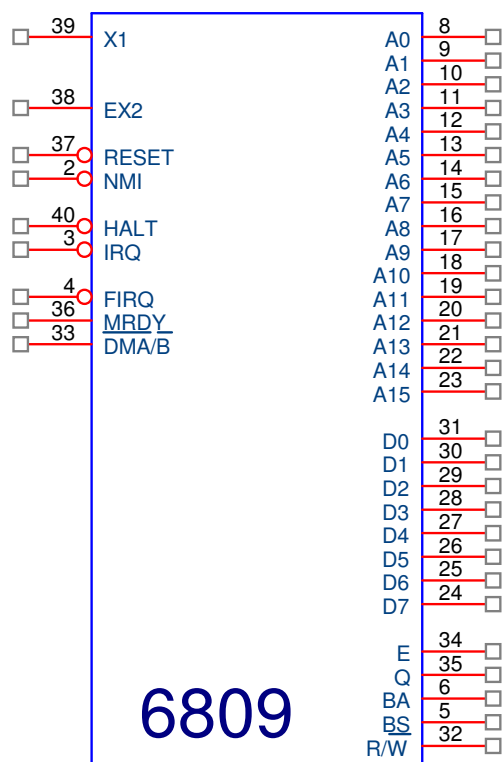


Microprocessore MC6809 Motorola	2
COMPATIBILITÀ CON CPU M6800 - HARDWARE	2
ARCHITETTURA INTERNA DELLA CPU M6809	2
CARATTERISTICHE HARDWARE DELLA CPU M6809	2
Caratteristiche Software della CPU M6809 10 modi di indirizzamento di cui :	3
MODELLO DI PROGRAMMAZIONE CPU M6809	4
DESCRIZIONE DELLE LINEE FACENTI CAPO ALLA CPU M6809	7
ADDRESS BUS	7
DATA BUS D0-D7	7
HALT ingresso CPU	7
Ingresso NMI	9
FUNZIONAMENTO DELLA CPU M6809	11
MODI DI INDIRIZZAMENTO	11
INDIRIZZAMENTO INERENTE	12
INDIRIZZAMENTO ASSOCIATO AI REGISTRI INTERNI DELLA CPU	12
INDIRIZZAMENTO IMMEDIATO	12
INDIRIZZAMENTO ESTESO	12
INDIRIZZAMENTO ESTESO-INDIRETTO	13
INDIRIZZAMENTO DIRETTO	13
INDIRIZZAMENTO INDICIZZATO	14
INDICIZZATO CON OFFSET 0	14
INDICIZZATO CON OFFSET COSTANTE	14
INDICIZZATO CON OFFSET IN ACCUMULATORE	15
INDICIZZATO CON INCREMENTO/DECREMENTO AUTOMATICO	15



Microprocessore MC6809 Motorola

La CPU M6809 è un microprocessore ad 8 bit di ottime prestazioni.

E' in grado di supportare le moderne tecniche di programmazione che richiedono la generazione di un codice-oggetto indipendente dalla posizione occupata in memoria dal programma, che richiedono la stesura di "routines" rientranti e la programmazione modulare strutturata.

La CPU M6809 dispone di una architettura potenziata (rispetto alla CPU M6800). Infatti essa è dotata di un gruppo di registri addizionali (non presenti nella CPU M6800) e di un set di istruzioni e modi di indirizzamento ampliati rispetto alla CPU M6800.

Il set di istruzioni della CPU M6800 è stato fortemente potenziato dalla introduzione di modi di indirizzamento molto potenti. La CPU M6809 dispone del più completo set di modi di indirizzamento disponibili oggi nel settore dei microprocessori a 8 bit .

La CPU M6809 ha caratteristiche hardware e software che la rendono idonea a supportare linguaggi ad alto livello, oppure all'impiego in controllori industriali di tipo standard.

COMPATIBILITÀ CON CPU M6800 - HARDWARE

La CPU M6809 si interfaccia con tutti i periferici della famiglia M6800.

SOFTWARE Compatibilità a livello di linguaggio sorgente Assembler sia per quanto concerne i simboli mnemonici sia per quanto concerne i modi di indirizzamento.

Questo non significa che un programma oggetto M6800 "funzioni" su un sistema M6809.

La compatibilità è garantita solo a livello di "moduli sorgente", per cui il sorgente M6800 può essere assemblato (senza sostanziali modifiche) su un sistema M6809, ottenendo così un oggetto M6809.

ARCHITETTURA INTERNA DELLA CPU M6809

La CPU M6809 dispone internamente di 4 registri a 8 bit e 5 registri a 16 bit :

- Due registri accumulatori A e B a 8 bit i quali possono essere concatenati per formare un accumulatore a 16 bit
- Un registro di pagina diretta DPR (Direct Page Register) a 8 bit
- Un registro di stato o CCR
- Un registro Program Counter PC a 16 bit
- Due registri Indice X e Y a 16 bit
- Due registri Stack Pointer S, U a 16 bit utilizzabili pure come registri indice.

CARATTERISTICHE HARDWARE DELLA CPU M6809

- Generatore di CLOCK incorporato: (oscilla alla frequenza $4 \times f_0$ ove f_0 è la frequenza di esercizio della CPU).
- Ingresso DMA/BREQ : consente il funzionamento della CPU in DMA o il rinfresco delle Memorie RAM dinamiche del sistema.
- Ingresso FIRQ: Fast Interrupt Request provoca la memorizzazione nello stack del CCR e del PC.
- Ingresso MRDY: estende il tempo di accesso sul DATA-BUS e consente l'impiego di memorie lente.
- Stato di INT-ACKN: consente la vettorizzazione di un interrupt da parte del dispositivo periferico.
- Stato di SYNC-ACKN: consente la sincronizzazione del funzionamento della CPU su eventi esterni.
- Segnale di Reset che agisce in un solo ciclo .
- Funzionamento con alimentazione singola a 5 V.
- Richieste di interrupt NMI bloccate dopo una operazione di RESET finchè non viene caricato lo STACK POINTER Hardware S.

- Indirizzi sull'Address Bus validi 1/4 di periodo di clock prima dell'inizio della fase E (ciò consente l'impiego di memorie più lente).

Caratteristiche Software della CPU M6809 10 modi di indirizzamento di cui :

6 modi di indirizzamento compatibili M6800 :

- Inerente
- rif. Accumulatori
- Diretto in pagina 0 (da 0000 a 00FF)
- Esteso
- Immediato
- Indicizzato
- Relativo

Il Modo di indirizzamento diretto è consentito su tutta la mappa di memoria di 64K grazie all'introduzione di un registro DPR che mantiene all'interno del CPU l'indirizzo di pagina (la parte più alta dell'indirizzo).

L'indirizzamento relativo a BRANCH lunghi e corti, permette salti superiori a +127 o -128 bytes attraverso l'impiego delle istruzioni Long-Branch (-32768 +32767).

Indirizzamento relativo al Program-Counter permette in alcuni casi di impiegare il registro PC come registro indice.

Indirizzamento indiretto : il registro puntatore è esterno alla CPU ed è costituito da due zone consecutive il cui contenuto rappresenta l'indirizzo base del dato.

Espansioni al modo di indirizzamento indicizzato:

- Offset costante pari a 0 oppure a 5 bit (4 bit con segno)
- 8 bit (7 bit con segno)
- 16 bit (15 bit con segno)
- Offset variabile a 8 bit costituito dal contenuto degli Accumulatori A e B o a 16 bit costituito dal contenuto del registro D.
- Possibilità di autoincrementare o autodecrementare il contenuto dei registri puntatori
- Manipolazione degli stack (relativa agli Stack pointers S e U) migliorata.

