Appunti IoT Systems

Capitolo 2

La memoria dei microcontrollori

Simone Coco

Memoria dei controllori

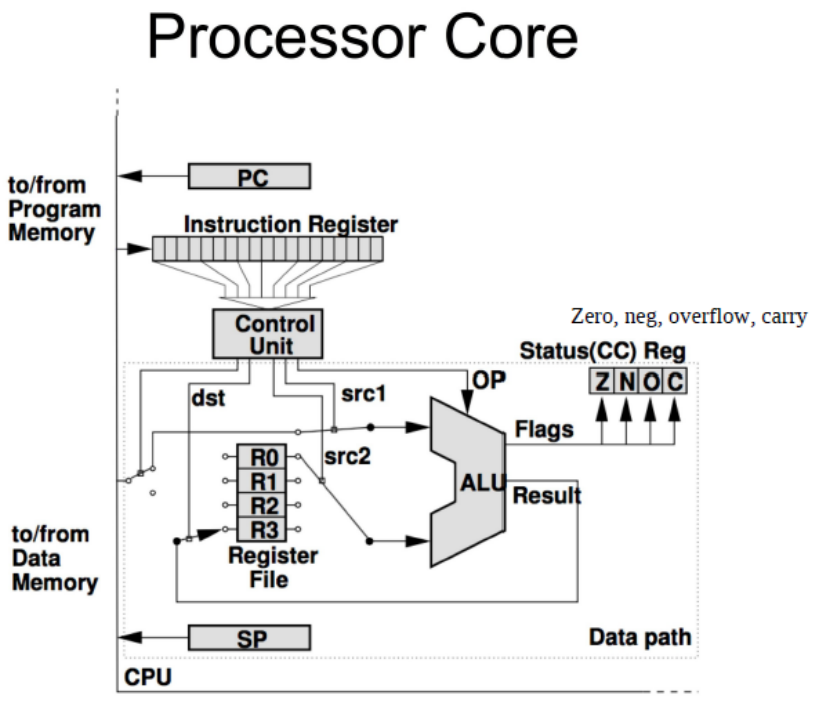
Abbiamo visto che tra i blocchi integrati di un microcontrollore vi sono molti blocchi inerenti alle memorie. Vediamo le loro strutture, il loro funzionamento all'interno del contesto in cui si trovano (quello del microcontrollore, appunto) e la loro caratterizzazione.

Processor core

Abbiamo parlato del processor core e abbiamo evidenziato che erro è lo stesso per microcontrollori appartenenti alla stessa famiglia; esso è integrato nel microcontrollore.

Gli elementi fondamentali del processor core sono, praticamente, quelli tipici dei processori:

- **Program counter (PC):** registro contenente un numero, ovvero l'indirizzo di memoria della prossima istruzione da eseguire. Dopo che l'istruzione in esecuzione finisce, dalla memoria viene estratta l'istruzione da eseguire e viene messa nell'instruction register (IR)
- **Instruction register (IR):** registro contenente la codifica dell'istruzione da eseguire (16 bit o 32 bit a seconda dell'instruction set). Dopo di che la gestione passa alla Control Unit
- **Control Unit (CU):** in base all'istruzione la control unit capisce l'operazione da eseguire (una somma, un prodotto o il settaggio di un determinato valore…), in base a questo può interpellare diversi componenti: registri, memoria, ALU…
- **Unità aritmetico-logia (ALU):** unità addetta alle operazioni
- **R0…R3:** registri addetti alle operazioni
- **Flags:** sono bit utili alle operazioni
- **Stack pointer (SP):** contiene l'indirizzo delle informazioni messe in stack

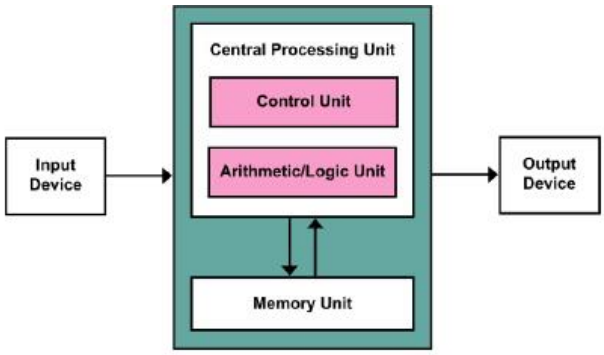


Architettura di Von Neumann contro l'architettura Harvard

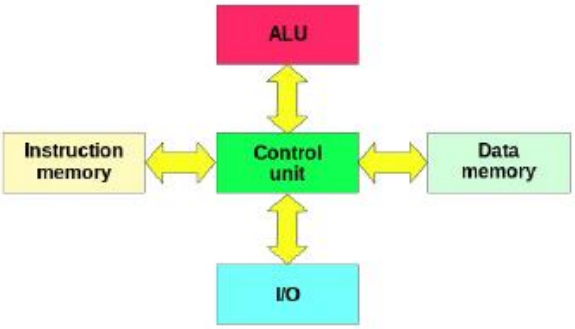
In memoria ci possono stare i dati: le temperature lette, i numeri da sommare ma anche le operazioni da effettuare: in memoria ci stanno sia i dati che le istruzioni, ovviamente localizzati in parti differenti ma comunque a risiedono nella memoria.

