CAPITOLO XIV
LE MEMORIE E I MICROPROCESSORI

14.1) Introduzione.

Un peso notevolissimo nel campo dell'integrazione a larga scala e' ricoperto al giorno d'oggi dalle memorie, che rappresentano attualmente circa i due terzi delle totalita' dei circuiti integrati prodotti. Ancora oggi si assiste ad un continuo aumento del numero di bit disponibili sul singolo chip, che approssimativamente quadruplica ogni tre anni.

Non esiste stranamente sulle memorie a semiconduttore quell'ampiezza di informazione che ci si potrebbe aspettare per componenti che rivestono una cosi' grande importanza. Probabilmente tale situazione prende origine, a differenza dei circuiti MSI e SSI, da considerazioni di competitivita' tra le industrie produttrici che non hanno alcun interesse a divulgare informazioni troppo dettagliate sui propri prodotti.

14.2) Classificazione delle memorie.

Le memorie a semiconduttore sono dei dispositivi elettronici costituiti da un insieme di celle elementari, ciascuna delle quali e' in grado di immagazzinare un'informazione binaria. Questo insieme di celle elementari e' dotato di apposite linee di comunicazione per accedere a ogni singola cella. Una possibile classificazione delle memorie e' riportata nella tabella sottostante.

Tale classificazione vede una prima grande suddivisione in memorie ad accesso casuale (RAM) e memorie di sola lettura (ROM). Nelle prime un dato puo' essere scritto e letto in qualsiasi cella, con un tempo di accesso che e' praticamente lo stesso sia che l'operazione sia di scrittura che quando l'operazione e' di lettura.

TABELLA 1

RAM	---------- dinamiche	(MOS)
	---------- statiche	(MOS) (bipolari)
ROM	---------- a maschera	(MOS)
	---------- PROM	(bipolari)
	---------- EPROM	(MOS)
	---------- EEPROM o EAROM	(MOS)

Nelle seconde invece la normale operazione e' solo quella di lettura in quanto o la scrittura non e' permessa o richiede un tempo troppo lungo rispetto a quella di lettura. Un vantaggio delle ROM tuttavia risiede nel fatto che l'informazione si conserva anche quando l'alimentazione viene a mancare, a differenza delle RAM in cui l'informazione viene persa. Per tale motivo le ROM vengono anche dette memorie non volatili.

Le ROM possono poi essere suddivise ulteriormente in alcune sottoclassi, per ciascuna delle quali e' bene accennare ad alcune delle caratteristiche piu' importanti.

-- **ROM a maschera**. Sono memorie di sola lettura nelle quali i dati vengono immagazzinati mediante un opportuno processo di mascheratura durante la fabbricazione. E' impossibile modificare i dati di un dispositivo finito. L'utente, all'atto dell'ordinazione, deve specificare con un'opportuna codifica tutta l'informazione che deve venir scritta nella memoria. La struttura fisica del componente e' particolarmente semplice e l'organizzazione di tale tipo di memorie e' sempre a matrice. In commercio si possono trovare ROM a maschera con diversi gradi di parallelismo, di cui i piu' comuni sono comunque 4 e 8.

-- **PROM o ROM programmabili**. Sono memorie a sola lettura in cui tuttavia i dati vengono inseriti dall'utente finale. La scrittura di un dato avviene o per la fusione di opportuni collegamenti o con la perforazione di una giunzione. Il dato, una volta scritto, non puo' essere in alcun modo cancellato. Vantaggio sostanziale rispetto alle ROM e' l'eliminazione dei tempi di attesa tra ordinazione della ROM e la sua consegna, malgrado che il loro costo unitario sia superiore a quello delle ROM a maschera. E' necessario tuttavia tener presente che la produzione di ROM a maschera non e' economicamente conveniente per quantitativi inferiori ai 1000 pezzi.

-- **EPROM**. Sono memorie a prevalente lettura, utilizzate quando siano necessarie memorie non volatili, ma in cui i dati memorizzati possano venir di volta in volta modificati. In questo caso i dati, scritti con mezzi elettrici, possono venir cancellati, rendendo la memoria nuovamente disponibile, esponendo il chip a radiazione ultravioletta attraverso un' apposita finestra in vetro di quarzo.

-- **EEPROM** (spesso dette anche **EAROM**). Sono memorie molto simili alle EPROM, ma la cancellazione di un dato viene effettuata elettricamente, permettendo quindi di mantenere il dispositivo nel sistema in cui e' installato. Le operazione di scrittura tuttavia sono normalmente molto piu' lente delle operazioni di lettura. Dal punto di vista dell'impiego e delle terminazioni EPROM e EAROM sono del tutto analoghe alle ROM e alle PROM. Per alcuni modelli addirittura esiste con queste ultime una completa compatibilita' sia elettrica che meccanica (compatibilita' pin to pin).

14.3) Struttura base di una memoria.

Una memoria e' essenzialmente costituita delle seguenti parti:

1) Una matrice di celle;
2) Un decodificatore di indirizzo;
3) Un circuito di controllo dell'ingresso e dell'uscita.

In memorie di piccole dimensioni, ad esempio in memorie da 16 celle, il circuito di indirizzamento puo' essere realizzato con un semplice selettore (linear select), come illustrato in fig. 14.3.1.

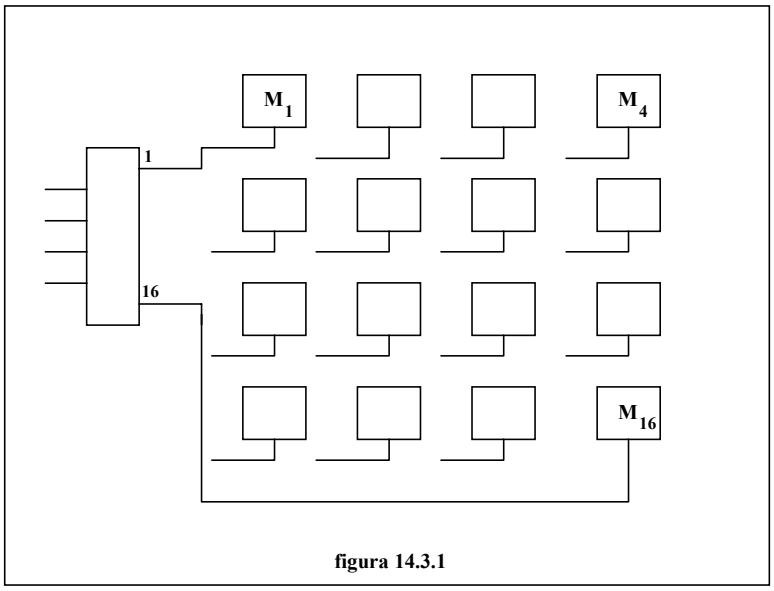


figura 14.3.1

E' opportuno far notare in tale struttura circuitale la complessita' dovuta alla presenza dei 2^n collegamenti elettrici necessari all'abilitazione delle singole celle. La complessita' inoltre aumenta in modo esponenziale all'aumentare del numero n di bit.

Una valida alternativa e' l'indirizzamento a matrice, illustrato in fig. 14.3.2 (coincident select).

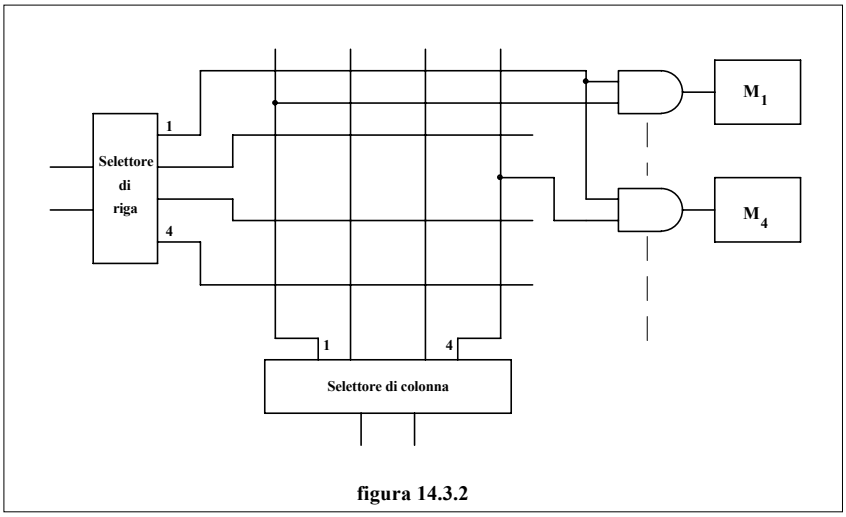


figura 14.3.2

Si ottiene in tal modo una notevole semplificazione della rete dei collegamenti. Pertanto si puo' ritenere che, tranne per memorie di minime dimensioni, la struttura di base sia quella di fig. 14.3.3.

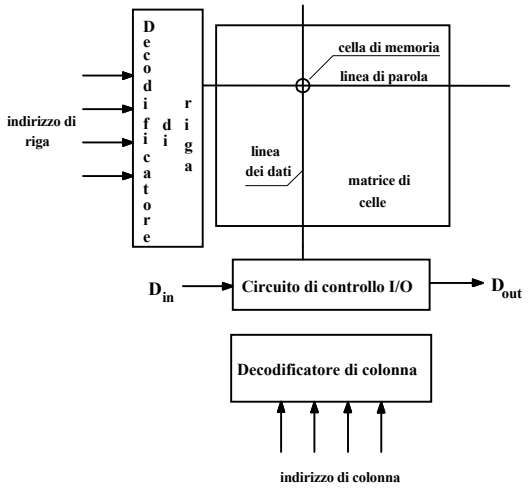


figura 14.3.3

Il decodificatore di riga selezione un'intera riga di celle. I dati immagazzinati nelle celle connesse con la linea selezionata sono trasferiti al circuito di controllo dell'ingresso e dell'uscita.

A questo punto il dato che interessa viene trasferito all'uscita selezionandolo con il decodificatore di colonna.

Ogni dispositivo di memoria possiede almeno i seguenti segnali:

1) Indirizzo (**Address**) composto dai segnali da applicare ai decodificatori. All'utente non e' normalmente possibile distinguere tra i bit di selezione di riga e di colonna. L'indirizzo deve pertanto essere considerato come un tutto unico.
2) Lettura/scrittura (**read/write**). E' presente ovviamente solo quando necessario e permette di scegliere tra un'operazione di scrittura o una di lettura in o da una determinata cella.
3) Selezione di chip (**chip select**) che permette o meno di abilitare la memoria in questione. Tale segnale si rende indispensabile quando si vogliano realizzare memorie di grandi dimensioni, superiori a quelle del singolo chip, oppure quando memorie ed altri dispositivi accedono allo stesso bus per il trasferimento dei dati.
4) Ingresso e uscita (**I/O**). In certe realizzazioni le linee di ingresso e di uscita possono coincidere.
5) Alimentazione (**supply**). Alcune realizzazioni, quali EPROM e EAROM, possono richiedere piu' di una tensione di alimentazione; tuttavia nella maggior parte dei casi e' sufficiente un'unica tensione.

La combinazione di tutti questi segnali organizza il dispositivo di memoria.

Piu' matrici del tipo descritto possono essere utilizzate per accedere ad informazioni di lunghezza superiore al bit. Si avranno ovviamente tante linee di ingresso/uscita quanti sono i bit di informazione cui si vuol accedere contemporaneamente. Si parla in tal caso di parallelismo della memoria, intendendo con tale termine il numero di bit per parola, dove parola e' l'insieme di bit cui si accede contemporaneamente con un unico indirizzo. Nelle memorie standard il grado di parallelismo e' di solito una potenza di due.

14.4) Struttura delle celle delle memorie RAM.

Prima di descrivere i principi di funzionamento di una memoria RAM e' opportuno accennare alle regole fondamentali cui bisogna attenersi nel progetto di una cella di memoria:

1) Una cella di memoria deve occupare sul chip la minima superficie possibile.
2) I processi di fabbricazione per la sua realizzazione devono essere i piu' semplici possibile.
3) Le operazioni di lettura e scrittura devono essere le piu' rapide possibile.
4) La potenza dissipata va minimizzata.

I primi due punti sono di carattere prevalentemente economico, mentre gli ultimi due riguardano le prestazioni del dispositivo.

Le memorie RAM si suddividono poi in memorie bipolari e MOS, statiche e dinamiche. Le memorie bipolari sono piuttosto rare, di capacita' ridotta, e vengono utilizzate solo in quelle applicazioni in cui sono richieste elevate velocita' di accesso. La distinzione tra dispositivi statici e dinamici e' gia' stata illustrata al capitolo XIII. Nel caso delle memorie tuttavia tale distinzione si riflette anche in profonde modificazioni strutturali e di impiego.

14.4.1) Struttura di una cella RAM statica.

Una possibile configurazione logica di una cella RAM statica e' riportata in fig. 14.4.1.

Il cuore della cella e' un semplice circuito bistabile che comunica con le linee dati attraverso buffer 3-state. Tali buffer, rispettivamente di lettura e di scrittura, sono controllati da due porte AND i cui ingressi sono il segnale di selezione di riga, quello di colonna e quello di read/write (R/W).

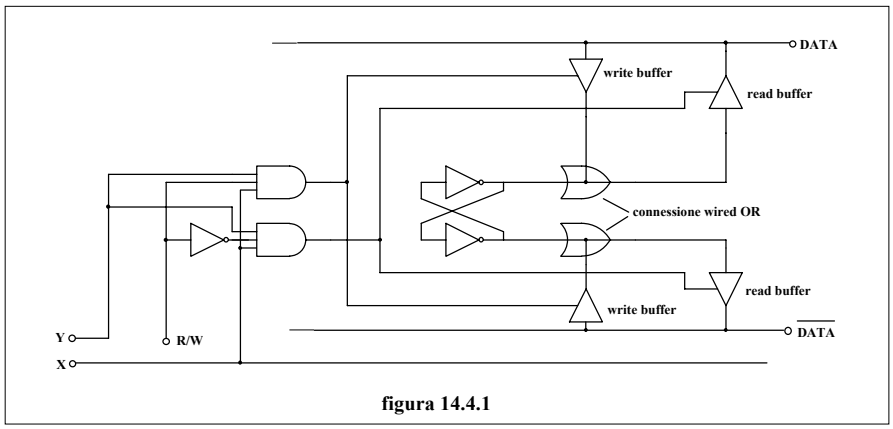


figura 14.4.1

Quando sia X che Y sono a livello basso, ambedue le uscite degli AND si trovano a livello basso e pertanto i quattro buffer 3-state risultano disabilitati. Quando invece X, Y e

R/W sono alti risultano abilitati i write buffer permettendo un'operazione di scrittura nella cella; cambiando lo stato della linea R/W si disabilitano i write buffer mentre vengono abilitati i buffer di lettura. Il valore contenuto nella cella viene in tal caso trasferito alle linee dati.

Realizzare una memoria RAM utilizzando la cella appena descritta e' molto semplice. In fig. 14.4.2 e' riportato lo schema di interconnessione per una memoria da 4 bit, per la quale sono necessarie solo quattro porte in piu' rispetto a quelle necessarie per la realizzazione delle singole celle.

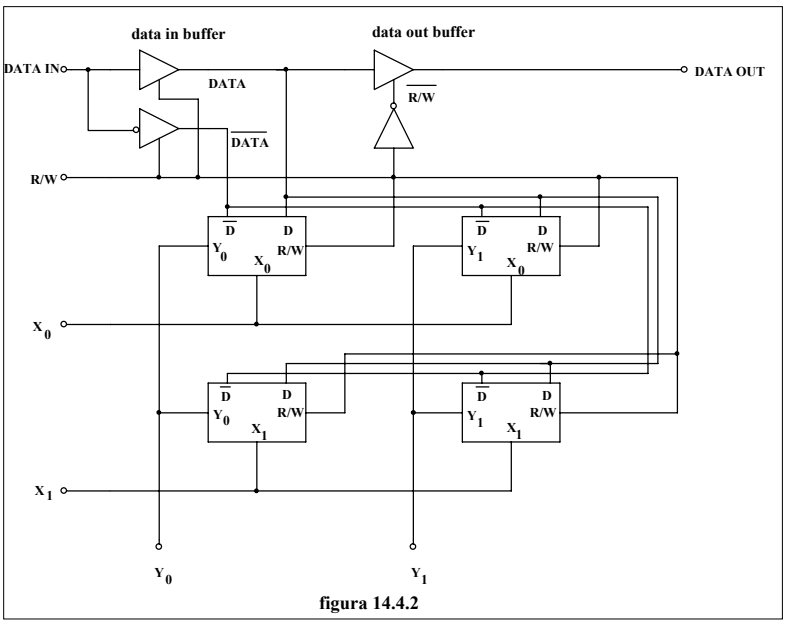


figura 14.4.2

Un'osservazione piu' attenta della cella proposta fa tuttavia osservare che le esigenze di progetto enunciate all'inizio del paragrafo 14.4 non sono sufficientemente rispettate. Infatti un circuito costituito da 8 porte, (escludendo le connessioni "wired-or" che non occupano uno spazio significativo) da 5 linee di segnale e da almeno 2 linee di alimentazione non puo' certo considerarsi efficiente in termini di superficie di silicio occupata.