-1464 codici macchina diversi.

-Moltiplicazione "unsigned" a 8 bit -Aritmetica a 16 bit -Istruzioni di trasferimento TFR e di scambio EXG che operano su tutti i registri della CPU.

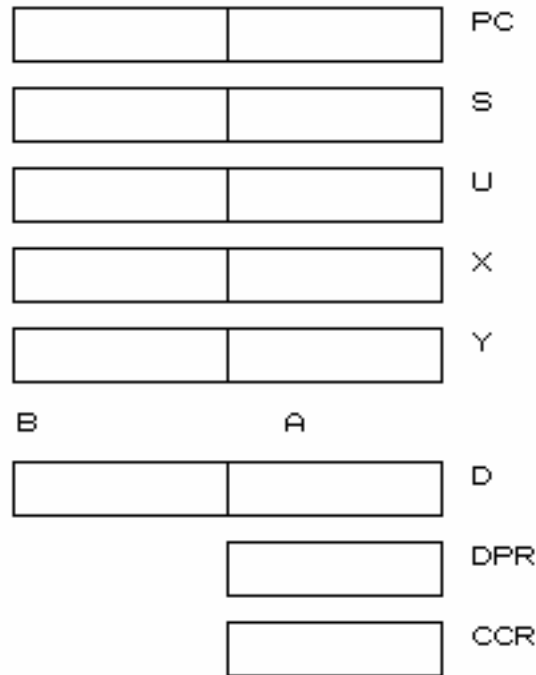
-Istruzioni PUSH e PUL che operano su tutti i registri della CPU

-Istruzioni LEA (Load Effective Address)

MODELLO DI PROGRAMMAZIONE CPU M6809

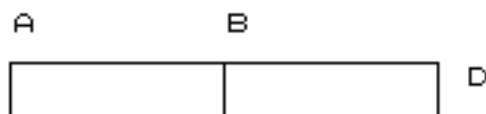
Rispetto alla CPU M6800 sono presenti in più:

- Un registro (ad 8 bit) DPR (Direct Page Register)
- Uno Stack Pointer-Utente U (a 16 bit)
- Un altro registro indice Y (a 16 bit)



ACCUMULATORI A,B

I registri A e B sono registri accumulatori per impieghi generali che vengono usati per calcoli aritmetici e per effettuare manipolazioni sui dati. Alcune istruzioni della CPU M6809 prevedono il concatenamento dei registri A e B per formare un unico accumulatore a 16 bit, denominato accumulatore D.



Accumulatore D

Il contenuto dell'accumulatore A costituisce il MSB Byte dell'accumulatore D.

REGISTRO DPR

Il registro DP della CPU M6809 è stato introdotto per estendere il modo di indirizzamento diretto (in pagina 0) a tutta la mappa di memoria della CPU M6809. Il contenuto del registro DPR viene collocato sull'Address-Bus da A8 a A15 nel corso di esecuzione di istruzioni che prevedono il modo di indirizzamento diretto.

L'impiego del modo di indirizzamento diretto in luogo del modo di indirizzamento esteso consente una riduzione del volume complessivo del programma ed una maggiore velocità di esecuzione complessiva.

Al fine di rendere la CPU M6809 compatibile con la CPU M6800, il contenuto del registro DPR viene azzerato nel corso dell'operazione di Reset dell'MPU.

REGISTRI INDICE X e Y

I registri indice X e Y vengono usati nel modo di indirizzamento indicizzato.

Il contenuto a 16 bit di questi registri viene utilizzato nel calcolo dell'effettivo indirizzo del dato su cui opera l'istruzione.

Il contenuto dei registri X e Y può essere usato per puntare direttamente a un dato, oppure può venir sommato ad un offset a 8 o 16 bit o al contenuto degli accumulatori A o B, per determinare l'indirizzo effettivo del dato. In alcuni modi di indirizzamento indicizzato i contenuti dei registri indice vengono incrementati o decrementati automaticamente per consentire il puntamento di zone memoria consecutive nell'ambito di una tabella.

Tutti i 4 puntatori a 16 bit: X Y U , S possono essere usati come registri indice.

PUNTATORI DI STACK S e U

Lo Stack Pointer Hardware S viene usato automaticamente dalla CPU durante lo svolgimento di subroutine o di routines di interrupt. I puntatori di stack della CPU M6809 puntano ad una locazione di memoria stack occupata da un dato mentre, nel caso della CPU M6800 lo stack pointer S punta alla successiva locazione libera nello stack.

Lo stack pointer U è controllato esclusivamente dal programmatore. Ciò consente lo scambio di dati fra vari sottoprogrammi con estrema facilità. Entrambi gli stack pointer U e S possono venire impiegati come registri indici X e Y con le stesse identiche caratteristiche. In più supportano, com'è ovvio le istruzioni di push PSHS, PSHU e di pull PULS, PULU. Ciò consente di impiegare, in modo efficiente, la CPU M6809 come "Stack Processor" caratteristica di fondamentale importanza per supportare, con buona efficienza, linguaggi ad alto livello e tecniche di programmazione strutturata.

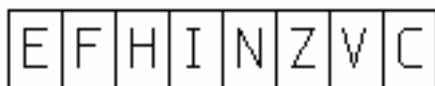
PROGRAM COUNTER

Il registro PC viene utilizzato dalla CPU per indicare l'indirizzo della successiva istruzione che deve essere eseguita dalla MPU.

È possibile in alcuni casi impiegare il registro PC come registro indice qualora si utilizzi il modo di indirizzamento relativo riferito al PC.

REGISTRO CCR (Condition-Code-Register)

Il registro CCR permette di definire lo stato interno della CPU in qualsiasi istante.



- C : Carry
- V : Overflow
- Z : Zero
- N : Negative
- I : IRQ Mask
- H : Half Carry
- F : Firq Mask
- E : Entire Flag

DESCRIZIONE DEI BIT ASSOCIATI AL CCR -CCRO (C)

Il bit CCRO o Carry Flag rappresenta il bit di riporto proveniente dallo svolgimento di operazioni aritmetiche entro l'ALU.

CCRO viene usato pure per rappresentare il riporto negativo derivante dallo svolgimento delle istruzioni di sottrazione quali (CMP,NEG,SUB,SBC). In questo caso il "carry-flag" rappresenta il complemento del carry proveniente dalla unità ALU.

-CCR1 (V) Il bit CCR1 flag di overflow viene portato a ALU da una operazione che produce un overflow di tipo aritmetico nella rappresentazione numerica in complemento a 2.

Questo overflow viene rilevato dalla unità ALU ogni volta che, a seguito di una operazione aritmetica, il riporto relativo al MSB non concorda con il riporto relativo al bit MSB-1.

-CCR2 (Z) Il bit CCR2 e' lo "zero-flag" e viene portato a LLA ogniqualvolta una operazione conduce ad un risultato identicamente nullo.

-CCR3 (N) Il bit CCR3 e' il "flag-negativo". In esso viene trasferito il valore del MSB (B7) del risultato dell'ultima operazione eseguita dalla CPU. Pertanto un risultato negativo (con B7 a LLA) produrrà un passaggio a LLA di CCR3, mentre un risultato positivo (con B7 a LLB) produrrà un passaggio a LLB di CCR3.

-CCR4 (I) Il bit CCR4 rappresenta il flag di mascheramento degli "interrupt hardware" che giungono alla CPU attraverso la linea IRQ .