Vi sono due approcci fondamentali:

- **Architettura Von Neumann:** I dati e le istruzioni stanno tutti in una memoria, poi sta al processore capire quali siano i dati e quali le istruzioni; vi è lo stesso canale di comunicazione (bus) da cui si possono prendere dati o istruzioni



- **Architettura Harvad:** I dati e le istruzioni sono in memorie separate logicamente e fisicamente, con due bus diversi. Il **vantaggio** sta nel fatto che non si hanno conflitti tra gli accessi ai dati e gli accesi alle istruzioni mentre nell'architettura Von Neumann si deve sempre stabilire cosa fare. Lo **svantaggio** sta nell'utilizzo di più hardware.



Classificazione della memoria

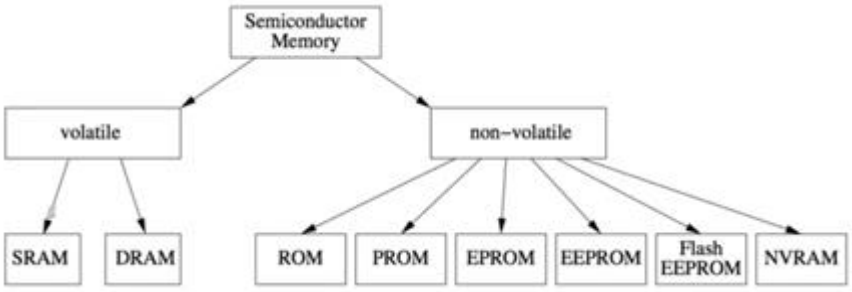
Possiamo classificare la memoria in due schemi logici diversi:

- Dal punto di vista **funzionale** possiamo suddividere la memoria in base alle funzioni che svolge:
  1. **Register File (o semplicemente Registri) :** "contenitori" di bit che si trovano nel datapath del microprocessore e, in alcuni sistemi, sono gli unici cablabili con l'ALU per fare operazioni, sono pochi e stanno dentro il microprocessore (possono essere 32 o 64).
  2. **Memoria dati:** memoria che si riferisce ai dati che devono essere processati
  3. **Memoria istruzioni:** memoria per le istruzioni da eseguire sui dati

- Dal punto di vista delle **caratteristiche fisiche** della memoria; le caratteristiche fisiche sanciscono uno specifico utilizzo delle memorie che, data la preziosità dello spazio dei costi e del tipo di sistema che si sviluppa, è essenziale ai fini del sistema da sviluppare. (ricorda: non possiamo prendere più memorie possibile, sarebbe dispendioso) a tal proposito è fondamentale capire con che memoria lavorare. Possiamo suddividere le memorie come:

1. **Memoria volatile:**
   Memoria che dal momento in cui non è più alimentata, perde il proprio contenuto informativo. Ne fanno parte:
   - **SRAM**
   - **DRAM**
2. **Memoria non volatile:**
   Memoria che mantiene il contenuto informativo anche se non è alimentata. Ne fanno parte:
   - **ROM**
   - **PROM**
   - **EPROM**
   - **EEPROM**
   - **Flash EEPROM**
   - **NVRAM**

Sono due memorie avente caratteristiche differenti, non vi sono migliori o peggiori, si adattano a determinate caratteristiche e si possono adoperare in diverse quantità, a seconda del bisogno del sistema.



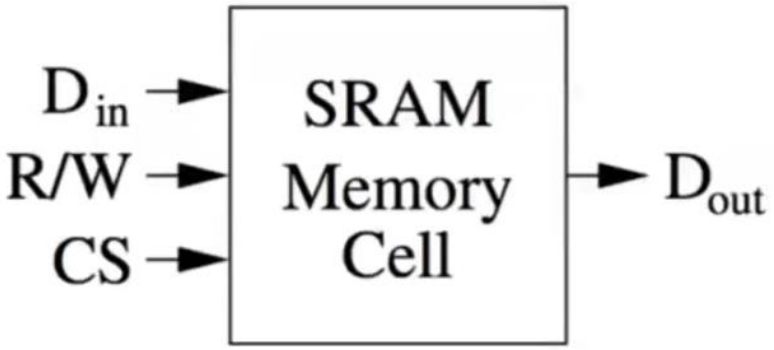
1.Le memorie volatili

Le memorie volatili sono delle memorie che memorizzano un dato fintato che esse sono alimentate. Quando si stacca l'alimentazione esse perdono tutti i dati. Possiamo suddividerle in Memorie SRAM e memorie DRAM.

L'acronimo RAM sta per "Random Access Memory"; tuttavia l'uso del termine "Random" è improprio poiché l'accesso in memoria non è casuale. Si usò il termine "random" per distinguere queste memorie dalle memorie sequenziali. Queste memorie consentivano di decidere una posizione in memoria e di scriverci, quelle sequenziali consentivano solo di scrivere in memoria in maniera sequenziale.

