Introduzione ai sistemi a microprocessore

1. Come si è arrivati ai microprocessori

Se si è provato a progettare e successivamente a sviluppare un circuito combinatorio o sequenziale appare immediatamente evidente sia la complessità sia il lungo tempo necessario. Non esistono soluzioni facilmente praticabili che possano semplificare la vita al progettista. Ad ogni errore, di progettazione o di cablaggio, l'attività di riparazione o correzione non è breve e può a volte indurre a riprogettare il tutto.

Ogni circuito così ottenuto è "dedicato", nel senso che svolge un compito specifico, ben definito, e non può fare altro; non c'è modo di sfruttare molto di quanto realizzato per usarlo in progetti futuri; si segue uno schema:

un progetto ←→ un circuito

Qualunque piccola variazione dell'idea iniziale, o anche un piccolo errore, comportano spesso il rifacimento del progetto, a partire dalla parte teorica sino ad arrivare al circuito finito. Ciò comporta un ingente impiego di risorse umane e di denaro! Nei progetti puramente combinatori i problemi che si incontrano sono minori, ma siamo sempre in presenza di circuiti costituiti completamente da ***hardware*** *(HW) ovvero fili, resistenze, circuiti integrati, condensatori e altri componenti elettronici. L'approccio con cui vengono risolti tali problemi prende il nome di* ***logica cablata*** *e con ciò si intende semplicemente dire che si impiega solo HW.*

LOGICA CABLATA: PROGETTO → HW

Nel tempo si è fatta strada un'altra maniera di affrontare i problemi. Si è pensato di costruire circuiti integrati "intelligenti" che potessero svolgere più compiti; si è quindi passati ad uno schema:

più progetti ←→ un circuito

Come sia possibile è presto detto: questi nuovi circuiti intelligenti dovranno essere istruiti a svolgere il loro compito mediante ***istruzioni*** *date dal progettista. Lo scenario cambia radicalmente: bisogna sempre conoscere l'elettronica ma, in aggiunta, si deve imparare il* ***linguaggio*** *con cui impartire le istruzioni ai circuiti.*

Abbiamo di fronte ogni giorno un esempio di come ciò sia possibile: il ***Personal Computer*** *(PC). Esso è costituito da circuiti di notevole complessità e risulta subito evidente che può svolgere più compiti molto differenziati: per esempio potete usare il PC come una macchina da scrivere; alcuni di voi lo avranno impiegato come calcolatrice o come foglio elettronico; sono molto diffusi i giochi per PC; in edicola si trovano prodotti multimediali che riguardano l'arte, l'economia, la scienza. In tutti questi esempi, molto diversi tra loro, l'HW - cioè i circuiti elettronici più le* ***periferiche*** *come la tastiera, il video, il mouse, il disco rigido – è sempre lo stesso! Ogni volta, però, cambiano le istruzioni impartite al PC ed esso può così svolgere nuove funzioni. Senza queste istruzioni il PC risulta un ammasso di ferraglia inservibile.*

L'insieme delle istruzioni impartite a tali circuiti viene detto ***programma*** *e questo giustifica il nome di* ***logica programmata*** *per questo nuovo approccio nell'affrontare e risolvere i problemi. I circuiti intelligenti prendono invece il nome di* ***HW programmabile****. Le istruzioni vanno inoltre impartite seguendo certe regole ben precise che prendono il nome di* ***linguaggio di programmazione****. È anche in uso, come sinonimo di programma, il termine* ***software*** *(SW). Ecco come cambia lo scenario visto precedentemente:*

LOGICA PROGRAMMATA: PROGETTO ➔ HW programmabile+SW

Di fronte ad un nuovo progetto come ci si deve comportare nella scelta tra logica cablata e logica programmata? Dipende dal problema; se viene privilegiata la velocità allora la soluzione cablata non ha rivali; se invece affidabilità e versatilità sono requisiti primari una soluzione programmata è preferibile.

Una cosa deve essere subito chiara: le competenze del progettista sono nei due casi differenti ed egli si dovrà orientare, nel caso di soluzioni programmate, verso un'ottica di integrazione di HW e SW con problematiche tipiche del SW, dell'HW e dell'interazione tra i due.

CPU e microprocessore

Il primo HW programmabile è stata la ***CPU*** *(Central Processing Unit, ovvero unità di elaborazione centrale) che è il cuore del computer; essa all'inizio consisteva di circuiti separati per svolgere le proprie funzioni di base; successivamente è stato costruito un circuito integrato completo di tutto che ha preso il nome di* ***microprocessore***[1] *(in inglese: microprocessor), abbreviato in* μ*P.*

Per capire i concetti espressi in questa introduzione conviene fare un esempio concreto: un semplice computer. Precisiamo subito che un computer è costituito da due entità principali: un

[1] *L'invenzione del microprocessore è attribuita ad un italiano: Federico Faggin.*

*complesso circuito elettronico che controlla tutto (detto **sistema a microprocessore** e conosciuto nel modo dei PC come **mother board**) ed una serie di periferiche quali il mouse, la tastiera, il video, la stampante, il disco rigido, il drive per CD-ROM. Quanto esamineremo nel seguito è il sistema a microprocessore che, essendo il cuore del computer, viene denominato **microComputer** e abbreviato in µC. La sigla µC viene attualmente utilizzata per indicare i **microcontrollori** che sono a tutti gli effetti dei microcomputer realizzati su un unico chip.*

2. I microComputer e l'architettura a bus

La struttura di un computer moderno è abbastanza complessa e frequenti sono le innovazioni offerte dai costruttori; ci limiteremo pertanto ad uno schema molto generale. Consideriamo innanzitutto come sono organizzati i componenti fondamentali che costituiscono un semplice computer (fig. 2.1).

Architettura a bus

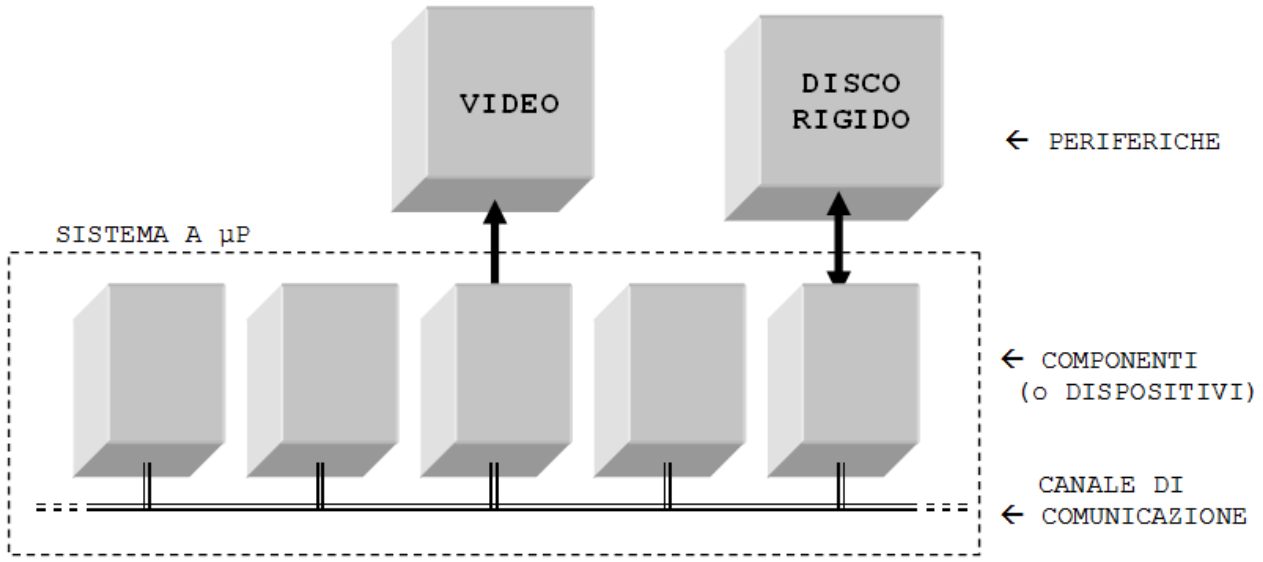

Figura 2. 1 Tipica architettura a bus

*Il modo in cui sono collegati i vari componenti del sistema a uP viene detta **architettura a bus** ed è caratterizzata da un canale di comunicazione chiamato **bus** condiviso da tutti i dispositivi: ognuno di essi è connesso a tutti gli altri. Questa organizzazione ha un vantaggio rispetto ad altre architetture: è facilmente espandibile[2], cioè un nuovo componente si introduce semplicemente connettendolo al bus senza alterare le altre connessioni. Su questo bus viaggiano le informazioni che i vari componenti si scambiano e poiché siamo in un mondo digitale tale scambio avviene sotto forma di bit, cioè 0 e 1; ogni linea trasporta un bit. I componenti che hanno il compito di gestire una periferica comunicano con essa con linee esterne al sistema a µP; questi segnali esterni non fanno parte del bus.*

*Lo scambio di informazioni viene detto **comunicazione**. Generalmente ogni comunicazione coinvolge due componenti, uno che invia l'informazione, il trasmettitore (TX), l'altro che la*

[2] *Oggi è molto in uso il termine "scalabile", con lo stesso significato*

riceve, il ricevitore (RX); in alcune situazioni un TX può comunicare con più RX. Un esempio alla figura seguente:

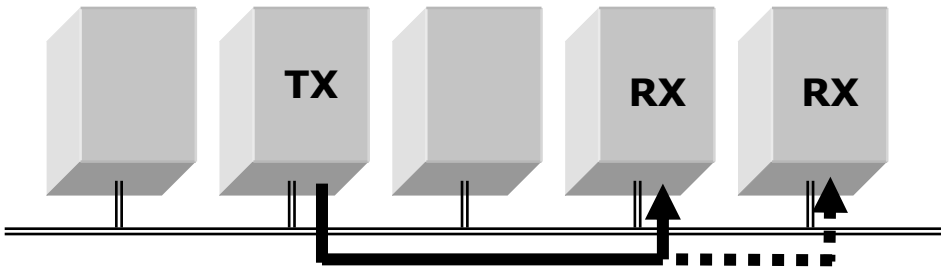

Figura 2. 2 La comunicazione tra TX e RX

Componenti attivi e segnali di abilitazione

I componenti coinvolti nella comunicazione sono detti attivi o abilitati, gli altri invece sono non attivi o disabilitati. Come molti integrati digitali che avete incontrato nel corso di Elettronica, questi dispostivi hanno un particolare segnale che serve per attivarli: prende il nome di "Chip Enable" (CE) o Chip Select (CS) o Strobe (STB); se il Chip Enable non è attivato il componente si comporta come se non esistesse, e non influenza altre comunicazioni.