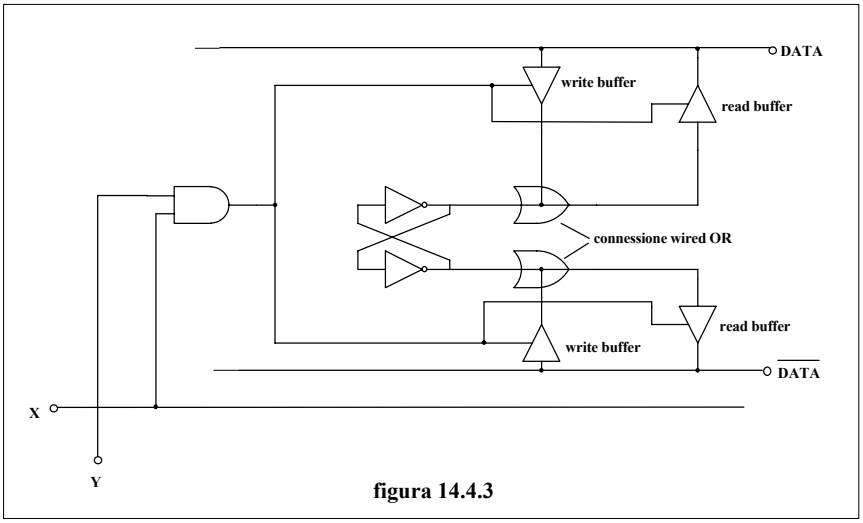


figura 14.4.3

Un primo miglioramento si puo' ottenere eliminando la necessita' di una linea di read/write per ogni cella, secondo quanto illustrato in fig. 14.4.3.

Le due porte AND a tre ingressi della cella originaria si riducono ad un'unica porta AND a due ingressi che controlla sia il buffer di lettura che quello di scrittura. In questo caso si puo' allora sostituire la coppia di buffer con un semplice transistore MOS utilizzato come interruttore.

La cella che cosi' si ricava potrebbe gia' essere accettabile, ma un ulteriore livello di semplificazione puo' essere raggiunto eliminando la porta AND e riorganizzando la configurazione della memoria in modo tale che tutte le celle di una riga siano parzialmente accessibili, mentre solo la cella della colonna selezionata lo sia totalmente. La nuova configurazione circuitale e' illustrata in fig. 14.4.4 mentre in fig. 14.4.5 e' riportato lo schema completo di una RAM da 4 bit.

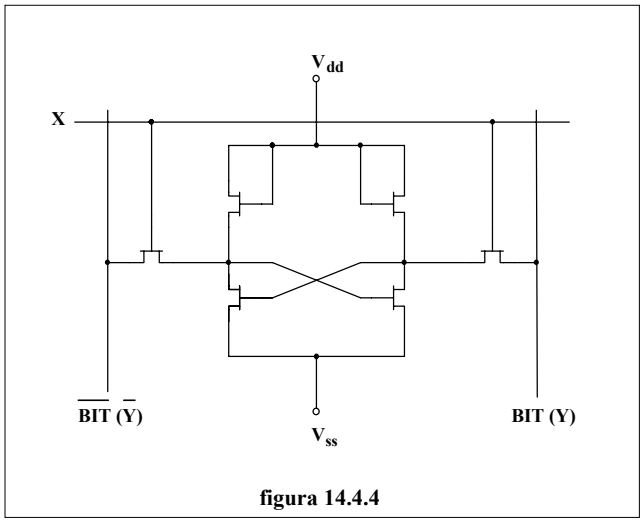


figura 14.4.4

La cella cosi' ottenuta e' realizzata con soli sei transistori e le linee di segnale si sono ridotte a tre. E' questa la struttura tipo della cella utilizzata nella maggior parte dei casi per realizzare memorie RAM statiche.

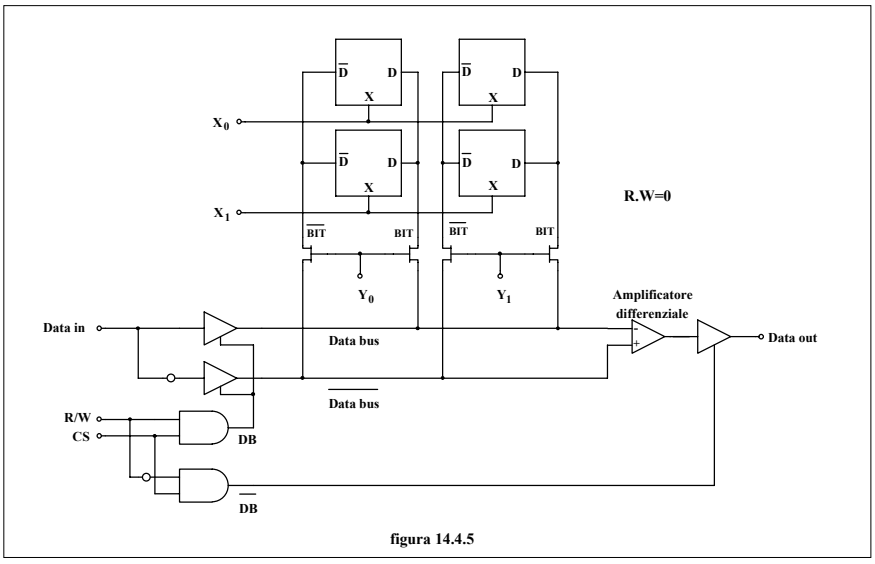


figura 14.4.5

La costruzione di una memoria completa, che utilizzi la cella a sei transistori, introduce alcuni nuovi concetti e terminologie. Significativa e' la sostituzione delle linee dei dati con le

cosiddette "bit line" che alimentano un bus dati comune. Le "bit line" sono isolate dal bus dati da interruttori MOS comandati dall' indirizzo di colonna.

Il bus dati a sua volta e' pilotato da una coppia di buffer 3-state in fase di scrittura, mentre in fase di lettura comunica con l'esterno tramite un amplificatore differenziale e un ulteriore buffer 3-state. Il controllo della RAM e' modificato dalla presenza del segnale CS di "chip select", che negli esempi fatti precedentemente non era presente. Tale segnale si rende necessario in quanto le RAM, in un sistema di memoria, sono disposte in modo matriciale come le celle che costituiscono la singola RAM ed i vari chip condividono lo stesso bus dati per trasmettere le informazioni. Ovviamente deve essere selezionato un solo chip per volta, mentre gli altri devono essere completamente ininfluenti sia per quanto riguarda i dati di ingresso che quelli di uscita. Questo risultato puo' essere raggiunto tramite il segnale CS di fig. 14.4.5.

Si supponga, a titolo di esempio, di voler modificare il contenuto della cella X_0, Y_1 da "0" a "1". I segnali da applicare alla memoria saranno allora X_0, Y_1 e CS alti, mentre X_1 e Y_0 dovranno essere tenuti bassi. Con questa combinazione di ingresso vengono attivati i buffer di ingresso e di conseguenza DB si mettera' a livello alto e $\overline{DB}$ a livello basso. Y_1 poi chiude gli interruttori della relativa colonna, mentre X_0 chiude gli interruttori MOS della cella X_0, Y_1.

Le "bit line" a questo punto si trovano in una situazione di conflitto per effetto del dato gia' memorizzato in precedenza nella cella e del nuovo valore da scrivere; tuttavia con un opportuno dimensionamento della parti componenti si puo' far si' che il nuovo dato prevalga, forzando la cella nella nuova situazione.

Durante queste operazioni il segnale R/W, a livello basso, mantiene il buffer di uscita disattivato, in modo che durante il ciclo di scrittura il dato presente sul bus dati non venga trasferito in uscita.

Si noti che gli interruttori MOS della cella X_0, Y_0 sono anch'essi chiusi, trasferendo quindi il contenuto della cella sulle "bit line" della colonna Y_0. Tuttavia il valore basso di X_1 impedisce un'interazione con la cella X_1, Y_0, mentre il livello basso di Y_0 impedisce il trasferimento del dato dalle "bit line" al bus dati.

In modo del tutto analogo puo' essere compiuta l'operazione di lettura.

14.4.2) Struttura di una cella RAM dinamica.

Una cella RAM dinamica si puo' considerare derivata da quella statica di fig. 14.4.4, in cui tuttavia siano stati eliminati i transistori di carico. Il dato viene in questo caso memorizzato nelle capacita' parassite di gate, come illustrato in fig. 14.4.6 (a).

In questa struttura circuitale si tende pertanto a perdere il dato se non si interviene con un'operazione di riscrittura prima che la tensione ai capi della capacita' sia scesa al di sotto di un determinato livello. Quest'operazione di riscrittura e' detta rinfresco (refresh) del dato.

Nella cella a quattro transistori tuttavia quelli T_1 e T_2 funzionano in modo complementare e di conseguenza si e' in presenza di una ridondanza di informazione.

Una cella ottenuta eliminando uno di questi transistori e' la cella a tre transistori di fig. 14.4.6 (b). In fase di lettura il dato e' decodificato dallo stato del transistore T_1, in conduzione o meno in funzione della carica immagazzinata nella capacita' parassita gate-source. Pertanto quando T_3 e' in conduzione T_1 controlla il passaggio di corrente sulla linea dei dati (R) e funge quindi da convertitore tensione-corrente.

Quando poi non si consideri la funzione di scrittura del dato, si puo' giungere alla cella a 1 transistore sempre illustrata in fig. 14.4.6 (c).

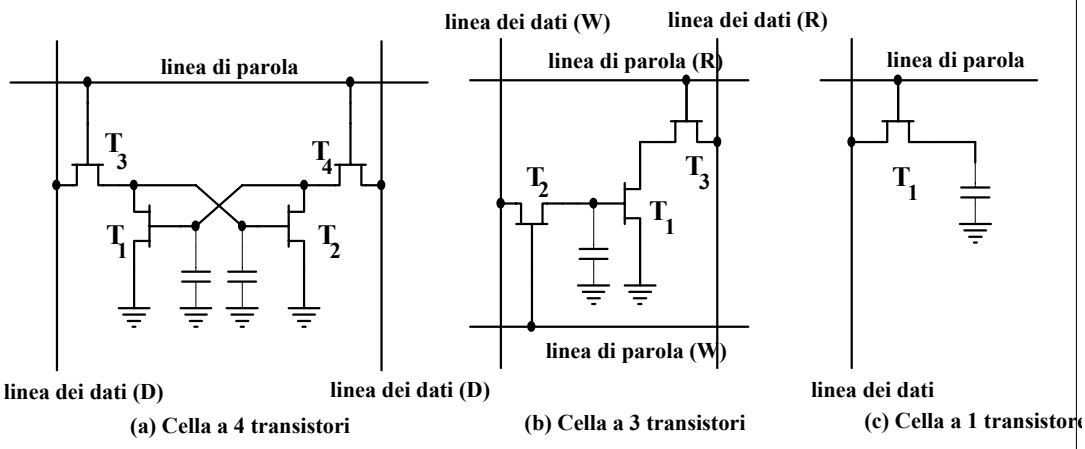


figura 14.4.6

L'operazione di lettura in tal caso avviene come segue. In un primo tempo la tensione sulla linea dei dati viene mantenuta allo stato alto; successivamente, quando T_3 viene portato in conduzione alzando il livello di tensione della linea di parola, un amplificatore a soglia (sense amplifier) legge uno "0" o un "1", come illustrato in fig. 14.4.7, in funzione della variazione di tensione che viene a verificarsi o meno sulla linea dei dati.

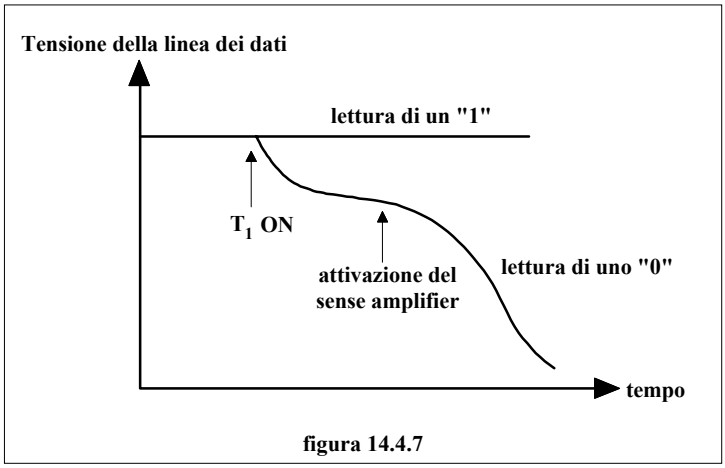


figura 14.4.7

Ogni linea dei dati e' collegata a un "sense amplifier", che corrisponde al circuito di controllo di I/O di fig. 14.3.3.

Per ottenere elevate sensibilita' e per questioni di stabilita' il "sense amplifier" e' spesso realizzato in configurazione differenziale.

14.4.3) Struttura di una memoria RAM dinamica.

A titolo di esempio verranno illustrati in questo paragrafo i principi secondo i quali funziona una semplice memoria RAM dinamica a 4 bit, il cui schema e' riportato in fig. 14.4.8.

Le forme d'onda che controllano questo circuito sono riportate in fig. 14.4.9.

Si osservi che un ciclo, sia esso di lettura oppure di scrittura, puo' essere suddiviso in due periodi distinti, detti rispettivamente periodo attivo e periodo di precarica.

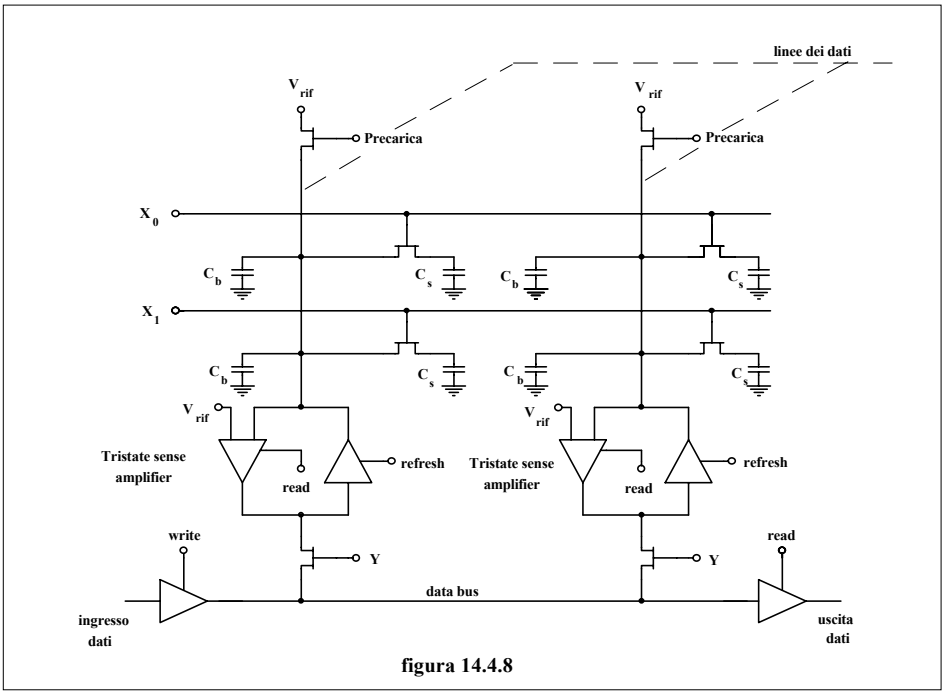


figura 14.4.8

Durante il periodo attivo vengono eseguite le operazioni richieste, cioe' la lettura o la scrittura di un dato, mentre durante il periodo di precarica la memoria viene preparata per le operazioni successive. Durante quest'ultimo periodo, con riferimento alla fig. 14.4.8, non si fa altro che connettere le linee dei dati, mediante i rispettivi MOS controllati dall'impulso di precarica, alla tensione di riferimento V_{rif}, di valore intermedio tra i livelli di tensione corrispondenti allo "0" e all'"1" logico.

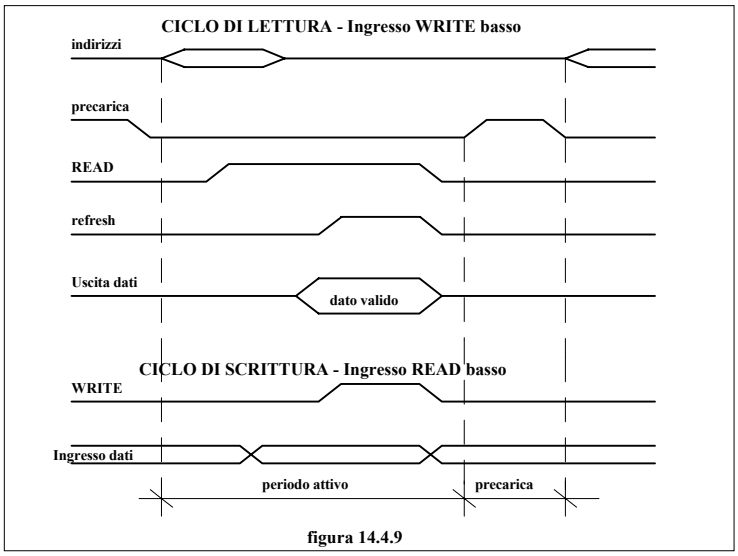


figura 14.4.9

Alla fine del periodo di precarica la linea dei dati si trovera' ad una tensione che in pratica coincide con V_{rif}, poiche' l'amplificatore di rinfresco si trova nel suo stato ad alta impedenza.

