4. STRUTTURA DI SISTEMA A MICROPROCESSORE

4.1. Introduzione

Nel capitolo 3 è stato esaminato un particolare tipo di circuiti a logica programmata: le strutture programmate.
Nell'esposizione di tali circuiti sono stati introdotti alcuni concetti tipici della logica programmata che vengono ripresi nei sistemi a microprocessore. Volendo fare una sintesi di quanto esposto, sono stati messi in evidenza i concetti di:

— *programma*, contenuto in una memoria ROM costitutito da una sequenza di *istruzioni*;
— *scansione del programma* mediante una successione di *indirizzi* alla memoria;
— possibilità di gestione del programma nel senso di poter compiere *salti incondizionati o condizionati*.

Il microprocessore si può pensare come una evoluzione della struttura microprogrammata che racchiude in un unico circuito integrato la logica di gestione del programma, la decodifica delle istruzioni che possono pertanto essere scritte in modo più sintetico e corrispondenti a simboli mnemonici atti a far comprendere al programmatore il significato dell'istruzione stessa, e una capacità di elaborazione numerica e logica.
Il microprocessore è pertanto una generalizzazione della struttura microprogrammata, che avendo molte funzioni già inoorporate facilita il lavoro dell'utente (dal punto di vista della programmazione e della progettazione circuitale), ma che in alcune prestazioni particolari, come tutte le generalizzazioni, risulta meno efficiente di una struttura progettata ad hoc per la particolare applicazione, sia dal punto di vista della velocità di scansione del programma, che nel numero di componenti utilizzato.
I sistemi a microprocessore, del resto, come già visto, nelle logiche microprogrammate, hanno una struttura circuitale ben definita che risponde alla filosofia del microprocessore stesso.
Uno schema molto sintetico è quello mostrato in figura 4.1.1, che rappresenta i blocchi funzionali fondamentali per il funzionamento del sistema e precisamente l'unità centrale CPU (o microprocessore in senso stretto), le memorie e le unità di ingresso-uscita. La CPU, che è il cuore di tutto il sistema

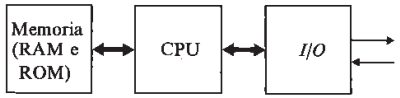


Fig. 4.1.1.

nel senso che contiene l'intelligenza e le capacità di calcolo e di elaborazione, è normalmente un unico circuito integrato che a sua volta contiene vari blocchi funzionali predisposti a vari tipi di elaborazioni (accumulatore, registri, unità logico aritmetiche, ecc.); ogni tipo di microprocessore ne possiede in un certo numero, con connessioni che possono essere diverse (architettura del microprocessore).
Da qui le differenze nel funzionamento e del linguaggio di programmazione tra un tipo e l'altro.
Il blocco di memoria comprende la memoria ROM (a sola lettura), dove risiedono i programmi da eseguire e le tabelle di dati che si riferiscono ai programmi stessi, e la memoria RAM (a lettura e scrittura) utilizzata come memoria di lavoro dove memorizzare: risultati parziali delle elaborazioni, i puntatori del programma e tutti i parametri necessari.
La quantità di memoria da impiegare naturalmente dipende dal numero di istruzioni del programma e dalla lunghezza delle tabelle e dei parametri per quanto riguarda la memoria ROM, e dalla complessità del programma per quanto riguarda la capacità della memoria RAM.
Per quanto riguarda le unità di ingresso/uscita (I/O), esse possono essere le più svariate e comunque servono a trasferire o ad acquisire dati dall'esterno del sistema a microprocessore; il numero di tali unità dipende dalla complessità del sistema a microprocessore e la loro natura dipende dall'applicazione richiesta (interfacce tipo serie o parallelo).
Esempi di tipi di interfacce I/O sono quelli che collegano il calcolatore ad una stampante o ad una tastiera, interfacce verso un registratore magnetico o un floppy disk ovvero per sistemi di controllo industriali, verso i circuiti di potenza che devono attivare certe condizioni o da trasduttori che fanno acquisire al sistema a microprocessore segnali elettrici proporzionali a grandezze fisiche sotto controllo.
Il tipo di collegamento tra i blocchi funzionali descritti deve naturalmente essere in qualche modo sincronizzato secondo regole ben precise; questa sincronizzazione viene svolta dalla CPU che presiede alla gestione delle informazioni e scambi di dati tra i vari blocchi, oltre che ad eseguire le istruzioni ed elaborazioni contenute nel programma.

4.2. Collegamento a BUS

Il tipo di collegamento normalmente utilizzato nei sistemi a microprocessori è quello a BUS: tale collegamento, come sarà visto nella descrizione di funzionamento, è quello che presenta i massimi vantaggi sia dal punto di vista della gestione dello scambio di informazioni tra i vari blocchi funzionali che da quello della loro modularità del sistema, nel senso di poter collegare ulteriori unità funzionali con il minimo di cambiamenti circuitali.
Per BUS si intende un insieme di fili, corrispondenti a segnali, cui si collegano in parallelo i vari blocchi funzionali (fig. 4.2.1).

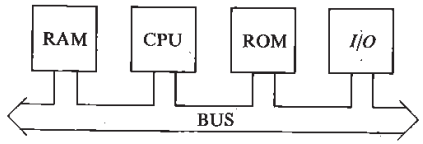


Fig. 4.2.1.

In questo modo viene messo a disposizione dei vari blocchi un canale di collegamento che, a divisione di tempo, attua la connessione dei due punti: in altre parole per un certo intervallo di tempo, uno solo dei blocchi trasmette dei dati sul BUS e un altro è predisposto in ricezione.
Naturalmente in questo intervallo di tempo gli altri blocchi non interessati al trasferimento di dati devono essere non attivi.
Dato infatti che la linea di collegamento è unica e deve essere condivisa da tutti i blocchi del sistema, deve risultare attivo un trasmettitore alla volta e gli altri disabilitati; altrimenti si avrebbe una situazione, detta *conflitto del BUS*, in cui due circuiti cercano di inviare sulla stessa uscita segnali logici anche diversi.
In questo caso, oltre al deterioramento dei circuiti le cui uscite risulterebbero in parallelo (come visto più avanti), il livello si porterebbe a valori intermedi tra quelli tipici dei valori logici e si avrebbe comunque un malfunzionamento.
Questa continua attivazione di collegamenti sul BUS deve essere naturalmente gestita secondo una sincronizzazione ben precisa: tale gestione, nello schema di figura 4.2.1, è effettuata dalla CPU che assume il ruolo di *master* del BUS mentre gli altri blocchi hanno un ruolo di *slave*.
Per la struttura mostrata in figura 4.2.1, tutti gli scambi di informazioni avvengono tramite la CPU, che a sua volta può funzionare sia da trasmettitore che da ricevitore.
Uno scambio di dati tra due blocchi slave, ad esempio tra la memoria e l'unità I/O, avviene normalmente in due fasi, la prima che trasferisce il dato delle memorie alla CPU, la seconda dalla CPU all'unità I/O.
Come sarà visto nel seguito, per aumentare l'efficienza sotto particolari punti di vista, si possono attivare trasferimenti diretti di dati tra due slave, senza

interessare la CPU (cicli DMA): in questo caso allora il controllo del bus deve essere preso da altri blocchi (che assumono il ruolo di master).
Risulta chiaro il vantaggio in base a quanto esposto, dalla struttura a bus, in quanto invece di realizzare tutte le possibili connessioni tra i vari punti tra cui deve esistere un collegamento, si predispone un'unica via cui possono accedere in tempi diversi gli utilizzatori interessati.
Ciò porta chiaramente a una struttura circuitale espandibile: ove si voglia ad esempio aggiungere ulteriori unità funzionali è sufficiente collegarne le uscite al bus, curando però la sincronizzazione e cioè disponendo i comandi opportuni per le abilitazioni sul bus nei momenti che non creano conflitto.