Static Random Access Memory (SRAM)

Le Static Ram sono delle memorie **molto veloci** e abbastanza **costose** che consistono in celle, ogni cella memorizza un bit ed è realizzata fisicamente attraverso l'utilizzo di 6 transistors. Con 6 transistors si realizza un flip-flop, necessario alla memorizzazione di un bit.



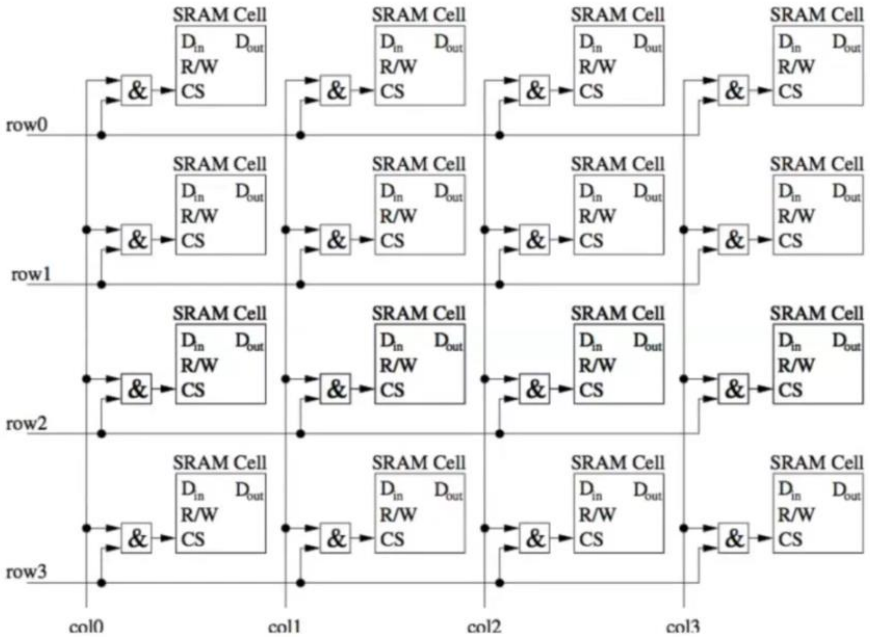
Definiamo le interfacce esterne per capirne il funzionamento:

- **Segnale Din:** è il segnale di input.
- **R/W:** bit che segnala se l'operazione da effettuare è una lettura o una scrittura; se esso **vale 0** significa che la variabile contenuta in **Din** deve essere scritta in memoria (memorizzata), se esso **vale 1** allora significa che la cella deve essere letta e quindi viene settata **Dout** per salvare il valore.
- **Cell Select (CS):** il segnale cell select serve per capire quando effettuare le operazioni di lettura o scrittura: il segnale raggiungerà, ad esempio, un picco in 1 e solo in quel momento viene eseguita l'operazione di scrittura o lettura. Quindi la cella non è sempre attiva, si "attiva" quando il segnale nella Cell Select arriva al suo picco.

Matrici di Celle di memoria in una SRAM

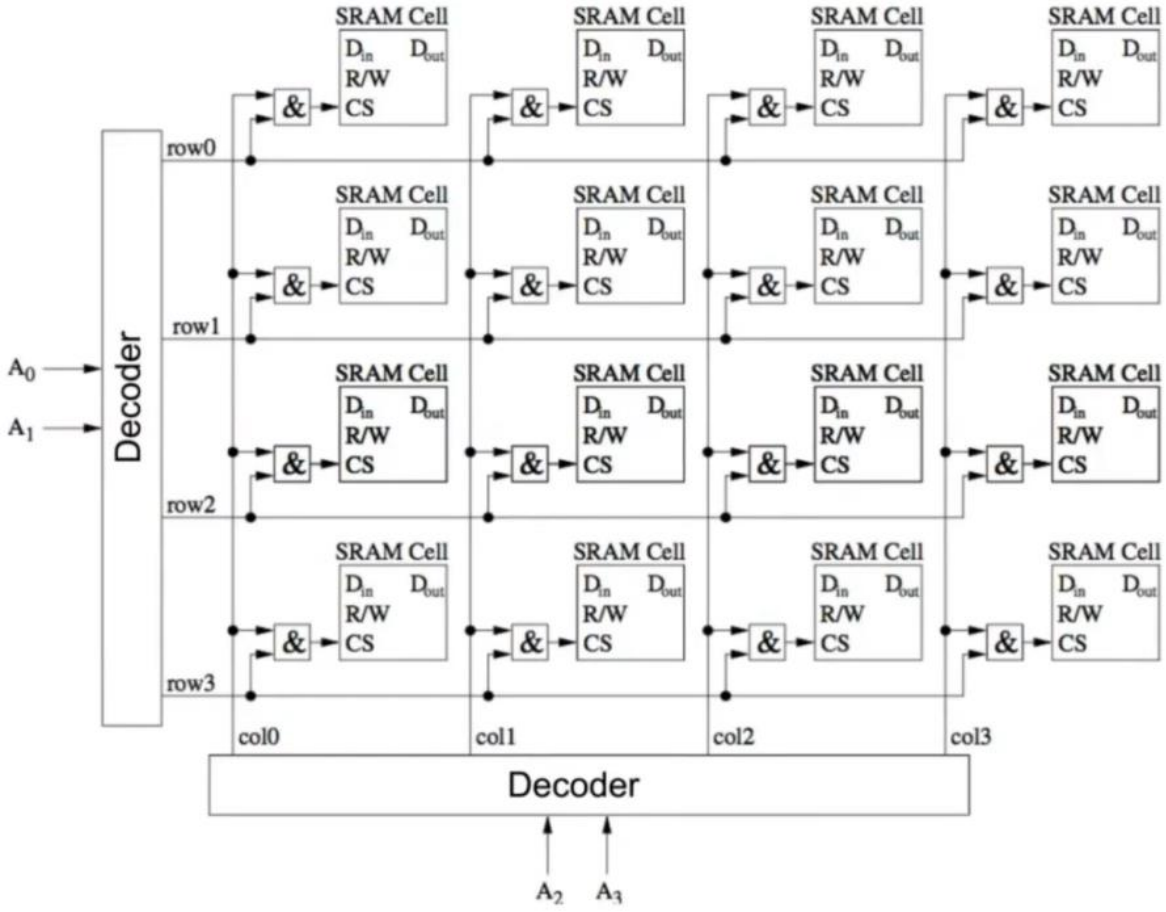
Questa "scatola nera" vale per una sola cella di memoria, in grado di memorizzare un solo bit. Supponiamo adesso che volessi implementare una memoria avente 16bit.

