Enrico Tombelli
Docente presso
ITC "A. Volta" - Bagno a Ripoli - Firenze
(e.tombelli@libero.it)

IL MICROPROCESSORE

8 BIT (µP)

IL MICROPROCESSORE A 8 BIT (µP)

- LA LOGICA PROGRAMMATA

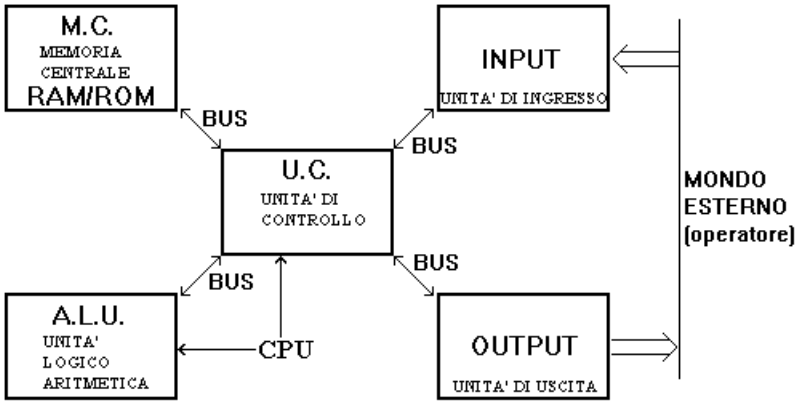
La logica programmata è un tipo di logica sequenziale nella quale l'evoluzione del processo non dipende solo dalla rete fisica (porte logiche e connessioni), ma anche dalle informazioni in essa memorizzate tramite lo stato della rete. L'informazione, influenza quindi lo svolgimento del calcolo ed è composta da una serie di direttive (istruzioni) poste in sequenza. Questa sequenza logica è detta "PROGRAMMA"[1].

A differenza di una logica cablata (composta esclusivamente dalla parte fisica della rete e definita dalle connessioni nonché dai dispositivi che ne fanno parte [HARDWARE]), la logica programmata è più versatile in quanto permette la modifica di volta in volta della funzione realizzata variando semplicemente le istruzioni che compongono il programma (e quindi non solo i dati d'ingresso). Ciò è possibile in quanto il programma può essere memorizzato in una memoria RAM nella quale è possibile riscrivere più volte, oppure in una memoria EPROM.

Lo svantaggio della logica programmata è comunque la lentezza, dovuta al fatto che le istruzioni, per essere eseguite, devono prima essere acquisite (fetch) e successivamente interpretate (decodifica) e tali operazioni comportano un certo tempo d'attuazione.

- L'ELABORATORE ELETTRONICO O COMPUTER

L'evoluzione della logica programmata ha portato alla realizzazione di reti sequenziali sempre più complesse, fino ad arrivare ad apparati che permettono di elaborare grossissime quantità di dati in tempi brevissimi. La tecnologia è così arrivata a realizzare gli ELABORATORI ELETTRONICI DIGITALI che funzionano secondo lo schema a blocchi di figura. In esso si rileva la presenza di varie parti:



SCHEMA A BLOCCHI DI UN ELABORATORE ELETTRONICO

- U.C. (Unità Centrale). Ha il compito di coordinare il lavoro delle altre parti.
- M.C. (Memoria Centrale). È la parte che memorizza i dati iniziali, finali e quelli temporanei. Essa è composta da circuiti integrati di due tipi ovvero sia ROM (che ospita il FIRMWARE, cioè il programma iniziale di lavoro) che RAM (quest'ultima, all'accensione, è completamente vuota che quindi serve ad ospitare il programma caricato dall'operatore di volta in volta, nonché i dati del sistema).
- A.L.U. (Unità Logico-Aritmetica). È la "CALCOLATRICE" del sistema e ha il compito di svolgere calcoli sia logici (Confronti, AND, OR, NOT ecc.) che aritmetici (Somme, sottrazioni ecc.).
- INPUT (Unità d'Ingresso). Serve per immettere i dati dall'esterno (p.e. tastiera)
- OUTPUT (Unità d'Uscita). Serve per comunicare all'esterno i risultati dell'elaborazione (p.e. Video e stampante)
- Tutte le parti comunicano con l'Unità di Controllo tramite un sistema di BUS.