La CPU non svolge la procedura di interrupt IRQ qualora CCR4 si trovi a LLA.

Le richieste di interrupt hardware NMI, FIRQ, IRQ, RESET e lo svolgimento di software interrupt SWI portano a LLA il flag CCR4. Le istruzioni SWI2 e SWI3 non modificano lo stato di CCR4.

-CCR5 (H) Il bit CCR5 rappresenta il flag di "half-carry".

Viene usato per segnalare la presenza di un riporto dal bit 3 nell'unità ALU solamente a seguito dello svolgimento di una istruzione di addizione ADD oppure ADC.

Il bit CCR5 viene utilizzato dall'istruzione DAA per effettuare l'operazione di "aggiustamento decimale" in modo da riconvertire in codice BCD, il risultato binario derivante dalla somma di termini BCD.

Lo stato del flag CCR5 risulta indefinito in tutte le istruzioni che fanno uso di operazioni di sottrazione.

-CCR6 (F) Il bit CCR6 rappresenta il flag di mascheramento degli interrupt hardware che vengono alla CPU attraverso la linea FIRQ.

La CPU non svolge la procedura di interrupt FIRQ qualora CCR6 si trovi a LLA.

Interrupt hardware del tipo NMI, FIRQ, RESET o software del tipo SWI portano sempre a LLA il bit CCR6, mentre interrupt hardware del tipo IRQ o software del tipo SWI2 e SWI3 non modificano lo stato del flag CCR6.

-CCR7 (E) CCR7 rappresenta il flag di utilizzazione dello stack nel corso di svolgimento di interruzioni.

Quando CCR7 si trova a LLA, ciò significa che è stato depositato nello stack lo stato completo della CPU M6809. Va osservato che in caso di svolgimento di una routine di interrupt di tipo FIRQ si verifica un deposito parziale nello stack del contenuto dei registri interni della CPU (in tal caso infatti viene collocato nello stack solo il contenuto del PC e del reg. CCR).

Il bit CCR7 del registro CCR collocato nello stack viene quindi utilizzato dall' MPU nel corso di svolgimento della istruzione RTI per stabilire l'entità dello stack da recuperare alla fine del servizio di interrupt. Pertanto il valore attuale del bit CCR7, presente all'interno della CPU M6809, e' rappresentativo dell'ultima operazione di scaricamento nello stack (completa se CCR7=1 , parziale se CCR7=0) .

DESCRIZIONE DELLE LINEE FACENTI CAPO ALLA CPU M6809

POWER (VSS, VCC) Alimentazione Due pin della CPU vengono usati per le necessità di alimentazione: Vss rappresenta la massa (a potenziale zero) mentre VDD rappresenta il positivo dell'alimentazione +5 V +5% .

ADDRESS BUS

(A0 - A15 Sedici pin vengono usati per consentire all' MPU di emettere un indirizzo a 16 bit sull' Address-bus)

Quando l' MPU non richiede il data-bus per trasferire dati, (cicli MPU con indirizzo non valido sull' Address-Bus) essa emette l'indirizzo esadecimale \$FFFF, la linea R/W si porta a LLA e l'uscita BS si porta a LLB. Da notare che, gli indirizzi sono validi sul fronte di salita del segnale Q (vedi fig.2 e 3 pag.).

Figura 11

Tutti i driver associati all'address-Buffer vengono posti in condizione di alta impedenza quando l'uscita BA della CPU si trova a LLA. Ciascun pin e' in grado di pilotare un carico TTL standard (oppure quattro carichi TTL LS) ed una capacità di 90 pF in parallelo.

DATA BUS D0-D7

Le otto linee del data-bus sono linee bidirezionali che consentono di comunicare con il data-bus del sistema.

Ciascuna linea del data-bus e' in grado di pilotare un carico TTL standard (oppure quattro carichi TTL LS) con in parallelo una capacità di 130 pF.

La linea R/W e' una uscita della CPU M6809. Attraverso questo segnale l' MPU indica la direzione di trasferimento dei dati sul data-bus. Se la linea R/W si trova a LLB ciò indica che l' MPU sta scrivendo sul data-bus.

La linea R/W viene posta in condizione di alta impedenza quando l'uscita BA si trova a LLA. Il segnale R/W è valido sul fronte di salita del segnale Q. (vedi fig. 1).

RESET Si tratta di un ingresso "triggerato" che se posto a LLB per un tempo maggiore di un ciclo MPU resetta l' MPU medesima.

Il vettore di Restart viene recuperato dalle locazioni ROM \$FFFE, \$FFFF allorquando la CPU si pone in condizione di interrupt or Reset Acknowledge BA=0 BS=1 fig.1.

Nella fase di applicazione iniziale dell'alimentazione, la linea di reset deve essere mantenuta a LLB fintantoche' il generatore di clock non sia completamente operativo

(vedi fig.2).

Poichè l'ingresso Reset della CPU M6809 è di tipo triggerato con una soglia di scatto superiore UTL maggiore di quella delle periferiche standard, è possibile usare una semplice rete RC per resettare l'intero sistema.

Il fatto che le periferiche abbiano una soglia di scatto UTL inferiore a quella della MPU M6809, assicura che quando le periferiche escono dalla situazione di Reset, il processore risulta ancora resettato.

HALT ingresso CPU

Un LLB sull'ingresso HLT provoca l'arresto dell'attività CPU a conclusione dell'istruzione che è in corso di svolgimento. La CPU rimane in stato di Halt indefinitamente senza perdita di dati. Quando la CPU si trova in stato di Halt, l'uscita BA si porta a LLA. Ciò indica che sia i Buffer del Data-Bus, sia i Buffer dell' Address-Bus, si trovano in condizione di alta impedenza. L'uscita BS si porta a LLA,

ciò indica che la CPU si trova in una delle situazioni: HALT o BUS GRANT (segnale di concessione del BUS che garantisce la disponibilità del Bus nel funzionamento DMA).

Mentre la CPU si trova in stato di Halt, non risponde a richieste di interrupt esterne (FIRQ, IRQ) nonostante i segnali di ingresso alla linea DMA/BREQ vengano sempre accettati e le richieste di interrupt NMI o RESET vengano memorizzate (latched) per successive risposte.

Mentre la CPU si trova in stato di Halt, non risponde a richieste di interrupt esterne (FIRQ,IRQ) nonostante i segnali di ingresso alla linea DMA/BREQ vengano sempre accettati e le richieste di interrupt NMI o RESET vengano memorizzate (latched) per successive risposte.

Mentre la CPU si trova in condizioni di Halt, le uscite Q ed E funzionano normalmente. Se la CPU non sta svolgendo alcuna istruzione (stato RESET o stato DMA/BREQ), è possibile passare in stato di Halt ($BA \cdot BS = 1$) portando a LLB l'ingresso Halt, mentre l'ingresso RESET è ancora a LLB. Se gli ingressi DMA/BREQ e HALT si trovano entrambi a LLB il uP raggiungerà l'ultimo ciclo della istruzione (by reverse cycle-stealing) quindi l'MPU passerà nello stato Halt. (vedi fig.2). Ciò indica che la CPU si trova in una delle situazioni: HALT o BUS GRANT (concessione del Bus) segnalazione che garantisce la disponibilità del Bus nel funzionamento DMA.

Mentre la CPU si trova in stato di Halt, non risponde a richieste di interrupt esterne (FIRQ,IRQ) nonostante i segnali di ingresso alla linea DMA/BREQ vengano sempre accettati e le richieste di interrupt NMI o RESET vengano memorizzate (latched) per successive risposte. (vedi fig.2).