In tal caso provvedo ad implementare una **matrice di bit**, in questo caso implemento una matrice (4x4).



Ma come faccio a segnalare esattamente quale cella leggere o in quale cella scrivere?

- Potrei affidarmi ai CS delle celle: ogni cella ha una porta AND all'ingresso del CS che combina la riga e la colonna, quando entrambi sono true allora anche il CS è true e sono in grado di individuare la cella avente il picco true nel CS. È una soluzione valida ma per grandi matrici di celle (ad esempio 1024x1024) non è la soluzione migliore. Pertanto, nella realtà in un dato momento io seleziono solo una riga e solo una colonna, quindi non ho bisogno di 4 linee ho semplicemente bisogno di un decoder che in base alla combinazioni di due bit individui se la cella che sto cercando sta nella linea 1,2,3 o 4 (poiché $2^2=4$)



Tuttavia, ancora resta ignota una cosa: è davvero conveniente implementare delle memorie volatili SRAM utilizzando ben 6 transistori (ricordando inoltre che ne bastino due per codificare uno stato on/off)? Sicuramente lo spazio occupato in questo modo sarà uno svantaggio ma sono memorie molto veloci quindi si può trascurare lo svantaggio dello spazio in vista di questo vantaggio.

Dynamic Random Access Memory (DRAM)

Le DRAM utilizzano un transistor per una sola cella di memoria. In questo modo si riduce lo spazio occupato, a parità di spazio, una DRAM avrà sicuramente più memoria di una SRAM in quanto avrà più celle. Ciò è una conseguenza del fatto che non vengono utilizzati flip-flop (e quindi i transistori per crearli), bensì viene utilizzato un capacitore (condensatore) che viene caricato applicando una tensione ai suoi capi e scaricato, applicando un valore logico quando è carico e quando è scarico.

In questa configurazione quindi, per memorizzare un bit basta un condensatore che mantenga l'eventuale carica e un transistor per caricarlo.

Il problema sta nel fatto che quando un condensatore è carico si hanno delle **correnti di perdita** per cui un condensatore carico, seppur lentamente, perde la carica facendo cadere il valore in uno stato di valore indefinito (non è al massimo della soglia per valere 1 ma non è nemmeno completamente scarico, quindi non può valere 0)

Quindi per queste memorie si hanno grandi velocità e, a parità di spazio, maggior memoria rispetto le SRAM ma il prezzo da pagare è che **si devono continuamente refreshare: vi è bisogno di una parte hardware aggiuntiva che refreshi il valore del condensatore, per ovviare al problema della corrente di perdita.**

Questo implica due notevoli svantaggi:

1. **Costo aggiuntivo dell'hardware:** si deve aggiungere una parte hardware che influisce nei costi di produzione
2. **Momenti di inattività:** vi sono momenti in cui si deve refreshare la memoria in cui non è possibile accedervi, questo avviene ogni pochi millisecondi

Quale memoria volatile usare per i microcontrollori

Questi svantaggi segnano una importante condizione: è meglio preferire le SRAM quando si parla di microcontrollori e spieghiamo il perché:

Vantaggio:

Le DRAM hanno il **vantaggio di avere una quantità maggiore di memoria** a parità di spazio rispetto alle SRAM.

