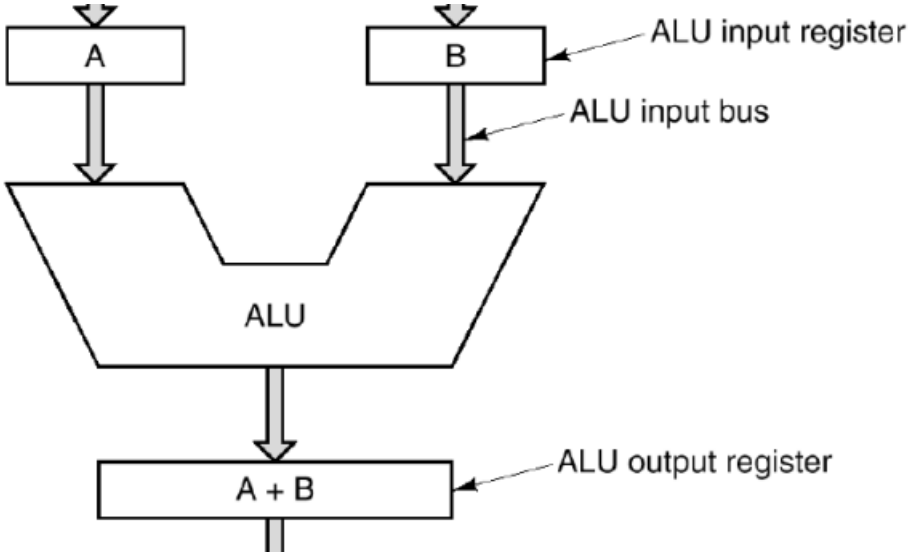
📖 Un **registro** è un elemento di memoria costituito da n flip-flop.

MIPS: 1 WORD = 4 byte ∀ registro ⇒ 32 flip-flop.

I registri si organizzano nel **Register File**.

MIPS: Il Register File consta di 32 registri (⇒32·32=1024 flip-flop).

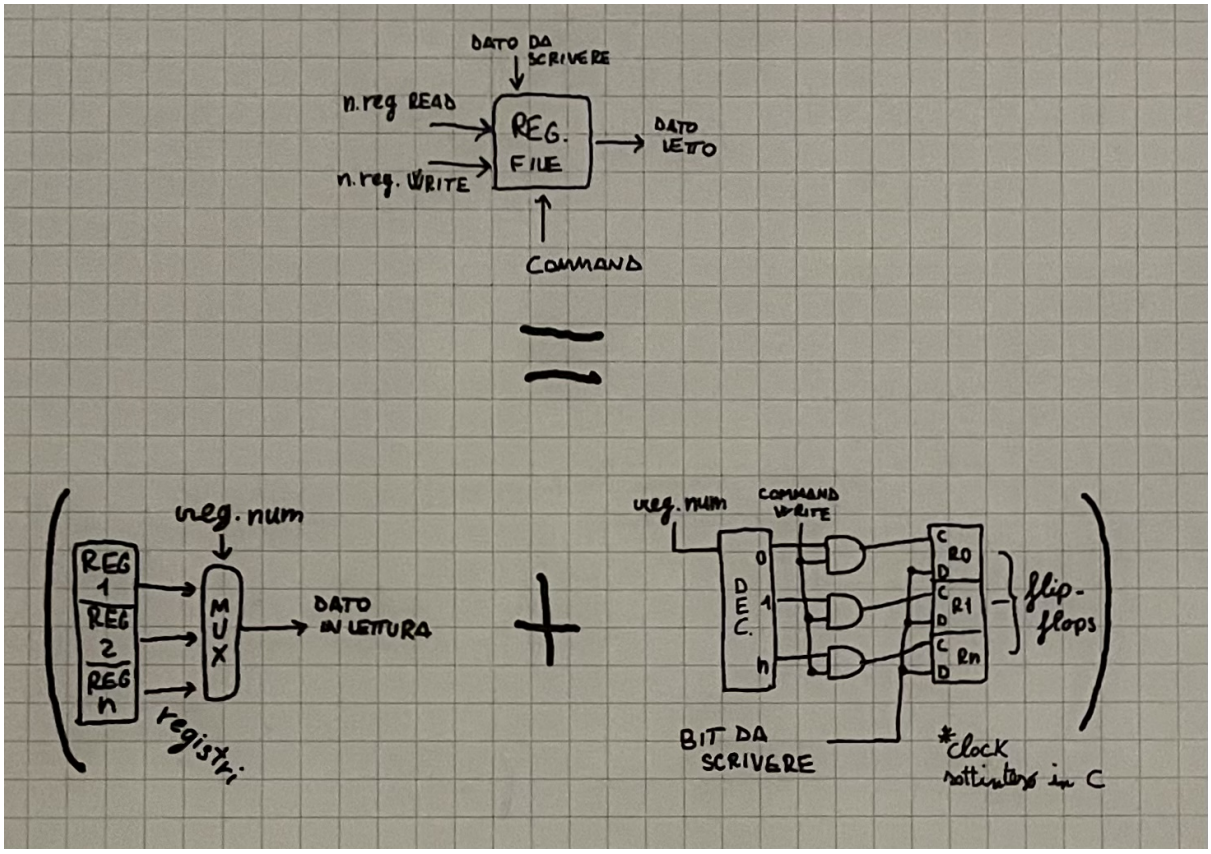
Il Register File consente, di solito, la lettura di 2 registri e la scrittura di un registro.