4.2.1. BUS indirizzi e BUS dati

Come già visto da una grossolana spiegazione del funzionamento di un sistema a microprocessore, la CPU genera un indirizzo cui corrisponde una cella di memoria contenente un'istruzione del programma: tale dato viene acquisito dalla CPU, interpretato e quindi eseguite le elaborazioni corrispondenti all'istruzione; in tali esecuzioni naturalmente possono avvenire ulteriori scambi tra CPU e memoria di lavoro (sia in lettura che in scrittura). Una lettura o scrittura nella memoria di lavoro presuppone naturalmente che la CPU generi nuovamente l'indirizzo corrispondente alla cella di memoria interessata.
Risulta pertanto chiara l'esigenza di avere due BUS distinti, il BUS indirizzi e il BUS dati, corrispondenti a un certo numero di linee.
Oltre ad essi vi sono altre linee atte a gestire il controllo di tali bus, onde consentire, in base a quanto già visto, il corretto scambio di informazioni tra i vari blocchi: queste linee di controllo vengono dette *bus di controllo*, anch'esso generato dalle CPU.
Il numero di linee costituenti il bus indirizzi dipende dalla capacità massima di indirizzamento della memoria: e cioè dal numero massimo di celle di memoria che possono essere individuate.
Molti microprocessori hanno il bus indirizzi di 16 bit con il quale si possono indirizzare 64K di memoria*.
Il bus dati è costituito da un numero di bit che specifica la profondità delle parole di programma e dei dati sulla memoria: tipicamente il bus dati è da 8 bit o 16 bit.
La notazione normalmente usata per specificare la potenzialità dei microprocessori, e cioè microprocessori a 8 bit o a 16 bit, si riferisce appunto alla costituzione del bus dati.
Un'altra osservazione che si può fare riguardo ai bus è che nel bus indirizzi,

* Il K è l'unità di misura della capacità di memoria che corrisponde a 1024 celle; 64K corrisponde quindi a 65536 celle di memoria, ogni cella è costituita da un certo numero di bit, tipicamente 8 o 16.

generato dalla CPU, il flusso di informazioni ha sempre lo stesso verso e cioè il trasmettitore è nella CPU, mentre i blocchi periferici devono acquisire tali segnali (salvo quel particolar modo di funzionamento DMA, in cui viceversa assume il ruolo di trasmettitore un altro blocco funzionale).
Viceversa nel bus dati il flusso delle informazioni può avvenire in due sensi in corrispondenza delle operazioni di scrittura o lettura della memoria o dalle altre unità.
I segnali del BUS di controllo invece hanno un verso ben definito, alcuni in uscita e altri in ingresso, a seconda delle funzioni che devono svolgere.

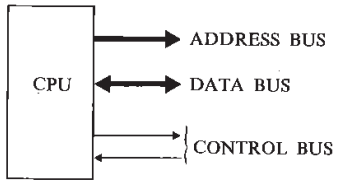


Fig. 4.2.2. I segnali di una CPU

In figura 4.2.2 è mostrato in modo sintetico l'insieme dei segnali che fanno capo a una CPU.
Mentre i segnali dell'address bus e del data bus sono sicuramente presenti in tutti i tipi di microprocessori, per, come visto, la filosofia di funzionamento di tali dispositivi, quello che differenzia un microprocessore da un altro, è la natura dei segnali del bus di controllo; possono esserci cioè da un tipo all'altro segnali che consentono un funzionamento più efficiente con modalità particolari nella gestione del programma.
La descrizione particolareggiata di tali segnali, con riferimento ai tipi di microprocessori che verranno esaminati, sarà svolta in seguito. Comunque alcuni sengali del bus di controllo sicuramente presenti sono, ad esempio, il segnale $\overline{WR}$ e $\overline{RD}$ (fig. 4.2.3).

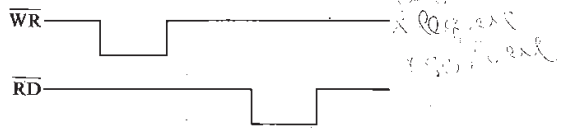


Fig. 4.2.3.

Quando, ad esempio, è attivo (a livello basso) il segnale $\overline{WR}$, ciò indica che è in corso un'operazione di scrittura, che i dati contenuti sul bus dati hanno valori stabili (le transizioni sono già esaurite) e che è stato stabilito il corretto scambio di informazioni sul bus.
In alcuni casi dai microprocessori non escono direttamente i bus indirizzi e

dati: si utilizza una struttura detta **bus multiplexato**, in cui cioè sulle stesse linee, in tempi diversi, sono presenti il bus dati e il bus indirizzi (o una parte di essi).
Naturalmente dovrà essere presente in questo caso un segnale che indica quando su tali linee è presente il bus indirizzi o quello dei dati.
Questo segnale viene utilizzato come clock per un registro che immagazzina i dati relativi al bus.
Una struttura di questo tipo è impiegata nel microprocessore 8085 Intel, in cui viene multiplexata la parte bassa degli indirizzi ($A_0 \div A_7$) con il bus dati (fig. 4.2.4).

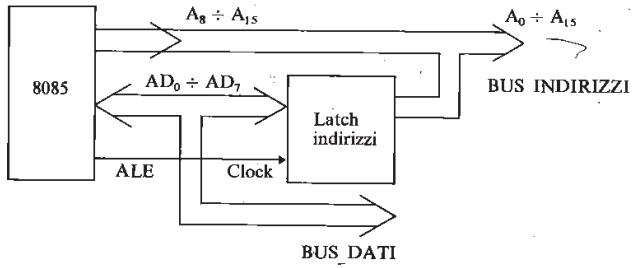


Fig. 4.2.4.

Il segnale ALE (che sta per *Address Latch Enable*) è a livello alto quando sulle linee $AD_0 \div AD_7$ sono presenti gli indirizzi $A_0 \div A_7$.
Quando è a livello basso invece, su queste linee sono presenti i segnali sul bus dati.
Tramite il latch che viene caricato con il segnale ALE, si forma il bus degli indirizzi completo.
In figura 4.2.5 è mostrata la corrispondente temporizzazione.

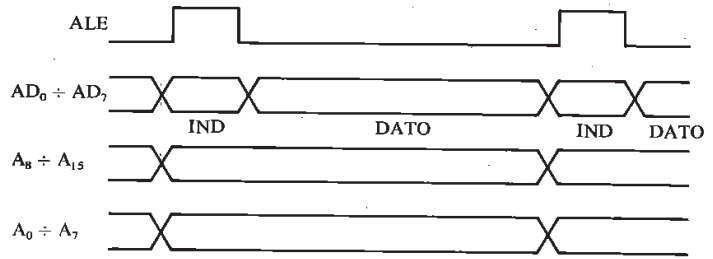


Fig. 4.2.5.

In questo caso per ogni ciclo di istruzione deve essere prevista questa fase per la formazione dell'intero bus indirizzi, che può portare ad un rallentamento della velocità di esecuzione.

D'altra parte però questa soluzione è importante per ridurre il numero di piedini dell'integrato microprocessore, consentendo così di arricchire, a parità di piedini, i segnali del bus di controllo. Nei microprocessori a 16 bit (cioè con il bus dati di 16 bit) vengono multiplexati i due bus (fig. 4.2.6), indirizzi e dati, sempre nell'ottica di tenere in un numero contenuto i piedini dell'integrato (il numero tipico di piedini dell'integrato microprocessore è di 40).

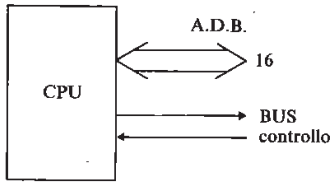


Fig. 4.2.6. I due BUS multiplexati con il bus dati di 16 bit

Inoltre, per aumentare le funzioni del microprocessore stesso (ad esempio, prevedere la possibilità di connessione in sistemi multimicroprocessore, cioè con più CPU o con memorie di capacità molto grande), alcuni dei segnali di uscita assumono significati diversi e funzioni diverse a seconda della scelta della modalità: scelta effettuata ponendo particolari livelli logici su alcuni segnali di selezione del modo.

4.2.2. Caratteristiche di componenti collegati al BUS

I circuiti logici che si collegano alle linee del bus devono avere caratteristiche particolari, in quanto, come visto, sulla stessa linea cui possono essere collegati diversi circuiti, uno solo alla volta deve trasmettere su tale linea il segnale, mentre gli altri devono essere non attivi, nel senso che non devono alterare il livello logico imposto e cioè dovrebbero essere equivalenti ad un circuito aperto (fig. 4.2.7).

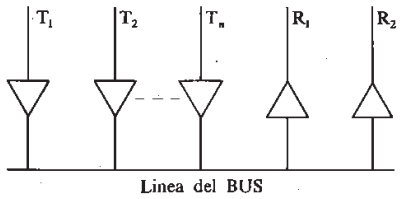


Fig. 4.2.7.

Si trovano in questo caso diversi circuiti con le uscite collegate in parallelo. Facendo riferimento alla logica TTL, ciò non sarebbe possibile con circuiti con *uscita standard* **(totem pole)**.

Ricordando infatti la struttura circuitale della logica TTL, si immagini di collegare in parallelo due uscite TOTEM-POLE; le specifiche di funzionamento prevedono, per tale tipo di uscita, una corrente massima a livello basso (I_{OL}) di circa 16 mA, che garantisce il livello logico basso.
Si supponga che le uscite mostrate in figura 4.2.8, per come sono i livelli di ingresso, debbano presentare un livello basso la prima e uno alto la seconda.

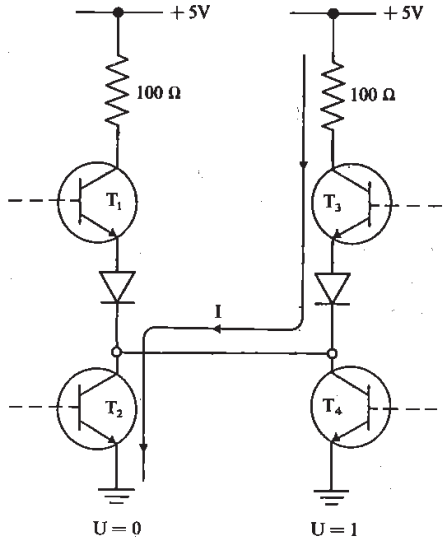


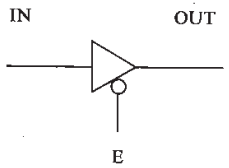
Fig. 4.2.8.