Svantaggio:

Tuttavia, **hanno lo svantaggio di essere più lente** delle SRAM per via delle operazioni di refresh e svilupparle ha un **costo maggiore** a causa dell'hardware addetto al refresh.

Quando parliamo di microcontrollori non necessitiamo di grandi banchi di memoria: non essendo dispositivi general purpose (in cui la quantità di memoria è fondamentale), la memoria di cui necessitano è limitata a quantità minime e quindi **si preferisce la SRAM poiché basta una memoria limitata e non serve affrontare costi maggiori in produzione (hardware per il refresh) e in prestazioni (tempi di inattività per il refresh).**

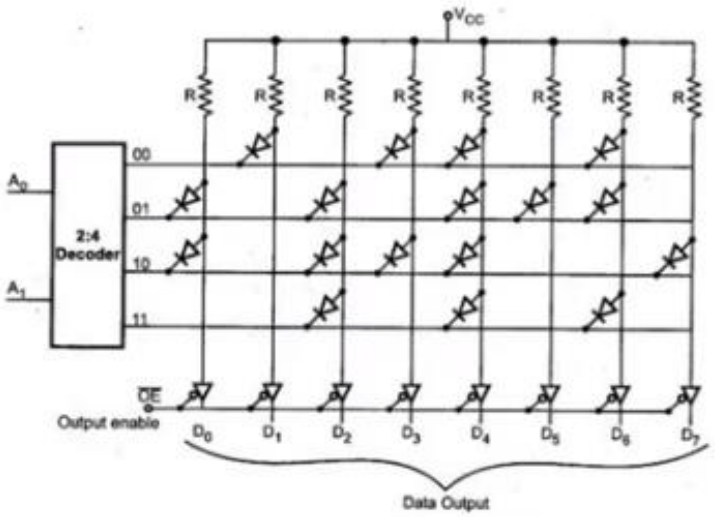
Le memorie non volatili

Il contenuto delle memorie non volatili resta memorizzato anche quando esse non sono alimentate. Esse sono più complesse di quelle volatili poiché hanno un requisito in più: l'immagazzinamento "permanente" dei dati.

Read only Memory (ROM)

La memoria ROM è una memoria non volatile ed è sostanzialmente identica ad una RAM: è formata da una matrice di bit. Tuttavia, rispetto la RAM, essa ha la capacità di memorizzare i dati in maniera permanente (in quanta memoria non volatile) ed è una memoria di solo accesso in lettura

In fase di fabbricazione vado a saldare fisicamente i bit 1 e 0 nella memoria; analizzando uno schema (semplificato) possiamo notare come funziona una ROM da 4 byte:

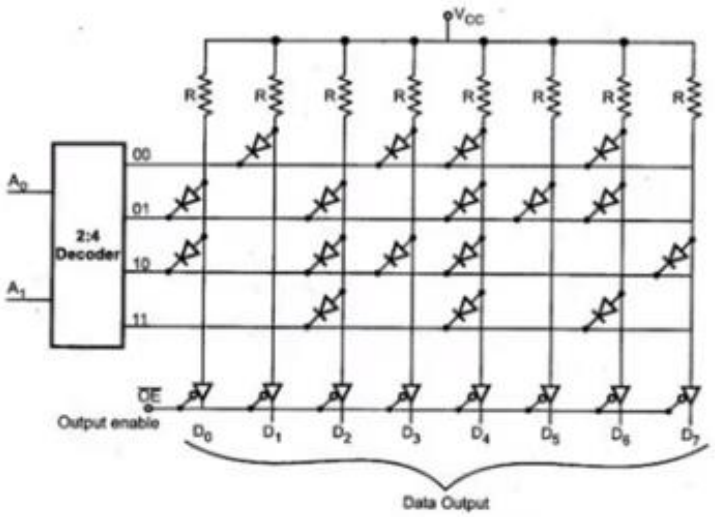


il decoder che prende in input i bit A1 e A2 e dà quindi in uscita un massimo di 4 combinazioni (poiché 2^2= 4), ognuna delle soluzioni possibili costituisce una riga, per un totale di 4 righe. Ogni riga è collegata ad 8 colonne, l'unione con le colonne daranno in uscita un bit, per un totale quindi di 8 bit (= 1byte) per riga. Ogni riga quindi dà in uscita un byte, per un totale di 4 byte su 4 righe.

La caratteristica della ROM è che, una volta che viene selezionata la riga, il fatto che da essa escano 1 o 0 è determinato permanentemente in fase di fabbricazione; Questo funzionamento è stabilito inserendo dei diodi che cortocircuitano determinate parti.