USCITE BA (Bus available-Bus disponibile) e BS (Bus Status)

L'uscita BA fornisce una indicazione esterna sullo stato del segnale di controllo interno che porta i buffer interni del Data-Bus e dell'Address-Bus in condizione di alta-impedenza.

Questo segnale non implica che il bus sia disponibile per più di un ciclo.

Quando l'uscita BA passa a LLB, deve trascorrere un ciclo MPU per così dire "morto" prima che l'MPU riacquisti il controllo sul Bus.

Lo stato logico dell'uscita BS rappresenta unitamente a quello dell'uscita BA lo stato interno della unità M6809. (I segnali BA e BS sono validi sul fronte di salita dell'uscita Q).

BA	BS	STATO INTERNO MPU M6809
0	0	Normal Running
0	1	Int. Acknowledge, Reset
1	0	Sync Acknowledge
1	1	Halt or Bus Grant

Interrupt Acknowledge

Questa segnalazione è presente durante entrambi i cicli di fetch dei vettori di interrupt (RES, NMI, FIRQ, IRQ, SWI, SWI2, SWI3).

Questo segnale, unito alla decodifica delle 4 linee più basse dell'address-bus (A0-A3) fornisce all'utente una indicazione sul livello di interrupt che viene servito in un certo istante dall'MPU e consente altresì la vettorizzazione di un interrupt da parte di una periferica.

Mappa di Memoria dei Vettori di Interrupt

Locazione di Memoria dei Vettori	Vettore di Interrupt	Tipo	Mask
\$FFFE - \$FFFF	RESET	HD	NO
\$FFFC - \$FFFD	NMI	HD	NO
\$FFFA - \$FFFB	SWI	SF	
\$FFF8 - \$FFF9	IRQ	HD	SI
\$FFF6 - \$FFF7	FIRQ	HD	SI
\$FFF4 - \$FFF5	SWI2	SF	
\$FFF2 - \$FFF3	SWI3	SF	
\$FFF0 - \$FFF1	Reserved		

HD=Interrupt Hardware SF= Interrupt Software

Sync Acknowledge

Questo stato di funzionamento indica che l'MPU attende di sincronizzarsi su un evento esterno (rappresentato da un interrupt proveniente da una linea HDW esterna: IRQ, FIRQ, NMI).

Halt-Bus-Grant

In questa situazione la CPU segnala di trovarsi in situazione HLT (ingresso HLT a LLB) o segnala alle unita' DMA la concessione del Bus per effettuare l'accesso diretto alla memoria.

Ingresso NMI

Un fronte di discesa su questo ingresso, segnala all'MPU l'arrivo di una richiesta di interrupt non mascherabile. Una richiesta di interrupt NMI non può essere inibita a livello di programma ed ha priorità più elevata rispetto a richieste IRQ, FIRQ o a interrupt di tipo software. Nel corso di un servizio di interrupt NMI, l'intero stato interno della CPU viene posto in salvataggio nello stack-hardware.

A seguito di una operazione di reset, effettuata sulla CPU M6809 va osservato che le richieste di interrupt NMI non daranno luogo a servizi di interrupt finchè non viene eseguita una istruzione di caricamento dello stack-pointer hardware S (LDS).

La durata del segnale NMI (tempo in cui NMI rimane a LLB), deve essere per lo meno uguale ad un ciclo completo del segnale E.

Se l'ingresso NMI non raggiunge il minimo set-up rispetto al segnale Q, la richiesta di interrupt NMI non viene acquisita fino al successivo ciclo (vedi fig.9 Data-Sheet p.10).

FIRQ

Un LLB su questo ingresso dell'MPU M6809 da inizio ad una sequenza di interrupt mascherabile FIRQ a condizione che il flag F del CCR si trovi a LLB.

La sequenza di interrupt FIRQ ha priorità superiore rispetto a richieste interrupt di tipo IRQ ed è più veloce rispetto a quest'ultima perchè nel corso del suo svolgimento è previsto il deposito nello stack dei soli registri PC e CCR.

La routine di servizio di interrupt deve eliminare la segnalazione di interrupt prima che l'MPU esegua l'istruzione RTI.(fig.10 Data Sheet p.10).

IRQ

Un LLB su questo ingresso dell'MPU M6809 da inizio ad una sequenza di interrupt mascherabile IRQ a condizione che il flag I del CCR si trovi a LLB.

Dato che la sequenza di interruzione mascherabile IRQ provvede a depositare nello stack l'intero stato interno dell'MPU M6809 ne consegue che la risposta ad un interrupt IRQ risulta più lenta rispetto ad una risposta FIRQ.

La sequenza di interrupt IRQ ha priorità inferiore rispetto alla sequenza FIRQ.

Da notare che la routine di servizio di interrupt deve eliminare la segnalazione di interrupt prima che l'MPU esegua l'istruzione RTI. (fig.9 Data sheet p.10)

XTAL EXTAL

Questi ingressi vengono utilizzati per collegare l'oscillatore incorporato nell'MPU M6809 ad un quarzo esterno. Il pin EXTAL può essere usato come ingresso TTL per collegare l'MPU ad una sorgente esterna di clock; in tal caso l'ingresso XTAL deve essere collegato a GND (0 V.).

La frequenza di oscillazione del quarzo è 4 volte quella del Bus (vedi fig.7 Data sheet p.8).

Nel progetto del circuito stampato devono essere rispettate tutte le prescrizioni richieste da un circuito generatore di Radio-Frequenze.

E, Q (uscite MPU)

Il segnale E è simile al segnale DBE del bus M6800. Il segnale Q può anche definirsi "clock in quadratura con E". Q non ha alcun corrispondente nel bus M6800.

Gli indirizzi provenienti dalla CPU M6809 sono validi sul fronte di salita del segnale Q.

Il dato a 8 bit presente sul "DATA-BUS" viene memorizzato sul fronte di discesa del segnale E. Il Timing dei segnali E e Q è mostrato in fig.12 pg.11.

MRDY (input MPU)

Attraverso questo ingresso di controllo è possibile estendere la durata dei segnali E e Q allo scopo di incrementare il tempo di accesso in memoria da parte della CPU.

Quando l'ingresso MRDY si trova a LLA le uscite E e Q dell'MPU funzionano normalmente; quando MRDY si trova a LLB è possibile estendere la durata dei segnali E e Q di un numero intero di quarti di ciclo MPU (ciclo di clock esterno). Ciò consente l'interfacciamento tra MPU M6809 e memorie lente come mostrato in fig.12A p.13.

La massima estensione consentita è pari a 10 u.sec considerata la natura dinamica dei registri interni dell'MPU M6809.

Durante i cicli MPU in cui il uP non indirizza alcuna unità esterna (cicli MPU con VMA=0) l'ingresso MRDY non provoca alcun allungamento alla durata dei segnali Q e E; ciò consente di mantenere massima la velocità della CPU durante le fasi in cui l'MPU non effettua accessi in memoria.

L'ingresso MRDY può essere usato anche per allungare la durata dei segnali di clock (in caso di memorie lente) qualora il controllo del bus venga trasferito ad un dispositivo esterno (attraverso l'impiego dei segnali HALT e DMA/BREQ).

Si noti che quattro delle più recenti maschere di produzione dell'MPU M6809 (contrassegnate G7F, T5A, P6F, T6M) richiedono la sincronizzazione del segnale MRDY con il segnale di clock esterno (a frequenza 4xfo). Per effettuare tale sincronizzazione e' necessario ricorrere ad un oscillatore esterno come mostrato in fig. 12B p.13) La transizione negativa del segnale MRDY, normalmente derivata dal segnale di chip-select proveniente da una decodifica, deve rispettare segnale di chip-select proveniente da una decodifica, deve rispettare il "timing" richiesto dal parametro TPCSM.