Esemplificazione dell'uso dei registri: due in input, uno in output.

Le porte **read** prendono in input il numero di registro e, attraverso *multiplexor* (in numero pari ai registri in lettura, grandi come il numero di registri nel Register File), restituiscono in output il dato letto.

Le porte **write** prendono in input il numero di registro, il comando WRITE e il dato da scrivere, attraverso un *decoder* identificano il registro il cui stato sarà modificato se il valore del comando WRITE è affermato (i due dati in ingresso si interfacciano tramite porte AND) e, infine cambierà lo stato del registro (flip-flop) mediante il dato da scrivere e lo stato del *clock*.



Register File semplificato = sistema di lettura del dato + sistema di scrittura del dato

⚠️ ATTENZIONE! In read, il register file fornisce sempre una **coppia** di registri in output.

⚠️ ATTENZIONE! Il **periodo** del clock deve essere sufficiente da permettere al segnale di propagarsi dall'output del register file al suo input (l'output del register file deve avere il tempo di essere immagazzinato nei registri).

⚠️ ATTENZIONE! Il **tipo di dato** processato è stabilito dal programmatore, che lo utilizza coerentemente con le sue caratteristiche.

MEMORIA

Oltre ai registri, esistono altri tipi di memorie, che si distingono per **dimensione**, **velocità** di recupero dei dati, **consumo** elettrico e **costo** per bit.

La **gerarchia** delle memorie stabilisce che, solitamente, le memorie più piccole, veloci e costose sono poste a **livelli alti** (interfacciate con la CPU); le memorie più grandi e lente sono poste a **livelli bassi**.

I **registri** sono posti al livello più alto, la **cache** e la **memoria principale** sono di livello più basso, mentre **supporti magnetici**, **ottici** ed **a nastro** sono al livello infimo della gerarchia.

RAM

La memoria prinicipale è la **RAM** (lenta e capiente).

> 📖 La **RAM** si compone di celle di 1 byte l'una, individuate da un *indirizzo di memoria*.

Gli **indirizzi** sono contenuti nel registro indirizzi: utilizzando n bit, si indirizzano 2^n celle di memoria.