Appena terminato il periodo di precarica le linee di indirizzo selezionate si portano a livello alto. La linea X fara' condurre i MOS delle due celle situate sulla medesima riga, connettendo le corrispondenti capacita' C_s alle linee dei dati. Di conseguenza le cariche della capacita' C_s e della capacita' C_b si ridistribuiscono in modo che sulla linea dei dati si abbia una tensione pari a:

$$V_R = \frac{(C_s . V_s + C_b . V_{rif})}{(C_s + C_b)}$$

Tale tensione viene confrontata dal "sense amplifier" con V_{rif} e quindi la rivelazione del dato memorizzato in ciascuna cella avviene senza difficolta'. Il dato della colonna selezionata passa poi sul bus dati e, tramite il "data out buffer", passa al terminale di uscita.

Se a questo punto il ciclo terminasse, i dati presenti nelle celle lette risulterebbero fortemente degradati; la lettura sarebbe cioe' di tipo distruttivo. Gli amplificatori di refresh ovviano a tale inconveniente, ripristinando i corretti valori di tensione sulle celle. E' ovvio che tutte le celle accessibili dalla stessa linea devono venir rinfrescate contemporaneamente.

Si noti che nel ciclo di lettura descritto, durante il periodo attivo, l'indirizzamento termina prima di qualsiasi altro segnale, allo scopo di proteggere i dati contenuti nelle celle, che potrebbero deteriorarsi se piu' segnali commutassero contemporaneamente prima che la cella fosse stata isolata dal resto del sistema.

L'operazione di scrittura e' relativamente piu' semplice. Il segnale **write** abilita il dato presente in buffer, che attraverso l'amplificatore di refresh viene trasferito sulla linea dati desiderata.

Si noti che ciascuna cella richiede solo due componenti, due linee di segnale e una linea di alimentazione (la linea di massa). Inoltre in condizioni di riposo la cella non dissipa potenza. A fronte di questi vantaggi si ha un aumento del tempo di accesso, determinato dal periodo di precarica e dalla necessita' di assicurare che ciascuna riga sia periodicamente indirizzata per assicurare il refresh dei dati.

Oltre alla necessita' di refresh e' inoltre di considerevole importanza l'istante di inizio del funzionamento del "sense amplifier". RAM dinamiche veloci richiedono che il "sense amplifier" inizi a funzionare il prima possibile; tuttavia anticipare troppo l'inizio del funzionamento riduce eccessivamente l'ampiezza della tensione sulla linea dei dati e puo' dar luogo a fenomeni di instabilita'. La relazione ricavata per V_R vale infatti solo a regime.

La miglior temporizzazione e' funzione di diversi fattori, quali il rapporto tra la capacita' delle linee dei dati e quelle delle celle di memoria, la sensibilita' del "sense amplifier", la generazione interna di rumori e altre condizioni.

Le considerazioni sul rumore, molto importanti poiche' il "sense amplifier" deve operare su livelli di tensione abbastanza bassi, portano a particolari geometrie nella disposizione delle celle di memoria, che tendono ad ottimizzare la velocita' di funzionamento e l'immunita' al rumore. Non si ritiene tuttavia utile in questa sede approfondire tale argomento.

14.5) Esempio di utilizzazione di memorie ROM.

Nel presente paragrafo verra' esaminato l'impiego di memorie ROM al fine di utilizzare un monitor televisivo come display alfanumerico.

Il principio di funzionamento di tale sistema, capace di visualizzare 16 righe, ciascuna contenente un massimo di 64 caratteri, puo' essere visto con l'ausilio dello schema a blocchi di fig. 14.5.1.

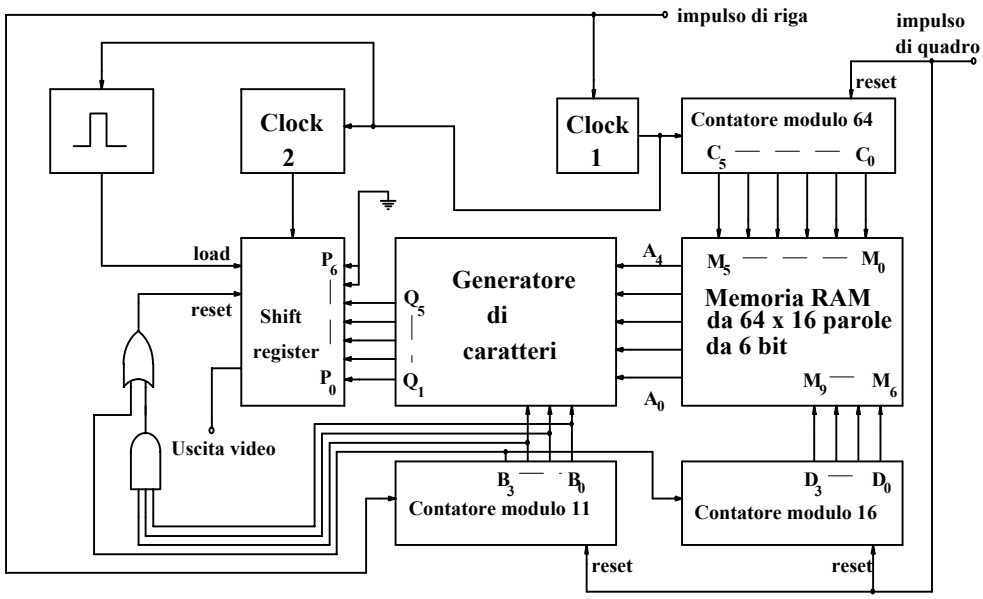


figura 14.5.1

Ogni carattere sia separato da quello successivo da due spazi, e ogni linea di caratteri sia separata dalla successiva da quattro spazi, come illustrato in fig. 14.5.2.

In tal modo, supponendo che ciascun carattere sia realizzato con una matrice 5 x 7, l'effettivo spazio riservato ad ogni carattere, comprensivo dell'area di separazione tra caratteri e tra righe, risulta essere di 7 x 11.

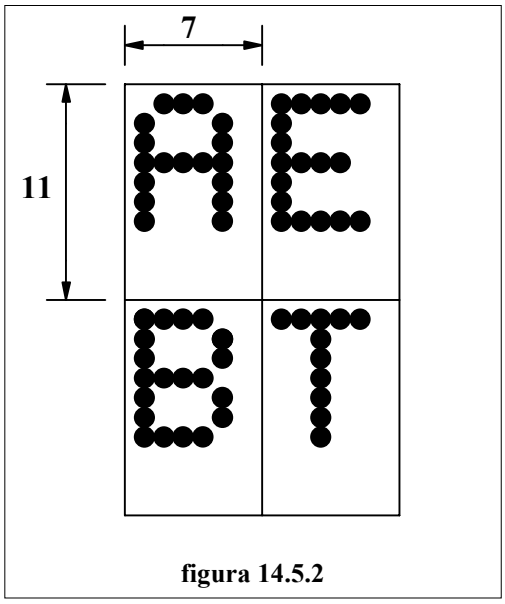


figura 14.5.2

L'intera pagina alfanumerica visualizzata venga poi prelevata da una memoria RAM da 64 x 16 = 1024 parole da 6 bit, in ciascuna delle quali e' contenuto il codice del carattere da

visualizzare. I bit di indirizzo di questa memoria saranno pertanto 10 (M_0-M_9) di cui i primi 6 (M_0-M_5) individuano la posizione sulla riga in cui visualizzare il carattere, gli ultimi quattro (M_6-M_9) una delle 16 righe costituenti la pagina. I 6 bit del dato (A_0-A_5), uscita della parola indirizzata, vengono inviati all'ingresso del generatore di caratteri.

Il funzionamento puo' essere descritto nel modo seguente. L'impulso di quadro azzera:

1) Il contatore modulo 64, necessario a indirizzare i 64 caratteri che formano ciascuna riga.
2) Il contatore modulo 16 necessario ad indirizzare ciascuna riga della pagina.
3) Il contatore modulo 11 utilizzato per la scansione delle sette righe che costituiscono ciascun carattere (ingressi R_0, R_1, R_2 del generatore di caratteri) e le quattro righe che costituiscono la spaziatura verticale.

L'impulso di riga e' inviato al contatore modulo 11 e al generatore di clock ($clock_1$) che fornisce un treno di 64 impulsi all'ingresso del contatore modulo 64.

Un secondo generatore di clock ($clock_2$), sincronizzato con il $clock_1$, permette di serializzare, utilizzando uno shift-register, l'uscita parallela (Q_1,..,Q_5) del generatore di caratteri. Q_1-Q_5 sono i cinque bit che formano una linea del carattere e vengono inviati ai primi cinque ingressi paralleli (P_0, .., P_4) dello shift-register, mentre gli ultimi due ingressi paralleli (P_5 e P_6) vengono mantenuti costantemente a zero in modo da realizzare la spaziatura orizzontale tra i diversi caratteri della stessa riga.

Pertanto per ogni impulso fornito dal $clock_1$ si hanno 7 impulsi di $clock_2$.

I sette bit presenti agli ingressi paralleli dello shift-register vengono caricati nel registro stesso con un opportuno ritardo in modo da evitare errori di caricamento per mezzo di un segnale di "load" ricavato dal $clock_1$.

Stato del contatore	B_3 B_2 B_1 B_0	Reset
0	0 0 0 0	0
1	0 0 0 1	0
2	0 0 1 0	0
3	0 0 1 1	0
4	0 1 0 0	0
5	0 1 0 1	0
6	0 1 1 0	0
7	0 1 1 1	1
8	1 0 0 0	1
9	1 0 0 1	1
10	1 0 1 0	1

$$Reset = B_3 + B_0 . B_1 . B_2$$

figura 14.5.3

Il contatore modulo 11 effettua il conteggio degli impulsi di riga e fornisce i tre bit di indirizzamento B_0, B_1 e B_2 agli ingressi R_0, R_1 e R_2 del generatore di caratteri, in modo da

individuare in sequenza, durante i primi 7 impulsi di ciascun gruppo di 11 impulsi, le sette righe che costituiscono il carattere, mentre durante i successivi quattro impulsi e' necessario azzerare totalmente lo shift-register, in modo da realizzare la spaziatura verticale.

E' necessario pertanto realizzare la semplice funzione logica illustrata in fig. 14.5.3.

Il contatore modulo 16 totalizza i gruppi di 11 linee video, cioe' incrementa il suo stato ogni 11 impulsi di riga. Le sue uscite (D_0, ..., D_3) vengono inviate alla memoria RAM (M_6-M_9) in modo da indirizzare successivamente le 16 righe che costituiscono la pagina.

Infine i sei bit di uscita del contatore modulo 64 (C_0-C_5) sono connessi ai primi sei bit di indirizzo della RAM, per individuare in successione la locazione che contiene il codice di ciascuno dei 64 caratteri che costituiscono la riga.

14.6) I microprocessori.

Al capitolo X sono stati presentati i problemi relativi ai sistemi a larga scala e si e' pervenuti a un possibile formalismo per la rappresentazione di sistemi sequenziali con un elevato numero di stati. E' opportuno ricordare che la soluzione proposta, a partire da un modello di circuito suddiviso in una parte sequenziale di controllo e in una parte di memorizzazione ed elaborazione dei dati, conduceva alla messa a punto di una sorta di linguaggio di programmazione in grado di descrivere in modo non equivoco il funzionamento di qualsiasi circuito, per quanto complesso esso fosse.

Il logico sviluppo di questo modo di affrontare il problema consiste evidentemente nel passare dai circuiti in logica cablata, sia pure descritti attraverso un linguaggio [regolare], a dispositivi LSI effettivamente programmabili, in modo che le funzioni logiche di un certo dispositivo non siano piu' realizzate da un hardware, ma da un programma, cioe' da una sequenza di istruzioni memorizzate in un qualche dispositivo (normalmente una RAM). Tali considerazioni hanno condotto i costruttori a realizzare circuiti LSI utilizzabili dal massimo numero di clienti nella applicazioni piu' svariate. Nascevano in tal modo ,verso l'inizio degli anni 70, i microprocessori, la cui struttura e', al livello minimo, quella di un'unita' centrale di un calcolatore di uso generale. Sostanzialmente la struttura e' quella schematicamente illustrata in fig. 14.6.1 in cui si nota immediatamente l'analogia con il modello introdotto al paragrafo 10.1.

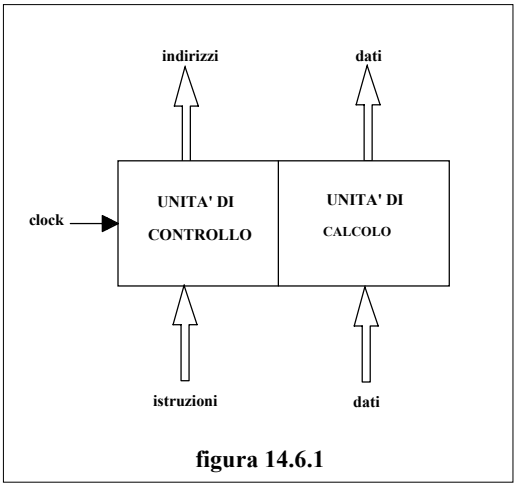


figura 14.6.1

La flessibilita' del sistema, alternativo rispetto a logiche cablate TTL o CMOS, e' molto elevata, in quanto con la sostituzione del programma si puo' cambiare completamente la funzione svolta o comunque adattarla facilmente a nuove esigenze nel frattempo intervenute.

Malgrado la scarsa potenzialita' dei primi microprocessori introdotti, il successo delle logiche programmate e' stato immediato e si e' via via espanso in rapporto ad una serie di vantaggi conseguibili, che sono sintetizzabili nelle voci:

1) Costo

2) Tempo di sviluppo di un progetto

3) Affidabilita' del sistema

In effetti la tendenza del mercato, a partire dal momento in cui i microprocessori sono stati introdotti, e' stata quella di offrire dispositivi sempre piu' sofisticati e potenti a costi sempre minori. Non e' azzardato affermare che i microprocessori moderni offrono a prezzi contenutissimi prestazioni addirittura superiori a quelle che solo poco tempo prima erano proprie dei minicalcolatori.

Oltre a cio' un sistema basato sull'utilizzo del microprocessore presenta dei notevoli vantaggi per quanto riguarda la velocita' del progetto e della messa a punto. Molto spesso si puo' fare riferimento a un'unita' standard e a schede modulari utilizzabili in diversi progetti, mentre il massimo sforzo progettuale va rivolto alla messa a punto del software. In ogni caso i microprocessori presenti sul mercato offrono dei supporti software non indifferenti, permettendo una notevole riduzione dei tempi di sviluppo.

Un sistema a microprocessore infine presenta un'affidabilita' maggiore che non un sistema a logica cablata. Infatti la maggior parte delle funzioni logiche richieste sono realizzate in un unico chip; diminuisce pertanto il numero di componenti e di interconnessioni e diminuisce corrispondentemente la probabilita' di guasto.

Oltre ai vantaggi citati e' appena opportuno far notare che i sistemi a microprocessore possono essere modificati, in modo anche pesante, con estreme facilita'. Tale fatto permette un adeguamento molto rapido alle richieste del mercato, giocando un ruolo molto favorevole in relazione all'obsolescenza dei prodotti.

14.7) Approccio al progetto di sistemi a microprocessore.

Da quanto e' stato appena detto discende immediatamente che un progettista che voglia far uso di microprocessori deve conoscere non solo i problemi relativi alle caratteristiche elettriche e funzionali dei dispositivi usati, ma deve possedere anche una buona conoscenza delle tecniche fondamentali della programmazione. La conoscenza delle caratteristiche elettriche e funzionali si rende necessaria per realizzare, a partire dal microprocessore un microcalcolatore, la cui struttura tipica e' riportata in fig. 14.7.1.

Le nozioni di programmazione si rendono invece necessarie per la messa a punto del programma da inserire in ROM. Le conoscenze in campo hardware e quelle relative alla programmazione non sono tuttavia slegate tra di loro, ma interagiscono nel corso del progetto; una determinata realizzazione circuitale vincola nello sviluppo del software, cosi' come la scelta di un determinato algoritmo puo' rendere obbligate determinate soluzioni circuitali. Non e' pertanto corretto affermare, come ancora qualcuno fa, che l'avvento dei microprocessori abbia ridotto il progetto logico ad una pura messa a punto di nuovi software.

Quale che sia l'impostazione che si vuole dare al progetto, o con i soli circuiti a bassa e media scala di integrazione o con i microprocessori, e' necessario in primo luogo analizzare la funzione che il progetto vuol realizzare, in modo da ricavarne uno schema di flusso in cui siano evidenziati gli stati fondamentali della macchina. Per specificare le transizioni tra uno stato e l'altro e' necessario poi conoscere la configurazione dei segnali di ingresso (comandi e

dati) e le loro temporizzazioni. Infine si deve valutare la quantita' dei dati che devono essere disponibili alla macchina per le successive elaborazioni, in modo da definire la quantita' di memoria necessaria.