Alcuni dispositivi possono solo trasmettere, altri solo ricevere, altri possono fare l'uno e l'altro ovvero essere rice/trasmettitori (RX/TX). Si dice anche che: possono solo scrivere, altri solo leggere, altri possono leggere/scrivere.

In una architettura a bus due TX non possono mai trasmettere insieme! Sul bus viaggiano infatti segnali elettrici e con due TX attivi ci si trova in una situazione di potenziale cortocircuito (figura successiva), cosa che deve essere assolutamente evitata; anche se le tensioni in gioco sono basse c'è un'alta probabilità di distruggere gli stadi di uscita dei TX.

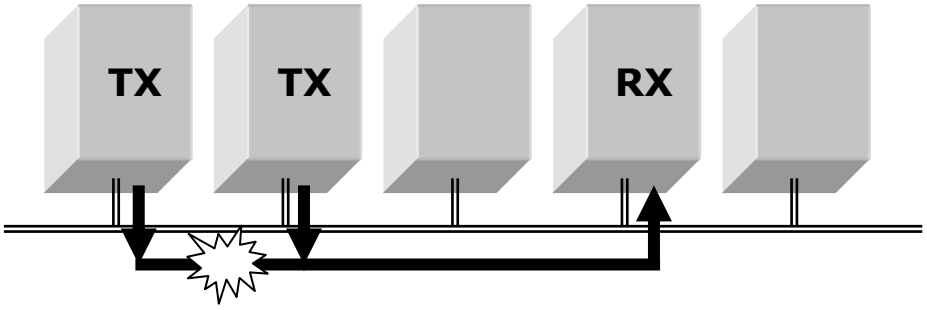

Figura 2. 3 Due TX attivi contemporaneamente possono provocare conflitti

*Esiste in questo caso una situazione di conflitto che prende il nome di **bus contention** (contesa del bus).Il conflitto è non solo elettrico ma anche logico: quale sarà il segnale dei due TX che arriverà al ricevitore? Vedremo più avanti come risolvere il conflitto.*

Non crea problema, invece, la presenza di più RX attivi contemporaneamente[3].

[3] *I problemi che insorgono in questo caso sono di interfacciamento: può succedere che il TX non abbia la potenza sufficiente per pilotare più di un componente.*

Modalità di trasferimento

Possiamo chiederci quali sono, un po' più nel dettaglio, le modalità del trasferimento delle informazioni sul bus. Può aiutare un esempio tratto dal sistema postale: spedire e ricevere una lettera, schematizzata nella figura seguente.

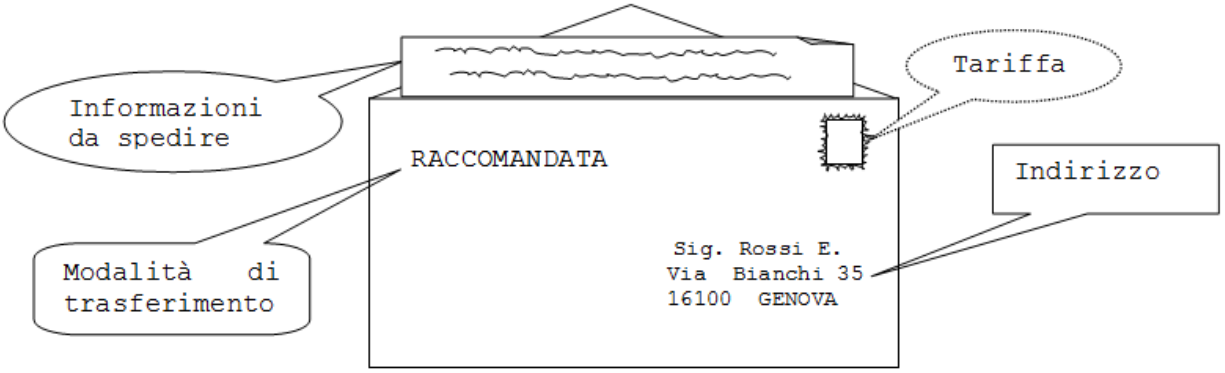

Figura 2. 4 Una lettera e le sue parti principali

Nel sistema postale sono in comunicazione più utenti; quando una persona vuole comunicare con un altro utente per amicizia, lavoro o altro (le informazioni da spedire) deve conoscerne l'indirizzo, che è univoco: ogni utente ha un indirizzo personale, differente da tutti gli altri. Se la lettera è particolarmente importante può scegliere di usufruire di una particolare modalità di trasferimento, RACCOMANDATA o ASSICURATA, altrimenti la lettera sarà considerata NORMALE; la lettera sarà trattata diversamente a seconda di questa indicazione.

Nel µC succedono più o meno le stesse cose. Ogni componente ha il proprio ***indirizzo*** *che lo distingue dagli altri; può ricevere contenuti di vario tipo (****dati*** *o* ***istruzioni****); il trasferimento è governato da segnali che servono a gestirne il flusso (linee di* ***controllo****). La figura seguente rappresenta questo flusso di informazioni sui bus.*

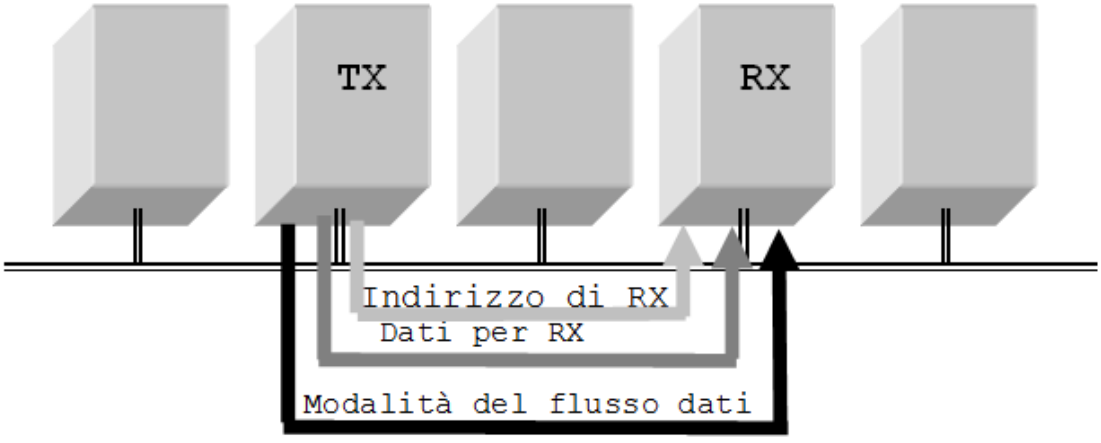

Figura 2. 5 Il flusso di segnali sul bus

Il bus è formato da più linee e queste possono essere suddivise in base alla loro funzione:

Suddivisione funzionale del bus

- ***ADDRESS BUS*** *(**AB**)* *la parte del bus che trasporta gli indirizzi*
- ***DATA BUS*** *(**DB**)* *la parte del bus che trasporta dati/istruzioni*
- ***CONTROL BUS*** *(**CB**)* *la parte del bus che governa il trasferimento*

Poiché ogni singola linea del bus può portare un solo bit di informazione (0 oppure 1) è normale che ogni singolo bus abbia più linee[4]; altrimenti avremmo a disposizione solo due indirizzi, o solo due dati o due soli controlli. Il numero di linee dell'Address Bus, del Data Bus e del Control Bus dipende dal μP.

Abbiamo visto in precedenza il problema del bus contention; a tale proposito nasce spontanea una domanda: chi decide chi deve trasmettere? Più in generale: chi governa il flusso delle informazioni sui bus? Ci vuole un arbitro, un componente che abbia il ruolo e la facoltà di pilotare opportunamente i bus. Questo componente prende il nome di "Master" (padrone) e deve essere unico, altrimenti subentrano ulteriori conflitti di competenza; tutti gli altri dispositivi saranno "Slave" (schiavi) e subiranno le decisioni del Master (figura seguente).

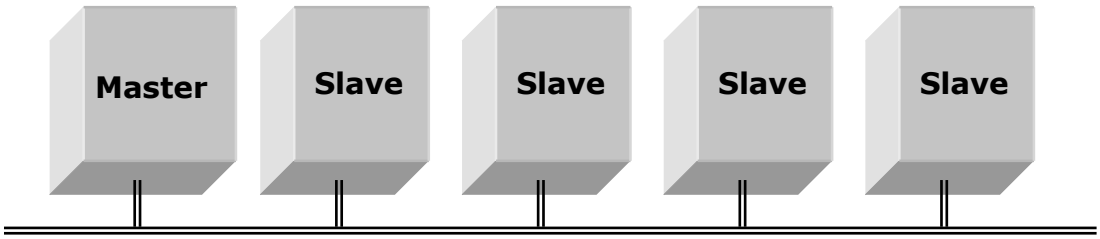

Figura 2. 6 Architettura Master-Slave.