Le due operazioni possibili sono:

- **Read**: si copia all'interno del *registro dati* il contenuto della *cella* indirizzata dal *registro indirizzi*.
- **Write**: si copia all'interno della *cella* indirizzata dal *registro indirizzi* il contenuto del *registro dati.*

Esistono due principali tipologie di RAM:

- **DRAM** (dynamic RAM): è meno veloce (50-70 ns) ma più capiente, è realizzata tramite un **condensatore** e, perciò, il suo contenuto deve essere periodicamente sottoposto a **refresh** (si leggono e si riscrivono i dati).

  Il processo di memorizzazione dei dati consta di un condensatore e di un transistor associato, detto pass transistor. Il condensatore ha carica 0/1; quando il pass transistor viene chiuso, il potenziale elettrico del condensatore si trasferisce sulla bit line se la word line è asserted (è attivata in base all'inirizzo di memoria che si richiede).

Nelle DRAM gli indirizzi si suddividono in due blocchi: la parte alta permette di accedere alla riga di memoria, la parte bassa seleziona la colonna specifica, ottenendo così la singola cella cercata. In questo modo si siminuisce sensibilmente la complessità del decoder.

- **SRAM** (static RAM): è più veloce (0,5-2,5 ns) ma meno capiente, è realizzata tramite una **matrice di latch** (*altezza: numero di celle* x *ampiezza: latch in ogni cella*). Si utilizza un unico indirizzo per lettura e scrittura e le due operazioni non possono essere svolte contemporaneamente.

  Esempio: una SRAM da 32mila celle da 8 latch (1 byte) ognuna avrà capienza 32mila kilobyte, 15 linee di indirizzo ($\log_2 32\,000 = 15$) e 8 linee di output.

  A differenza della DRAM, il segnale è mantenuto da due porte invertenti (latch) e permane, in presenza di alimentazione elettrica, senza bisogno di refresh.

  In input la SRAM riceve **Din** (1 linea lunga quanto una cella, implementata per l'operazione write), gli **indirizzi**, il **chip select** (*asserted* per poter leggere o scrivere), l'**output enable** (*asserted* per poter leggere) e il **write enable** (*asserted* per poter scrivere).

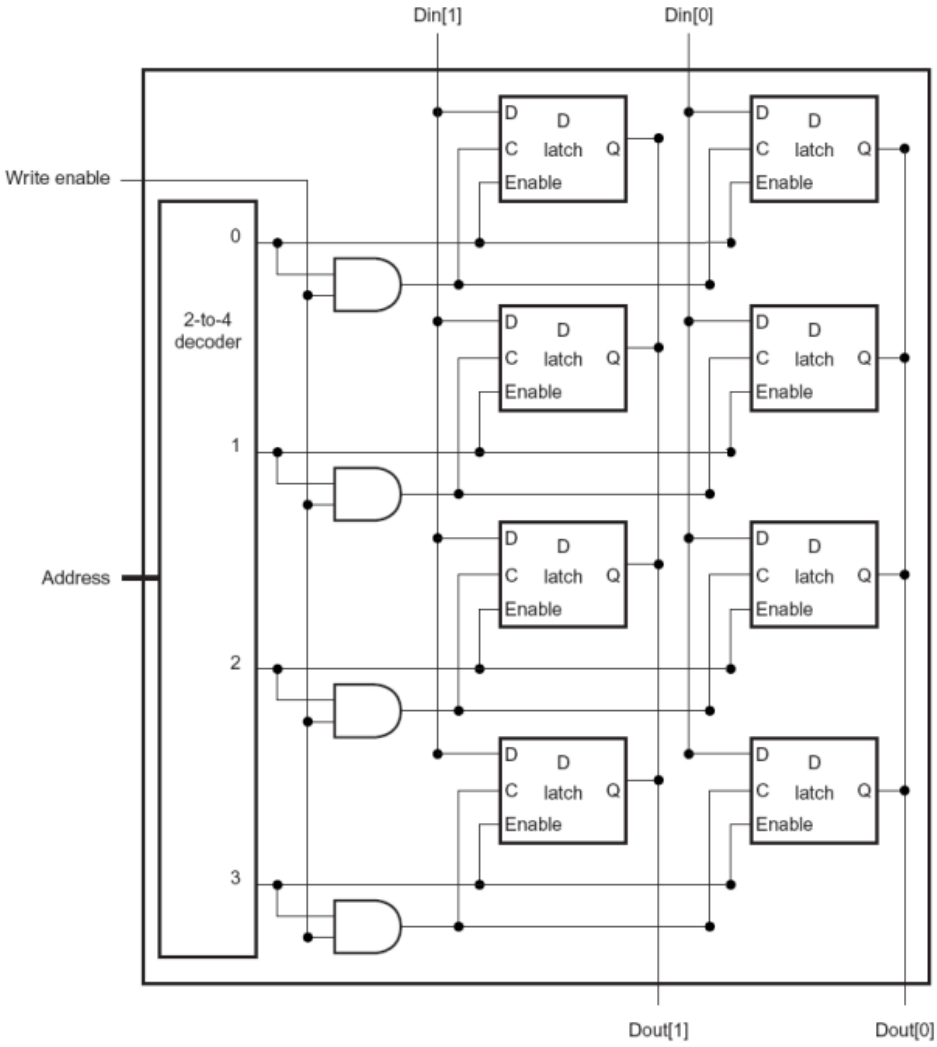
  In output la SRAM restituisce **Dout** (1 linea lunga quanto una cella, implementata per l'operazione read). Se output enable è *disasserted*, l'operazione non sarà effettuata.