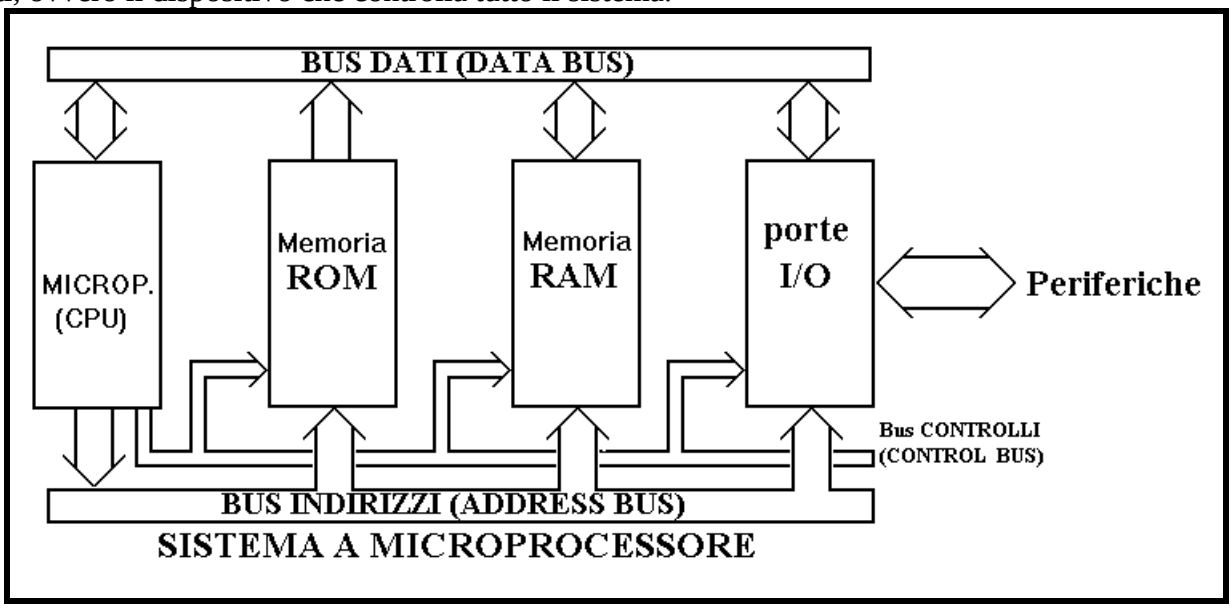
[1] Il termine "PROGRAMMA" fa parte di una terminologia più estesa associata alla parola "SOFTWARE". Quest'ultima deriva dalla contrapposizione alla parola "HARDWARE" il cui significato corrisponde alla parte fisica del sistema. Per software non si intende solo il programma, ma tutto ciò che è INFORMAZIONE e che quindi risiede in una memoria.

NB: i due blocchi costituiti dalla Unità di Controllo (UC) e dalla Unità Logico-Aritmetica (ALU) formano l'UNITÀ CENTRALE DEL PROCESSO (C.P.U.= Central Processing Unit)

(CU) + (ALU) = (CPU)

- IL MICROPROCESSORE

Il MICROPROCESSORE è un componente elettronico integrato (Circuito integrato) capace di eseguire una sequenza d'istruzioni, detta "programma", residenti in una memoria esterna (RAM/ROM) e codificate in codice binario. In pratica il µp è una CPU (Central Processing Unit - UNITÀ CENTRALE) a tutti gli effetti, ovvero il dispositivo che controlla tutto il sistema.