Adottando il circuito di fig.12B p.13, la transizione positiva del segnale MRDY può capitare sul fronte di salita del segnale di clock esterno (freq. 4xfo).

Inoltre nella produzione con le maschere citate il segnale MRDY non provoca l'allungamento dei segnali E e Q se l'MPU sta eseguendo sia un'istruzione TFR che EXG nel corso del passaggio da LLA a LLB del segnale HALT.

Se l'MPU sta eseguendo una istruzione CWAI, il uP colloca nello stack il contenuto di tutti i suoi registri interni, quindi si pone in attesa di un interrupt.

Durante la fase di attesa è possibile porre l'MPU in situazione HALT allo scopo di sconnettere l'MPU dal Bus; in questa situazione l'ingresso MRDY non provoca alcun allungamento nei segnali di clock.

DMA/BREQ

L'ingresso DMA/BREQ permette di sospendere l'attività della CPU e di acquisire il Bus dell'MPU per altri usi come mostrato in fig.13 p.14.

Tipiche applicazioni si hanno nelle tecniche DMA e nelle tecniche di "refreshing" delle memorie dinamiche.

La transizione del segnale DMA/BREQ deve verificarsi durante la fase attiva del segnale Q.

Un LLB presente sull'ingresso DMA/BREQ arresta l'esecuzione dell'ultima istruzione a conclusione del corrente ciclo MPU.

L'MPU quindi, in risposta alla richiesta DMA/BREQ porta a LLA sia l'uscita BA sia l'uscita BS.

Il dispositivo che ha richiesto l'accesso al Bus ha quindi a disposizione fino ad un massimo di 15 cicli MPU prima che il uP riprenda il controllo sul Bus per l'operazione di rinfresco dei registri interni che richiede un ciclo MPU preceduto e seguito da cicli morti MPU. (fig.14 p.14).

In situazioni tipiche, il controllore DMA richiederà l'uso del Bus portando a LLB l'ingresso DMA/BREQ dell'MPU M6809 sul fronte di discesa del segnale E. Quando l'MPU risponde, portando a LLA entrambe le uscite BA e BS, quel ciclo è un ciclo morto e viene utilizzato dall'MPU per trasferire il controllo del Bus al controller DMA.

Si devono prevenire falsi accessi alla memoria che possono verificarsi durante qualsiasi ciclo-morto, costruendo un segnale DMA*VMA che si trova a LLB per l'intera durata di un ciclo successivo all'istante in cui l'uscita BA è variata.

Quando il segnale BA passa a LLB (sia per effetto di un LLA presente sull'ingresso DMA/BREQ, sia per effetto di una operazione di refreshing automatico dell'MPU, il dispositivo DMA deve essere

mantenuto sconnesso dal Bus. Un altro ciclo morto deve trascorrere prima che l'MPU possa accedere alla memoria per consentire il trasferimento del controllo del bus senza contenzione.

FUNZIONAMENTO DELLA CPU M6809

Nel suo normale funzionamento la CPU M6809 recupera una istruzione alla volta dalla memoria e quindi la esegue. Questa sequenza inizia con la sequenza di RESET e si ripete indefinitamente a meno che non venga alterata da una istruzione speciale o da un particolare evento Hardware.

Le istruzioni Software che alterano il normale funzionamento della CPU sono : SWI, SWI2, SWI3, CWAI, RTI e SYNC.

Un interrupt, una situazione HALT o DMA/BREQ possono anch'esse alterare la normale esecuzione delle istruzioni.

La fig.15 p.15 mostra il flow-chart per la CPU M6809.

MODI DI INDIRIZZAMENTO

Le istruzioni-base della CPU M6800 sono state potenziate fortemente nella CPU M6809 per la presenza di potenti modi di indirizzamento.

La CPU M6809 dispone del più completo set di modi di indirizzamento disponibili oggi su qualsiasi microcomputer.

La CPU M6809 dispone di un set di 59 istruzioni che risulta ridotto rispetto a quello della CPU M6800 (72 istruzioni).

Tuttavia i codici operativi diversi che si possono comporre risultano 1464 per l'M6809 contro i 197 della CPU M6800.

L'elevato numero di codici operativi possibili nel caso di CPU M6809, e' da ricercare nella potenza dei modi di indirizzamento di questa nuova CPU.

Da notare che i nuovi metodi di indirizzamento introdotti con l'MPU M6809 e peraltro non presenti nella CPU M6800 consentono di supportare le moderne tecniche di programmazione.

La CPU M6809 consente di utilizzare i seguenti "Modi di Indirizzamento":

- Indirizzamento Inerente (compreso l'indir. su Accumulatori)
- Indirizzamento Immediato
- Indirizzamento Esteso
- Indirizzamento Esteso Indiretto
- Indirizzamento Diretto
- Indirizzamento associato ai reg. MPU
- Indirizzamento Indicizzato:
 - con offset 0
 - con offset costante 5-8-16 bit
 - con offset A,B,D
 - con auto-incremento del puntatore
 - con auto-decremento del puntatore
 - indiretto
- Indirizzamento Relativo:
 - Branch lunghi e corti
 - al contenuto del PC

INDIRIZZAMENTO INERENTE

In questo modo di indirizzamento il codice operativo dell'istruzione contiene tutte le informazioni necessarie per lo svolgimento dell'istruzione medesima.

Esempi :

```
ABX
DAA
SWI
ASRA
CLRB
```

INDIRIZZAMENTO ASSOCIATO AI REGISTRI INTERNI DELLA CPU

I codici operativi associati alle istruzioni TFR, EXG, PSHS, PULS, PSHU, PULU devono essere seguiti da un apposito byte che definisce su quale registro o gruppo di registri interni all'MPU opera l'istruzione; questo byte viene chiamato post-byte.

Esempi :

```
TFR X,Y
EXG A,B
PSHS A,B,X,Y
PULU X,Y,D
```

INDIRIZZAMENTO IMMEDIATO

Nel modo di indirizzamento immediato, l'effettivo indirizzo del dato è la locazione immediatamente successiva al codice operativo. Il dato che deve essere usato dall'istruzione segue immediatamente il codice operativo dell'istruzione. La CPU M6809 usa valori immediati sia a 8 che a 16 bit a seconda della capacità del registro cui si riferisce l'istruzione.

Esempi di istruzioni con modo di indirizzamento immediato sono:

```
LDA #$20
LDX #$F000
LDY #TABLE
```

Ove # sta per immediato e \$ sta per esadecimale.

INDIRIZZAMENTO ESTESO

Nel modo di indirizzamento esteso, il contenuto dei due bytes che seguono immediatamente il codice operativo dell'istruzione, specificano in modo completo l'indirizzo effettivo a 16 bit dell'operando su cui opera l'istruzione. Si noti che l'indirizzo utilizzato da una istruzione tradotta con modo di indirizzamento esteso, definisce un indirizzo assoluto che non è indipendente dalla posizione occupata dal programma in memoria.

Sono esempi di indirizzamento esteso:

```
LDA TABLE
STX TEMPX
LDD $2000
```

INDIRIZZAMENTO ESTESO-INDIRETTO

L'indirizzamento esteso-indiretto e' stato implementato nell'MPU M6809 come caso particolare di indirizzamento indicizzato (che verrà preso in esame in seguito).