Questa situazione delle uscite comporta che i transistor T1 e T4 siano interdetti e T2 e T3 saturi.
Collegando insieme le uscite, la corrente circola nel modo indicato in figura 4.2.8: ricordando che la tensione V_{ce} di un transistor saturo è di circa 0,2 V e la caduta di tensione su un diodo in conduzione di circa 0,7 V si trova

$$I = \frac{5 - 0{,}7 - 0{,}2 - 0{,}2}{100} = 39 \text{ mA}$$

Tale valore di corrente è largamente maggiore di quella massima tollerabile da T2 per garantire un livello logico basso; oltre alla possibilità di danneggiare il componente, questa corrente può portare T2 fuori dalla zona di saturazione, con conseguente aumento della tensione di uscita, che si porta a valori non più interpretabili a livello logico basso.
Questo tipo di connessione, con uscite in parallelo va quindi assolutamente evitato.

Una variante delle porte TTL che invece consente questo tipo di collegamento, e quindi i collegamenti a bus, è la TTL con uscite *three state*.
In essa oltre alle due uscite consuete corrispondenti a livello logico basso (0) e alto (1), esiste un terzo stato detto di alta impedenza (Z): in questo stato, l'impedenza di uscita è molto elevata e quindi tale uscita è equivalente a un circuito aperto.
Naturalmente in questo caso occorre un ulteriore segnale di controllo (ENABLE) che quando attivo fa funzionare il circuito normalmente e quando non attivo invece lo porta in questo stato di alta impedenza.
In figura 4.2.9 è mostrato lo schema di un buffer three state e la sua corrispondente tabella di verità.



E	IN	OUT
0	0	0
0	1	1
1	0	Z
1	1	Z

Fig. 4.2.9.

La variazione dello schema della TTL, per consentire questo tipo di funzionamento consiste nell'imporre che entrambi i transistor di uscita siano interdetti, in corrispondenza del segnale di abilitazione non attivo (alto nell'esempio precedente).
Ciò può essere ottenuto con lo schema di figura 4.2.10.

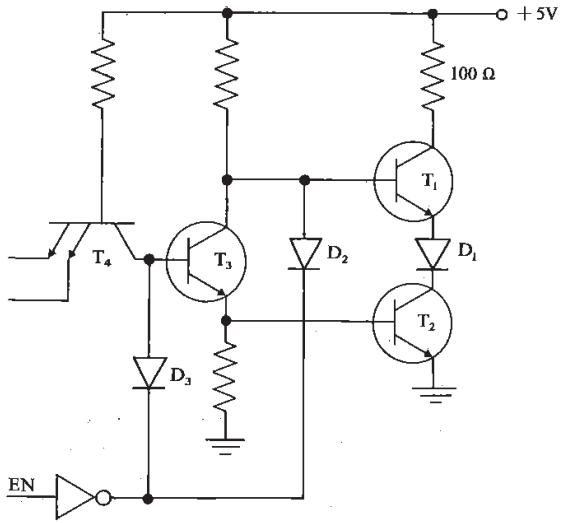


Fig. 4.2.10.

Quando il segnale ENABLE è alto i catodi di D3 e D2 si trovano a livello basso (circa 0,2 V) e quindi saranno in conduzione (caduta di tensione di circa 0,7 V).
Pertanto le basi dei transistor T1 e T3 si trovano ad un potenziale di circa 0,9 V, insufficiente a farli condurre; quindi T3, T1, e T2 si trovano interdetti e il morsetto di uscita equivale a un circuito aperto.
In un collegamento a bus, come quello mostrato in figura 4.2.11, si attiva un solo circuito per volta che manda segnali sul bus (trasmettitore) tramite il corrispondente segnale di abilitazione.

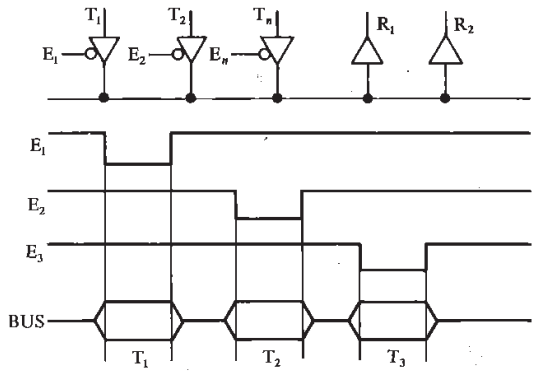


Fig. 4.2.11.

Come visto in figura 4.2.11, quando nessuno dei trasmettitori è attivo, la linea del bus si trova sospesa (cioè equivalente ad un circuito aperto) e il livello di tensione che si forma dipende dagli ingressi cui è collegata; se ad esempio gli ingressi sono dei circuiti TTL tale tensione si porta a circa 2 V.
Naturalmente il circuito ricevitore deve ricevere il comando che acquisisce il dato dal bus, negli intervalli di tempo in cui vi sono stati validi, e cioè quando uno dei trasmettitori è abilitato.
Deve quindi esserci una sincronizzazione tra abilitazione dei trasmettitori e acquisizione dei dati corrispondenti da parte del ricevitore interessato.

4.3. Organizzazione delle memorie

Come già visto, i dispositivi essenziali con cui deve scambiare informazioni la CPU sono le memorie, le memorie di programma (normalmente una ROM) e le memorie di lavoro RAM.
Tali dispositivi sono collegati al bus del sistema e devono ricevere dal bus di controllo i necessari segnali di sincronizzazione per il corretto trasferimento dei dati.

Ogni componente di memoria è caratterizzato da una capacità ben precisa, cioè da un numero di celle, ciascuna di un certo numero di bit; per ottenere un banco di memoria avente una certa capacità, bisogna utilizzare diversi dispositivi di memoria smistando in modo opportuno i segnali di sincronismo. Ad esempio se si vuole una memoria di capacità complessiva 8K·8, disponendo di componenti di memoria ciascuno da 1K·8, bisognerà utilizzare 8 componenti.

Per come è organizzato il sistema a microprocessore, deve esserci una **mappa** della memoria, cioè ogni banco di memoria deve rispondere ad indirizzi ben precisi (e di questo deve tener conto il programmatore), indirizzi generati dalla CPU tramite il bus indirizzi.

Con un bus indirizzi a 16 bit e cioè con 64K di memoria indirizzabili, in sede di progettazione del sistema a microprocessore, si destinano le varie zone ai banchi di memoria ROM, RAM, alle periferiche ecc.

Da un punto di vista circuitale questo corrisponde a generare i segnali che comandano le memorie in corrispondenza di un certo valore degli indirizzi secondo le tecniche che verranno ora prese in considerazione. Prima di esaminare questo problema, la cosidetta decodifica degli indirizzi, è utile ricordare come si presenta ai morsetti un componente di memoria.

In figura 4.3.1 è mostrata una memoria (RAM) con i relativi segnali di ingresso e uscita.

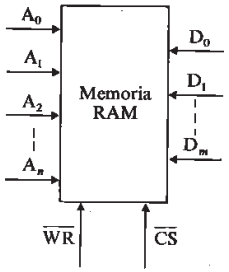


Fig. 4.3.1. Memoria RAM con relativi segnali

Come si può osservare essa è caratterizzata da un certo numero di indirizzi ($A_0 \div A_n$) che servono ad individuare all'interno del componente la cella di memoria interessata; il numero di tali indirizzi è funzione della capacità della memoria stessa, secondo la solita relazione

$$n \text{ (celle)} = 2^n \qquad [4.1]$$

Questi segnali, per la memoria, sono segnali di ingresso e si collegano direttamente al bus degli indirizzi.

I dati ($D_0 \div D_m$) sono in numero variabile a seconda del componente usato; nei componenti più usati sono 8, pari al numero di bit del bus dati della maggior parte dei microprocessori.

Questi segnali si collegano anch'essi direttamente al bus dati del sistema. Per rendere possibile questo collegamento, ovviamente questi segnali devono uscire da un buffer three-state, in modo che le memorie non interessate al trasferimento dati restino in alta impedenza e non creino conflitti sul bus.

Il segnale di abilitazione ad eseguire cicli di scrittura e di lettura e in fase di lettura trasmettere il corrispondente dato sul bus è il segnale $\overline{CS}$ (*chip-select*), mentre il segnale $\overline{WR}$ indica se in un ciclo in cui viene selezionata una certa memoria (segnale $\overline{CS}$ attivo) deve essere eseguita un'operazione di lettura o scrittura.