Facciamo un esempio:
Si consideri l'insieme delle colonne: esse sono alimentate dalle tensione Vcc.
Prendiamo in esame la prima riga (la riga 00), essa viene attraversata da 8 colonne. La prima colonna non tocca la riga 00 (lo si può notare poiché dallo schema sono "una sopra l'altra", esse si toccano solo grazie al diodo che le unisce), quindi sostanzialmente nell'unione della riga con la colonna non si ha caduta di tensione, il valore che si ha in uscita è quello di Vcc stesso, che a livello di informazione assumiamo sia il valore 1.
Nella seconda colonna possiamo notare la presenza di un diodo che diventa un cortocircuito: fa si che si ha una differenza di potenziale, ovvero un passaggio di corrente (se vi è d.d.p. vi è anche corrente), quindi in

uscita avrò un valore diverso da Vcc: avrò il valore (Vcc - x), dove x è la caduta di tensione dovuta al cortocircuito, possiamo assumere questo valore come 0.
[I valori 1 o 0 assunti in questo esempio sono solo convenzionali: si deve decidere a monte se il valore di Vcc originale è 0 o 1 e di conseguenza (Vcc - x) assumerà il valore opposto a Vcc]



Tralasciando i dettagli di fabbricazione, possiamo dire che in un determinato momento solamente una delle 4 righe può essere vera poiché il decoder ne sceglie solamente 1, un'altra particolarità da tenere ben a mente è che queste memorie escono così di fabbrica, quindi non è possibile cambiarne lo schema, sono appunto Read-Only.

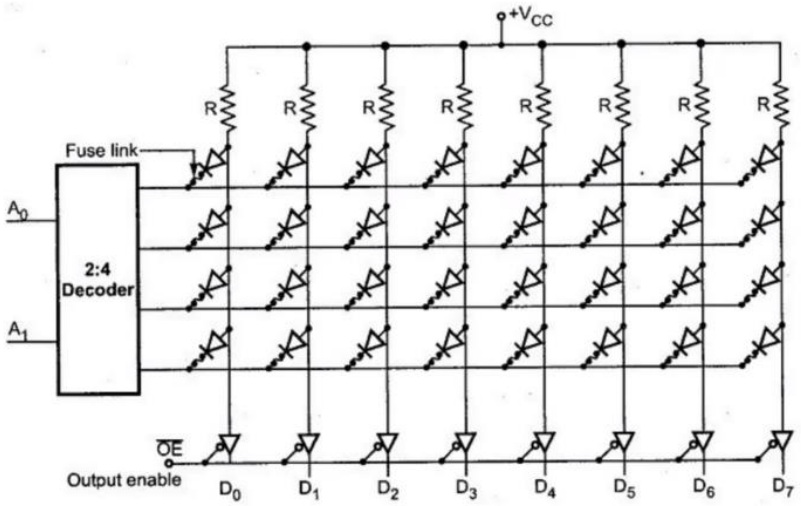
Un difetto di questa memoria è che in caso di aggiornamento, come il cambiamento di un bit, si dovrebbe rifabbricare l'intera memoria.

Programmable Read Only Memory (PROM)

Le PROM sono delle memorie rom che possono essere programmabili.

Analizzando lo schema di una PROM possiamo subito notare che essa ha diodi ovunque: essi collegano tutte le colonne a tutte le righe, questo già suggerisce che fabbricarla sarà meno costoso: non si necessita dell'accuratezza di mettere un diodo in una posizione piuttosto che in un'altra, come accadeva per le ROM, i diodi si possono mettere dappertutto, creando una **maschera che contiene diodi dappertutto.**

Si parte dunque da una situazione in cui si hanno tutti i diodi, dopo di che si applicano delle alte tensioni su alcuni di essi, in modo tale da fondere i fusibili collegati ai diodi, disattivando quest'ultimi in modo permanente. [NB: i fusibili nello schema sono collegati direttamente ai diodi: "Fuse link"]



Differenze tra ROM e PROM:

- La **ROM deve essere creata "su misura"** con diodi saldati su determinate posizioni specifiche.
- La **PROM** offre una "**soluzione generale**" che permette ad un progettista di ricreare la propria ROM su misura, fondendo i diodi che non servono.

La PROM è "programmabile" nel senso di "modellabile", ma in maniera **irreversibile**

Vantaggi:

√ Economicamente vantaggiosa: il vantaggio di vendere una soluzione "standard" per ogni utilizzo rende le PROM meno costose delle ROM poiché queste devono essere create ad hoc.

Svantaggi:

× Le PROM non possono essere "aggiornate", cioè non sono vantaggiose ai fini dello sviluppo, poiché una volta programmate non si può tornare indietro.

Erasable Programmable Read Only Memory (EPROM)

La EPROM è praticamente una PROM che può essere cancellata e riprogrammata, quindi sopprimendo lo svantaggio appena indicato. La differenza sta nel fatto che, invece di distruggere i diodi per ricreare lo schema della memoria, vengono assunti dei **FETs (Field Effect Transistors)**, dei **transistor che possono essere fisicamente resettati mediante alte tensioni:** le alte tensioni creano un cortocircuito nel transistor permanente, senza fonderlo. Con questo metodo si garantisce la **reversibilità** del sistema: è possibile tornare indietro e riprogrammare la memoria.