I due bytes successivi al post-byte di una istruzione estesa indiretta contengono quindi l'indirizzo ove risiede il puntatore del dato su cui opera l'istruzione stessa.

Esempi :

```
LDA [CAT]
LDX [$FFFE]
STU [DOG]
```

INDIRIZZAMENTO DIRETTO

Il modo di indirizzamento diretto è simile a quello esteso: nel modo diretto l'indirizzo dell'operando e' costituito da un solo byte che indica gli otto bit meno significativi dell'indirizzo assoluto dell'operando (da A0 ad A7) , gli 8 bit più significativi (da A8 ad A15) sono forniti dal contenuto del registro DPR.

Questo modo di indirizzamento richiede un solo byte per definire l'indirizzo, ne consegue quindi che sia la dimensione di un programma in memoria, sia il tempo necessario per la sua esecuzione risultano ridotti (naturalmente se posti a confronto con istruzioni o programmi analoghi impieganti il modo di indirizzamento esteso). Ovviamente è possibile accedere solamente a 256 locazioni di memoria diverse (costituenti una pagina) senza modificare il contenuto del registro DPR.

Il registro DPR viene azzerato ogniqualvolta l'MPU M6809 esegue la routine di RESET, l'indirizzamento diretto della CPU M6809 risulterà quindi compatibile con quello della CPU M6800. In questo caso non è consentita la forma indiretta.

Esempi :

```
LDA $30
SETDP $10 Direttiva Assembler
LDB $1030
LDD <CAT Il simbolo "<" è una direttiva dell'assembler M6809 che forza il modo di indirizzamento diretto .
```

INDIRIZZAMENTO INDICIZZATO

In tutti i modi di indirizzamento indicizzato, uno dei registri puntatori (X, Y, U, S e talvolta PC) viene usato per calcolare l'indirizzo effettivo dell'operando che viene utilizzato dalla istruzione.

Esistono 5 tipi fondamentali di indicizzazione che verranno discussi più avanti.

Il post-byte di una istruzione indicizzata, viene collocato nel programma subito dopo il codice operativo, esso specifica sia il tipo di indicizzazione, sia il registro puntatore usato.

In fig.xx p.YY è presentata una tabella che riassume i possibili formati consentiti per il post-byte.

PostByte								Modo di Indirizzamento Indicizzato
7	6	5	4	3	2	1	0	
0	R	R	d	d	d	d	d	EA = ,R + 5 Bit Offset
1	R	R	0	0	0	0	0	,R+
1	R	R	i	0	0	0	1	,R++
1	R	R	0	0	0	1	0	,-R
1	R	R	i	0	0	1	1	,--R
1	R	R	i	0	1	0	0	EA = ,R + 0 Offset
1	R	R	i	0	1	0	1	EA = ,R + ACCB Offset
1	R	R	i	0	1	1	0	EA = ,R + ACCA Offset
1	R	R	i	1	0	0	0	EA = ,R + 8 Bit Offset
1	R	R	i	1	0	0	1	EA = ,R + 16 Bit Offset
1	R	R	i	1	0	1	1	EA = ,R + D Offset
1	x	x	i	1	1	0	0	EA = ,PC + 8 Bit Offset
1	x	x	i	1	1	0	1	EA = ,PC + 16 Bit Offset
1	0	0	1	1	1	1	1	EA = [Address]

In Tab.2 p.17 sono riassunti i formati assembler previsti, il numero di cicli MPU ed il numero di bytes supplementari che devono essere aggiunti ai cicli MPU e al numero di bytes base a seconda del tipo di indicizzazione prescelta.

INDICIZZATO CON OFFSET 0

In questo modo di indirizzamento il puntatore selezionato contiene l'indirizzo effettivo del dato utilizzato dalla istruzione.

E' il modo di indirizzamento indicizzato più veloce.

Sono esempi :

```
LDD 0,X
LDA S
```

INDICIZZATO CON OFFSET COSTANTE

In questo modo di indirizzamento il contenuto del registro puntatore prescelto viene sommato ad un offset (rappresentato in complemento a 2). Dalla somma nasce l'indirizzo effettivo dell'operando coinvolto nell'istruzione.

Il contenuto iniziale del reg. puntatore non viene modificato dall'addizione citata.

Sono consentite tre possibilità per l'offset:

5 bit (da -16 a +15) 8 bit (da -128 a +127) 16 bit (da -32768 a +32767) In caso di offset a 5 bit esso viene incluso direttamente nel post-byte (si tratta del modo più efficiente di questo gruppo).

In caso di offset a 8 bit, esso viene collocato in memoria nel byte che segue il post-byte . In caso di offset a 16 bit, esso viene collocato in memoria nei due bytes che seguono il post-byte.

Nella maggior parte dei casi al programmatore non interessa scegliere tra offset a 8 o a 16 bit dato che a ciò provvede automaticamente l'assembler.

Esempi :

```
LDA 23,X
LDX -2,S
LDY 300,X
LDU CAT,Y
```

INDICIZZATO CON OFFSET IN ACCUMULATORE

In questo modo di indirizzamento il contenuto (in complemento a 2) di un registro accumulatore (A,B,o D) viene sommato al contenuto di un registro puntatore (X,Y,S o U) per formare l'indirizzo effettivo dell'operando. Sia il contenuto dell'accumulatore che quello del registro puntatore rimangono inalterati ad operazione eseguita . Il post-byte specifica quale accumulatore usare come offset . Non è richiesto alcun byte addizionale oltre al post-byte. Il vantaggio di un offset in accumulatore è che il valore dell'offset può essere calcolato nella fase di esecuzione del programma.

Esempi :

```
LDA B,Y
LDX D,Y
LEAX B,X
```

INDICIZZATO CON INCREMENTO/DECREMENTO AUTOMATICO

In questo modo di indirizzamento, il registro puntatore prescelto può operare in modo "post-increment" o "pre-decrement".

In caso di auto-incremento l'indirizzo effettivo è rappresentato dal contenuto del registro puntatore prescelto (stabilito nel post-byte); successivamente il reg. puntatore viene automaticamente incrementato (della quantità stabilita nel post-byte) : si parla più correttamente di post-incremento del registro puntatore.

In caso di auto-decremento il registro puntatore prescelto (indicato nel post-byte) viene innanzitutto decrementato (della quantità stabilita nel post-byte); successivamente il contenuto del reg. puntatore viene utilizzato come indirizzo effettivo su cui opera l'istruzione specifica.

L'entità dell'incremento o del decremento può essere di 1 o 2 unità : ciò consente all' MPU di accedere a tabelle di dati sia a 8 che a 16 bit .

Il modo "post-increment" o "pre-decrement" permette di gestire STACKS addizionali di tipo software di natura identica a quelli gestiti dai puntatori S e U.

Alcuni esempi di indirizzamento ad incremento/decremento automatico sono:

```
LDA ,X+
STD ,Y++
LDB ,-Y
LDX ,--S
```

L'entità dell'incremento o del decremento (+/-1 o +/-2) viene precisata nel post-byte. Si deve fare attenzione nell' operare su registri puntatori a 16 bit (X,Y,U,S), quando lo stesso registro viene usato per calcolare l'indirizzo effettivo.