La logica del circuito di decodifica sulla memoria deve generare il segnale $\overline{CS}$ per la singola memoria, tra tutte quelle presenti, e naturalmente, deve essere selezionata una sola memoria alla volta ad evitare situazioni di conflitto sul bus dei dati.

In figura 4.3.2 è mostrata una temporizzazione che si riferisce a un ciclo di scrittura e lettura nella memoria.

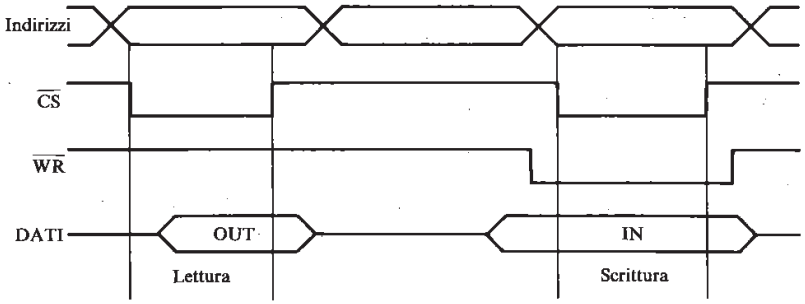


Fig. 4.3.2.

Da questo diagramma temporale, si può osservare che quando viene attivato il segnale di selezione della memoria, il bus indirizzi deve trovarsi in una situazione stabile, cioè gli indirizzi devono essere posizionati in corrispondenza del valore desiderato (esaurite cioè le trasmissioni); altrimenti si potrebbero avere erronee scritture o letture da celle indesiderate.

Nella fase di lettura, inoltre, da quando viene attivato il segnale CS, i dati risultano disponibili in uscita dopo un certo ritardo tipico della memoria stessa. Questo tempo di ritardo deve essere naturalmente compatibile con i tempi caratteristici di scansione del ciclo di lettura della CPU; nel caso di memorie troppo lente rispetto alla CPU, esiste la possibilità, come sarà visto in seguito, di introdurre nella gestione del programma dei cicli di attesa, per rendere compatibili questi tempi e quindi eseguire in modo corretto il ciclo di lettura.

Analogo discorso vale per il ciclo di scrittura: in questo caso prima della selezione dell'indirizzo, devono essere stabili sia gli indirizzi che i dati. Anche in questo caso la selezione deve durare almeno per il tempo necessario affinché il ciclo di scrittura della memoria venga completato.

WAIT STATES

4.3.1. *Decodifica della memoria*

Il circuito di decodifica della memoria deve generare, sulla base degli indirizzi e dei segnali di controllo, forniti dalla CPU, i segnali di selezione $\overline{CS}$ alle varie memorie del sistema, in corrispondenza dei valori prefissati dalla mappa stabilita. Il segnale di lettura/scrittura $\overline{WR}$, generato sempre dalla CPU, viene collegato in parallelo a tutte le memorie.
Esistono naturalmente diversi modi di progettare e realizzare una tale rete, trattandosi di rete combinatoria.
Ad alimentare tale rete sono gli indirizzi che non entrano direttamente nella memoria (questi ultimi infatti determinano la cella desiderata all'interno della memoria selezionata).
Ad esempio una memoria di 1 K di capacità occupa 1024 locazioni e cioè 400 esadecimale; se si vuole che risponda all'indirizzo 1000 (esadecimale) il circuito di decodifica genera un segnale di selezione in corrispondenza di tutti gli indirizzi compresi tra 1000 e 13FF e pertanto i primi 10 bit di indirizzo (che entrano direttamente nella memoria) risultano indifferenti al fine della generazione della selezione. Uno schema generale di un banco di memoria può essere pertanto quello mostrato in figura 4.3.3.

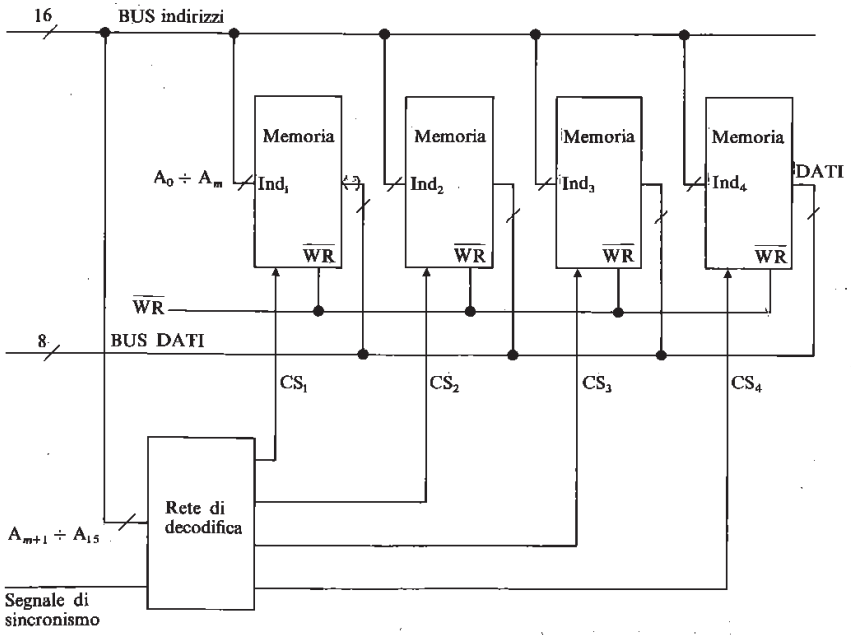


Fig. 4.3.3.

Il segnale di sincronismo che entra nella rete di decodifica può essere diverso a seconda del tipo di microprocessore impiegato, ma deve essere comunque un segnale che indica, quando attivo, che gli indirizzi sono stabili e sono esaurite le transizioni.
Altrimenti, dato che la rete di decodifica è puramente combinatoria, in corrispondenza della variazione degli indirizzi potrebbero verificarsi situazioni in cui si generano dei segnali di selezione, anche se di durata molto breve, che però potrebbero generare indesiderate scrittura o lettura della memoria.
Tale segnale per la CPU Z80 potrebbe essere $\overline{MREQ}$, che, quando attivo (a livello basso), indica che è in corso un ciclo di lettura o scrittura dalla memoria, mentre per la CPU 8085 potrebbe essere generato combinando insieme i segnali $\overline{RD}$ e $\overline{WR}$ (entrambi attivi bassi) (fig. 4.3.4).

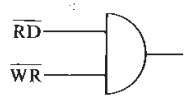


Fig. 4.3.4.

In quest'ultimo caso, l'uscita della porta AND diventa bassa quando è in corso un ciclo di lettura ($\overline{RD}$) o di scrittura ($\overline{WR}$), con gli indirizzi sicuramente stabili.
Stabilito il campo di indirizzi cui deve rispondere la singola memoria e quindi essere generato il corrispondente segnale $\overline{CS}$, è sufficiente cambiare gli indirizzi che entrano nella rete di decodifica a livello diritto o negato, e tramite porte generare un livello basso sull'uscita.

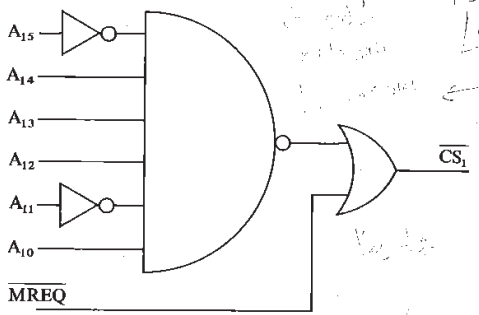


Fig. 4.3.5.

Nell'esempio di figura 4.3.5 viene generato il segnale $\overline{CS1}$ a livello basso quando:
— il segnale $\overline{MREQ}$ è basso (segnale di sincronismo);

— gli indirizzi più alti si trovano nella combinazione:

A15 A14 A13 A12 A11 A10
0 1 1 1 0 1

La memoria in questione allora, cui arriva il segnale di selezione $\overline{CS1}$, risponde agli indirizzi compresi tra

7400 e 77FF

È chiaro che con questa soluzione, per ogni segnale di selezione $\overline{CS}$, occorre una struttura circuitale del tipo di quella mostrata in figura 4.3.5. Una soluzione più compatta dal punto di vista del numero di componenti è quella che utilizza un decodificatore (o demultiplexer).
Questo componente nella famiglia degli integrati a media scala di integrazione, ha *m* uscite, delle quali una sola per volta è a livello basso, in corrispondenza di una selezione di ingresso; esistono inoltre dei bit di abilitazione che se non attivi tengono tutte le uscite alte.
Un decoder della famiglia TTL, largamente usato nei sistemi a microprocessore è il 74138 con 8 uscite codificate da tre bit di selezione, più tre ingressi di abilitazione (fig. 4.3.6).

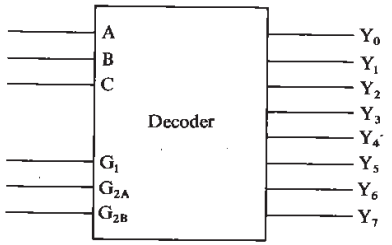


Fig. 4.3.6.