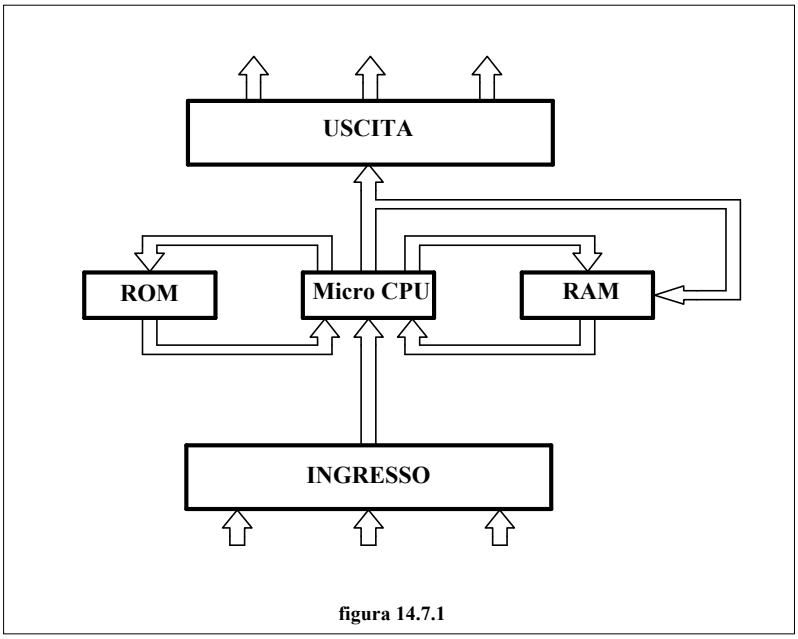


figura 14.7.1

Stabilite queste caratteristiche fondamentali, si deve poi scegliere la tecnologia con cui realizzare il progetto. Se fino a qualche anno fa la scelta poteva poteva venir fatta essenzialmente sulla base della dissipazione di potenza (tecnologie NMOS o PMOS, CMOS, I^2L) e della velocita' (TTL, STTL, ECL), l'avvento dei microprocessori ha introdotto un nuovo livello di scelta; occorre cioe' scegliere se sia piu' opportuno realizzare il progetto con circuiti SSI e MSI oppure con circuiti microprocessori LSI.

Il criterio fondamentale di scelta e' il rapporto costo-prestazioni, risultato di molti fattori. Uno schema operativo semplificato per compiere tale scelta e' riportato in fig. 14.7.2.

E' bene tuttavia commentare, sia pure molto sinteticamente, alcuni dei fattori che condizionano la decisione.

a) **Quantita' delle funzioni da realizzare.**

Tale parametro e' approssimativamente proporzionale al numero di elementi SSI e MSI necessari. Usando invece microprocessori tale proporzionalita' si sposa invece alla dimensione della memoria di programma.

b) **Parallelismo dei segnali e dei dati.**

Questo parametro gioca un ruolo notevolmente importante quale criterio di scelta. I microprocessori sono infatti strutturati per elaborare dati con parallelismo 4, 8 , 16 o piu'. E' abbastanza evidente che ottenere le stesse prestazioni con circuiti SSI e MSI e' notevolmente piu' difficoltoso e costoso.

c) **Espandibilita' del sistema.**

Utilizzando microprocessori l'espansione il piu' delle volte si riduce all'aggiunta di circuiti di I/O e di nuova memoria. Con circuiti SSI e MSI la cosa e' invece notevolmente piu' difficoltosa.

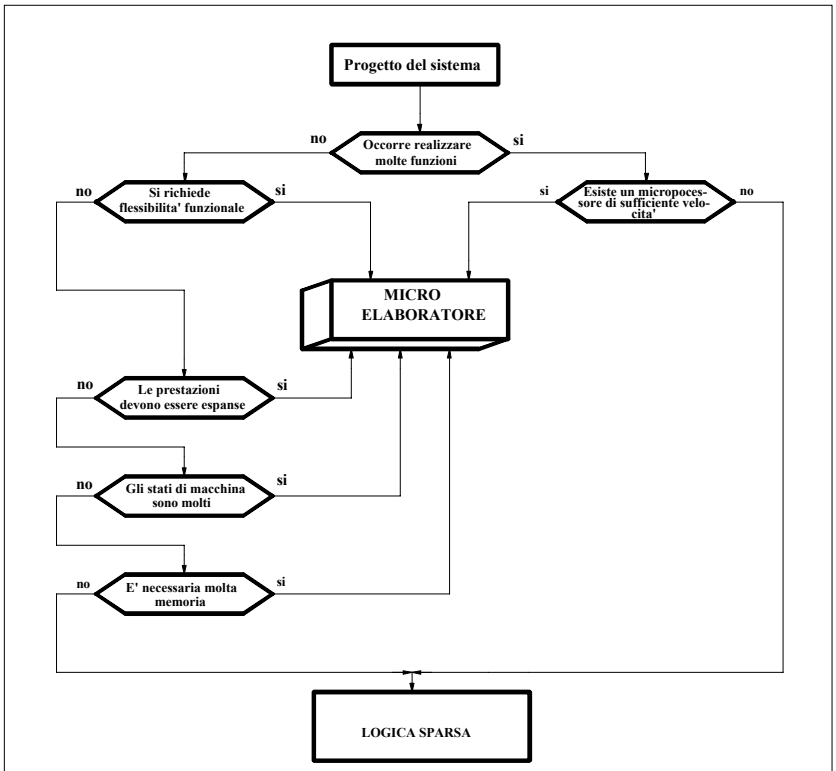


figura 14.7.2

d) **Necessita' di utilizzare temporizzazioni complesse.**

Quando sorga la necessita' di misurare con una certa precisione intervalli di tempo l'uso di microprocessori si rivela un metodo poco efficiente. E' preferibile in tal caso usare logiche SSI e MSI. Tuttavia per superare tale difficolta' sono stati sviluppati dispositivi programmabili di temporizzazione utilizzabili da microprocessore.

e) **Memoria dati.**

Se il numero di dati da elaborare e' notevole, l'uso di un microprocessore diventa pressocche' indispensabile. I microprocessori infatti contengono al loro interno tutti i circuiti (registri di indirizzo e di dati) necessari alla gestione di memorie anche di notevole dimensione.

f) **Capacita' decisionali.**

Quando il problema da risolvere prevede la necessita' di eseguire salti condizionati, cioe' di scegliere tra due stati successivi di macchina in funzione di certe condizioni di ingresso (cfr. capitolo X), l'uso di microprocessori diventa praticamente obbligato. Realizzare le stesse funzioni con circuiti MSI/SSI e' abbastanza complicato, come e' stato illustrato al capitolo X,. Un discorso del tutto analogo si puo' fare anche per i trasferimenti condizionati.

g) **Tempo di sviluppo.**

Lo sviluppo di un sistema a microprocessore puo' sembrare a prima vista piu' difficile e lungo che non quello dei tradizionali circuiti in logica cablata.

E' vero viceversa che, con l'uso degli opportuni sistemi di sviluppo hardware e software, la realizzazione di un sistema a microprocessore si ottiene normalmente in tempi piu' brevi che non quelli del corrispondente circuito in logica cablata.

Va tuttavia osservato che hardware e software non possono essere sviluppati indipendentemente, ma interagiscono tra di loro secondo lo schema di fig. 14.7.3.

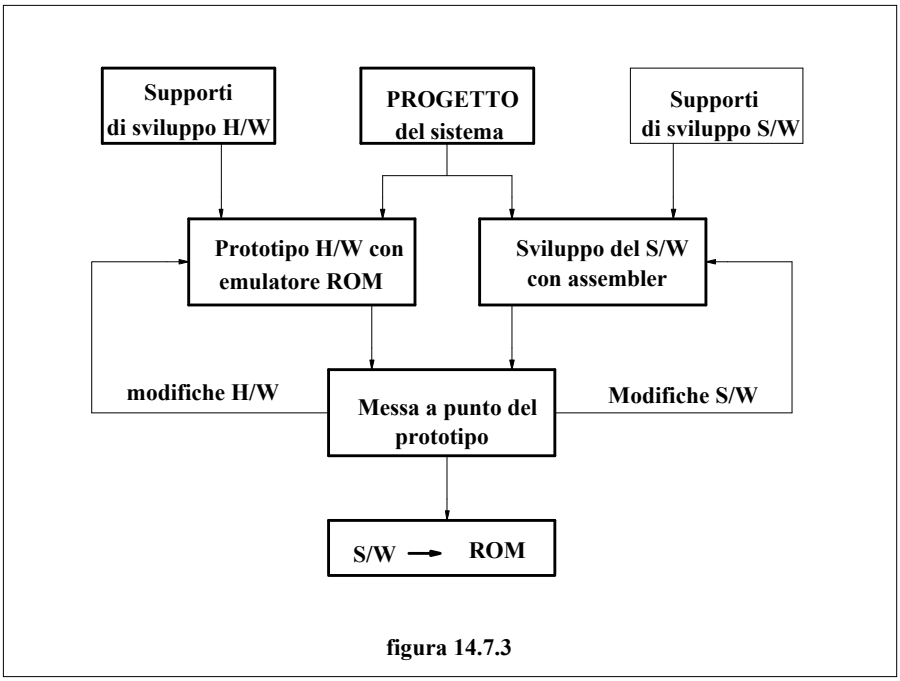


figura 14.7.3

E' opportuno far notare che caratteristica fondamentale dei sistemi di sviluppo hardware e' quella di condividere con il prototipo del sistema in progetto una memoria RAM, che emula la memoria ROM del dispositivo finale. In tale memoria e' possibile caricare, modificare quando necessario, ed esaminare in qualsiasi momento il programma che realizza le funzioni desiderate. Tale programma, ottenibile con i supporti software, puo' essere sviluppato sia nel linguaggio assemblatore del microprocessore utilizzato, come indicato in fig. 14.7.3, che in linguaggi a livello piu' elevato, quali Fortran, Pascal, C, PLM o simili.

h) **Costi.**

Nella valutazione dei costi e' necessario evidentemente, come e' stato accennato al capitolo X, tener conto dei costi accessori, quali quello dell' alimentatore e quelli dell'interconnessione. L'uso di microprocessori, portando ad una drastica riduzione di componenti, permette sotto questo punto di vista di ottenere delle notevoli economie.

i) **Affidabilita'.**

Anche per l'affidabilita' l'uso dei microprocessori comporta notevoli vantaggi, legati soprattutto alla diminuzione di saldature, contatti, cavi e connettori.

14.8) Struttura elementare di un microprocessore.

Si e' gia' visto al paragrafo precedente che un microprocessore puo' essere considerato suddiviso in due blocchi fondamentali, chiamati rispettivamente unita' di calcolo e unita' di controllo.

E' conveniente a questo punto esaminare con maggior dettaglio l'unita' di calcolo; a questo scopo e' opportuno appoggiarsi ad un semplice esempio che permetta di esaminare in modo non eccessivamente astratto le strutture interne del microprocessore.

Si prenda pertanto in considerazione un microcalcolatore che debba eseguire le quattro operazioni aritmetiche su numeri decimali di otto cifre. La struttura di tale microcalcolatore sara' quella illustrata in fig. 14.8.1.

I dati, cioe' gli operandi e i comandi di operazione, pervengono al microprocessore tramite l'ingresso I; dati e risultati parziali saranno registrati nella memoria M, i risultati finali inviati all'uscita U. Ingresso e uscita possono essere ad esempio una tastiera e un display con cifre a 7 segmenti. Infine i dati siano espressi in codice BCD.

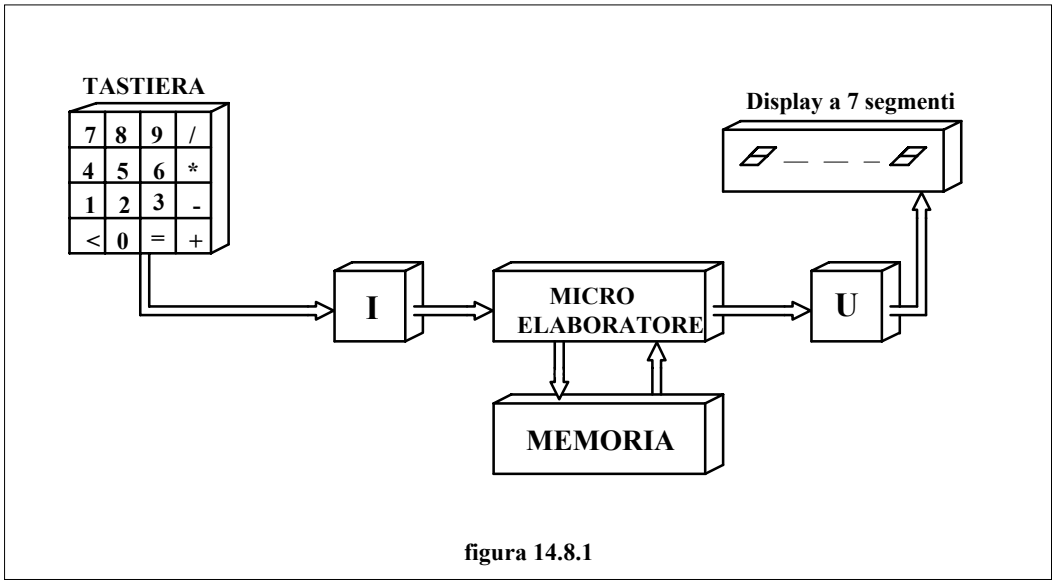


figura 14.8.1

La prima cosa su cui e' necessario fissare l'attenzione e', come si e' detto, se la scelta di un microprocessore sia effettivamente preferibile ad una realizzazione SSI/MSI. Nel presente caso essa si rivela opportuna per i seguenti motivi:

1) Parallelismo dei dati pari a 4, facilmente gestibile con un microprocessore, molto piu' difficile da affrontare invece con una realizzazione in logica sparsa.
2) Gestione dei dati. Ogni dato e' composto da 8 cifre e il risultato puo' avere fino a 16 cifre. E' necessario pertanto gestire una memoria di almeno 32 parole da 4 bit. La cosa e' semplice utilizzando un microprocessore, non altrettanto in logica sparsa.
3) Espandibilita' del sistema. Nel caso di un microprocessore l'aggiunta di nuove operazioni consiste nell'espansione della memoria di programma e nella messa a punto di nuovi algoritmi e del relativo software. E' invece pressocche' impossibile con dispositivi realizzati in logica cablata.
4)Costo. Esistono sul mercato microprocessori con il grado di parallelismo richiesto a prezzi paragonabili a quelli della singola parte MSI.

Il flusso elementare delle operazioni e' riportato in fig. 14.8.2.

Con i classici metodi di progetto in logica sparsa ad ogni blocco del diagramma di flusso viene associato uno stato di macchina ed a ciascuno di questi stati viene associata una sequenza piu' o meno lunga di operazioni elementari.

In una realizzazione a microprocessore invece, ad ogni blocco del diagramma viene associata una routine, cioe' un insieme di istruzioni. Tali istruzioni tuttavia lavorano su reti hardware, che devono venir specificate assieme al programma. In altre parole dallo schema di flusso di fig. 14.8.2 e' necessario estrarre non solo sequenze di istruzioni, o in alternativa uno schema circuitale, ma le due cose contemporaneamente.

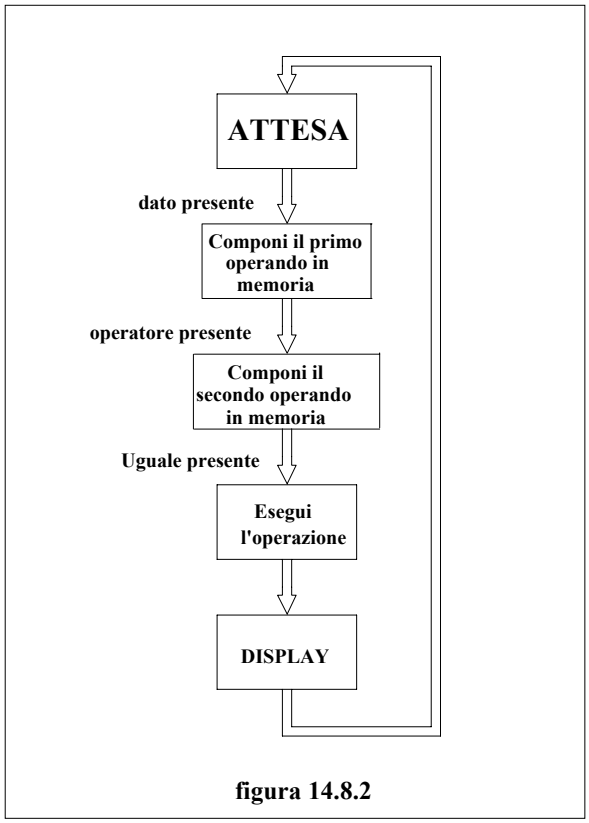


figura 14.8.2

Con riferimento al microcalcolatore di cui ci si sta occupando, si stabilisca che:

a) I due operandi di otto cifre siano allocati in memoria alle posizioni 0-7 e 8-15 rispettivamente, mentre il risultato sia memorizzato alle posizioni 16-31.

b) Le cifre meno significative siano rispettivamente agli indirizzi 0,8 e 16.