Esaminiamo più in dettaglio le funzioni del Master tenendo presente che può avvenire una sola comunicazione alla volta. Abbiamo un TX ed un RX con indirizzi noti; ma il bus indirizzi è uno solo: come possono viaggiare due indirizzi contemporaneamente sullo stesso bus? Ecco una prima soluzione: il Master è sempre uno dei due interlocutori (o il TX o il RX) e quindi serve solo l'indirizzo dell'altro componente. Analogia col sistema postale: il mittente (uno Slave) spedisce una lettera ad un destinatario (anch'esso Slave); questa lettera passerà comunque dalla Centrale Postale (Master): il primo trasferimento è tra mittente e Centrale (e il Master in questo caso riceve), il secondo è tra Centrale e destinatario (il Master trasmette): si scrive forse sulla busta l'indirizzo della Centrale?

Se volete scrivere due righe al vostro vicino di casa il percorso che seguirà la lettera passerà comunque dalla Centrale. Così è anche nei sistemi a μP; nella figura seguente viene mostrato il flusso dei dati per scambiare informazioni tra due Slave.

[4] *Oggi sono in commercio bus seriali; le informazioni sono veicolate su un'unica linea. Ciò comporta che l'informazione venga inviata un bit per volta lungo questa linea. L'USB – come dice il nome - è un bus seriale.*

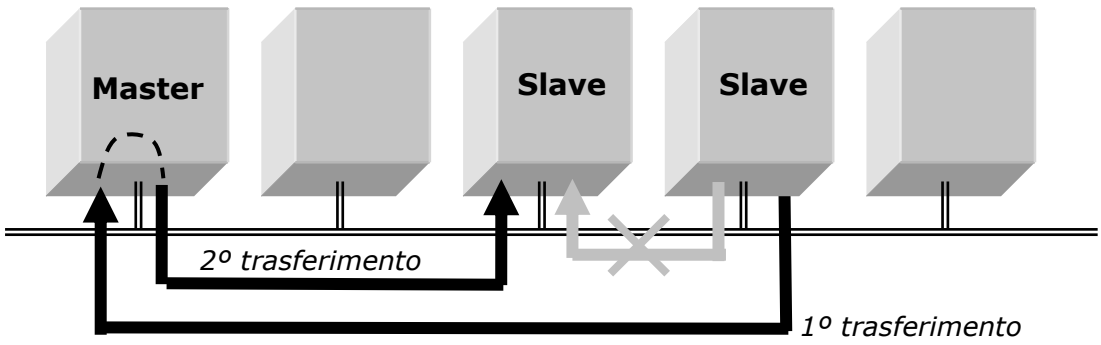

Figura 2. 7 Flusso delle informazioni per lo scambio di dati tra due Slave.

Il Master deciderà di volta in volta il dispositivo con cui comunicare ponendo il suo indirizzo sull'Address Bus, quindi tale bus è unidirezionale, dal Master agli Slave. Il Data Bus è invece bidirezionale; a volte i segnali arriveranno al µP, altre volte ne usciranno (in un paragone stradale si chiamerebbe senso unico alternato). Il Control Bus è linea per linea unidirezionale cioè alcune linee sono solo in uscita al µP mentre le altre sono in ingresso; studieremo nel dettaglio i compiti del Control Bus più avanti.

Se lo scambio di informazioni è tra Slave e Master ci sarà evidentemente un solo trasferimento sul Data Bus.

Come avrete già intuito nei µC il ruolo di Master viene assunto dal µP!

Un terzo livello: il 3-state

Le linee bidirezionali del Data Bus saranno occupate a volte nella direzione Master→Slave, a volte nella direzione Slave→Master; ma ci saranno dei momenti che non saranno occupate da nessuno, per esempio nei tempi morti tra un trasferimento e il successivo. In questi casi le linee del Data Bus non sono ne' a livello logico 0 ne' a livello logico 1; dal punto di vista elettrico si comportano come se fossero staccate dai dispositivi che si affacciano sul bus e si dice che si trovano in ***three-state*** *(siglato anche 3-state o tri-state, oppure Hi-Z ad indicare High Impedance, cioè alta impedenza) cioè in un "terzo stato", intendendo che i primi due sono lo 0 e l'1. Quando il Data Bus è in 3-state nessun TX impegna il bus; bisogna quindi che tutti i TX sappiano fare tre cose: imporre uno 0, imporre un 1, astenersi dall'imporre 0/1 mettendo le proprie uscite in 3-state.*

Il 3-state è compatibile con gli altri valori logici; vediamo meglio il senso di questa frase con il prossimo argomento.

Cosa succede al Data Bus durante una comunicazione

Cosa succede ai dispositivi attaccati al Data Bus durante una comunicazione? Quando c'è un trasferimento TX→RX, un solo TX impone uno stato logico (0 oppure 1); tutti gli altri TX si astengono dal comunicare, per non creare bus-contention, e quindi sono in 3-state; tutti gli RX sono in ascolto del bus ma solo uno è interessato a raccogliere il dato dalle linee del Data Bus (figura successiva).

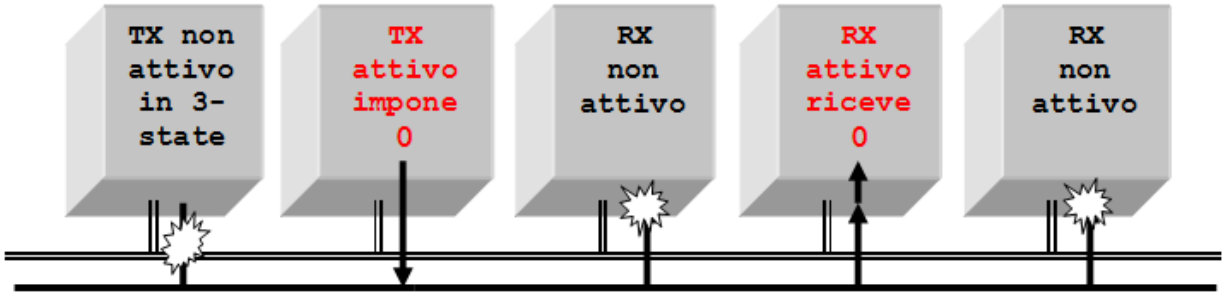

Figura 2. 8: Uno "0" viaggia su una linea del Data Bus

L'uscita dei TX non attivi è in 3-state e quindi il bus può assumere il valore imposto dal TX attivo, senza creare alcun bus contention.

Le memorie nei µC

Il µP-Master dovrà essere istruito con un opportuno programma (SW) sui trasferimenti o sulle elaborazioni di informazioni. Questi programmi si possono trovare su hard disk, CD-ROM o altro dispositivo; questi dispositivi sono periferiche dette ***memorie di massa*** *perché hanno il compito di memorizzare, cioè di contenere, i programmi (e anche i dati quando serve). In realtà, per poter essere eseguiti dal µP, i programmi devono essere trasportati dentro il sistema a µP ed introdotti nella* ***memoria centrale****, ospitata in dispositivi che prendono il nome di* ***RAM*** *(Random Access Memory) e* ***ROM*** *(Read Only Memory). La ROM è una memoria particolare che serve per le prime istruzioni di avvio del computer (da qualche parte il µP deve pur incominciare...); per questo motivo la ROM mantiene le istruzioni al suo interno anche dopo lo spegnimento del computer. La memoria RAM invece perde tutto il suo contenuto allo spegnimento della macchina e pertanto è anche definita memoria "****volatile****"; la RAM è memoria a lettura/scrittura, mentre la ROM (come suggerisce il nome) è a sola lettura..*

Nella RAM il µP va a leggere le istruzioni ma può anche leggere o scrivere dati; dalla ROM generalmente legge istruzioni (più raramente dati). Possiamo ora aggiungere qualche dettaglio allo schema del µC (figura successiva).

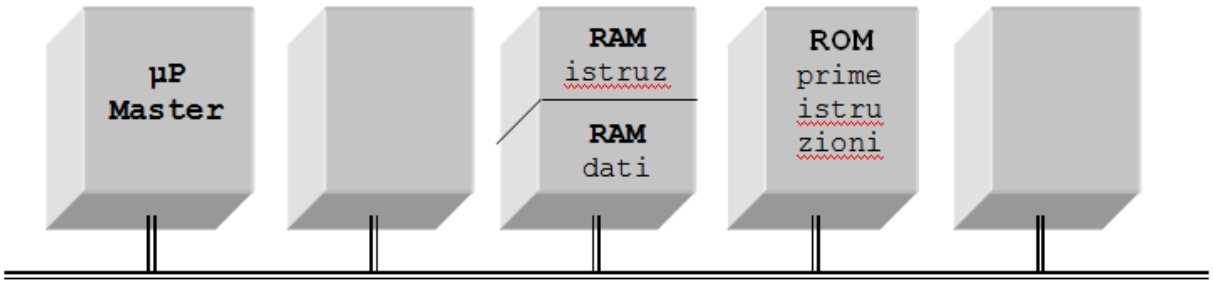

Figura 2. 9 Uno schema più dettagliato del µC

Dati e istruzioni

Le istruzioni ed i dati di cui stiamo parlando viaggiano entrambi sul Data Bus; ne discutiamo brevemente la differenza. Possiamo prendere, come esempio, la comunicazione tra un

direttore d'ufficio ed un segretario, una ipotetica mattinata quando il principale consegna il lavoro da svolgere al suo dipendente lasciando sulla sua scrivania due fogli (figura successiva).