La corrispondente tabella di verità è la tabella 4.3.I.

Tab. 4.3.I

G1	G*	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0

dove $G^* = G2_A + G2_B$

Con questo componente il circuito di decodifica degli indirizzi atto a generare i segnali di selezione per otto memorie, può essere strutturato collegando gli indirizzi più significativi (diritti o negati) ai bit di selezione e di abilitazione.

Esempio

Il circuito di figura 4.3.7 è atto a selezionare un banco di memoria, ciascuna da 1 K × 8 a partire dall'indirizzo (1000)H.

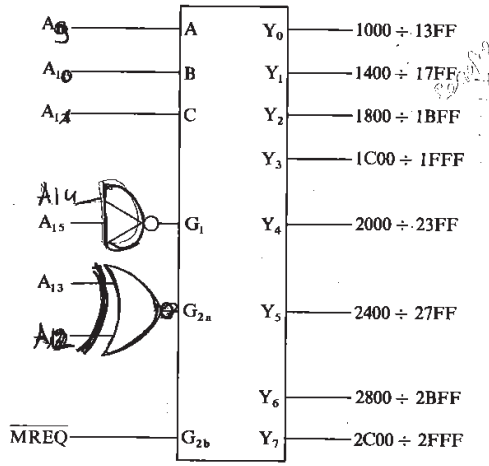


Fig. 4.3.7.

Sulle uscite sono indicati gli spazi di memoria attivati da ciascun segnale, che corrispondono ai segnali $\overline{CS}$ destinati alle 8 memorie del banco in questione. Il metodo di progettazione di questa rete con l'uso del decoder presuppone la conoscenza dei componenti di memoria impiegata (e quindi della loro capacità) e dell'indirizzo di partenza cui deve rispondere tale banco.
In base a queste informazioni, si può stilare la tabella degli indirizzi, sulla base della quale, predisporre il collegamento con il decoder.

Si debba ad esempio realizzare il circuito di decodifica di un banco di memoria di capacità complessiva di 16 K, disponendo di memorie di cui ciascuna di capacità di 2K×8, a partire dall'indirizzo (4000)H e quindi fino all'indirizzo (7FFF)H.
Ciascuna memoria occupa uno spazio di (800)H e riceve dal bus i primi 11 bit di indirizzo ($A_0 \div A_{10}$).
Le zone di memoria occupate da ciascun componente e in corrispondenza delle quali generare i relativi segnali di selezione sono:

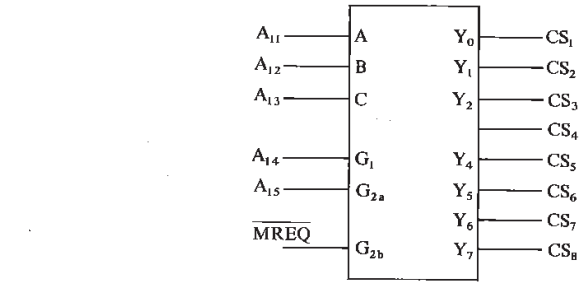
4000 ÷ 47FF CS1
4800 ÷ 4FFF CS2
5000 ÷ 57FF CS3
5800 ÷ 5FFF CS4
6000 ÷ 67FF CS5
6800 ÷ 6FFF CS6
7000 ÷ 77FF CS7
7800 ÷ 7FFF CS8

A interessare il circuito di decodifica sono i bit di indirizzo da A_{11} a A_{15}. Nella tabella 4.3.II, sono riportate le combinazioni degli indirizzi che generano i segnali $\overline{CS}$.

Tab. 4.3.II

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
(4000)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CS1
(47FF)	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	
(4800)	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	CS2
(4FFF)	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	
(5000)	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	CS3
(57FF)	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	
(5800)	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	CS4
(5FFF)	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
(6000)	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	CS5
(67FF)	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	
(6800)	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	CS6
(6FFF)	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	
(7000)	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	CS7
(77FF)	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	
(7800)	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	CS8
(7FFF)	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Dalla tabella 4.3.II, si può assumere che i bit da A_0 e A_{10}, nell'ambito di selezione di una memoria, possono variare e quindi sono indifferenti ai fini della rete di decodifica (infatti sono i bit che entrano direttamente nel componente di memoria).
I bit A_{11}, A_{12}, e A_{13} invece variano in tutti i modi possibili secondo le combinazioni binarie e quindi vanno collegati ai segnali di selezione del decoder (A,B,C), mentre A_{14} e A_{15} mantengono dei valori fissi e vanno quindi collegati ai segnali di abilitazione del decoder.
Lo schema risulta pertanto come in figura 4.3.8.



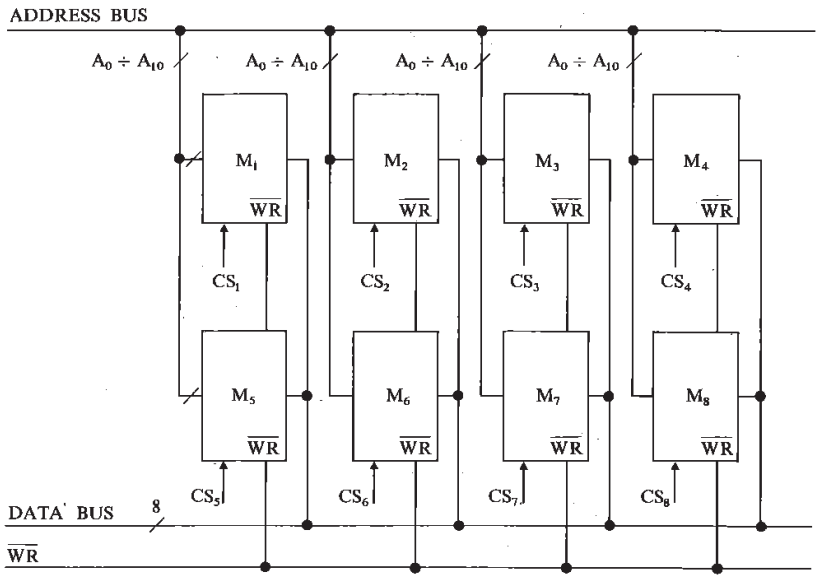


Fig. 4.3.8.

4.3.2. Decodifiche ad indirizzo variabile

In molte applicazioni è necessario avere la possibilità di poter variare l'indirizzo di partenza del banco di memoria: questo accade per esempio nei sistemi in cui i vari blocchi funzionali sono realizzati su singole schede, connesse tra loro mediante cablaggio esterno (fig. 4.3.9).

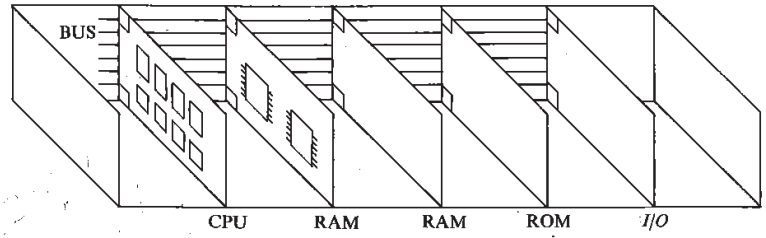


Fig. 4.3.9.

In questo tipo di realizzazione sul cablaggio esterno sono presenti i bus, mentre i circuiti di decodifica degli indirizzi sono sulle schede di memoria.
Trattandosi, in questo caso di un sistema modulare in cui si può, ad esempio, espandere la capacità di memoria (fino alla massima capacità), l'indirizzo base della singola scheda non può essere fisso, ma variabile; in questo modo l'utente che assembla il sistema, può allocare i vari sottosistemi ad indirizzi diversi (per quanto già detto ad un indirizzo deve rispondere una sola unità, altrimenti si creano situazioni di conflitto sul bus).
Per poter variare l'indirizzo base in modo semplice, bisogna prevedere nel circuito di decodifica una serie di interruttori che l'utente posiziona in modo opportuno a seconda dell'indirizzo desiderato. Come già visto negli esempi precedenti, nel progetto della rete di decodifica (tralasciando i bit di indirizzo che entrano direttamente nella memoria, indifferenti ai fini della decodifica), degli altri bit di indirizzo alcuni servono a selezionare la singola memoria ed agiscono sui segnali di selezione del decoder, mentre gli altri, che mantengono nella tabella valori costanti, vanno alle abilitazioni del decoder.
Sono proprio questi ultimi bit, i più significativi degli indirizzi, che determinano l'indirizzo base.
Con riferimento all'esempio di par. 4.3.1 tali bit sono A_{15} e A_{14}; se quindi si consente l'abilitazione del decoder in corrispondenza di una delle quattro possibili combinazioni di questi due bit, si ottengono quattro possibili indirizzi base che sono

A_{15}	A_{14}	*Indirizzo base*
0	0	0000
0	1	4000
1	0	8000
1	1	C000

Con le connessioni al decoder di figura 4.3.8, ciò corrisponde ai collegamenti di figura 4.3.10 *a* e *b*.