Esaminando ora il diagramma di flusso di fig. 14.8.2 e interpretandolo sulla base di un possibile algoritmo di somma, si ricava che le reti hardware associate all'operazione voluta sono:

1) Un registro R_c in cui caricare la costante "8", lunghezza degli operandi.
2) Un flip-flop di riporto.
3) Un registro indice RI (di almeno tre bit).
4) Un registro degli indirizzi di memoria (almeno 5 bit).
5) Una memoria RAM di almeno 32 parole.
6) Un registro accumulatore AC.
7) Una rete logico-aritmetica (ALU - Aritmetic Logic Unit) in grado di eseguire le operazioni sui due operandi e sul riporto.
8) Un bus da memoria verso accumulatore e ALU.
9) Un bus da accumulatore a memoria.
10) Un bus da registro indirizzi a memoria.
11) Una rete per decidere quando la routine e' finita.
12) Una rete che abiliti le vie logiche richieste dalle singole istruzioni (decodifica delle istruzioni).

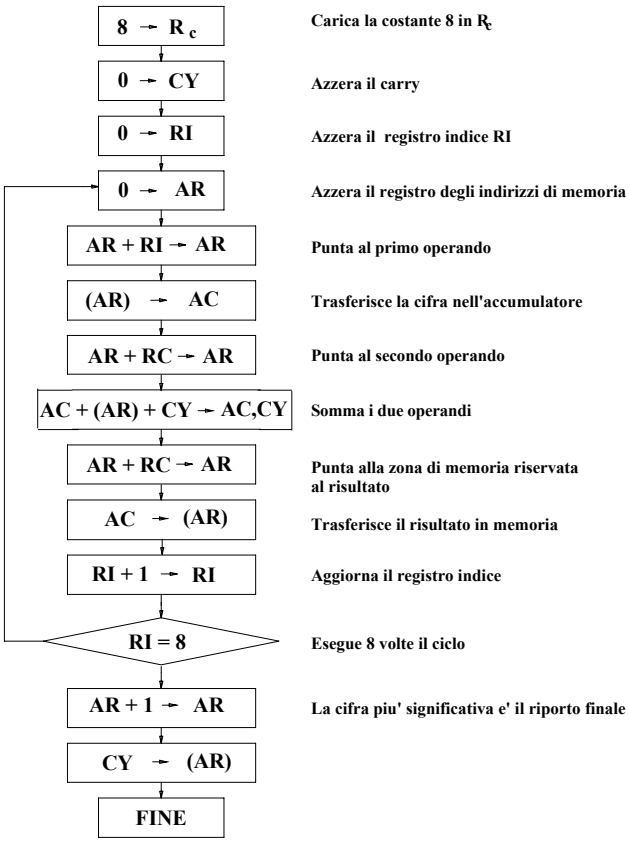


figura 145.8.3

Lo schema di flusso delle singole operazioni della routine di somma, riferite alla struttura circuitale proposta, e' riportata in fig. 14.8.3 mentre in fig. 14.8.4 e' riportato lo schema a blocchi funzionale del relativo circuito.

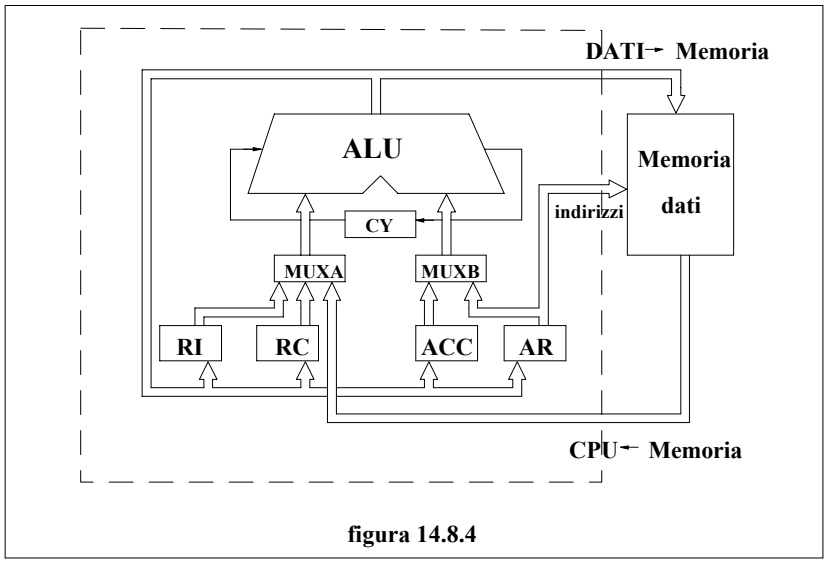


figura 14.8.4

Si nota immediatamente che in questa struttura circuitale non compare ne' la rete che decide quando la routine e' terminata, ne' quella che abilita le vie richieste dalle singole istruzioni. La parte circuitale che si vuol prendere in considerazione e' infatti unicamente l'unita' di calcolo, mentre di quella di controllo si discutera' piu' avanti.

14.8.1) Unita' di calcolo di un microprocessore.

Un esame appena un po' attento dello schema di fig. 14.8.3 rivela che esso in sostanza descrive unicamente dei trasferimenti tra registri con un'eventuale trasformazione dei dati da parte dell'ALU, che e' una rete puramente combinatoria.

Le operazioni piu' comuni eseguite sugli operandi da un'unita' logico-aritmetica sono:

- Somma binaria (con o senza riporto)
- Complementazione
- Incremento (+1)
- Decremento (-1)
- Spostamento verso destra o verso sinistra
- Somma logica bit a bit
- Prodotto logico bit a bit
- OR esclusivo bit a bit
- Confronto

Ciascuna operazione viene scelta sulla base di un opportuno codice. In sostanza l'ALU puo' eseguire, con una o piu' operazioni elementari, le operazioni scalari e vettoriali descritte al capitolo X. Cio' significa che la rete di decodifica delle istruzioni, parte del centro di controllo, deve generare:

1) Codici di operazione per l'ALU.
2) Codici di selezione per i multiplexer.
3) Codici di destinazione dell'operato.

La potenza di un microprocessore e' legata alla liberta' nelle scelta della coppia di operandi e delle destinazione del dato in uscita. Quest'ultima puo', in linea di principio, essere:

a) L'accumulatore.
b) Uno dei possibili registri interni.
c) La RAM.
d) L'uscita esterna.

In fig. 14.8.5 e' riportato uno schema semplificato dell'unita' di calcolo del micro-processore INTEL 8080, ormai abbondantemente superato, ma ancora utile a fini didattici.

Si noti che esiste un unico canale dati, in questo caso bidirezionale, su cui i dati viaggiano nella direzione stabilita dal segnale DBIN. Pertanto sono presenti alcuni registri temporanei in cui caricare gli operandi prima di utilizzarli.

Altra particolarita' notevole e' il parallelismo 16 del registro degli indirizzi di memoria, ottenuto unendo due registri H e L da 8 bit.

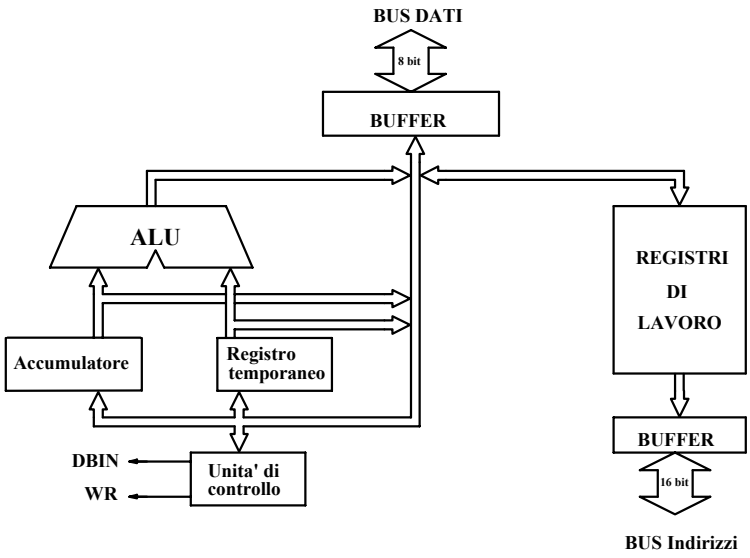


figura 14.8.5

La struttura del microprocessore presentata in fig. 14.8.5 e' di uso del tutto generale. Si vuole intendere con cio' che e' possibile risolvere ogni algoritmo in opportune sequenze di operazioni elementari di aritmetica e logica binaria. Pertanto programmando opportunamente le reti interne del microprocessore e' possibile realizzare qualsiasi progetto senza ricorrere a nessuna altra rete logica se non a registri di staticizzazione. Unica limitazione di quest'impostazione e' la velocita' operativa, in quanto il microprocessore opera serialmente.

Si ricordi che, come gia' detto, ogni operazione elementare richiede che vengano specificati operandi, operazione e destinazione dell'operato. Con riferimento ad una struttura di microprocessore ulteriormente semplificata, in cui oltre all'accumulatore vi sia un unico registro (fig. 14.8.6) si vogliano ricavare le tavole di verita' dei multiplexer, dell'ALU e quella per la destinazione del risultato.

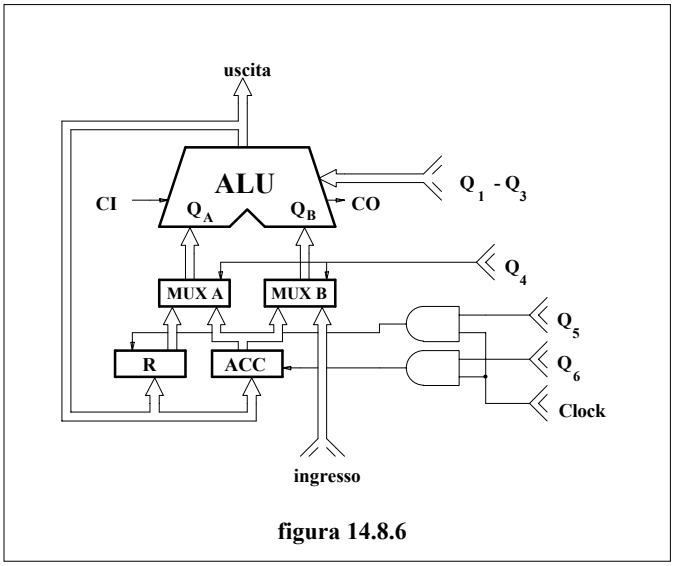


figura 14.8.6

L'insieme di bit che compare in queste tavole di verita' verra' detto "microistruzione". Ad esempio, nella rete di fig. 14.8.6 le operazioni eseguibili dall'ALU sono selezionate secondo la tabella di verita' di fig. 14.8.7.

Gli operandi trasferibili all'ALU sono:

- L'accumulatore.
- Il registro R.
- Un ingresso esterno.

$Q_1Q_2Q_3$	Operazione
000	$Q_A + Q_B$
001	$Q_A - Q_B$
010	$Q_A + 1$
011	Scorrimento
100	$Q_A \cap Q_B$
101	$Q_A \cup Q_B$
110	$Q_A \oplus Q_B$
111	$\overline{Q_A}$

figura 14.8.7

La selezione degli operandi puo' venir fatta con un ulteriore bit Q_4 secondo la seguente tabella.

Q_4	Q_A	Q_B
0	Accumulatore	Ing. esterno
1	Registro R	Accumulatore

Infine il risultato generato dall'ALU, sempre presente in uscita puo' venir caricato, utilizzando un ulteriore bit Q_5, nell'accumulatore ($Q_5 = 1$) e/o nel registro R se un sesto bit Q_6 vale 1. L'insieme dei bit Q_1 - Q_6 (microistruzione) deve essere definito ad ogni ciclo macchina per mantenere il controllo nel flusso delle operazioni ed agisce direttamente sui gate dell'unita' di calcolo.

Si puo' notare che, rispetto al modello semplificato presentato, le unita' di calcolo dei microprocessori presenti sul mercato differiscono essenzialmente per il maggior numero di registri e per la maggior liberta' che si ha nello scegliere tra questi registri gli operandi dell'ALU.

Si ritorni ora per un momento allo schema di flusso di fig. 14.8.3. L'istruzione

$$AC + [AR] + CY \rightarrow AC, CY$$

puo' evidentemente essere eseguita in un unico ciclo macchina. Istruzioni piu' complesse, che dovessero operare su dati scelti tra memoria, RC e RI (fig. 14.8.5), richiedono piu' di una microistruzione. Allo stesso modo, con riferimento alla struttura circuitale di fig. 14.8.6, mentre e' possibile con un'unica microistruzione eseguire operazioni aritmetiche o logiche tra accumulatore e ingresso esterno, non e' possibile fare la stessa cosa tra registro R e dato di ingresso.

In questo caso e' necessario risolvere l'operazione in due successive microistruzioni.

1) Dato di ingresso $\rightarrow$ Accumulatore

2) Accumulatore + registro R $\rightarrow$ Accumulatore

Questo semplicissimo esempio permette tuttavia di mettere in evidenza come, programmando opportunamente una rete logica di semplice struttura, e' possibile risolvere anche algoritmi complessi.

Vi sono evidentemente approcci alternativi per risolvere il problema di eseguire operazioni piu' complesse di quelle permesse dalla rete di fig. 14.8.6 con un'unica istruzione.

Un primo modo di operare e' quello di realizzare microprocessori con reti di "multiplexing" piu' complesse. Si ha in tal modo una scelta maggiore tra gli operandi, ma la microistruzione diventa piu' complessa in quanto e' necessario un maggior numero di bit di controllo. I programmi diventano evidentemente piu' semplici in quanto diminuisce il numero totale di istruzioni. Questa via non puo' tuttavia essere spinta oltre un certo limite; non si puo' infatti aumentare eccessivamente la complessita' dei dispositivi LSI senza imbattersi in problemi di integrabilita' e di testabilita' dei circuiti.

Un secondo modo di operare consiste nell'associare al microprocessore (nella sezione di controllo) una rete di decodifica dei comandi capace di eseguire due o piu' microistruzioni in sequenza a partire da un unico codice di istruzione. L'esecuzione dell'istruzione (non piu' microistruzione) coinvolge evidentemente piu' di un ciclo macchina e permette operazioni di complessita' medio-alta.

L'adozione di sequenziatori sempre piu' complessi e' la soluzione piu' comunemente utilizzata dai progettisti di microprocessori MOS. In piu' l'introduzione di sequenziatori che permettono il controllo del flusso dei dati risolve anche il problema del numero di piedini del dispositivo.

Un microprocessore, che appartiene gia' alla storia di questi dispositivi, ma e' sufficientemente significativo per gli scopi che ci si propone, e', come gia' accennato, l'INTEL 8080. Per tale microprocessore il parallelismi dei dati e' 8, quello degli indirizzi 16, mentre le istruzioni sono codici da 8 bit. In totale, considerando distinti i dati in ingresso e in uscita, si hanno in totale 40 segnali. Se ad essi si dovessero aggiungere tutti i segnali di controllo necessari, il numero di piedini sul chip diverrebbe eccessivo. Si e' pertanto scelta la soluzione di avere un solo bus dati bidirezionale, la cui direzione di funzionamento e' determinata dalla stato del sequenziatore interno (fig. 14.8.8).

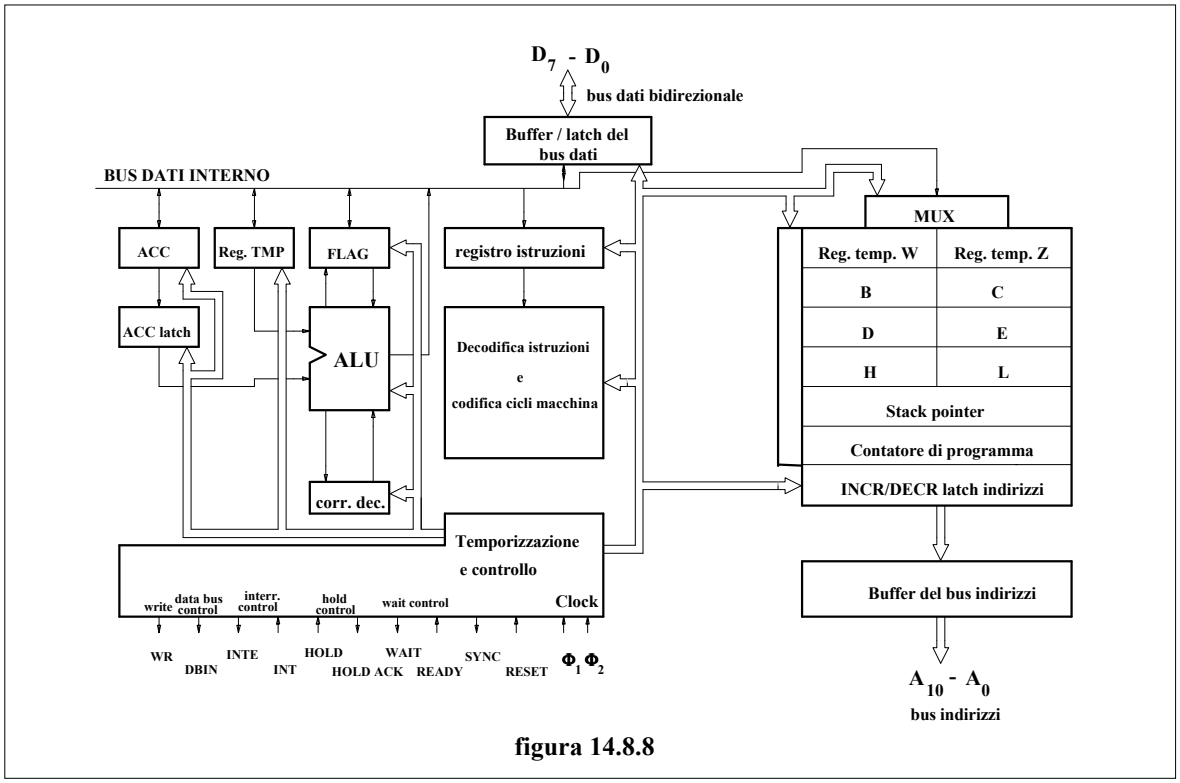


figura 14.8.8

Questo sequenziatore non solo genera i segnali interni per il controllo dei circuiti di ingresso e di uscita, ma anche quelli per i dispositivi esterni (ad esempio memorie RAM), che assieme al microprocessore formano il microcalcolatore.