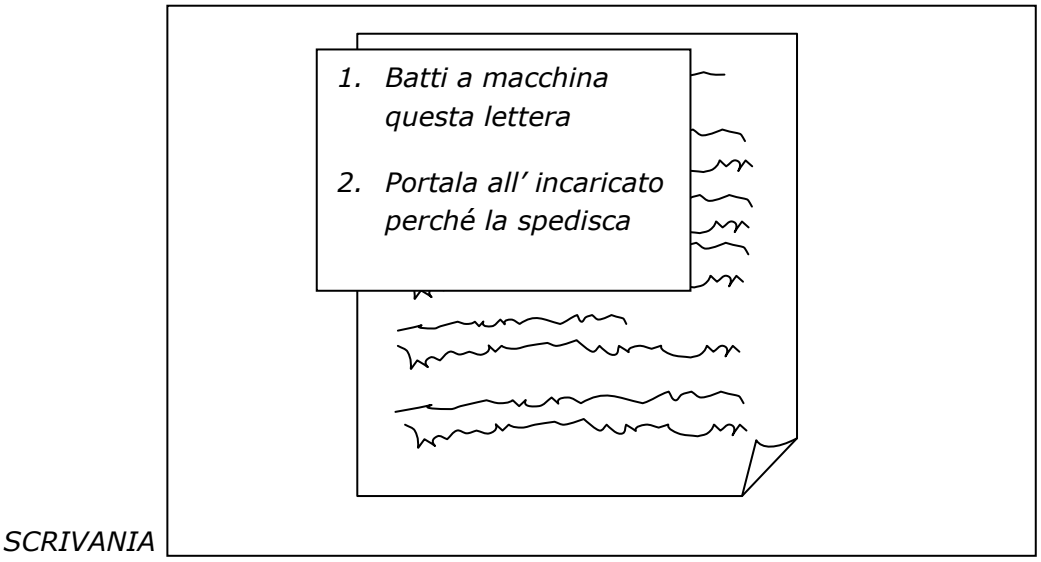

Figura 2. 10 Il lavoro del segretario in un ufficio

Il primo foglio contiene in questo caso due istruzioni (il programma di lavoro), l'altro foglio contiene informazioni (i dati) da elaborare in base alle istruzioni; entrambi i fogli sono stati depositati sulla scrivania e identico è stato il canale di comunicazione (non sono stati consegnati in modo differente). Avviene qualcosa di simile in un sistema a µP. La scrivania è la RAM, che contiene dati ed istruzioni, entrambi veicolati attraverso il data bus.

Alla domanda: "Volendo mantenere il paragone, nel sistema a µP chi fa la funzione del direttore?" alcuni potrebbero rispondere: il µP; invece no! Il µP, essendo un lettore-esecutore di istruzioni e un manipolatore di dati, è il segretario! Chi ricopre il ruolo del direttore, di colui che scrive le istruzioni? Il programmatore, ovvero quel professionista che ha imparato ad istruire il computer.

Il µP è un esecutore di istruzioni

Il concetto di µP come esecutore di istruzioni è molto importante: bisogna infatti sfatare un luogo comune che vede i µP e i computer come entità capaci di fare tutto. Sono sempre circuiti elettrici e, per quanto "intelligenti" e molto veloci, sono in realtà molto rigidi: bisogna seguire rigorosamente il loro linguaggio e la minima imperfezione o approssimazione nell'impartirlo ha conseguenze assai negative: il µP o non capisce o capisce male e di conseguenza fa cose non previste o dannose. La programmazione dei µP è difficoltosa proprio per questo: è difficile parlare con una macchina! E non c'è modo di evitare questa fatica. Essendo molto rigidi e meccanici nel loro ruolo di esecutori, i µP sono di fatto abbastanza stupidi; l'intelligenza è solo del programmatore, che è un uomo! In conclusione: il µP non "capisce" nulla, solo "esegue".

A questo punto abbiamo definito quasi tutti i blocchi del µC. Mancano solo i componenti che hanno la funzione di ***interfaccia***[5] *con le periferiche (figura successiva).*

[5] *Interfacciare due dispositivi significa trovare un modo affinché possano comunicare correttamente, senza errori di tipo elettrico o logico. Il problema dell'interfacciamento sorge spontaneamente ogni volta che si devono collegare dispositivi differenti (per esempio: una porta logica ed un LED).*

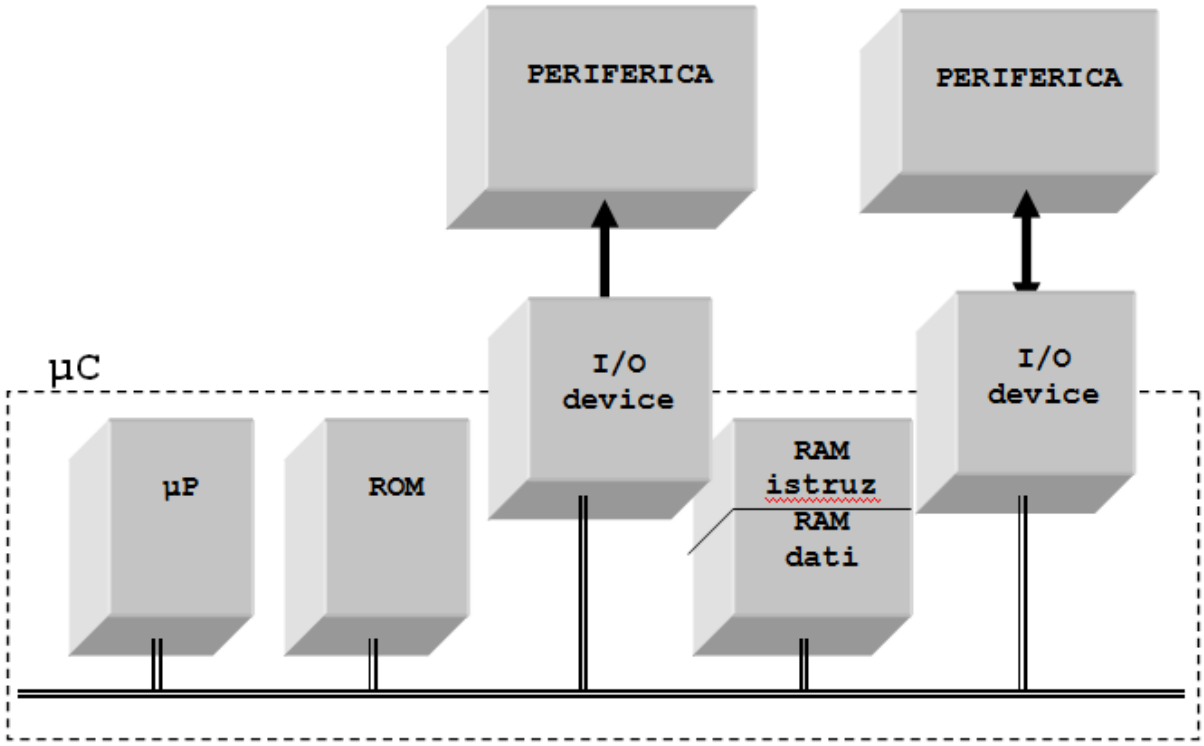

Figura 2. 11 Schema a blocchi dettagliato di un µC

*Tali componenti prendono il nome di **I/O device** (dispositivi di input/output) e sono ora stati disegnati sulla periferia del µC poiché, pur essendo saldati sulla piastra del µC, i loro circuiti verso la periferica non sono sempre digitali e comunque risentono della struttura del componente al quale si devono **interfacciare**.*

Un altro schema, perfettamente equivalente al precedente e spesso usato, è riportato nella figura successiva.

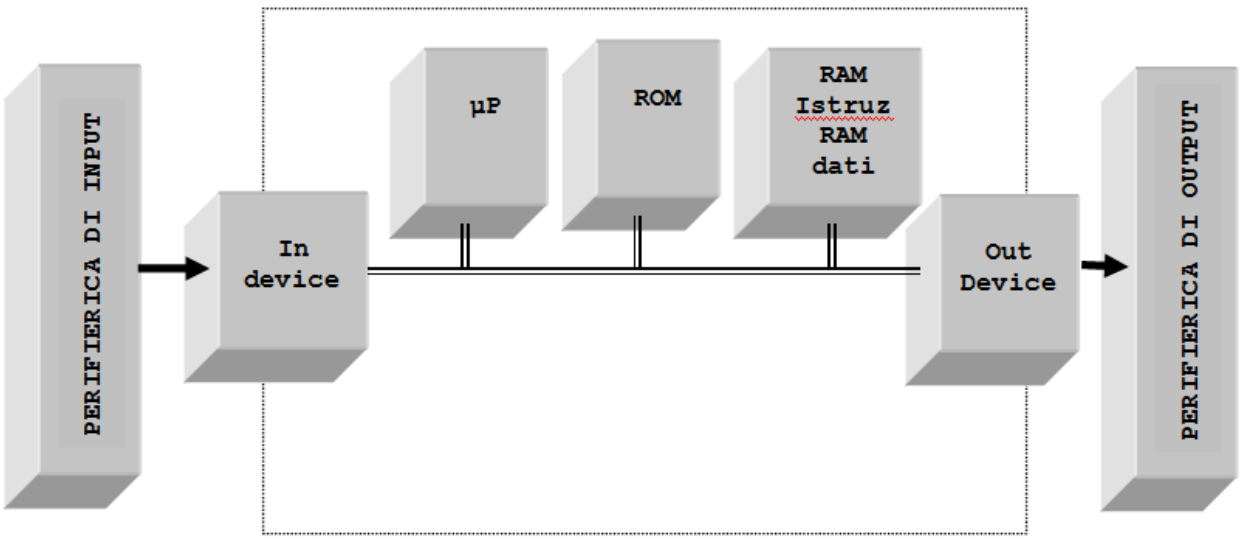

Figura 2. 12 Un altro schema a blocchi dettagliato del µC

La periferica standard di input è una tastiera (o un tastierino numerico o anche una qualunque pulsantiera), mentre la periferica standard di output è un display (uno schermo o una serie di display a 7 segmenti o un display LCD). Nello schema di prima possono ovviamente trovare posto più periferiche; come regola generale ogni periferica ha il proprio I/O device[6]*. Gli I/O device più semplici sono i* ***buffer*** *(se connessi ad una periferica di input) e i* ***latch*** *(se connessi ad una periferica di output). Per motivi che a questo punto dovrebbero essere chiari gli I/O device di ingresso devono avere uscite 3-state.*

Avendo visto la struttura essenziale di un μC è arrivato il momento di indagare più approfonditamente il funzionamento del μP.