Si noti che il cortocircuito **non è permanente nel lungo periodo: esso si degrada negli anni fino a tornare nella condizione iniziale** (ovvero quella del transistor in condizioni normali). Questo processo di "invecchiamento" potrebbe essere un difetto ma può essere utilizzato in maniera positiva: **applicando una luce ultravioletta il processo di "invecchiamento" può essere accelerato.**

Questo significa che, applicando una luce ultravioletta, di accelera il processo di **cancellazione** dei dati. Per questo motivo ogni memoria EPROM ha un piccolo vetro che consente la cancellazione attraverso

una luce ultravioletta. Quindi con questo “invecchiamento precoce” si può resettare la memoria nel breve periodo per riutilizzarla.

Vantaggi:

- √ Le EPROM possono essere riprogrammate. Inoltre, è possibile accelerare il processo di decadimento del cortocircuito del transistor attraverso l’uso di una luce ultravioletta.

Svantaggi:

- × Non è consigliabile l’impiego delle EPROM in sistemi che devono garantire una lunga durata poiché, seppur lentamente, il processo di “invecchiamento” porta la EPROM ad esaurire il cortocircuito nei transistor che la compongono, portandola a tornare nello stato di fabbrica originale (quindi cancellandone automaticamente il contenuto).

Electrically Erasable and Programmable Read Only Memory (EEPROM)

Le EEPROM hanno un funzionamento sostanzialmente identico alle EPROM ma per cancellarle non vi è alcun bisogno dei raggi ultravioletti, basti bensì un impulso elettrico ad alto voltaggio.

Questo processo può essere effettuato un numero finito di volte poiché in un numero elevato di volte l’alto voltaggio danneggerebbe irreversibilmente il sistema. Quindi il processo di cancellazione va effettuato in un numero limitato di volte e in casi specifici.

Vantaggi:

- √ È cancellabile come le EPROM ma, a differenza di queste che possono essere cancellato attraverso l’utilizzo di una luce UV, esse possono essere cancellate elettricamente.
  Questo vantaggio implica che non vi è “l’invecchiamento” durante il corso temporale

Svantaggi:

× Le cancellazioni dei dati in memoria possono essere effettuati solo in un numero limitato di volte poiché esse avvengano con alte tensioni e se si sottopone la memoria a continui picchi di tensione elevati la si danneggia irreversibilmente.

Flash Electrically Erasable and Programmable Read Only Memory (Flash EEPROM)

La flash EEPROM è la "versione moderna" della EEPROM ed è quella che **viene integrata nei microcontrollori.** Sono una semplificazione delle EEPROM in cui ciò che si cancella è il contenuto intero. Le differenze sostanziali tra EEPROM e Flash EEPROM sono:

- **EEPROM:** permettono la **cancellazione in vari contenuti**
- **Flash EEPROM:** viene **cancellata tutta la memoria** come unica entità (viene flashata la memoria)

Dato che nella Flash EEPROM i dati vengono inseriti in un'unica volta e, se volessimo inserire altri contenuti dovremmo praticamente eliminare l'intero contenuto precedente, in esse **va messo il programma che il microcontrollore deve eseguire.**
La motivazione è semplice: **il microcontrollore deve eseguire un solo programma** quindi questo implica che esso deve essere cancellato solamente se, ad esempio, si deve aggiornare il programma per aggiornarlo ad una nuova versione.

Chiaramente anche la Flash avrà dei limiti, essendo una memoria EEPROM "particolare" avrà gli stessi limiti di queste memorie: può essere flashata un numero finito di volte poiché gli impulsi ad alta tensione la danneggerebbero. Si parla di circa 100.000 flashate possibili, considerando che il programma viene aggiornato un numero limitato di volte, tale limite non è particolarmente vincolante.

Vantaggi:

√ Cancellano interamente il contenuto, questo è vantaggioso nei microcontrollori.

Svantaggi:

× Può essere resettata un numero limitato di volte visto che è comunque una EEPROM (svantaggio lieve poiché il reset nei microcontrollori che devono compiere esattamente un compito è una soluzione poco effettuata).

Non-Volatile Random Access Memory (NVRAM)

Le memorie NVRAM sono memorie RAM non volatili; l'implementazione consiste in due modalità:

1. Una **RAM con una Batteria**, in modo tale che, quando si interrompe l'alimentazione, la RAM resta comunque alimentata dalla batteria, in questo modo si ha la memorizzazione "permanente" dei dati.
   [Esempio: il BIOS è una memoria alimentata da batteria, se si scarica la batteria o non funziona correttamente si perde il contenuto].

2. Si instaura un **sistema formato da una SRAM e una EEPROM**, quando viene acceso il dispositivo il contenuto delle EEPROM viene caricato nella SRAM, in esecuzione viene utilizzata la SRAM poiché è molto veloce e può essere scritta varie volte e, nel momento in cui si termina l'esecuzione e viene spento il dispositivo, viene effettuata la copia del contenuto nella EEPROM, attraverso un sistema hardware.