Il µp, per funzionare deve essere assemblato con altri componenti elettronici:

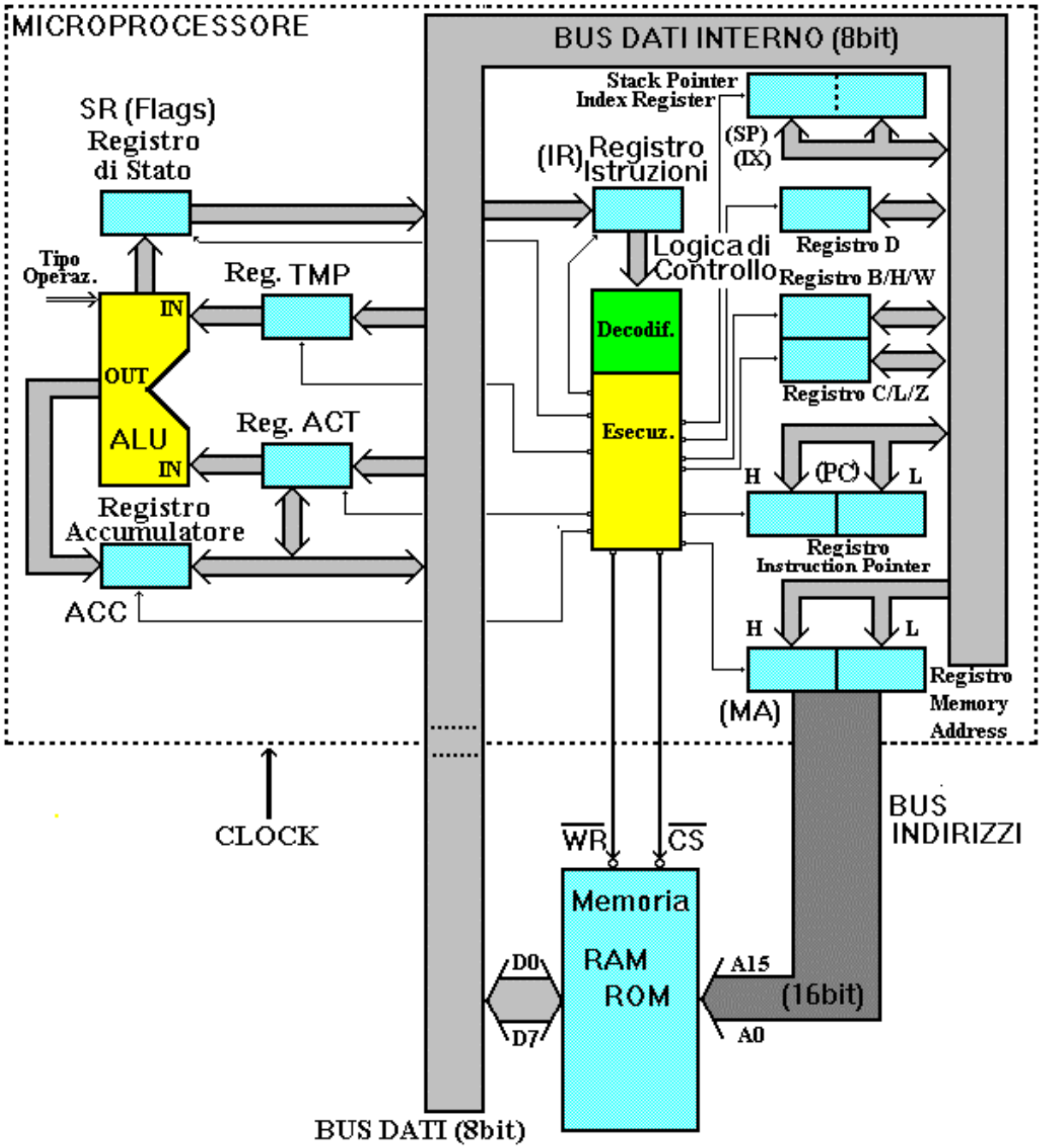
- Memoria ROM. Contiene le istruzioni del programma (FIRMWARE) che la CPU o µp deve eseguire.
- Memoria RAM. È la memoria di lavoro dove sono memorizzati i dati d'ingresso, i risultati intermedi e i dati d'uscita che variano di volta in volta e durante tutta l'esecuzione del programma.
- Unità di I/O. Costituisce l'interfaccia del sistema verso l'esterno (ovvero verso le periferiche).

Tutti questi dispositivi comunicano fra loro tramite 3 tipi di bus.

- Bus indirizzi (address bus). Unidirezionale, trasporta gli indirizzi dal µp agli altri componenti.
- Bus controlli (control bus). Unidirezionale, trasporta i segnali di comando del µp agli altri componenti e i segnali di stato nel senso opposto.
- Bus dati (data bus). Bidirezionale, trasporta i dati da un componente all'altro sotto la supervisione della CPU (microprocessore).