DOMANDE:

- *Quali sono gli elementi essenziali in una architettura a bus?*
- *Quali sono i motivi per cui due TX non possono trasmettere contemporaneamente?*
- *Quali sono le tre funzioni principali del bus?*
- *Qual è la funzione del Master in un μC?*
- *Chi svolge la funzione di Master nel μC?*
- *Qual è lo scopo del 3-state nella gestione di un bus?*
- *Qual è la funzione della RAM e della ROM in un μC?*
- *Quali sono i possibili percorsi di un'istruzione all'interno di un μC? e di un dato?*
- *Qual è la funzione di un I/O device?*
- *Quante periferiche sono collegate ad un I/O device?*
- *Quali sono gli I/O device più semplici?*

ESERCIZI DI RIEPILOGO

1. *Sino a che punto si può spingere l'analogia tra sistema postale e sistema a μP? Esiste analogia per il francobollo? Chi fa il ruolo del postino nel μC? Perché nel sistema postale non è necessario l'indirizzo del mittente? Succede lo stesso nei μC?*
2. *Quanti trasferimenti sul bus sono necessari affinché un'informazione sia mandata da uno slave ad altri due slave?*
3. *Quando digitate un tasto sulla tastiera del Personal Computer lo vedete apparire immediatamente sul video. Disegnate il flusso dell'informazione tra tastiera e video supponendo che queste siano periferiche semplici. Quanti trasferimenti sono necessari sul bus?*

[6] *Quando la periferica è molto complessa (ad esempio il video) questi dispositivi di interfaccia sono a loro volta abbastanza complessi e prendono comunemente il nome di "adattatori" (inglese: adapter); un esempio nel Personal Computer sono gli adattatori video, meglio conosciuti col nome di schede-video.*

4. *Un programma contiene le seguenti istruzioni: prendere un dato dal dispositivo di input, moltiplicarlo per un numero che si trova in memoria e quindi inviarlo al dispositivo di output. Quanti trasferimenti sono necessari?*

PAROLE CHIAVE

***Address Bus (AB)**: è il bus che trasporta l'indirizzo dell'interlocutore del µP.*

***Architettura a bus**: modo di organizzare il collegamento tra più dispositivi. Richiede l'esistenza di un bus collegato a tutti i dispositivi. Prevede inoltre la presenza di un arbitro (o Master) che regoli il flusso tra chi trasmette e chi riceve le informazioni.*

***Buffer**: componente che presenta in generale 8 ingressi e 8 uscite; funziona come una porta di accesso: quando è abilitato fa passare le informazioni. Poiché è utilizzato in circuiti che si inseriscono nel bus è quasi sempre dotato della funzionalità 3-state e prende il nome di buffer 3-state.*

***Bus**: Insieme di segnali che connettono più dispositivi tra loro. In generale è costituito da più fili elettrici che scorrono in parallelo.*

***Bus contention**: (contesa del bus) situazione di conflitto che si viene a creare quando due o più trasmettitori vogliono comunicare contemporaneamente. Rischia di provocare un corto-circuito con danni irreparabili agli stadi di uscita dei trasmettitori e, se permane, a tutto il sistema.*

***Comunicazione**: scambio di informazioni tra due entità*

***Control Bus (CB)**: è il bus che trasporta i segnali di controllo del flusso di informazioni all'interno del µC.*

***CPU (Central Processing Unit)**: è il "cervello" di un computer, la parte che esegue le istruzioni e controlla il flusso delle informazioni; in generale consiste di più circuiti.*

***Data Bus (DB)**: è il bus che trasporta i dati tra i dispositivi di un µC.*

***Hardware (HW)**: la parte materiale, tangibile di un sistema elettronico, e cioè fili, componenti elettronici in generale (resistenze, condensatori ecc.) e circuiti integrati.*

***HW programmabile**: circuito che per funzionare richiede delle istruzioni*

***HW configurabile**: circuito che, mediante opportuni comandi, può predisporsi a funzionare in più modi. Spesso detto impropriamente "HW programmabile" si differenzia da esso perché non può eseguire istruzioni*

***I/O device**: dispositivi che permettono la comunicazione tra mondo esterno e bus del µC; funzionano come porte di accesso.*

***Interfaccia**: tutto ciò che collega due entità; in elettronica indica un circuito che permette un corretto collegamento fisico e logico tra due dispositivi.*

***Interfacciare**: collegare due dispositivi in modo tale che lo scambio di informazioni avvenga correttamente.*

***Latch**: dispositivo che memorizza dati. In generale indica un registro che mantiene sulle sue uscite un byte in modo stabile, finché non se ne memorizza un altro. È spesso utilizzato come porta di uscita per inviare all'esterno le informazioni.*

***Memoria centrale**: è la memoria RAM e ROM all'interno del µC.*

***Memoria di massa**: sono memorie esterne al µC. Termine generico per indicare hard disk, floppy disk, CD-ROM, nastri magnetici.*

***MicroComputer (µC)**: termine alternativo a "sistema a microprocessore".*

***Microprocessore (µP)**: circuito integrato che svolge le funzioni di una CPU con un unico dispositivo; è l'HW programmabile per eccellenza e costituisce la base per lo sviluppo di sistemi programmabili, come per esempio i personal computer. Esegue istruzioni in modo meccanico ma molto veloce. È l'arbitro della comunicazione all'interno di un µC.*

***Microcontrollore (µC)**: è un microcomputer embedded, ovvero realizzato su un unico circuito integrato; particolarmente efficiente per quanto riguarda le dimensioni è diffuso in tutti i campi dell'attività umana grazie alla flessibilità di impiego.*

***Periferiche**: componenti di supporto ad un computer, come video, tastiera, stampante, hard-disk, floppy disk, CD-ROM, mouse, modem, speaker, casse ecc., esterni al sistema a microprocessore.*

***Personal Computer (PC)**: è una macchina programmabile dotata di tastiera, video, mouse, dischi fissi o rimovibili e di un circuito che li gestisce; può svolgere molteplici attività che vanno dai calcoli alla videoscrittura, dai giochi al disegno tecnico.*

***Programma**: indica la sequenza di istruzioni che vengono impartite ad un circuito elettrico programmabile (o HW programmabile); sinonimo di Software.*

***RAM**: Random Access Memory (memoria ad accesso casuale); dispositivo destinato a contenere istruzioni e dati durante il normale funzionamento del µP. È definita "volatile" perché allo spegnimento della macchina perde il suo contenuto.*

***ROM**: Read Only Memory (memoria di sola lettura), una memoria che conserva le informazioni in essa contenute (istruzioni o dati) anche senza l'alimentazione. È anch'essa una memoria ad accesso casuale.*

***Sistema a microprocessore**: detto anche microComputer è l'insieme dei circuiti elettrici che permettono ad un µP di leggere ed eseguire istruzioni. È costituito in generale da µP, RAM, ROM, I/O device e altri circuiti come decoder, porte logiche, ecc.*

***Software (SW)**: è sinonimo di programma e indica le istruzioni che vengono impartite ad un circuito elettrico programmabile.*

***Three-state**: è lo stato di una linea elettrica digitale quando non vale né 1 né 0; è l'uscita di un TX quando non vuole interferire con il bus. Si scrive anche tri-state, 3-state o Hi-Z che significa alta impedenza (High Impedance). Alta impedenza, in elettronica, indica una situazione paragonabile ad un "filo staccato", ovvero impossibilità di imporre un segnale: ecco perché tale caratteristica è impiegata nei µC.*

3. Struttura essenziale di un semplice µP

Chi ha pratica con un linguaggio di programmazione cosiddetto ad alto livello, come il C, ha già un'idea abbastanza chiara di quello che può fare un µP. Infatti se esaminate un listato di un programma troverete sostanzialmente queste cose:

Cosa sa fare un µP

1. *Calcoli:*
   - *matematici (es.: 4*5-8/11)*
   - *logici (es.: Vero AND Falso OR Vero)*
2. *Trasferimenti di dati:*
   - *da/verso la memoria*
   - *acquisizione da un dispositivo di input (es.: tastiera)*
   - *invio a un dispositivo di output (es.: video)*
3. *Controllo del flusso del programma*

Inoltre bisogna considerare un'altra azione che è sottintesa ma è primaria, ovvero:

4. *interpretare ed eseguire le istruzioni*

Queste sono le operazioni fondamentali che sa fare il µP; ci sono anche altre operazioni più sofisticate ma per ora ci dedicheremo soltanto a queste; ribadiamo che il µP non comunica direttamente con la tastiera o il video ma attraverso gli I/O device ad essi collegati.