Elaboratori del tipo descritto vengono chiamati "a insieme fisso di istruzioni" in contrapposizione a quelli microprogrammati, di cui verra' dato in cenno nel seguito.

A titolo di esempio e con riferimento alla fig. 14.8.8 si consideri passo a passo il trasferimento di un dato dal registro B alla memoria.

All'inizio delle operazioni l'indirizzo dell'istruzione da eseguire e' contenuto in un registro chiamato "contatore di programma (PC)", mentre l'indirizzo della posizione di memoria in cui si vuole trasferire il contenuto del registro B e' contenuto nella coppia di registri a 8 bit H e L.

Gli stati successivi del sequenziatore, coincidenti ciascuno con un ciclo macchina, sono:

$$M_1\begin{cases} T_1 \ldots\ldots PC \rightarrow \text{bus indirizzi} \\ T_2 \ldots\ldots PC+1 \rightarrow PC \\ T_3 \ldots\ldots \text{Istruzione(bus dati)} \rightarrow \text{registro istruzioni} \\ T_4 \ldots\ldots \text{Registro B} \rightarrow \text{Registro temporaneo TMP} \end{cases}$$

$$M_2\begin{cases} T_5 \ldots\ldots H, L \rightarrow \text{bus indirizzi} \\ T_6 \\ T_7 \ldots\ldots \text{Registro temporaneo TMP} \rightarrow \text{bus dati} \end{cases}$$

Si osservi che:

1) Sono necessari 7 stati per eseguire l'istruzione.

2) Gli stati sono raggruppabili in due "fasi di macchina", ciascuna delle quali inizia con l'invio di un indirizzo sul bus indirizzi.

3) L'uso di un registro temporaneo, ne' visibile, ne' accessibile al programmatore, facilita tuttavia la scrittura del programma.

4) Il bus dati viene occupato due volte; una prima volta per trasferire da memoria a registro istruzioni l'istruzione stessa; una seconda volta per trasferire il contenuto del registro B alla voluta posizione di memoria. Sono necessari due tempi T_6 e T_7 poiche' il dato e' da 16 bit, mentre il bus dati e' da 8 bit.

Il codice di istruzione, ricavabile dalla specifiche dell' 8080, relativo alle operazioni appena descritte, e' il seguente:

$$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$$
$$0\quad 1\quad 1\quad 1\quad 0\quad 0\quad 0\quad 0$$

Gli ultimi 3 bit (D_2, D_1, D_0) specificano il registro su cui operare secondo la seguente tabella.

$D_2\ D_1\ D_0$	Registro
000	Registro B
001	Registro C
010	Registro D
011	Registro E
100	Registro H
101	Registro L
110	--------
111	Accumulatore

In generale si puo' affermare che ciascuna microistruzione o istruzione e' suddividibile in campi, il primo dei quali e' detto "codice operativo" e stabilisce l'operazione da eseguire, mentre gli altri sono usati per selezionare gli operandi o per il controllo di vie di comunicazione o di reti interne al microprocessore.

Ritornando ora alla routine di somma, illustrata nel diagramma di flusso di fig. 14.8.3 si puo' notare che le istruzioni vengono eseguite in sequenza fino al test del registro RI. Il valore

successivo del contatore di programma dipende dal risultato del test; quando il contenuto del registro e' pari a 8 il contatore di programma avanza di un passo, mentre in caso contrario viene riposizionato all'indirizzo relativo dell'istruzione

$$0 \rightarrow AR$$

14.8.2) Unita' di controllo.

Da quanto esposto al paragrafo precedente si deduce immediatamente che per l'unita' di controllo del programma esistono due modi di funzionamento (fig. 14.8.9):

1) Incremento, eseguito automaticamente ad ogni lettura della memoria di programma.

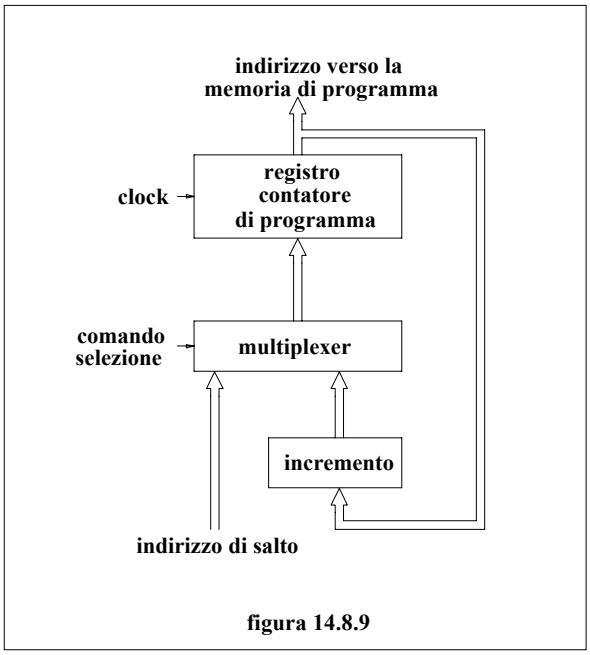


figura 14.8.9

2) Salto assoluto, eseguito sulla base di un'istruzione e ottenuto con la sostituzione del contenuto attuale del contatore di programma con l'indirizzo specificato nella relativa istruzione di salto.

Il salto puo' essere indipendente da condizioni (salto incondizionato) o essere eseguito solo al verificarsi di determinate condizioni (salto condizionato).

In quest'ultimo caso l'istruzione deve specificare il codice di selezione della condizione che deve essere verificata affinche' il salto venga eseguito. Tale informazione viene interpretata dalla rete di decodifica delle istruzioni, che genera il segnale di controllo verso l'ingresso del contatore di programma (fig. 14.8.10).

Le condizioni di salto C_1, C_2, ... , C_n sono normalmente memorizzate in un opportuno registro (flag) e il loro valore dipende dal risultato delle operazioni eseguite nell'ALU.

L'assieme dell'unita' di decodifica delle istruzioni, dell'eventuale sequenziatore che controlla la temporizzazione delle operazioni elementari necessarie all'esecuzione di un ciclo di istruzione, dei circuiti destinati all'incremento automatico del contatore di programma e della rete di verifica delle condizioni costituisce in sostanza l'unita' di controllo del micro-processore.

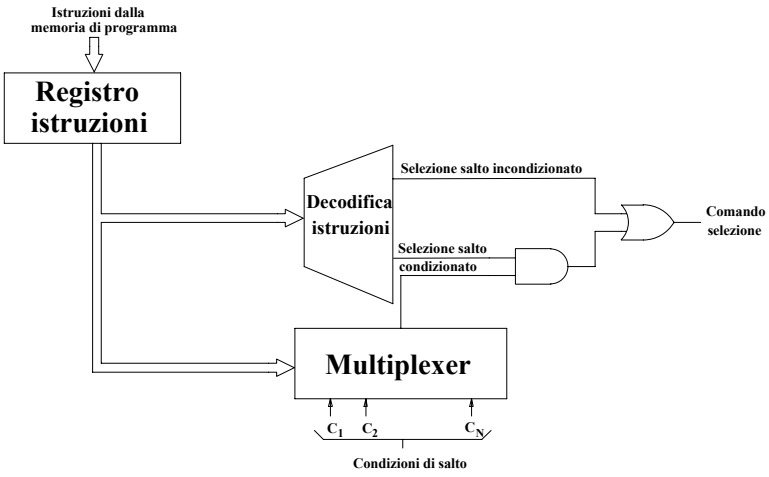


figura 14.8.10

Tale unita' di controllo puo' essere, come gia' accennato, cablata o microprogrammata. Il metodo classico di progetto associa ad ogni stato di macchina la configurazione di un certo numero N di flip-flop; di conseguenza il massimo numero di stati di macchina e' 2^N.

Ogni stato di macchina genera poi dei segnali che si traducono in sequenze di comandi e in elaborazione dei dati. Lo stato successivo dipende dallo stato attuale, dagli ingressi esterni e dai risultati delle elaborazioni. I flip-flop di stato e l'interconnessione ricavata avvalendosi di questa impostazione costituiscono un'unita' di controllo cablata (fig. 14.8.11).

Una struttura di questo tipo implica una notevole rigidita' di progetto e un costo elevato quando si presenta una qualche necessita' di modifica. E' inoltre necessario contenere il massimo numero di stati.

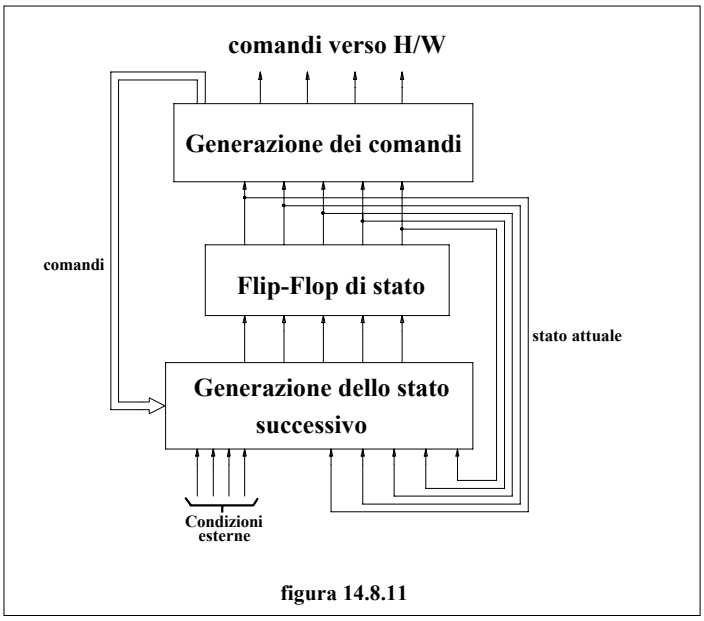
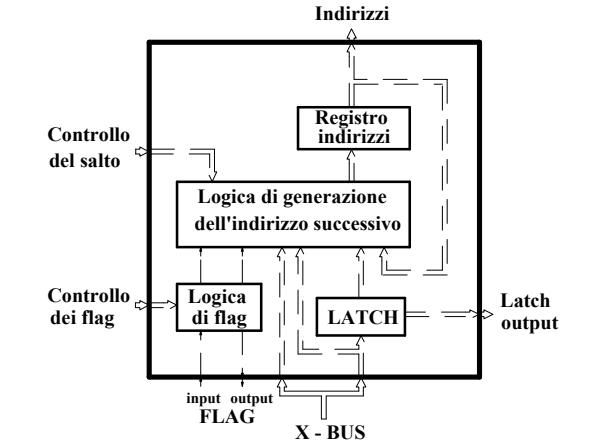
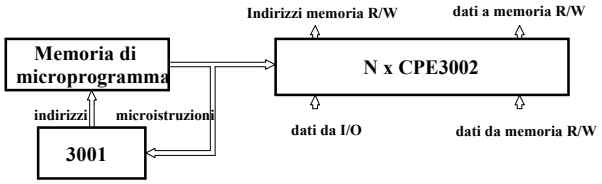


figura 14.8.11

Sostituendo i flip-flop di stato con un contatore di microprogramma e la rete di generazione dell'indirizzo successivo con i circuiti di fig. 14.8.12 (a) e (b) si ottiene un'unita' di controllo microprogrammata.



(a)



(b)

figura 14.8.12

In questo caso il controllo si trova situato nella memoria e ogni modifica di comandi o di flusso logico della routine si riduce ad una modifica dei codici di operazione residenti nella memoria di microprogramma.

14.9) Dal microprocessore al microcalcolatore.

Al paragrafo precedente sono state prese in considerazione le strutture interne del microprocessore, prendendo in esame la configurazione e le funzioni dell'ALU, dei registri, della rete di decodifica delle istruzioni e dell'unita' di controllo. E' stato inoltre messo in luce come possono esistere due grandi categorie di microprocessori; quelli con un insieme fisso di istruzioni e quelli microprogrammati.

Tra i primi, realizzati normalmente in tecnologia MOS, ricadono in pratica tutti i microprocessori con parallelismo 8 bit, quali l'8085, il 6502 e gran parte di quelli a 16 e 32 bit.

Nella seconda categoria, realizzata per lo piu' in tecnologia TTL-Schottky, sono compresi normalmente i microprocessori modulari (bit-slice) con cui possono essere realizzati microcalcolatori veloci con grado di parallelismo qualsiasi.

In ambedue i casi tuttavia la realizzazione di un microcalcolatore richiede l'interconnessione del microprocessore con tutto un insieme di circuiti ausiliari, quali circuiti

di memoria, circuiti di ingresso e uscita (I/O), circuiti speciali, quali ad esempio i circuiti di accesso diretto in memoria. E' quindi evidentemente necessario conoscere quali siano i circuiti accessori disponibili sul mercato, non solo per minimizzare i costi del progetto, ma anche per minimizzare l'hardware aggiunto, semplificare il software e rendere piu' affidabile l'intero dispositivo.

14.9.1) Connessione e organizzazione della memoria.

Fin dall'inizio del presente capitolo sono state prese in esame le principali caratteristiche degli elementi di memoria disponibili.

Scopo del presente paragrafo sara' quindi quello di dare delle indicazioni di massima su come la memoria debba venir scelta, anche se la rapidita' con cui i circuiti di memoria si sono evoluti e tuttora si evolvono verso livelli di integrazione sempre piu' elevati, rende il problema abbastanza difficile.

Nel progetto di un sistema microcalcolatore la memoria occupa una posizione fondamentale. Da un lato essa condiziona il costo dell'intero sistema, dall'altro essa e' indispensabile in quanto il microcalcolatore deve essere controllato da un programma e tale programma non puo' che risiedere in memoria.

Tralasciando per il momento i problemi legati alla messa a punto del programma e' opportuno osservare che e' bene che quest'ultimo sia registrato su un supporto non volatile, tranne per i calcolatori di tipo "general purpose"; il programma pertanto risiedera' normalmente in una memoria di sola lettura (ROM, PROM, EPROM, EAROM).

Per quanto riguarda invece i dati su cui il microelaboratore opera e' necessario viceversa prevedere una memoria di lettura/ scrittura (RAM statica e dinamica).

Nella sua piu' grossolana rappresentazione il sistema di memoria di un microcalcolatore puo' essere strutturato come illustrato in fig. 14.9.1.

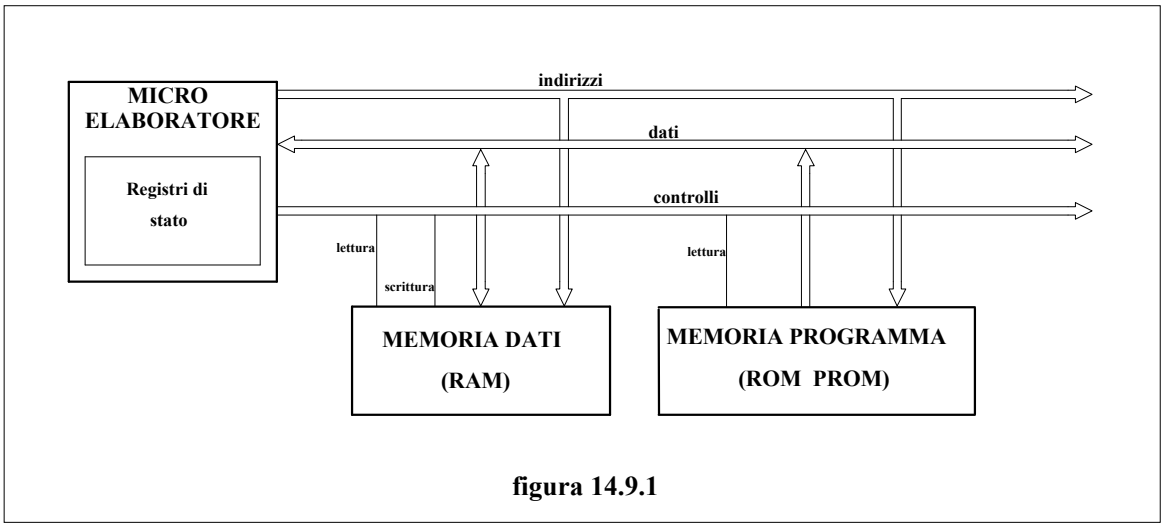


figura 14.9.1

E' opportuno notare che, durante lo sviluppo di un programma per un microelaboratore, la frequenza delle modifiche da apportare passa da un valor massimo iniziale praticamente a zero quando si e' giunti alla fase finale dello sviluppo. Pertanto e' preferibile usare inizialmente per la memoria di programma dei dispositivi RAM che emulino la ROM da montare alla fine dello sviluppo per passare poi a delle EPROM quando la frequenza delle modifiche tende a diminuire. Solo alla conclusione della fase di progetto il programma potra' essere riversato su PROM oppure su ROM a maschera, se il volume di produzione deve

raggiungere almeno qualche centinaio di esemplari. Infatti una ROM a maschera, come e' gia' stato fatto notare, diventa economicamente conveniente solo se realizzata in un numero notevole di esemplari. Disgraziatamente, nella malaugurata ipotesi che si rendesse necessaria una modifica, la ROM a maschera richiede un notevole impegno economico, non solo per la realizzazione della nuova maschera, ma anche per il costo degli elementi di memoria eventualmente gia' acquisiti e presenti a magazzino.