Si consideri il seguente caso :

```
STX 0,X++ (X è inizializzato a 0)
```

Si desidera immagazzinare il dato "0" nelle locazioni \$0000 e \$0001, poi incrementare il registro indice "X" in modo che punti alla locazione \$0002. In realtà succede quanto segue:

```
0 ---> TEMP * TEMP e' un registro di supporto X+2 ---> X *
```

Effettua l'incremento automatico X ---> TEMP *

Esegue l'operazione di immagazzinamento

INDICIZZATO INDIRETTO

Tutti i modi indicizzati ad eccezione di quello ad incremento/decremento automatico, facenti uso di 1 o +/- 4 bit di offset, possono avere un livello addizionale di indirezione. Nell'indirizzamento indiretto l'indirizzo effettivo è contenuto nella locazione specificata dal contenuto del registro indice più un qualsiasi offset.

Nell'esempio seguente l'accumulatore A è caricato indirettamente usando un indirizzo effettivo calcolato tra registro indice ed un offset.

PRIMA DELL'ESECUZIONE:

A=XX (Contenuto privo di significato)
X=\$F000

\$0100 LDA [\$10,X] * EA è ora in \$F010

\$F010 \$F1 * \$F150 è ora il
\$F011 \$50 * nuovo EA
\$F150 \$AA

Dopo l'esecuzione:

A=\$AA (dato attualmente caricato)

X=\$F000 Sono disponibili tutti i modi indicizzati indiretti eccettuati quelli privi di significato (es. incr./decr. automatico tramite un indiretto).

Alcuni esempi sono:

LDA [,X]
LDD [10,S]
LDA [B,Y]
LDD [,X++]

INDIRIZZAMENTO RELATIVO

Questo modo di indirizzamento viene usato quando si desidera effettuare salti nel corso della esecuzione del programma partendo dalla posizione corrente indicata dal PC. Se la condizione di salto è vera, viene calcolato l'indirizzo effettivo (PC + offset segnato) e il salto viene effettuato. Se la condizione di salto è falsa, la CPU prosegue eseguendo l'istruzione successiva.

Si osservi che quando viene calcolato l'indirizzo effettivo EA, il PC punta sempre all'istruzione successiva all'offset.

I salti di tipo BRANCH vengono usati nella stesura di programmi che a livello di codice macchina debbono risultare "position independent" ossia naturalmente rilocabili.

Sono disponibili due tipi di indirizzamento relativo:

- corto (un byte di offset)
- lungo (due bytes di offset)

Nel primo caso il byte successivo al codice operativo dell'istruzione BRANCH viene utilizzato come offset segnato a 8 bit per calcolare l'indirizzo effettivo EA della successiva istruzione da eseguire.

Evidentemente in caso di branch di tipo corto sarà possibile saltare +127 o -128 bytes partendo dalla locazione di programma successiva al byte di offset.

Nel secondo caso la coppia di bytes successivi al codice operativo dell'istruzione BRANCH vengono utilizzati come offset segnato a 16 bit per calcolare l'indirizzo effettivo EA della successiva istruzione da eseguire.

In caso di long-branch sarà possibile saltare +32767 o -32768 bytes partendo dalla locazione di programma successiva a quella dei bytes di offset.

Di seguito sono riportati alcuni esempi di indirizzamento relativo:

```
      BEQ  CAT (corto)
      BGT  DOG (corto)
CAT   LBEQ RAT (lungo)
DOG   LBGT RABBIT (lungo)
* * *
```

```
RAT      NOP
RABBIT   NOP
```

RELATIVO AL PROGRAM COUNTER

Il program counter PC può venire usato come registro puntatore congiuntamente ad offset segnati a 8 o 16 bit. Analogamente a quanto accade in caso di indirizzamento relativo, l'offset viene sommato al contenuto attuale del program counter, per costruire l'indirizzo effettivo EA su cui opera l'istruzione .

L'indirizzamento relativo al program counter viene utilizzato per scrivere programmi che a livello di codice macchina debbono risultare "position independent" ossia naturalmente rilocabili.

Sono esempi:

```
      LDA CAT,PCR
      LEAX TABLE,PCR
```

La posizione relativa della tabella TABLE rispetto al punto ove essa viene richiamata rimane invariata anche in caso di movimento del codice oggetto .

Poiché l'indirizzamento relativo al program counter appartiene alla famiglia dei modi di indirizzamento indicizzati , è disponibile un livello addizionale di indizione.

```
      LDA [CAT,PCR]
      LDU [DOG,PCR]
```

SET ISTRUZIONI

Il set di istruzioni dell' MC6809 è simile a quello dell' MC6800 ed a livello di codice sorgente l'unità MC6800 è compatibile con l'unità MC6809, ma non viceversa .

Il numero di codici operativi è stato ridotto da 72 a 59, ma a causa dell'espansione dell'architettura e degli ulteriori modi di indirizzamento, il numero di codici operativi disponibili (con differenti modi di indirizzamento) è salito da 197 a 1464 . Alcune delle istruzioni sono descritte in modo dettagliato successivamente.

PSHU / PSHS

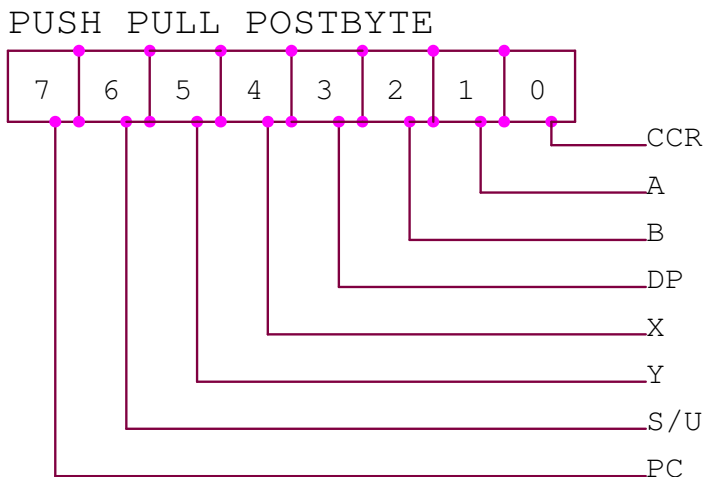
Le istruzioni PUSH consentono di scaricare o nell' HARDWARE STACK (S) o nello USER STACK (U), un solo registro o un qualsiasi gruppo di registri con un'unica istruzione.

PULU / PULS

Le istruzioni PULL si comportano in modo inverso alle istruzioni PUSH .

Il byte che segue immediatamente il codice operativo delle istruzioni PUSH o PULL viene denominato post-byte e determina quale registro o gruppo di registri debbano essere caricati o scaricati nello STACK.

La sequenza di PUSH/PULL è fissa: ogni bit e' associato ad un unico registro da caricare o scaricare come di seguito indicato:



Stacking Order

Pull Order

U/S Lo

PC Hi

PC Lo :

:

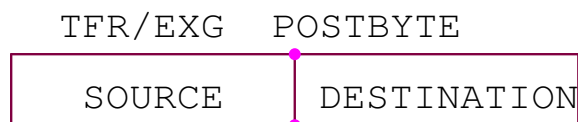
Push Order

Increasing Memory

TFR / EXG

All'interno della CPU M6809, e' possibile effettuare trasferimenti o scambi di contenuti fra registri delle medesime dimensioni (es.: 8 bit con 8 bit, 16 bit con 16 bit). Anche in questo caso dopo il codice operativo dell'istruzione TFR o EXG e' presente un post-byte :

i bits da 4 a 7 del post-byte definiscono il registro sorgente , mentre i bits da 0 a 3 rappresentano il registro di destinazione come di seguito indicato :



Value	Register
0000	D(A:B)
1000	A
0001	X
1001	B
0010	Y
1010	CCR
0011	U
1011	DPR
0100	S
0101	PC

NOTE Le combinazioni non presenti in tabella sono da considerarsi indefinite e perciò non consentite .