Uno schema a blocchi di massima dell'interno del µP è riportato nella figura 2.13.

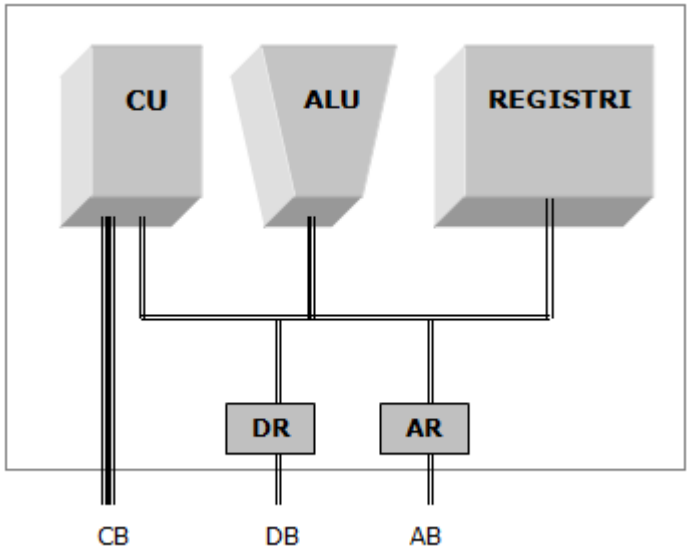

Figura 3. 13 Un primo schema a blocchi del µP

CU, ALU e Registri

La prima cosa che si nota immediatamente è la architettura interna: ancora una architettura a bus! Quindi anche internamente al µP ci vorrà un Master e infatti questa funzione è svolta dal blocco ***CU*** *(Control Unit) che, come suggerisce il nome, controlla il flusso delle informazioni all'interno del µP. Il blocco* ***Registri*** *è una piccola memoria interna (una piccola RAM) per gli usi interni del µP: per esempio è qui che vengono depositati i dati in transito nei trasferimenti da uno Slave all'altro. [Se manteniamo l'analogia con il "segretario" vista precedentemente le memorie di massa (hard-disk, CD-ROM ecc.) fanno la parte dell'archivio dell'ufficio, le memorie RAM e ROM rappresentano la scrivania con fogli e pratiche, i registri rappresentano invece la sua memoria personale del segretario]. La* ***ALU*** *(Arithmetic-Logic Unit) è il blocco che esegue tutte le operazioni aritmetiche e logiche, praticamente una piccola semplice calcolatrice. I blocchi alla periferia del µP servono per interfacciare il bus interno del µP con i bus del µC; l'AR (Address Register) è collegato con l'Address Bus mentre il DR (Data Register) è collegato al Data Bus.*

Vediamo ora nel dettaglio il funzionamento di questi blocchi affrontando una per una le 6 operazioni fondamentali prima descritte. D'ora in avanti vi accorgerete che sarà difficile spiegare un elemento HW senza ricorrere al SW che lo gestisce o descrivere un'istruzione (SW) senza spiegarne gli effetti sui segnali (HW) che coinvolge.

Per operare il µP deve innanzitutto essere istruito sui compiti da svolgere; facciamo quindi un esempio con una sequenza di istruzioni, cioè un programma.

> *Acquisire un numero dall'I/O device di indirizzo…*
> *Acquisire un altro numero dall'I/O device di indirizzo…*
> *Sottrarre i due numeri*
> *Inviare il risultato precedente all'I/O device di indirizzo…*
> ***Programma 3. 1***

É stato necessario specificare l'indirizzo degli I/O device coinvolti; il µP può distinguere tra i tanti slave coi quali deve può comunicare solo grazie al loro indirizzo.

Struttura di un'istruzione

Questo programma consta di alcune istruzioni e si nota che ogni istruzione è un ordine impartito al µP; assume sempre una forma del tipo:

AZIONE OGGETTO1 (**OGGETTO2**)

Questa articolazione di istruzioni come "azioni su oggetti" è fondamentale ed è tipica di molti linguaggi di programmazione. Se ricordate l'esempio del direttore e del segretario noterete che la sintassi degli ordini è più o meno la stessa.

Fasi di un'istruzione

Un altro concetto molto importante è che, esattamente come succedeva al segretario così spesso invocato, le istruzioni devono essere prelevate e lette prima di essere eseguite; quindi per ogni istruzione ci sono fasi successive:

PRELIEVO → LETTURA → ESECUZIONE

*La fase di prelievo è meglio nota col nome inglese di **fetch** (prelievo), la seconda fase prende il nome tecnico di **decode**, la terza di **execute**.*

Come viene eseguito un programma

*Le istruzioni risiedono normalmente nella RAM[7]. Quando vengono lette il µP diventa un RX e la RAM assume il ruolo di TX; il µP metterà sull'Address Bus l'indirizzo dell'istruzione e raccoglierà la stessa dal Data Bus; il Control Bus servirà a gestire tale trasferimento. Il posto preciso dove viene temporaneamente messa l'istruzione è il registro IR, **Instruction Register** (figura successiva). Una volta all'interno del µP l'istruzione viene "decodificata" cioè letta e interpretata dalla Control Unit; come il µP è il cervello del µC così la CU è il cervello del µP.*

Una volta letta e decodificata un'istruzione il µP dovrà "eseguirla"; nel caso della prima istruzione ("acquisire un numero dall'I/O device di indirizzo…") in qualità di RX dovrà andare a leggere dal dispositivo collegato alla periferica mettendone l'indirizzo sull'Address Bus e raccogliendo poi il dato dal Data Bus.

Un registro particolare: l'accumulatore

Il dato raccolto dal Data Bus sarà memorizzato all'interno del µP nella sua piccola memoria interna, ovvero in uno dei suoi Registri. Per i trasferimenti c'è un registro principale che viene quasi sempre coinvolto e che ha il nome di "accumulatore": è qui che si trova il dato alla fine della prima istruzione (figura seguente).

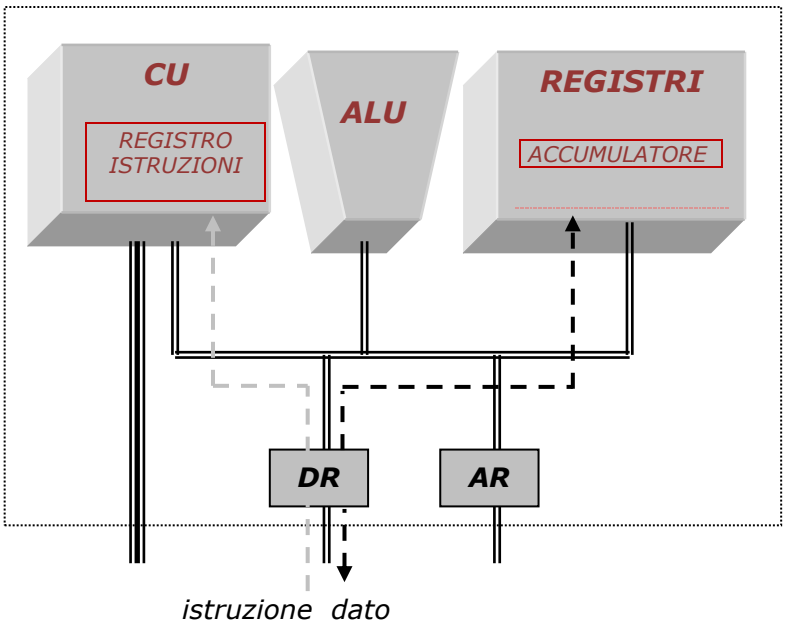

Figura 3. 14 Flusso dei dati e delle istruzioni all'interno del µP; notare le frecce

Passiamo alla seconda riga del programma: c'è un'altra istruzione di lettura dall'input. Il meccanismo per il fetch di questa istruzione è esattamente identico al caso precedente ed è comune a tutte le possibili istruzioni, quindi non lo ripetiamo. In questa istruzione viene ordinata la lettura di un altro numero; la destinazione di questo dato è ancora l'accumulatore e ma così prenderà il posto del numero precedente: c'è qualcosa che non va! Si desidera evidentemente prendere il secondo dato senza perdere il primo. Il problema è tipico quando si

[7] *Le istruzioni di avvio del sistema stanno invece nella ROM; all'accensione infatti la RAM è vuota.*

programma un μP ma la soluzione non è difficile: prima di prelevare il secondo dato bisogna spostare il primo da qualche altra parte, tipicamente in un altro dei registri interni del μP.

Il programma corretto è il seguente:

```
Acquisire un numero dall'I/O device di indirizzo…
Depositare tale numero in un registro del μP    *
Acquisire un altro numero dall'I/O device di indirizzo…
Sottrarre i due numeri
Inviare il risultato precedente all'I/O device di indirizzo…
```

Programma 3. 2

Rispetto al programma precedente è stata introdotta la riga evidenziata con un asterisco. Il fetch dell'istruzione avviene come al solito. Si ordina di depositare il dato in un altro registro. Questa operazione è interna al μP e pertanto non coinvolgerà ne' Address Bus, ne' Data Bus, ne' Control Bus; è pertanto un'operazione molto veloce perché non ci sono trasferimenti sui bus; però, essendo il numero dei registri limitato, bisognerà garantirsi che ce ne sia uno libero.