Per quanto riguarda la memoria per i dati si puo' ovviamente scegliere tra memorie statiche e dinamiche. E' opportuno ricordare che una memoria statica si avvale di una cella bistabile e mantiene l'informazione per tutto il tempo durante il quale l'alimentazione viene mantenuta. Nelle memorie dinamiche invece la cella, in cui l'informazione e' accumulata come carica elettrica, si comporta come un dispositivo monostabile rendendo necessaria l'operazione periodica di "refresh". Tuttavia grossi vantaggi delle memorie dinamiche nei confronti di quelle statiche sono:

-- Maggiori dimensioni in bit su singolo "chip", dovute alla maggiore integrabilita' delle celle dinamiche.

-- Dissipazioni notevolemente inferiori.

-- Minor costo a parita' di dimensioni.

A fronte di questi vantaggi le memorie dinamiche sono mediamente piu' lente di quelle statiche.

In generale si puo' anzi affermare che la velocita' di un chip di memoria, sia essa statica o dinamica, diminuisce al crescere delle dimensioni. L'uso dei chip di notevoli dimensioni permette tuttavia di semplificare i circuiti di decodifica degli indirizzi e la gestione dei banchi di memoria.

Tuttavia non sono solo la velocita' e le dimensioni a determinare qual'e' il tipo di memoria da connettere al microelaboratore, ma tutta una serie di altre considerazioni, tra le quali si possono citare le tensioni di alimentazione, la maniera con cui la memoria viene interfacciata con la CPU e i segnali di controllo necessari. Alcuni di questi punti riguardano semplicemente la funzionalita' della memoria, altri, quali le tensioni di alimentazione e il modo di interfacciamento possono avere rilevanti interazioni con i costi.

La massima velocita', cui una memoria puo' funzionare, e' determinata dal tempo di accesso della memoria stessa, definito come l'intervallo di tempo tra l'istante in cui l'indirizzo viene presentato e quello in cui il dato voluto e' disponibile. La conoscenza della velocita' della memoria e' uno degli elementi che permettono di stabilire se le prestazioni di un microcalcolatore sono tali da soddisfare le specifiche del progetto che si sta sviluppando. Normalmente non e' necessario che la memoria abbia un tempo di accesso tale da permettere al microprocessore di funzionare alla massima velocita' possibile; e' possibile nella grande generalita' dei casi utilizzare circuiti di memoria relativamente lenti, in quanto meno costosi. In tal caso tuttavia e' necessario rallentare il microprocessore in modo da sincronizzarne il funzionamento con il ciclo di funzionamento della memoria; tale operazione puo' venir fatta associando alla memoria un semplice circuito che generi un segnale di "**dato pronto**" quando il contenuto della memoria e' sicuramente disponibile. Il microprocessore, dopo aver fornito l'indirizzo della posizione di memoria da leggere esegue il test di tale segnale e finche' esso non si presenta rimane in uno stato di attesa (**wait state**), secondo lo schema illustrato in fig. 14.9.2.

Un meccanismo di tal genere esiste praticamente in tutti i microprocessori attualmente sul mercato, in quanto permette di utilizzare, ove possibile, dispositivi di memoria piu' lenti e meno costosi.

Le tensioni di alimentazione assumono una notevole importanza in quanto il costo dell'alimentatore puo' essere tra i piu' importanti nel progetto di un microcalcolatore. Per quanto riguarda i microprocessori, siano essi bipolari, MOS o CMOS la tendenza attuale e' quella di utilizzare un'unica tensione di alimentazione, normalmente di 5 volt.

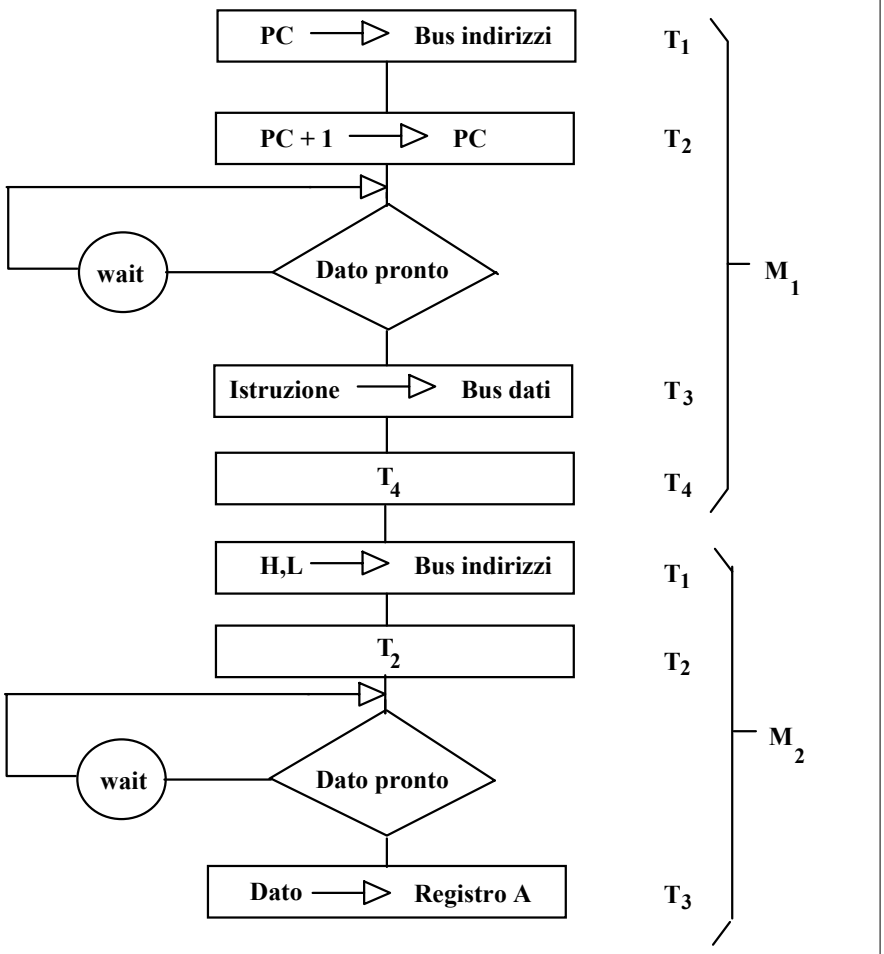


figura 14.9.2

Il discorso si complica quando al microprocessore vengono associati circuiti di memoria e di interfaccia. In tal caso possono rendersi necessarie diverse tensioni di alimentazione (+5 V, -5 V, +12 V) che complicano il progetto dell'alimentatore, anche se la tendenza odierna e' quella di passare a dispositivi con un'unica tensione di alimentazione compatibile con quella del microprocessore usato.

Infine l'interfacciamento tra memoria e microprocessore puo' essere logicamente suddiviso in tre parti: gli indirizzi, i dati e i controlli. Tuttavia questa suddivisione logica tra i diversi flussi di informazioni solo in alcuni casi corrisponde ad un'analoga suddivisione fisica dei relativi canali trasmissivi; solo in alcuni casi cioe' e' nettamente distinguibile un bus per gli

indirizzi, uno per i dati e uno per i controlli. Molto spesso infatti lo stesso supporto fisico viene utilizzato in tempi diversi per la trasmissione di informazioni diverse.

Per quanto riguarda i controlli, il minimo indispensabile e' dato dai segnali di:

-- READ che comanda un'operazione di lettura dalla memoria.

-- WRITE che permette di effettuare un'operazione di scrittura in memoria.

-- READY cioe' il segnale di dato pronto che permette di sincronizzare le operazioni tra microprocessore e memoria.

Pertanto la piu' semplice struttura che si possa immaginare per l'interfaccia tra memoria e microprocessore avra' l'aspetto di quella illustrata in fig. 14.9.3.

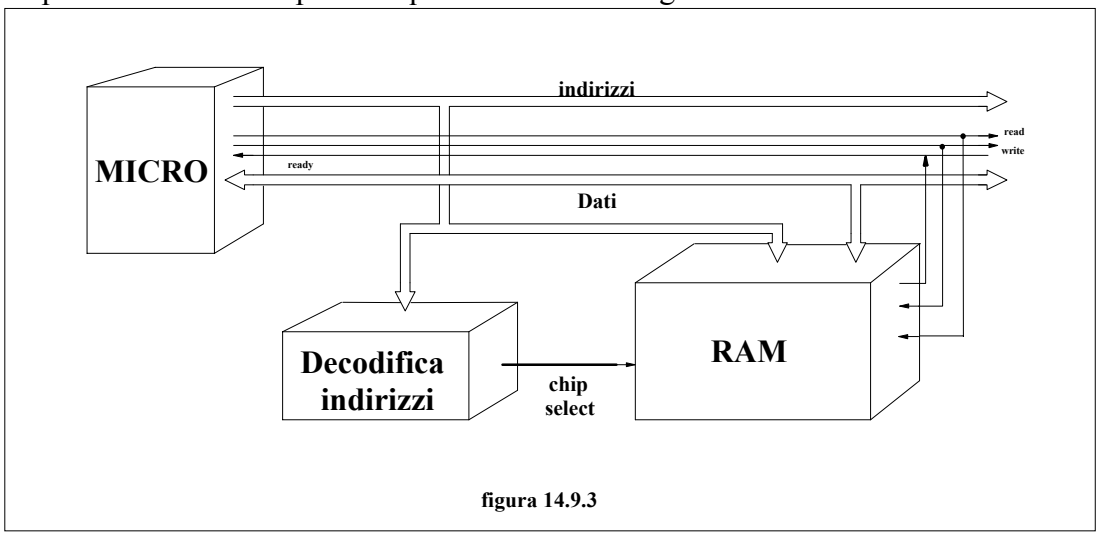


figura 14.9.3

I brevissimi cenni dati sull'organizzazione e l'interconnessione della memoria con il microprocessore non hanno alcuna pretesa di completezza ne' di sistematicita'; infatti il mondo dei circuiti di memoria a semiconduttore e' estremamente vario e in rapida espansione, con l'introduzione quasi quotidiana di nuovi dispositivi. Gia' in questo rapido excursus non si e' parlato di dispositivi largamente usati, quali le memorie CCD, quelle FIFO, ne' di dispositivi particolari, quali ad esempio le memorie a doppio accesso. Scopo del presente paragrafo era tuttavia quello di mettere in luce la necessita' di valutare attentamente tutte le possibili soluzioni, analizzando attentamente le possibilita' di dialogo tra memoria e microprocessore. Una buona organizzazione e un corretto modo di indirizzamento della memoria infatti permettono molto spesso di compensare la maggior lentezza di certi tipi di dispositivi, a vantaggio dei costi e delle prestazioni complessive del microcalcolatore.

14.9.2) I circuiti di I/O.

I circuiti di I/O collegano il microcalcolatore con il modo circostante e pertanto, in relazione alla molteplicita' delle applicazioni possibili, essi sono specializzati per ogni singola applicazione, malgrado che trovino impiego in un ambito in cui l'hardware e' del tutto generalizzato.

Le interfacce del microcomputer vanno ottimizzate, sia per ragioni di costo complessivo, sia perche' il microcalcolatore trova molto spesso i suoi limiti dinamici proprio nelle operazioni di I/O.

A differenza del microprocessore, campo di applicazione incontrastato di tecnologie LSI, la periferia richiede frequentemente un largo impiego di parti MSI/SSI; tuttavia per alcune applicazioni, ricorrenti in un numero molto elevato di progetti, sono stati realizzati circuiti LSI di input/output (porte di I/O) che hanno permesso di risolvere allo stesso tempo problemi sia di hardware che di software. Tali componenti sono andati ad arricchire le varie famiglie di microprocessori, rendendo possibile realizzare un microcalcolatore avvalendosi di blocchi funzionali appartenenti tutti alla stessa famiglia.

Si ricordi che tutti i microcalcolatori sono organizzati attorno a tre bus, detti rispettivamente dei dati, degli indirizzi e dei controlli.

Nelle configurazioni piu' semplici tutti gli scambi di informazioni tra mondo esterno e memoria avvengono tramite il microprocessore, al quale viene riservato il controllo di tutti i bus. Nelle configurazioni piu' complesse invece anche gli organi di I/O possono assumere il controllo dei bus e trasferire dati direttamente dalla/alla memoria. In ogni caso i dati viaggiano sul bus dei dati mentre sul bus degli indirizzi, oltre agli indirizzi delle locazioni di memoria interessate all'operazione, viene inviato il codice di selezione della porta di I/O utilizzata e sul bus controlli compaiono i segnali essenziali necessari al funzionamento della porta.

Come struttura, sia pure estremamente semplificata, le porte di ingresso possono essere ridotte a buffer con uscita ad alta impedenza (3-state) e quelle di uscita a registri (latch); la selezione della porta voluta richiede una rete di decodifica degli indirizzi, mentre la porta stessa viene attivata da segnali di controllo, che possono essere gli stessi segnali READ/WRITE utilizzati per la lettura/scrittura in memoria. Tutte le porte MSI/LSI hanno poi la possibilita' di generare un "interrupt" verso il microprocessore.

I piu' comuni circuiti di input/output sono:

1) Circuiti per la trasmissione e la ricezione seriale dei dati che possono funzionare sia in modo asincrono (UART) sia sincrono/asincrono (USART). La velocita' di trasmissione e di ricezione e il formato dei dati sono generalmente programmabili da software e in trasmissione viene aggiunto in modo automatico un eventuale bit di parita' e uno o piu' bit di stop, mentre in ricezione vengono controllati vari tipi di errore. L'estrema semplicita' di uso e la notevole complessita' delle funzioni eseguite fanno si' che i dispositivi UART/USART siano i dispositivi di interfaccia piu' usati, in quanto l'unico compito lasciato al progettista del sistema e' quello di ottimizzare le procedure di trasmissione.

2) Circuiti di interfaccia parallela programmabili, che permettono la trasmissione di un completo byte allo stesso tempo da/verso un'unita' periferica. Oltre ai bit del dato sono ovviamente presenti un certo numero di segnali di controllo che permettono di governare il flusso di informazioni tra periferia e calcolatore. Le varie linee di comunicazione con la periferia possono essere programmate in vario modo; ad esempio come "ingressi" o come "uscite", oppure come segnali di controllo del flusso informativo. La programmazione e' molto spesso individuale per ciascun bit e si puo' ottenere addirittura una procedura di colloquio "hand-shaking" in modo automatico. La procedura di caratterizzazione del funzionamento di una porta parallela viene normalmente fatta durante la fase di inizializzazione del microcalcolatore. Non e' possibile dare altri dati generali su questo tipo di porta, in quanto molte delle loro caratteristiche sono intimamente legate alle caratteristiche del microprocessore alla cui famiglia appartengono.

3) Altri circuiti LSI disponibili per operazioni di I/O piu' sofisticate sono le memorie FIFO e il DMA controller. Le memorie FIFO (**First In-First Out**) permettono di risolvere i problemi di sincronizzazione nella trasmissione parallela, diventando una coda d'attesa in cui i dati entrano quando si rende disponibile un posto ed escono quando vengono richiesti dall'utilizzatore. Si rivelano particolarmente adatte per la gestione di periferiche quali le stampanti in quanto permettono una notevole semplificazione del software e riducono il tempo di occupazione della CPU, necessario alle operazioni di stampa.

Il DMA (**direct memory access**) permette lo scambio di dati tra memoria e unita' periferiche senza l'intervento del microprocessore. Tutta l'operazione di trasferimento e quindi tutti i bus del microcalcolatore sono completamente controllati dal DMA controller, mentre il microprocessore e' temporaneamente in stato di attesa (**HOLD**). Vantaggio di un tal modo di operare e' la notevole accelerazione delle operazioni di trasferimento.

Prima di dare il via alle operazioni del DMA e' necessario che il microcalcolatore fornisca tre parametri: l'indirizzo di memoria da/in cui iniziare a trasferire i dati, la lunghezza del blocco di dati da trasferire e il codice identificativo della porta di I/O da usare.