Oltre alle parti sopra descritte, il microprocessore necessita, per funzionare, di un sistema di temporizzazione (CLOCK) e d'alimentazione elettrica. Un apparato così composto è detto "SISTEMA A MICROPROCESSORE". Nota: il n° di linee del bus dati (p.e. 8) definisce il n° di bit del sistema a µp.

- ARCHITETTURA DI UN MICROPROCESSORE



I tipi di dispositivi che formano l'interno di un µp sono 3:

- Unità logico aritmetica (ALU)
- Logica di controllo
- Registri vari

Questi dispositivi comunicano fra di loro e con l'esterno tramite un sistema di bus interni al µp.

- UNITÀ LOGICO-ARITMETICA (ARITHMETIC-LOGIC-UNIT: ALU)

L'unità Logico-Aritmetica è il dispositivo che effettua le operazioni logiche e aritmetiche; è una rete puramente combinatoria[2]. Essa presenta due porte d'ingresso (IN) e una porta principale d'uscita (OUT).

In base al tipo di funzione impostata dalla logica di controllo essa effettua svariati tipi di operazioni matematiche (Logiche: "L" o Aritmetiche: "A") sui dati posti nelle porte d'ingresso (IN), facendo apparire immediatamente i risultati sulla porta d'uscita (OUT). Tipiche operazioni sono: confronto bit a bit (L), somma (A), sottrazione (A), complemento (L), ecc.

Le ALU sono dispositivi che possono essere venduti a parte su circuiti integrati diversi dall'unità centrale (per esempio: ALU a CMOS MC 14581 B (Motorola). Può svolgere 16 operazioni aritmetiche + 16 operazioni logiche su dati di 4 bit ciascuno)

Non avendo, le ALU, memoria, qualsiasi dato applicato ai suoi ingressi appare immediatamente in uscita. Dato che non è disponibile in ingresso un DOPPIO BUS (architettura a BUS SINGOLO), è necessario bufferizzare sia le porte d'ingresso che quelle d'uscita. A questo scopo si dispone dei REGISTRI TEMPORANEI (TMP e ACT) e dell'ACCUMULATORE (ACC). Per sintetizzare il risultato delle operazioni logiche o aritmetiche appena svolte, l'ALU, dispone di tre linee d'uscita, che vanno a forzare tre bit di un registro, detto REGISTRO DI STATO (status register - SR).

I tre bit sono: risultato nullo (Z), risultato negativo (N), riporto presente (CY). Altri flag, presenti nello status registrer dipendono dal tipo di microprocessore.

- REGISTRI INTERNI

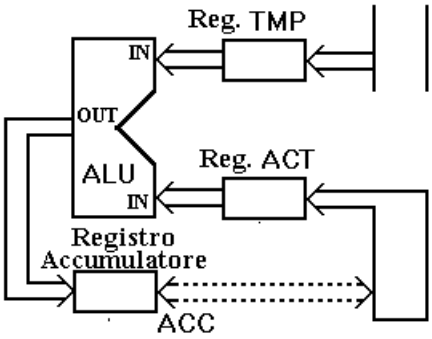
Ogni registro ha la funzione di memorizzare temporaneamente una informazione binaria (normalmente di 8 bit). Le informazioni possono essere di vario tipo:

- dati: numeri (BCD o binario semplice) - caratteri (in codifica ASCII)
- stato del sistema
- istruzioni (o parti di esse)
- indirizzi
- comandi

Sia il numero sia la capacità dei registri interni dipende dal µp utilizzato. Nota: dato che ipotizzeremo un µp a 8 bit, tutti i registri sono a 8 bit, il bus dati è a 8 bit e il bus indirizzi è a 16 bit.