LEAX / LEAY / LEAU / LEAS

L'istruzione LEA viene utilizzata per caricare l' indirizzo effettivo EA usato da un' istruzione indicizzata all' interno di un registro puntatore.

Questo gruppo di istruzioni consentono al programmatore di utilizzare tutta la potenza dell' hardware interno di indirizzamento della CPU M6809.

Alcune istruzioni di questo tipo vengono illustrate in tavola 3.

L' istruzione LEA permette all' utente l' accesso a dati e tabelle in modo indipendente dalla posizione occupata dal programma.

LEAX	MSG1,PCR
LBSR	PDATA (stampa la routine di messaggio) * *
MSG1 FCC	"MESSAGE"

Questo semplice programma scrive : "MESSAGE". La notazione "MSG1,PCR", serve per far calcolare all' assemblatore l'offset esistente tra la posizione corrente del PC ed il messaggio MSG1

In fase di esecuzione, l'istruzione LEAX, viene indicizzata dal PC perciò verrà calcolato e caricato entro X l'indirizzo assoluto di MSG1 .

Le istruzioni LEA sono molto potenti e usano un registro di supporto interno (TEMP). Bisogna prestare attenzione quando si usa l'istruzione LEA con modo di indirizzamento che prevede incremento/decremento automatico, e' opportuno tenere sempre presente la sequenza di operazioni interne .

La sequenza interna di LEA è schematizzata come segue:

LEA a ,b+

(a e b rappresentano uno qualsiasi dei 4 puntatori i a 16 bit X,Y,U o S)

- b ---> temp (calcola l' EA)
- b+1 ---> b (modifica b e post-incrementa)
- temp ---> a (carica a)

LEA a, -b

- b-1 ---> temp (calcolato l'EA con pre-decremento)
- b-1 ---> b (modifica b e pre-decrementa)
- temp ---> a (carica a)

TABELLA 3 - Esempi di LEA

Istruzione	Operazione	Commento
LEAX 10,X	X+10 --> X	Somma il valore 10 (5 bit) ad X
LEAX 500,X	X+500 --> X	Somma il valore 500 (16 bit) ad X
LEAY A,Y	Y+A --> Y	Somma il contenuto di A ad Y
LEAY D,Y	Y+D --> Y	Somma il contenuto di D ad Y
LEAU -10,U	U-10 --> U	Sottrae il valore 10 da U
LEAS -10,S	S-10 --> S	Riserva memoria nello Stack
LEAS 10,S	S+10 --> S	Riposiziona lo Stack (Cancellandolo)
LEAX 5,S	S+5 --> X	Trasferisce mentre effettua la somma

Le istruzioni di incremento o decremento automatico di due unità funzionano in modo simile.

Si noti che:

- a) LEAX ,X+ non cambia X;
- b) LEAX ,X- decrementa X;
- c) LEAX 1,X deve essere sempre usata per incrementare di uno X.

MUL

Moltiplica quantità binarie di tipo "unsigned" presenti dentro gli accumulatori A e B e pone il risultato "unsigned" entro l'accumulatore D a 16 bit.

SYNC

Qualora venga incontrata nel corso di un programma un'istruzione SYNC, la CPU entra in uno stato di "wait"; cessa la elaborazione delle istruzioni e inizia la fase di attesa di una richiesta di interrupt di tipo hardware .

In presenza di interrupt di tipo NMI oppure di tipo IRQ o FIRQ con bit di mascheramento F o I azzerato, la CPU esce dallo stato di "wait" ed esegue la routine di interrupt.

Poichè FIRQ ed IRQ non sono ingressi attivi sui fronti ma sui livelli , l'acquisizione di richieste IRQ o FIRQ implica una loro attività per almeno 3 cicli MPU.

In presenza di interrupt di tipo IRQ o FIRQ con bit di mascheramento F o I settato, la CPU esce dallo stato di "wait" e prosegue eseguendo l'istruzione successiva alla SYNC .

Figura 18 contiene il processo di sync.

- INTERRUZIONI SOFTWARE La CPU M6809 dispone di ben 3 istruzioni di Software Interrupt SWI, SWI2, SWI3.

Quando la CPU esegue una istruzione di Software Interrupt, lo stato interno della CPU viene scaricato entro lo Stack e successivamente viene eseguita la routine di interrupt corrispondente .

Le interruzioni software vengono comunemente usate in fase di debug dei programmi e per effettuare chiamate a routines del sistema operativo. La esecuzione di un servizio di interrupt di tipo SWI porta a 1 i bit F ed I del CCR per impedire che richieste di tipo FIRQ o IRQ diano luogo ai servizi di interrupt conseguenti.

La esecuzione di servizi di interrupt del tipo SWI2 o SWI3 non modifica affatto lo stato del CCR quindi eventuali richieste di interrupt di tipo hardware FIRQ o IRQ verranno prese in considerazione dalla CPU.

Ne consegue che SWI e' collocata su un livello di priorita' superiore a SWI2 ed SWI3 .

- OPERAZIONI A 16 BIT L'unita' M6809 pur essendo una CPU a 8 bit e' in grado di manipolare dati a 16 bit.

Il set di istruzioni a 16 bit comprende "Load" e "Store" , "Compare" , istruzioni di Somma, Sottrazione, trasferimento, ed infine operazioni a 16 bit sugli stacks S ed U.

FUNZIONAMENTO CICLO DOPO CICLO DELLA CPU

La tabella "Address Bus Cycle by Cycle Performance" di pag. 22 illustra le modalità d'accesso alla memoria per ogni istruzione della CPU M6809.

Ogni istruzione inizia con la fase di "fetch" del codice operativo. Mentre questo viene internamente decodificato, il program byte seguente viene recuperato ; La maggior parte di istruzioni infatti utilizza il byte successivo al codice operativo per trarre informazioni necessarie alla esecuzione dell'istruzione.

Poi l'esecuzione di ogni istruzione seguirà la carta di flusso. VMA (Valid Memory Address = indirizzo di memoria valido) è un'indicazione esadecimale corrispondente a \$FFFF sull'address bus, R/W = 1 e BS = 0. I seguenti esempi illustrano l'uso della carta di flusso.

Esempio 1:

LBSR (salto) prima dell'esecuzione lo stack pointer contiene il dato \$F000 (SP = F000)

**

\$8000 LBSR CAT

**

\$A000 CAT *

SEQUENZA DELLE OPERAZIONI CICLO DOPO CICLO

CICLO #	INDIRIZZO	DATO	R/W	DESCRIZIONE		
1	8000	17	1	Recupero OP CODE		
2	8001	20	1	Offset High		
3	8002	00	1	Offset Low		
4	FFFF	*	1	Ciclo VMA		
5	FFFF	*	1	Ciclo VMA		
6	A000	*	1	Indirizzo di salto		
7	FFFF	*	1	Ciclo VMA		
8	EFFE	80	0	Deposita nello stack	\$80	ADRH
9	EFFE	03	0	Deposita nello stack	\$03	ADRL

Esempio: DEC (esteso)

\$8000 DEC \$A000

\$A000 \$80

SEQUENZA DELLE OPERAZIONI CICLO DOPO CICLO

CICLO #	INDIRIZZO	DATO	R/W	DESCRIZIONE
1	8000	7A	1	Recupero OP CODE
2	8001	A0	1	ADRH operando
3	8002	00	1	ADRL operando
4	FFFF	*	1	Ciclo VMA
5	A000	80	1	Lettura Dato
6	FFFF	*	1	Ciclo VMA
7	A000	7F	0	Scrittura Dato