Si noti che con l'introduzione del DMA controller il microprocessore perde la sua funzione centrale nel microcalcolatore, ma diventa assieme alle porte di I/O uno degli utenti dei bus e della memoria.

Un dato di fatto e' che comunque i circuiti di I/O evolvono verso strutture complesse, anche se la nascita di circuiti LSI per risolvere problemi di interfaccia con il mondo esterno e' determinata dal mercato, cioe' dal numero di potenziali clienti. Essi tuttavia sono la strada che permette di risolvere certi conflitti di tempo nell'ambito del software, determinati dalle "ridotte" prestazioni dinamiche dei microprocessori.

La scelta tra una soluzione hardware e una soluzione software dello stesso problema di I/O e' tuttavia di solito determinata da una valutazione dei costi: costi di sviluppo, dei componenti LSI, dei bit di memoria ROM necessari in relazione alle prestazioni complessive del progetto.

La tendenza che tuttavia gia' da diversi anni si e' delineata e che sembra essere una strategia vincente e quella di trasferire un gran numero di funzioni dal software all'hardware, anche se tale tendenza sembra essere stranamente in contrasto con le esigenze che hanno portato alla nascita e allo sviluppo dei microprocessori.

E' infine necessario osservare che chi intendesse progettare con elementi LSI, tra cui anche microprocessori, deve non solo possedere le necessarie competenze nell'ambito dell'hardware e del software, ma anche avere la necessaria visione sistemistica che gli permetta di interconnettere blocchi logici di notevole complessita', quali le parti LSI, avendo tuttavia come obiettivo finale l'ottimizzazione dell'intero progetto.

14.9.3) I supporti software e hardware.

Come gia' accennato, lo sviluppo di un progetto con microprocessori richiede nuovi metodi e nuovi strumenti di lavoro rispetto a quelli correntemente usati nei tradizionali progetti di circuiti di commutazione.

Infatti gran parte dello sviluppo hardware tradizionale viene eliminato del fatto che e' lo stesso microprocessore che genera i segnali di indirizzo, i dati e i segnali di controllo con modalita' di durata, forma e livello logico che sono stati determinati a priori dal costruttore. La stessa standardizzazione delle interfacce verso la memoria e verso il mondo esterno riduce in misura considerevole la possibilita' di errori di interconnessione tra le diverse parti, che d'altra parte sono sempre in numero ridotto.

Il progetto e' caratterizzato in buona misura dalla messa a punto di un programma da inserire nelle memorie ROM; sono pertanto molto piu' frequenti errori determinati da errate sequenze di istruzioni che non errori circuitali veri e propri.

Di conseguenza l'uso di oscilloscopio, generatori di segnali, analizzatori di stati ecc. deve venir integrato con l'uso di nuovi strumenti sia hardware che software che permettano la verifica, nella sua totalita', della correttezza del dispositivo realizzato.

Supporti software di progetto.

Lo sviluppo di una qualsivoglia routine per un sistema a microprocessore richiede innanzitutto la conoscenza circuitale del sistema su cui la routine stessa deve lavorare. Per quanto riguarda le parti circuitali esterne al microcalcolatore e' necessario conoscere la funzione di ciascun segnale dell'interfaccia, il suo valore logico e le relative temporizzazioni. Per quanto riguarda il microprocessore e' necessario definire a priori:

1) L'indirizzo iniziale della memoria ROM in cui verra' registrato il programma.
2) Gli indirizzi della memoria RAM disponibile per i dati.
3) Gli indirizzi delle porte di I/O interessate ad operazioni di trasferimento di dati.

Quando l'hardware e' completamente definito risultano definiti anche i codici di istruzione in binario. Questo fatto, ovvio per i microprocessori ad insieme fisso di istruzioni, non lo e' altrettanto per i microprocessori "bit slice" in cui al progettista rimane la liberta' di definire le istruzioni sulla base delle microistruzioni disponibili.

Risultano inoltre definiti anche gli indirizzi della memoria e delle porte di input/output e le eventuali maschere per il test e l'elaborazione dei singoli bit.

L'insieme di tutti questi codici binari costituisce il linguaggio macchina ed e' in questo linguaggio che deve venir realizzato qualsiasi programma (programma oggetto) per il microprocessore, indipendentemente da quale sia il punto di partenza del processo di sviluppo.

E' abbastanza evidente che lo sviluppo di una routine direttamente in linguaggio macchina e' un procedimento difficile e scarsamente efficace; normalmente il programma viene scritto in un linguaggio composto da simboli mnemonici in corrispondenza biunivoca con i codici del linguaggio macchina. La programmazione in tale linguaggio simbolico e' piu' semplice e nettamente piu' efficiente; il programma inoltre risulta piu' facilmente leggibile e interpretabile.

Evidentemente il programma simbolico, per poter essere utilizzato sul microcalcolatore deve venir tradotto in linguaggio macchina per mezzo di un opportuno programma di traduzione detto ASSEMBLATORE (ASSEMBLER).

Generalmente i costruttori forniscono il linguaggio Assembler per i microprocessori a insieme fisso di istruzioni. Per quelli microprogrammati vengono forniti i cosiddetti linguaggi microassemblatori che possono essere personalizzati a seconda delle necessita', definendo nuovi codici di istruzione allo stesso modo in cui tale operazione e' possibile in campo hardware.

Comunque sia il linguaggio Assembler e' uno dei supporti software fondamentali per lo sviluppo dei programmi; piu' precisamente e' possibile operare una distinzione tra linguaggi assembler veri e propri, cioe' quei traduttori utilizzati su calcolatori (o microcalcolatori) che

hanno lo stesso linguaggio macchina del programma oggetto realizzato e transassembler, cioe' quei traduttori utilizzati su calcolatori con linguaggio macchina diverso da quello del programma oggetto prodotto.

Quest'ultima impostazione permette di sviluppare i programmi su calcolatori propri o di facile accesso, ma ha tuttavia il difetto che l'unica verifica possibile e' quella della correttezza formale del programma prodotto. Non e' infatti possibile eseguire un controllo in tempo reale del programma prodotto in quanto il calcolatore utilizzato per lo sviluppo e quello cui il programma e' destinato sono affatto diversi e comunque le loro temporizzazioni non coincidono.

L'impiego di linguaggi assembler propriamente detti, attraverso l'utilizzo di opportuni sistemi di sviluppo per microprocessori, appare attualmente la soluzione piu' attraente e completa.

Nello sviluppo di un programma tuttavia e' necessario disporre di altri strumenti che non semplicemente l'assembler per consentire un lavoro semplice e ordinato. Tra essi possono essere citati come elementi pressocche' essenziali il Monitor, il Sistema operativo, l'Editor e i Compilatori. E' opportuno per ciascuno di questi moduli dare un cenno delle principali funzioni svolte.

1) Il Monitor e' un programma, che puo' venir definito come la versione piu' ridotta ed elementare di un sistema operativo. Esso consente le seguenti operazioni:

- Inizializzare il sistema;
- Caricare in memoria programmi da un organo di input;
- Visualizzare il contenuto dei vari registri;
- Modificare il contenuto della memoria e dei registri;
- Eseguire programmi residenti in memoria.

Il monitor e' il minimo indispensabile per poter operare nella messa a punto del software di un microcalcolatore. Molto spesso tuttavia i sistemi specializzati per lo sviluppo di tale software sono dotati di unita' di memoria di massa, quali floppy disk, su cui sono registrati sia i programmi, che via via vengono messi a punto, sia gli strumenti software che durante tale processo potessero rendersi necessari.

2) Il Sistema Operativo e' un insieme di moduli costituenti un programma complesso che, oltre a svolgere le funzioni gia' descritte per il Monitor, permettono la gestione degli archivi su disco e il colloquio con il modo esterno in maniera ben piu' raffinata di quanto il Monitor non consenta.

3) L'Editor e' un modulo che permette di stendere e memorizzare successivamente su disco i programmi necessari allo sviluppo del progetto, utilizzando istruzioni in un opportuno linguaggio simbolico. Oltre alla possibilità di creare un testo ex novo, l'Editor permette evidentemente anche la correzione e la modifica di testi precedentemente memorizzati, inserendo o cancellando caratteri, parole o righe, ricercando o sostituendo sequenze di caratteri e infine memorizzando su un supporto permanente il risultato di tali operazioni.

4) Il programma simbolico (programma sorgente) messo a punto con l'Editor viene poi tradotto in programma oggetto con l'uso del linguaggio assemblatore, se il linguaggio simbolico utilizzato e' l'Assembler del microprocessore per cui si sta sviluppando il software, o utilizzando traduttori, chiamati in tal caso compilatori, a piu' alto livello, quali i compilatori FORTRAN, PASCAL, C, PLM ecc.

In tal caso tuttavia si perde la corrispondenza biunivoca tra istruzioni del linguaggio sorgente e istruzioni in linguaggio macchina, nel senso che un programma, di un certo numero di istruzioni, scritto in un linguaggio sorgente che non sia l'Assembler puo' dar luogo a un programma oggetto che contiene un numero di istruzioni nettamente superiore.

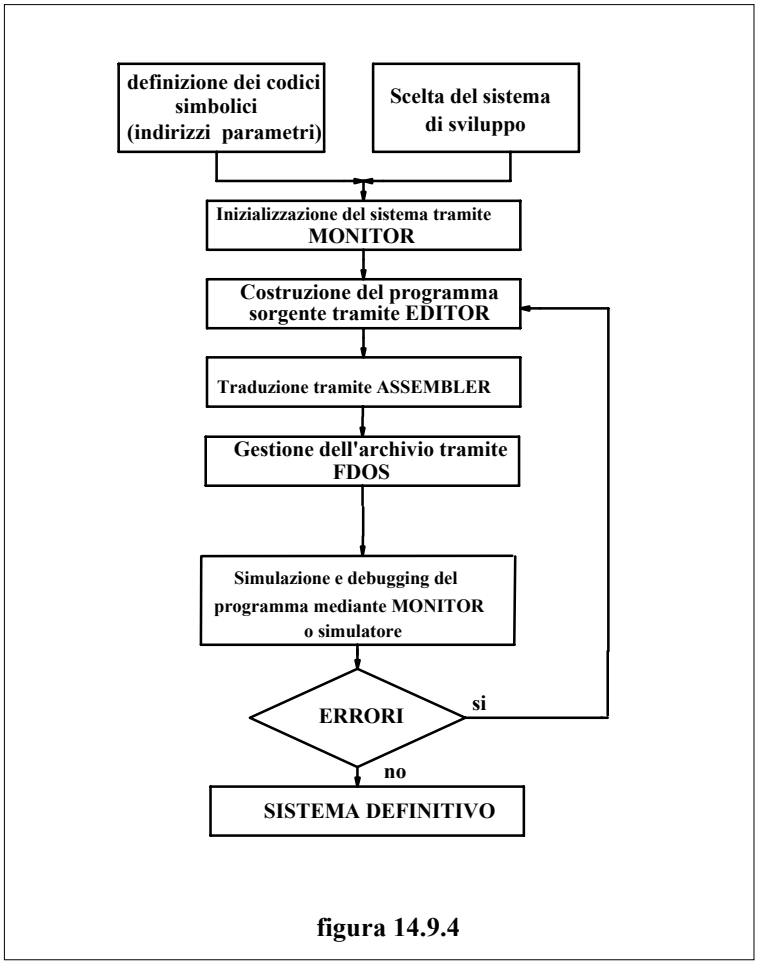


figura 14.9.4

I passi successivi necessari allo sviluppo del software applicativo di un microcalcolatore ed i programmi di base utilizzati sono illustrati in linea di massima in fig. 14.9.4.

Supporti Hardware di progetto.

Durante la messa a punto dei programmi con l'utilizzo dei supporti software appena descritti e' necessario provare il software dopo ogni correzione, possibilmente assieme all'hardware del progetto. Nel caso cio' non fosse possibile, soprattutto se per lo sviluppo non ci si avvale di sistemi specializzati, si puo' ricorrere a simulatori.

Tuttavia anche quando si puo' sottoporre a test contemporaneamente hardware e software del progetto, non e' pensabile di utilizzare delle memorie ROM per farvi risiedere il programma messo a punto. E' necessario infatti disporre di una memoria di programma che sia facilmente correggibile per poter apportare rapidamente le modifiche necessarie; le memorie ROM vengono utilizzate solo quando il progetto e' consolidato.

Un utile sistema di memoria di lettura/scrittura che viene utilizzato in questa fase e' l'emulatore di ROM, il cui schema di massima e' riportato in fig. 14.9.5.

L'emulatore e' realizzato con memorie RAM, ma gli indirizzi possono venir generati sia dal microprocessore del sistema progettato (in fase di lettura) che dalla logica di governo del sistema di sviluppo (in fase di scrittura).

Opportuni organi di input e output permettono sia un facile caricamento dei programmi in memoria che un'agevole verifica dei contenuti di quest'ultima in qualsiasi momento.

Poiche' l'emulatore di ROM e' realizzato con memorie a lettura/scrittura esso puo' venir impiegato per realizzare in fase di messa a punto tutta la memoria del microcalcolatore, con il vincolo che sia rispettata un corretta decodifica degli indirizzi.

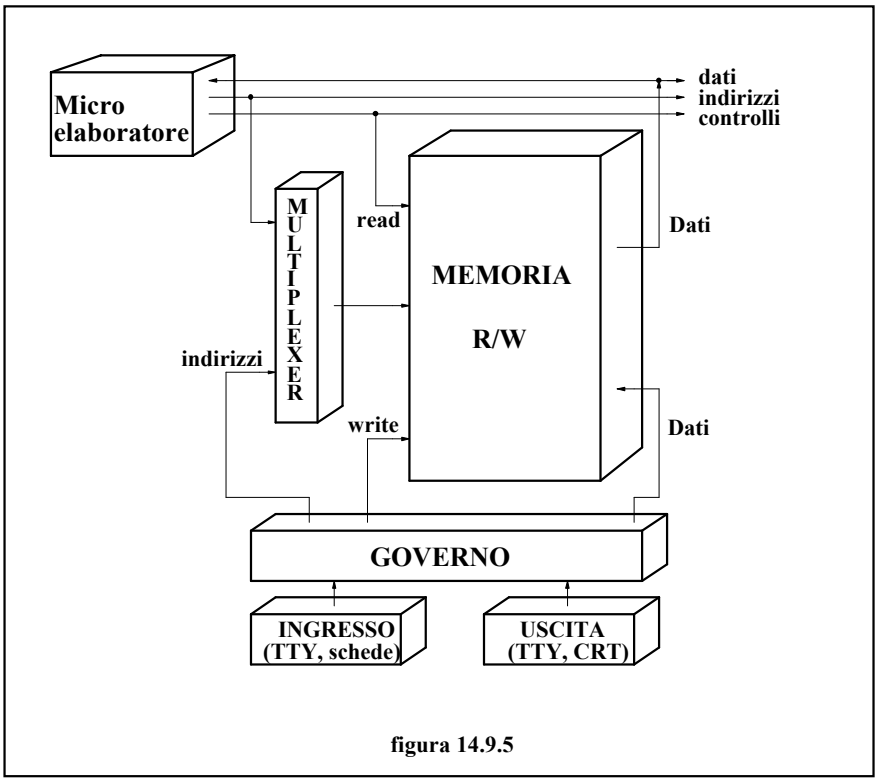


figura 14.9.5

Quando poi il programma e' stato sufficientemente verificato, esso viene trasferito su memorie che permettono la verifica del sistema in forma definitiva; potrebbe infatti essere rischioso trasferire un programma direttamente dall'emulatore di ROM alle ROM stesse. Si preferisce di solito fare ricorso a memorie EPROM compatibili pin to pin con le ROM scelte per il progetto.

Si fa notare che durante la fase di messa a punto di un programma possono presentarsi due esigenze, apparentemente antitetiche:

1) Si vuole che il programma venga eseguito passo a passo in modo da poter seguire in modo dettagliato il suo evolvere, come nel caso di algoritmi matematici, controllo di segnali statici, ecc. E' evidente che in tal caso la logica del sistema di sviluppo deve prevedere tale possibilita' e la periferia del microcalcolatore deve permettere tale tipo di funzionamento.

2) Il programma messo a punto prevede il controllo di un processo che non permette il funzionamento passo a passo. E' allora difficile capire quali sono le routines eseguite dal sistema, che in un secondo esegue per lo meno 10^4-10^5 istruzioni. Si ricorre in tal caso ad uno strumento, il tracciatore, che memorizza gli indirizzi generati nel Contatore di Programma, in modo che possa essere ricostruito il cammino percorso. Il tracciatore non e' altro che una memoria lettura/scrittura sincronizzata con l'evoluzione del contatore di programma.

Non e' ne' utile ne' conveniente avere una memoria di tracciamento di rilevanti dimensioni; e' spesso sufficiente memorizzare qualche decina di istruzioni per avere le informazioni sufficienti alla correzione di un programma. E' tuttavia necessario stabilire quando iniziare e quando terminare il tracciamento e quindi il tracciatore deve essere dotato di opportune logiche di confronto di indirizzi e di riconoscimento di stati. Sempre con queste logiche e' possibile impostare dei "**break-point**" in cui arrestare l'esecuzione del programma per procedere a delle verifiche.