Trasferimento di un dato

Chiariamo meglio cosa vuol dire trasferire un dato da un posto (locazione di memoria o registro) all'altro prendendo come esempio l'istruzione precedente e la figura che segue:

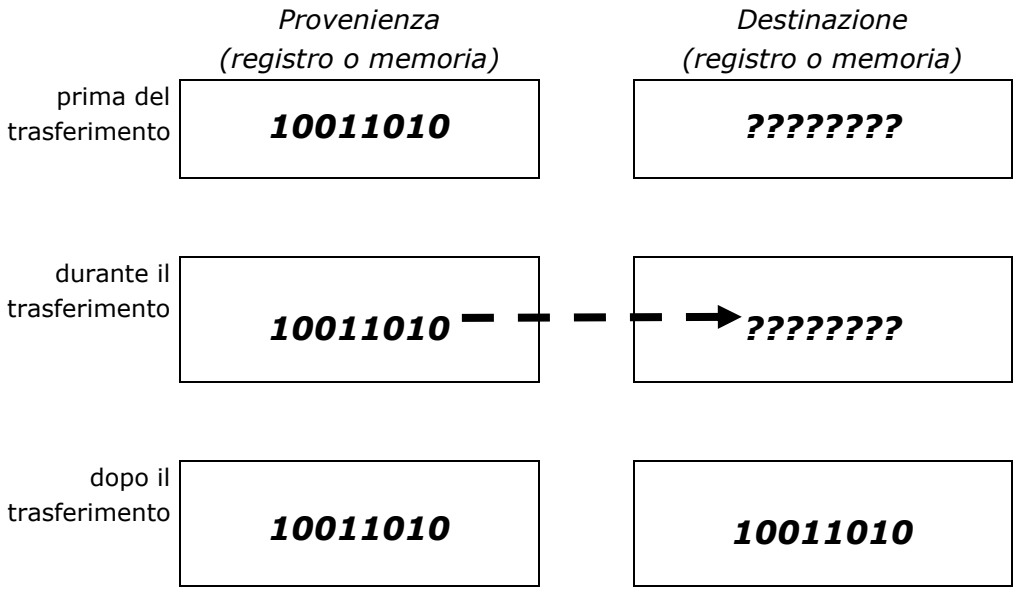

Figura 3. 15 Come viene trasferito un dato; è una copia!

Potete notare che alla fine il dato è rimasto anche nella locazione di provenienza! Ciò ha una spiegazione semplice; siamo di fronte a circuiti elettrici che contengono sempre dei valori in tensione (o 0 Volt o 5 Volt) e quindi non ha senso "far sparire" un dato; qualcosa resta sempre. Per esempio nella locazione di destinazione sono stati posti all'inizio dei "?" perché non si sa a priori che cosa contenga; però qualcosa è sicuramente presente! E l'unico modo per

"cancellare" un dato è quello di sovrascriverlo[8], proprio come, nell'esempio appena fatto, è accaduto alla locazione di destinazione. Invece di trasferimento sarebbe pertanto più corretto parlare di "copia", ma nel linguaggio comune dei programmatori si usano entrambi i termini.

Esaminiamo la terza riga del programma: essa è identica alla prima, dopo il fetch sarà eseguita esattamente nello stesso modo.

La quarta riga del programma è:

Sottrarre i due numeri

e questa operazione coinvolge direttamente la ALU, la calcolatrice interna al µP. La ALU esegue calcoli aritmetici e logici con un operando (cambiandone il segno o facendo una NOT) oppure con due (facendo somme, sottrazioni, OR, AND ecc.); gli operandi si trovano, in linea di massima, all'interno del µP; tutto il gioco è interno al µP e quindi avviene molto rapidamente.

Accumulatore: registro privilegiato

Il risultato dell'operazione è sempre depositato nell'accumulatore, e questa è un'altra caratteristica che ne fa il registro più importante in assoluto. Inoltre tale registro costituisce sempre uno degli operandi su cui agisce la ALU. Uno schema abbastanza generale è riportato nella figura seguente.

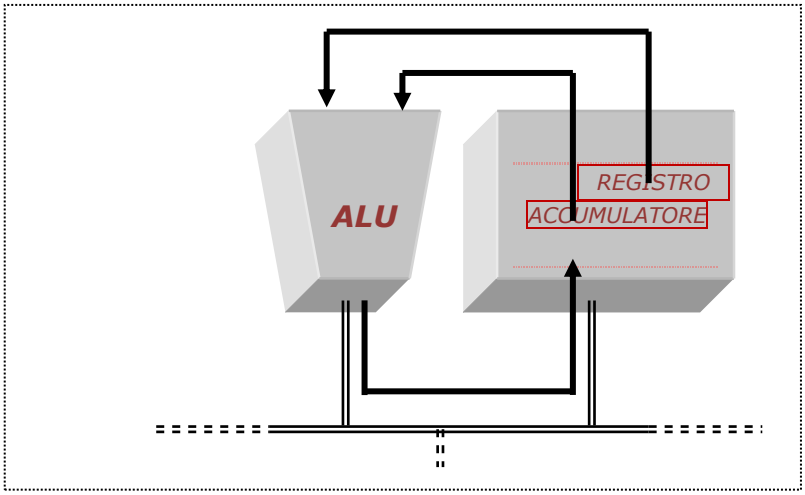

Figura 3. 16 La ALU in funzione con due operandi

Si nota che l'accumulatore è sia operando che risultato…

Modi di indirizzamento di un µP

Dobbiamo ora correggere leggermente, nel linguaggio, l'istruzione appena descritta; ciò ci introdurrà in un altro aspetto fondamentale dei µP.

Il µP è estremamente rigido nell'interpretare le istruzioni e pertanto va istruito con precisione assoluta. Una espressione del tipo:

Sottrarre i due numeri

[8] *Se volete veramente cancellare qualcosa dal vostro computer non crediate che metterlo nel cestino sia sufficiente; ci vogliono programmi appositi per cancellare definitivamente i vostri file.*

è ambigua: quali numeri? Il μP non è così furbo da capire che state parlando degli ultimi due numeri acquisiti; e comunque, anche dal punto di vista logico, questo riferimento ai due numeri rimane un po' vago. È necessario essere estremamente precisi e infatti è regola generale che un'istruzione opera su oggetti che devono essere specificati senza ambiguità, indicandone la posizione*. Questa posizione può essere specificata in più modi; per esempio, per la memoria si fornisce l'indirizzo; per un registro interno al μP bisognerà indicarne il nome; per un I/O device si fornisce l'indirizzo*[9]*. In termini tecnici queste diverse possibilità di accedere all'informazione vengono chiamate* ***modi di indirizzamento*** *e variano da un μP all'altro. Vedremo successivamente come deve essere formulata l'istruzione affinché sia accettata dal μP.*

Rimane l'ultima riga del programma:

> *Inviare il risultato precedente all'I/O device di indirizzo….*

L'istruzione viene prelevata (fetch) e decodificata. Il dato da inviare si trova nell'accumulatore; l'indirizzo dell'I/O device viene posto sull'Address Bus, il contenuto dell'accumulatore viene posto sul Data Bus ed il Control Bus sarà pilotato dal μP affinché il trasferimento risulti corretto. È lo stesso percorso, in senso inverso, già visto con l'input.

Per studiare in maggior dettaglio il μP è opportuno approfondire la conoscenza degli altri componenti del μC; memorie RAM, ROM e I/O device sono nati prima del μP ed esistono anche al di fuori dei μC; la loro struttura ha guidato ed influenzato il progetto dei μP. Per tutte queste ragioni i prossimi paragrafi tratteranno di questi dispositivi.

DOMANDE:

- *Quali sono le funzioni della CU, della ALU e dei Registri?*
- *Qual è la struttura essenziale di un'istruzione del μP?*
- *Quali sono le fasi di un'istruzione?*
- *Cosa significa interpretare un'istruzione?*
- *Quali sono le funzioni principali del registro accumulatore?*
- *Qual è lo scopo del registro IR?*
- *Quali sono i possibili percorsi di un'istruzione e di un dato all'interno del μP e del μC?*
- *Perché è più corretto parlare di copia di un dato invece che di trasferimento?*
- *Si può cancellare un dato in un registro? E in una locazione?*
- *Qual è il rapporto tra ALU e accumulatore?*

4. Bit, Byte e Word

Introduciamo in questo paragrafo alcune parole usate frequentemente nell'ambito dell'elettronica digitale.

[9] *si può specificare un indirizzo anche in maniera indiretta, indicando il registro che lo contiene*

Bit (binary digit) è una sigla che significa cifra binaria ed è l'unità elementare di informazione; quindi ogni elemento in un μC contiene, trasporta o elabora uno o più bit. Ogni bit può valere 0 oppure 1 e, elettronicamente, viene realizzato associando 0Volt alla cifra 0 e 5Volt alla cifra 1. Il motivo per cui si usano i bit è che i circuiti digitali costituenti le reti combinatorie e sequenziali (tra cui μP e μC) elaborano i segnali secondo le regole dell'algebra di Boole; questa algebra opera su variabili a due stati, 0 e 1, e utilizza operatori come AND, OR, NOT, che vengono chiamati "operatori logici" (o "booleani"); i circuiti con AND, OR, NOT vengono detti "circuiti logici" (o digitale). Ci si può chiedere come possa un circuito logico realizzare calcoli aritmetici; l'Half Adder, il Full Adder e i comparatori digitali sono esempi di circuiti che, utilizzando l'algebra di Boole, fanno calcoli di algebra tradizionale! Tali circuiti sono presenti nelle calcolatrici e ovviamente anche la ALU ne contiene molti al suo interno.