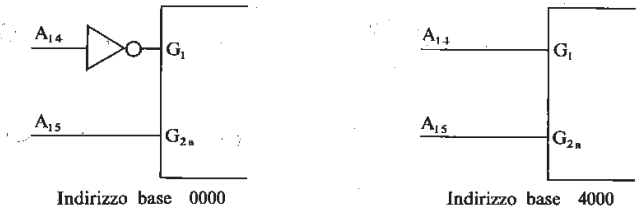


Fig. 4.3.10a

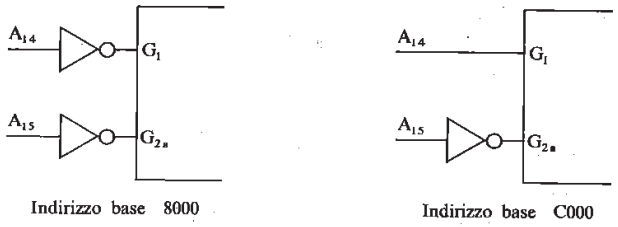


Fig. 4.3.10b

Volendo realizzare un circuito in cui si possa variare tale indirizzo base mediante switch, si può in questo caso ricorrere alla configurazione 4.3.11.

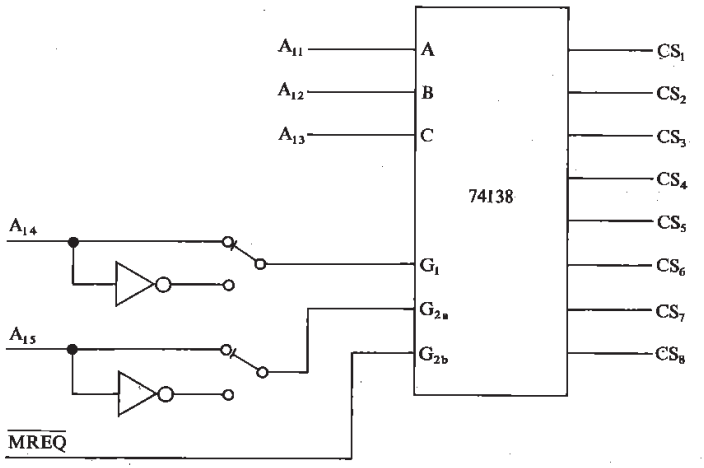


Fig. 4.3.11.

Un altro circuito che viene spesso utilizzato di carattere generale sfrutta un comparatore: si comparano i bit di indirizzo più significativi che determinano l'indirizzo base, con valori binari impostati tramite switch.
In caso di uguaglianza viene generato un segnale alto, che si collega con l'abilitazione G_1 del decoder.

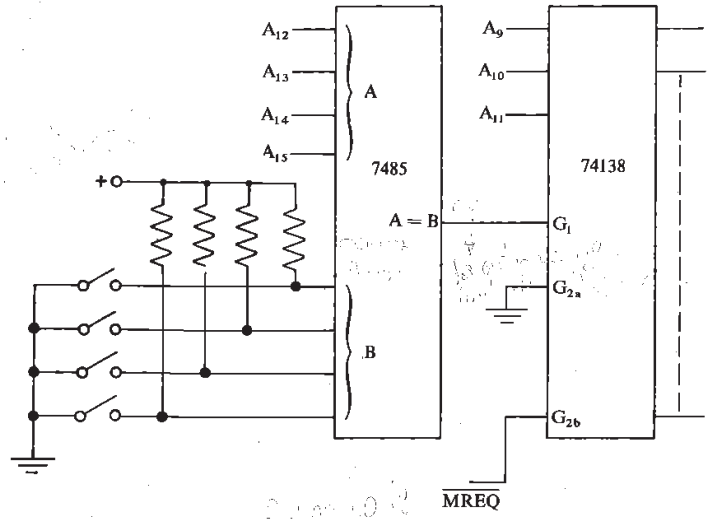


Fig. 4.3.12.

Nel circuito di figura 4.3.12 si decodifica un banco di memoria di 4 K con memorie da 512 bytes ciascuna, con indirizzo base variabile (in questo caso sono 16 i possibili valori).
La soluzione prospettata, prevede un comparatore integrato (nell'esempio il 7485 della famiglia TTL); la stessa funzione logica può essere naturalmente svolta con componenti più elementari, ad esempio con porte OR-esclusivo, che, come noto, forniscono in uscita un livello basso solo quando i due ingressi sono uguali.
Il circuito potrebbe essere quindi come in figura 4.3.13.
Il segnale G_1 è a livello alto (e quindi il decoder), solo quando gli ingressi A_{12}, A_{13}, A_{14}, A_{15} sono uguali ai livelli impostati tramite switch.
A seconda dell'applicazione, non sempre è necessario utilizzare tutti i bit di indirizzo per la decodifica degli indirizzi: in altre parole, se la totale area di memoria disponibile non è impegnata dai rispettivi componenti, si può fare

un circuito di decodifica più semplice in cui il componente risponde a più indirizzi (non occupati da altri componenti).

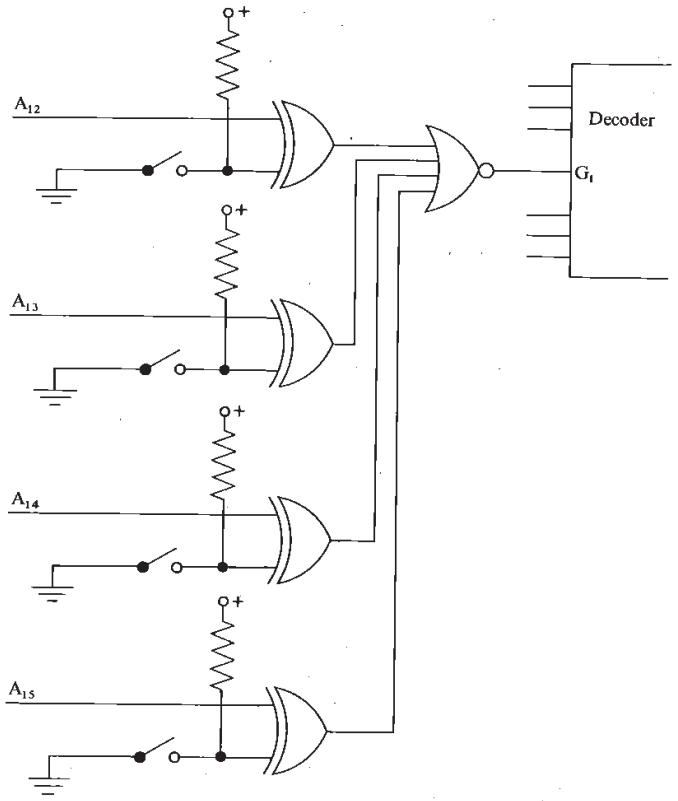


Fig. 4.3.13.

Ad esempio il circuito mostrato in figura 4.3.14 genera un segnale di selezione $\overline{CS}$, in corrispondenza degli indirizzi A_{12}, A_{13}, A_{15}, non avendo interessato nel circuito il bit di indirizzo A_{14}.

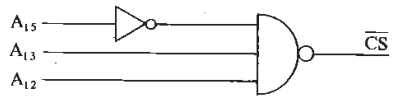


Fig. 4.3.14.

4.4. Gestione dei circuiti di Input-Output

I circuiti finora esaminati che consentono di generare un segnale di comando, in corrispondenza di un certo valore di indirizzo, sono di uso generale nel senso che sono atti, oltre che a comandare le memorie, come visto, anche a generare dei clock che comandano altri tipi di circuiti, ad esempio registri o buffer three state atti ad acquisire sul data-bus o a trasferire all'esterno dei dati.
Il circuito di figura 4.4.1, ad esempio, acquisisce i livelli logici impostati tramite switch in corrispondenza del segnale $\overline{CS1}$ (che corrisponderà ad un certo indirizzo) e può trasferire un dato da 8 bit su led luminosi in corrispondenza del segnale $\overline{CS2}$.

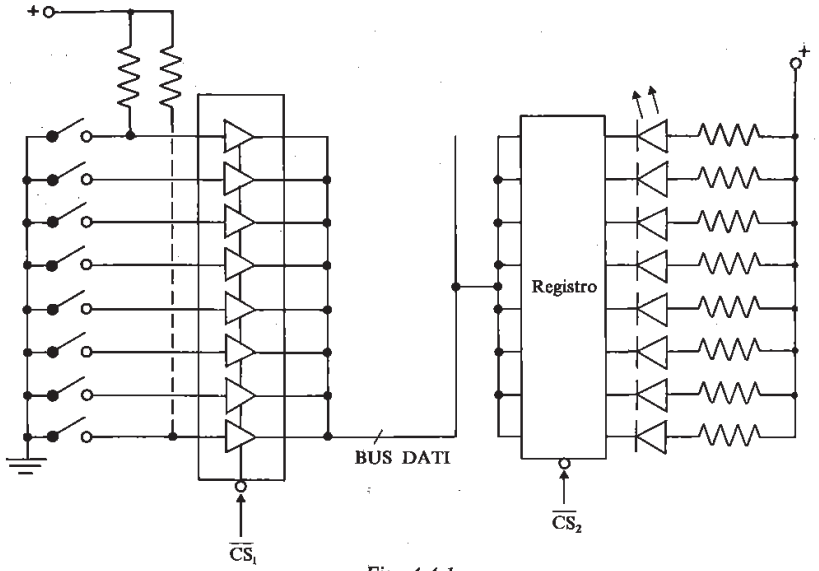


Fig. 4.4.1.