  La SRAM non impiega decoder in fase di writing o multiplexor in fase di reading (servirebbero decoder e multiplexor enormi e poco pratici per una memoria tanto capiente, oltre che più livelli di porte AND che rallenterebbero l'accesso alle informazioni).

  Il multiplexor in lettura è sostituito da una **bit line condivisa**. Per collegare gli input a tale bit line si utilizzano i **tri-state buffers**, che aprono o chiudono i collegamenti con l'output mediante l'**input Enable**, che deve essere asserito. In questo caso, solo un Enable per volta risulta vero.

  Ogni D Latch ha un proprio segnale Enable che consente di abilitare il tri-state buffer.

  Un decoder, che riceve in input l'address, abilita alla lettura/scrittura di una data *linea di memoria.*



Memoria SRAM 4x2 (semplificata, senza Output enable e Chip select). Si noti come il decoder consenta di operare solo su una riga di memoria: per esempio, se l'output del decoder risulta 0, si opera solo sui D Latch disposti più in alto nell'immagine.

In SRAM di dimension reali, lo spazio è diviso in più **blocchi in sequenza**. L'indirizzo in ingresso si divide in due parti:

- **la parte alta** dell'indirizzo individua la riga di ogni blocco attraverso un *decoder*;
- **la parte bassa** dell'indirizzo individua la singola cella di memoria attraverso un numero di *multiplexor* pari al numero di blocchi.

In questo modo si ottiene un numero di **Din/Dout** pari al numero di blocchi.

SSRAM E SDRAM

📖 Le **Synchronous SRAM e DRAM** consentono di aumentare la **banda di trasferimento** della memoria (ovvero, il numero di trasferimenti di un dato al secondo) della memoria.

In tali memorie è possibile specificare che si vogliono trasferire dalla memoria una *sequenza di celle consecutive*, detta **burst**.

Ogni burst è specificato da un **indirizzo di partenza** e dalla **lunghezza della sequenza**. La riga viene selezionata una sola volta tramite il decoder. La memoria fornisce, in ogni caso, una cella ad ogni ciclo di clock, ma il costo del decoder è pagato una sola volta (non bisogna ripetere l'indirizzo) e, dunque, si ha una migliore banda di trasferimento.