Ovviamente i risultati dei circuiti digitali sono espressi in forma di bit. Come è possibile passare alla rappresentazione tradizionale? Noi facciamo i calcoli rappresentando i numeri sempre in base 10 cioè utilizzando le cifre da 0 a 9, ma questa è una convenzione (e viene dal fatto che abbiamo dieci dita); nulla vieta di rappresentare i numeri in base 2 utilizzando le cifre 0 e 1 (⇒ Scheda 2.1 sulla notazione binaria)

Byte = 8 bit

*Il **byte** non è altro che un insieme di 8 bit; ne vediamo un esempio nella figura 2.17a. Questi 8 bit possono essere interpretati come otto informazioni digitali separate oppure come un numero naturale espresso in base 2: sta al programmatore adottare il punto di vista che gli conviene. $B_7...B_0$ è una convenzione per numerare i bit dal più significativo al meno significativo.*

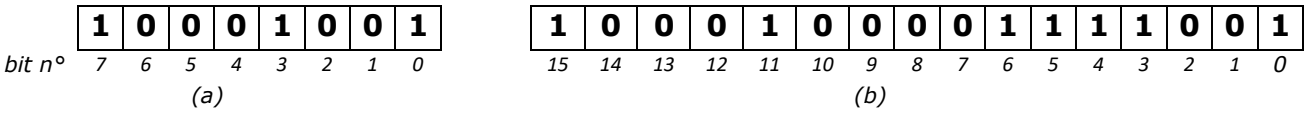

Figura 2. 17 Rappresentazione (a) di un byte e (b) di una word, con numerazione dei bit

Word = 2 byte

*La **word** è un insieme di 16 bit (2 byte); ne vediamo un esempio nella figura 2.17b. Analogamente al byte tali bit vengono numerati da B_{15} a B_0.*

*Esiste anche la dizione **double-word** a significare 2 word (quindi 32 bit) e **quad-word** per indicare 4 word (64 bit).*

5. Le memorie ROM

Le memorie sono realizzate mediante circuiti integrati digitali.

In particolare, le memorie ROM (Read Only Memory) sono memorie di sola lettura, cioè le informazioni in esse contenute, una volta scritte non possono essere modificate; sono memorie "non volatili", cioè mantengono il loro contenuto anche se prive di alimentazione ed è proprio per questo che vengono impiegate nei computer. Contengono in generale istruzioni per l'avvio del µC. Nei sistemi più piccoli (o "embedded", come i microcontrollori) memorizzano anche il programma che fa funzionare la macchina; per esempio le bilance che trovate oggi nei negozi richiedono l'impostazione del prezzo al chilo e della tara, poi in base al peso della merce fanno il calcolo del prezzo; in questo caso sarebbe alquanto scomodo dover caricare il programma ogni mattina all'accensione della bilancia e poi il programma non deve cambiare: quindi le istruzioni per il funzionamento della bilancia vengono poste in una ROM e sono immediatamente disponibili all'accensione della bilancia.

Tipologie di ROM

Ovviamente qualcuno deve pur riempire queste memorie una prima volta; le ROM sono scritte dal costruttore del chip[10] ed escono così, già pronte, dalla fabbrica; le PROM (Programmable ROM) sono invece scrivibili (si dice "programmabili") dall'utente che le acquista, però scrivibili una sola volta; le EPROM (Erasable Programmable ROM = ROM programmabili e cancellabili) possono essere scritte più di una volta previa cancellazione del loro contenuto con raggi ultravioletti[11]. Le EEPROM (Electrically Erasable Programmable ROM, siglate anche E^2PROM) sono riscrivibili come le EPROM, ma la cancellazione avviene elettricamente fornendo tensioni diverse da quelle del normale funzionamento. Quelle utilizzate attualmente sono per la maggior parte FLASHROM; differiscono dalle EEPROM per aspetti tecnologici, e risultano generalmente più veloci; le FLASHROM sono le memorie utilizzate nelle "pennette USB".

Per indicare una qualunque di queste tipologie si usa la notazione "xxROM" (intendendo ROM, PROM, EPROM, EEPROM, FLASHROM); intendendo in questo modo focalizzare l'attenzione non sulla particolare tipologia costruttiva ma sul fatto che tale dispositivo contiene informazioni non volatili, cioè permanenti.

Locazioni di memoria

Le ROM (e le RAM) possono contenere molte informazioni che vengono poste in locazioni (dette anche celle) al loro interno. In generale ogni locazione è predisposta per contenere un insieme di bit che verrà posto sul Data Bus quando richiesto; esistono memorie con locazioni da 1 bit, 2 bit, 4 bit, 8 bit, 16 bit ecc; xxROM hanno quasi sempre locazioni da 8 bit (un byte).

Le ROM non possono essere scritte e quindi, quando sono interpellate, funzionano da TX.

Una ROM può essere pensata come un grosso contenitore con al suo interno molti piccoli contenitori da 1 byte ciascuno; nella figura 2.18 vediamo una prima schematizzazione di una ROM di capacità pari a 8 byte, numerati non da 1 ad 8 ma da 0 a 7 perché questa è la convenzione che si usa.

---

[10] *Chip è una parola inglese ed indica il pezzo di silicio che sta dentro al circuito integrato e nel quale sono ricavati i circuiti veri e propri*

[11] *Le EPROM presentano una finestrella trasparente per permettere il passaggio dei raggi ultravioletti. Tale finestra viene coperta con un'etichetta opaca per impedire che, esposte ai raggi solari o ad altre emissioni elettromagnetiche, vengano accidentalmente cancellate*

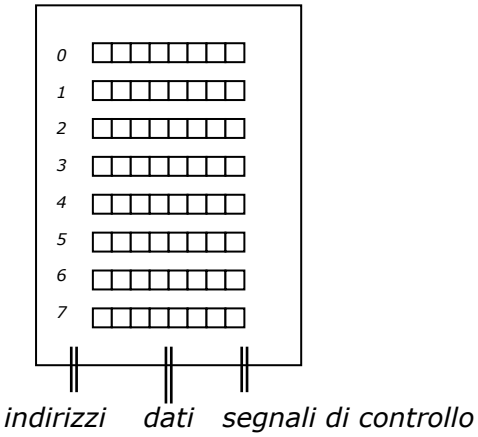

figura 2. 18 Schema a blocchi di una ROM

Per indicare quale locazione si vuole leggere è necessario fornire l'indirizzo della locazione stessa. Per esempio la quarta locazione ha indirizzo uguale a 3 (per effetto della numerazione che parte da 0), l'ultima ha indirizzo uguale a 7; tale indirizzo deve essere posto sull'Address Bus e deve essere formulato in binario. Ribadiamo che ogni entità in un µC è digitale e pertanto sarà sempre riconducibile ad una serie di bit; per fare un esempio l'indirizzo 3 sarà espresso come 011, l'indirizzo 7 come 111. Se una memoria ha 8 locazioni saranno necessarie 3 linee di indirizzo; se possiede 1024 locazioni le linee di indirizzo saranno 10; il calcolo che bisogna fare è il seguente:

Numero di locazioni e linee di indirizzo

$$2^{N° \ linee \cdot indirizzi} = N° \ locazioni$$

Internamente alla memoria c'è un circuito che, in base all'indirizzo binario impostato, selezionerà la locazione da attivare; il suo contenuto sarà poi posto sul Data Bus. Questo circuito è praticamente un decoder/demultiplexer che riceve in ingresso le linee di indirizzo e, selezionando una sola delle sue uscite, attiva una sola locazione (figura successiva).

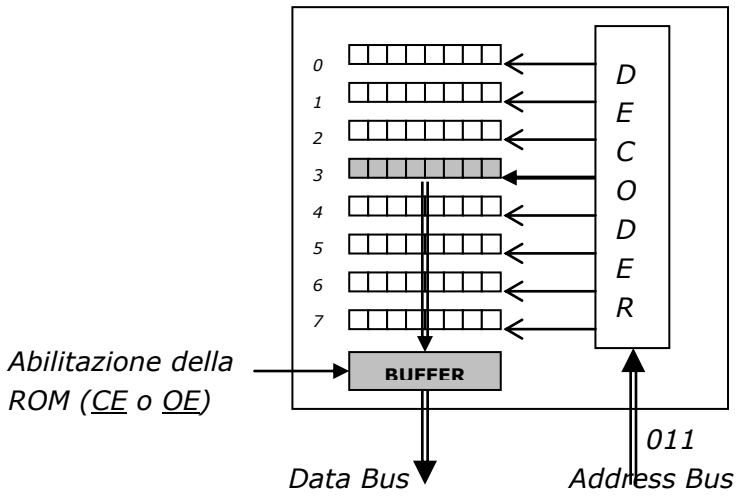

figura 2. 19 Schema a blocchi più dettagliato di una ROM

Il segnale di abilitazione di una ROM ha praticamente l'unico scopo di dare l'OK al passaggio del byte dalla zona interna al Data Bus. Infatti, una volta forniti gli indirizzi, il dato va in

*ingresso ad un componente detto **buffer** e poi, abilitando la ROM, si abilita tale buffer. Per quanto spiegato precedentemente questo buffer dovrà avere la possibilità del 3-state; è detto pertanto buffer 3-state.*

Considerate questa semplice analogia: ad un piano dell'albergo ci sono 8 camere chiuse che danno sul corridoio (bus); arriva l'addetto dell'albergo (segnale di abilitazione) che apre la porta (buffer) della camera 011 (indirizzo) per permettere alle persone (byte) di uscire. L'esempio è banale ma tutti i dispositivi che vogliano occupare (TX) il bus con i propri dati hanno sempre una porta, cioè un buffer, per accedervi; il motivo è il solito: solo un TX può avere la porta aperta, tutte le altre porte devono restare rigorosamente chiuse (3-state), per non creare ingorghi in corridoio (bus contention).

DOMANDE:

- *Come si può calcolare il numero di locazioni di una ROM conoscendo il numero delle linee di indirizzo? E viceversa?*
- *Qual è lo scopo del segnale di abilitazione in una ROM?*
- *Qual è la funzione del buffer interno alla ROM?*