In questo caso i due circuiti, che fanno parte dei circuiti di Input-Output, vengono considerati a tutti gli effetti come celle di memoria che rispondono quindi ad un indirizzo ben preciso; per acquisire il dato corrispondente alla posizione degli switch, il programmatore dovrà utilizzare una istruzione di lettura dalla memoria in corrispondenza dell'indirizzo per il quale si genera il segnale $\overline{CS1}$, e analogamente per accendere il LED di uscita.
Una gestione di questo genere dei circuiti di Input-Output si dice **memory-mapped**. L'area riservata alla memoria naturalmente diminuisce, dovendo riservare degli spazi nella mappa di memoria a questi componenti periferici; quest'area può essere di una cella, se si utilizzano tutti gli indirizzi necessari

per la generazione del segnale di comando $\overline{CS}$ e di più spazi se si usa un circuito semplificato (come visto prima).
Nei sistemi a microprocessore tuttavia, esiste una gestione particolare delle unità I/O, tramite particolari istruzioni nel programma (ad esempio istruzioni IN e OUT).
In questo modo si può riservare tutta l'area indirizzabile (64 K con 16 bit di indirizzo), alle memorie vere e proprie, creando un circuito di decodifica a parte per le periferiche.
Naturalmente anche per le istruzioni IN e OUT si specifica un indirizzo, che viene trasmesso sul bus indirizzi; in questo caso però l'indirizzo che deve essere indicato è a 8 bit.
Ciò significa che si possono individuare fino a 256 periferiche diverse.
Il circuito di selezione delle periferiche fa pertanto riferimento a 8 bit di indirizzo (i meno significativi) e ad un segnale di sincronismo particolare che indica che si sta utilizzando un indirizzamento a periferiche.
Nel caso della CPU Z-80 tale segnale si chiama $\overline{IORQ}$, che in questa circostanza diventa basso (mentre naturalmente $\overline{MREQ}$ resta inattivo, non consentendo così la selezione delle memorie in corrispondenza dell'indirizzo che comunque viene generato).
In tal modo si evitano situazioni di conflitti sul bus.
Utilizzando per il circuito di decodifica sempre il decoder 74138, nell'esempio riportato in figura 4.4.2, si generano segnali di comando per periferiche in corrispondenza degli indirizzi ripetuti nella tabella 4.4.I.

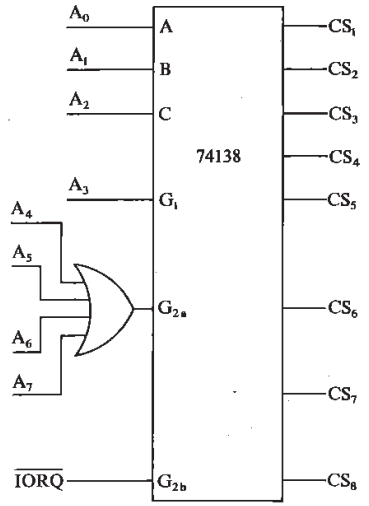


Fig. 4.4.2.

Tab. 4.4.I

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	*Segnale attivo*	
0	0	0	0	1	0	0	0	CS1	(08)
0	0	0	0	1	0	0	1	CS2	(09)
0	0	0	0	1	0	1	0	CS3	(0A)
0	0	0	0	1	0	1	1	CS4	(0B)
0	0	0	0	1	1	0	0	CS5	(0C)
0	0	0	0	1	1	0	1	CS6	(0D)
0	0	0	0	1	1	1	0	CS7	(0E)
0	0	0	0	1	1	1	1	CS8	(0F)

In questo caso avendo impiegato tutti i bit di indirizzo necessari, nella rete di decodifica, la singola periferica risponde ad un indirizzo ben preciso.
Normalmente, dato che il numero delle periferiche da comandare in un sistema a microprocessore non è mai molto elevato, si può omettere qualche bit di indirizzo nella rete di decodifica, rendendola così più semplice, tollerando che la stessa periferica risponda ad indirizzi diversi.
Se ad esempio il circuito di figura 4.4.3 viene modificato nel modo seguente, il segnale $\overline{CS1}$ viene generato, oltre che in corrispondenza dell'indirizzo 08, anche per 28, 48, 68, 88, A8, C8, E8.

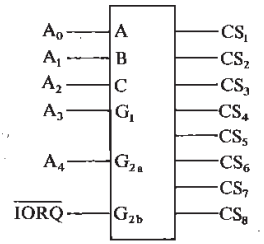


Fig. 4.4.3.

Se ad esempio il segnale $\overline{CS1}$ comanda una periferica di ingresso, la relativa istruzione che consente l'acquisizione del dato sarà del tipo

IN 08

Anche nella CPU 8085-Intel la gestione dei circuiti Input-Output è simile a quella dello Z80. In questo caso però esiste un solo segnale che distingue se l'indirizzamento è rivolto alla memoria o alle periferiche, segnale IO/$\overline{M}$ con il significato indicato in figura 4.4.4.
Per la sincronizzazione sul circuito di decodifica degli indirizzi questo segnale va combinato con altri segnali che indicano che gli indirizzi sono stabili, che nel caso specifico sono $\overline{RD}$ e $\overline{WR}$. I circuiti per la decodifica degli indirizzi della memoria e delle periferiche per la CPU 8085 possono essere realizzati nel modo indicato in figura 4.4.5.

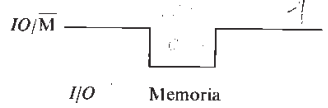


Fig. 4.4.4.

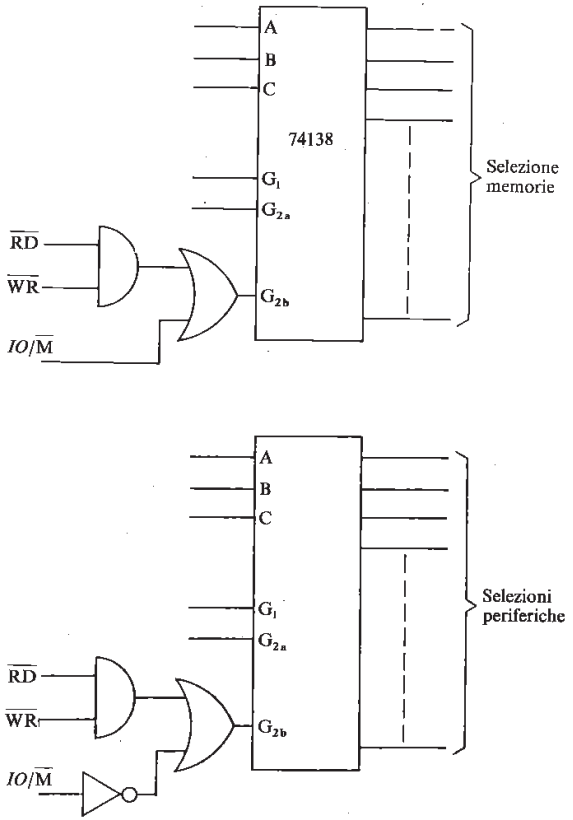


Fig. 4.4.5.

Esercizio 1.

In un sistema a microprocessore basato sulla CPU Z80, progettare il circuito di selezione per un banco di memoria di 8 K × 8, impiegando memorie di capacità 1 K · 4 (2114 Intel), con indirizzo base variabile.
In questo caso i componenti di memoria da utilizzare sono 16, infatti occorre mettere in parallelo due memorie per ottenere il dato da 8 bit. Data la capacità di 1 K di ciascuna memoria, i primi 10 bit di indirizzo vanno direttamente in esse, e quindi i restanti 6 bit comandano la rete di decodifica.
I tre bit più significativi A_{15}, A_{14}, A_{13} formano l'indirizzo base, mentre A_{12}, A_{11}, A_{10} devono controllare la selezione del decoder per smistare il comando $\overline{CS}$ a una delle 8 coppie di memoria del banco. Per la realizzazione circuitale, ci si può riferire a uno degli schemi già indicati in precedenza.
Gli indirizzi base, naturalmente scalati di 8 K e cioè espressi in esadecimale, sono:

0000
2000
4000
6000
8000
A000
C000
E000

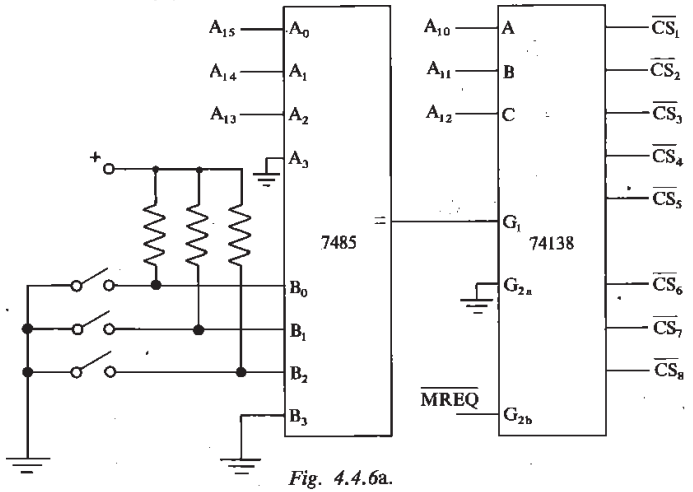


Fig. 4.4.6a.