Accesso alla memoria

Nei microcontrollori la program memory e la data memory sono integrate e di solito si utilizzano:

- Le **Flash-EEPROM come program memory**, per salvare il programma in modo permanente e, in caso di aggiornamento ad una nuova versione o di particolari necessità, esso può essere flashato (= eliminato interamente in una sola volta)
- Le **SRAM come Data memory**, per caricare delle variabili dinamicamente. Vengono impiegate esse piuttosto che le DRAM poiché sono più veloci ed economiche e, anche se possono caricare un numero minore di dati a parità di spazio, ciò non implica un sostanziale svantaggio perché nei microcontrollori non si devono caricare grosse quantità di dati.

Quindi conoscendo le variabili che deve allocare e il compito che deve svolgere un microcontrollore attraverso un programma, si può determinare quali microcontrollori scegliere a seconda dei componenti integrati su essi.

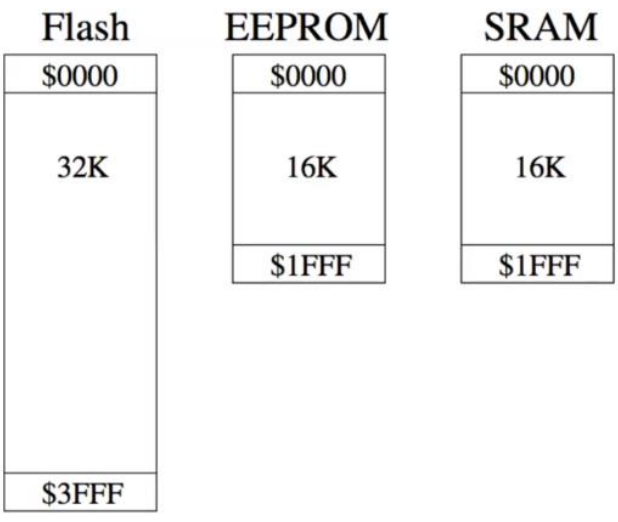
Il microcontrollore ha varie memorie quindi è lecito chiedersi: come avviene l'accesso in memoria avendo memorie tra loro eterogenee (volatili e non volatili)?

- Esistono due possibili approcci:

1. **Diversi spazi di indirizzamento:** gli indirizzi totali di memoria vengono gestiti separatamente: se si ha l'indirizzo "0" si deve specificare se questo è l'indirizzo "0" della memoria Flash, della EEPROM o della SRAM. Il programmatore deve specificare a quale memoria si deve fare riferimento, questo può essere effettuato con metodologie differenti: utilizzando istruzioni specifiche per la flash piuttosto che per la SRAM, con implementazioni Assembly differenti.

   Svantaggio: il programmatore deve esplicitamente occuparsi della specifica degli indirizzi (se si riferiscono alla Flash o alla SRAM), con possibili problemi nella decodifica delle istruzioni.

   Vantaggio: seppur più complesso, l'accesso alle memoria è, per natura, separato.

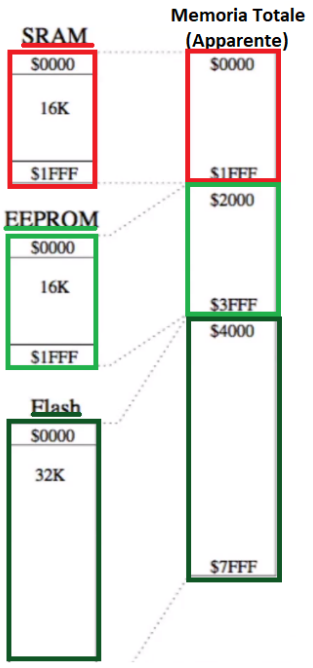


2. **Memorie fisicamente separate ma viste come un unico spazio di memoria:**

le memorie hanno indirizzi di memoria separati ma il programmatore li "mappa" in modo tale da vedere gli spazi eterogenei delle memorie come un unico spazio.

Esempio:
gli indirizzi nel range ($0000 - $1FFF) mappano gli indirizzi della SRAM che vanno da [$0000] a [$1FFF] , gli indirizzi nel range ($2000 - $3FFF) mappano gli indirizzi della EEPROM che vanno da [$0000] a [$1FFF], gli indirizzi nel range ($4000 - $7FFF) mappano gli indirizzi della Flash che vanno da [$0000] a [$3FFF].

Al programmatore interessa conoscere solo la distinzione degli indirizzi (sapere entro quale range si parla di SRAM, di EEPROM o di Flash), sarà poi il microcontrollore stesso che, attraverso la mappatura, lavorerà sugli indirizzi fisici a seconda dell'indirizzo mappato. Quindi per il programmatore è più semplice programmare ma il microcontrollore risulta più complesso poiché deve gestire la mappatura.



Vantaggio: Il vantaggio nell'utilizzo di una mappatura degli indirizzi di memoria sta nell'alleggerimento del carico di lavoro al programmatore

Svantaggio: L'errore numerico sul calcolo degli indirizzi può avere un impatto **devastante** per alcune componenti. Questa è una conseguenza dinamica, che si ripercuote durante l'evoluzione del programma, anche se questo staticamente sembra non presentare errori.