Esercizio 2.

In un sistema a microprocessore basato sulla CPU 8085, progettare il circuito di decodifica di 16 unità Input-Output, che rispondano agli indirizzi 00, 10, 20, 30, 40, 50, 60, 70, 80, 90, A0, B0, C0, D0, E0, F0 (naturalmente espressi in codice esadecimale).

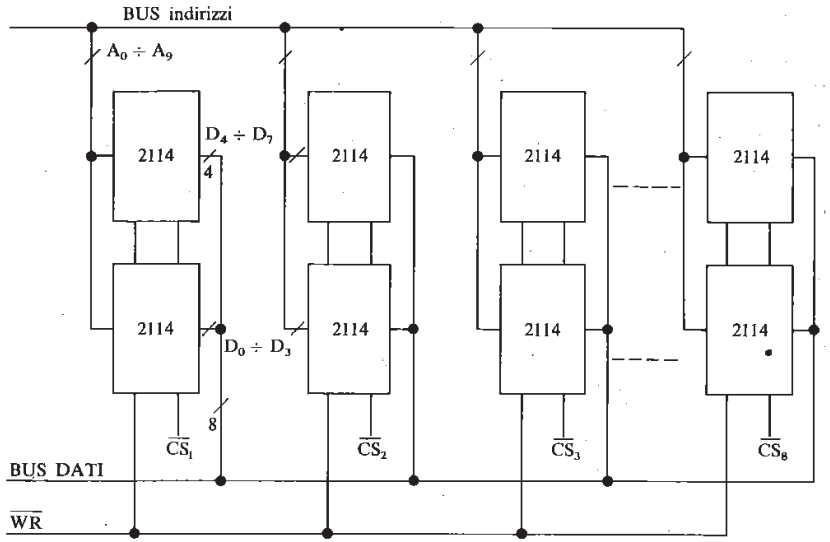


Fig. 4.4.6b.

Come già visto, con le istruzioni Input-Output, i bit di indirizzo interessati alla rete di decodifica sono i primi 8.

Tab. 4.4.II

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	*Segn. di selez.*
0	0	0	0	0	0	0	0	$\overline{CS1}$
0	0	0	1	0	0	0	0	$\overline{CS2}$
0	0	1	0	0	0	0	0	$\overline{CS3}$
0	0	1	1	0	0	0	0	$\overline{CS4}$
0	1	0	0	0	0	0	0	$\overline{CS5}$
0	1	0	1	0	0	0	0	$\overline{CS6}$
0	1	1	0	0	0	0	0	$\overline{CS7}$
0	1	1	1	0	0	0	0	$\overline{CS8}$
1	0	0	0	0	0	0	0	$\overline{CS9}$
1	0	0	1	0	0	0	0	$\overline{CS10}$
1	0	1	0	0	0	0	0	$\overline{CS11}$
1	0	1	1	0	0	0	0	$\overline{CS12}$
1	1	0	0	0	0	0	0	$\overline{CS13}$
1	1	0	1	0	0	0	0	$\overline{CS14}$
1	1	1	0	0	0	0	0	$\overline{CS15}$
1	1	1	1	0	0	0	0	$\overline{CS16}$

La tabella 4.4.II mostra gli indirizzi, e i corrispondenti segnali di selezione da generare sono mostrati in figura 4.4.7.
Se si tengono indifferenti i bit A0 ÷ A3, con il che la periferica risponde oltre che all'indirizzo specificato anche agli indirizzi che hanno tali bit di valori qualunque (ad esempio, oltre che all'indirizzo 00, anche a 01, 02, ... 0F), il circuito di selezione è la figura 4.4.7.

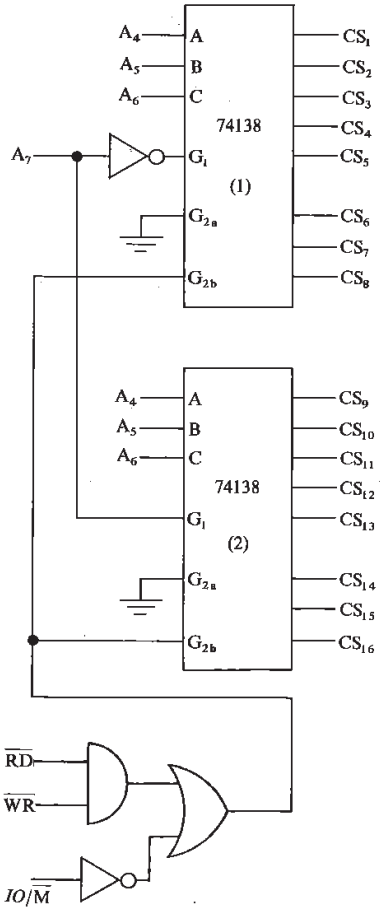


Fig. 4.4.7.

In questo caso sono stati raggruppati due decoder a 8 bit, per formarne uno da 16 agendo sull'abilitazione G_1, come quarto bit di selezione. Infatti, quando A_7 è basso, è abilitato il decoder 1 e disabilitato il 2; e viceversa quando A_7 è alto.

4.5. Architettura di un microprocessore

La differenza sostanziale tra una CPU e l'altra è visibile dalla loro struttura interna, detta anche *architettura*, e cioè dai blocchi funzionali di cui è costituita, anche se alcuni di essi devono necessariamente essere comuni a tutti i tipi di CPU.
La diversa architettura della CPU, comporta anche una differenza nel *linguaggio di programmazione* (**assembler**) e la diversa natura dei segnali di controllo, che chiaramente dipende dalla costituzione dell'hardware interno in grado di gestirli.
Questo spiega come non esista uno standard per il linguaggio ASSEMBLER, mentre ciò succede per altri tipi di linguaggi (tipo BASIC, FORTRAN, PASCAL ecc...) salvo piccole varianti: infatti il linguaggio ASSEMBLER, che in pratica è il linguaggio macchina della CPU, tramite le sue istruzioni presiede alle elaborazioni e ai trasferimenti di dati tra i vari blocchi funzionali della CPU o verso l'esterno, ed è quindi strettamente legato alla natura e alle funzionalità di tali blocchi.
Tuttavia, tra tutti i microprocessori esistenti sul mercato, è possibile fare una classificazione per famiglie; nell'ambito di una famiglia, il tipo di architettura simile, pin con funzioni in più e prestazioni diverse.
L'evoluzione delle CPU nell'ambito di una famiglia è avvenuta, creando componenti con prestazioni migliorate rispetto ai precedenti, ma conservando la cosidetta *compatibilità software*; ciò significa che il componente più evoluto della famiglia interpreta nello stesso modo i codici operativi corrispondenti alle istruzioni scritte per il componente precedente e quindi un programma già scritto, gira anche sul nuovo componente.
Ciò corrisponde naturalmente ad una grossa opportunità per l'utilizzatore che può sostituire l'hardware del sistema (in pratica la CPU), ottenendo prestazioni migliori, senza dover scrivere il programma esecutivo per l'applicazione particolare.
Questa evoluzione è avvenuta per la famiglia 8080 Intel cui sono succedute nel tempo le CPU 8085 Intel e Z80, conservando, come detto, la compatibilità software rispetto ai precedenti.
C'è da osservare però, che ogni componente più evoluto oltre ad avere migliori prestazioni dal punto di vista circuitale, ha anche un set di istruzioni più ricco, cioè conservando tutte le istruzioni dei precedenti della famiglia ne ha diverse nuove. Questo significa che un programma scritto per lo Z80, sfruttando appieno il suo set di istruzioni, non gira sulla CPU 8085.
Prima di entrare nei dettagli della descrizione di queste CPU, viene esaminato lo schema generale di una CPU con le parti fondamentali, comuni a CPU di vario tipo.
Come si può osservare dalla figura 4.5.1 nella CPU è presente un ALU (*Unità Logica Aritmetica*) che presiede ad elaborazioni di tipo aritmetico o logico