Accumulatore (ACC - 8 bit)

Si trova all'uscita dell'ALU ed è utilizzato generalmente come deposito per il risultato e per uno degli operandi delle operazioni aritmetiche, nonché come deposito per il trasferimento di dati da una cella di memoria a un'altra.
Nota: l'accumulatore non accede direttamente all'ingresso dell'ALU, ma è bufferizzato da un registro temporaneo (ACT).
P.e. operazione somma



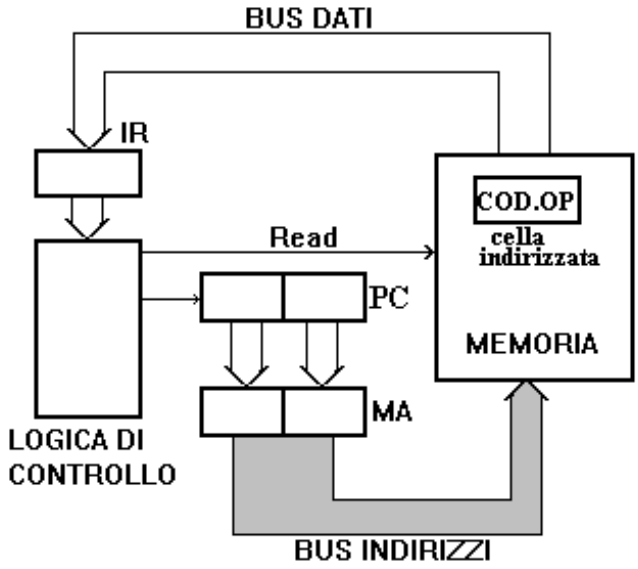
[2]Rete combinatoria: ogni istante l'uscita dipende esclusivamente dall'ingresso applicato attualmente e non dagli ingressi applicati in precedenza. Quindi l'ALU non ha memoria.

1. dato1 → ACC
2. dato2 → TMP
3. ACC (dato1) → ACT
4. risultato → ACC

Se non ci fosse l'ACT il risultato presente in uscita al momento del caricamento in ACC, si ripresenterebbe all'ingresso variando il risultato e causando un ciclo continuo. Il risultato sparisce proprio mentre lo si sta caricando in ACC.

Program Counter (PC - 16 bit) ovvero puntatore all'istruzione (è detto anche Instruction Poiter IP).

È uno dei registri più importanti del µp. Il PC contiene passo per passo l'indirizzo della cella di memoria in cui è memorizzata l'istruzione, o parte di essa, che deve essere eseguita o è in fase di esecuzione. Pertanto, il PC, trasferisce generalmente i suoi dati sul bus indirizzi che è a 16 bit. (Si possono quindi indirizzare 2^{16}=64K celle di memoria). Al momento dello START (Accensione o dopo ogni operazione di Reset) il PC è azzerato, quindi punta alla locazione di memoria d'indirizzo 0 (esadecimale 0000H). Il PC è incrementato automaticamente dal µp, senza necessità di comandi generati esternamente, dopo l'esecuzione di una qualsiasi istruzione. Istruzioni apposite possono influenzare quest'automatismo per variare, in base a certe condizioni, la normale sequenza d'esecuzione delle istruzioni (Salti). Pertanto se il contenuto del PC viene messo nel registro indirizzi (Memory Address - MA), viene indirizzata la cella di memoria che contiene la prossima istruzione da eseguire. Al comando di lettura, emesso dal µp, il contenuto della cella indirizzata viene riversato sul bus dati e quindi incanalato nel registro istruzioni (Instruction Register) per l'esecuzione. Nel frattempo il PC è incrementato per indirizzare la cella di memoria successiva.



Nota: il PC punta sempre all'istruzione successiva a quella appena recuperata.

Il PC può essere considerato costituito da due registri di 8 bit ciascuno:

- PC (H) - Parte alta più significativa
- PC (L) - Parte bassa meno significativa

Ciò allo scopo di rendere possibile il caricamento sul bus dati a 8 bit. Infatti il PC viene memorizzato in due volte.

Registro indirizzi di memoria - Memory Address (MA - 16 bit)

È il registro tampone che fa da confine fra il bus interno a 8 bit (Data bus) e il bus indirizzi esterno. Pertanto è anch'esso a 16 bit ed è diviso in due registri di 8 bit:

- MA (H) - Parte alta più significativa
- MA (L) - Parte bassa meno significativa

Il MA può essere caricato da molte sorgenti, non solo dal PC, visto che è collegato al bus dati interno.

Registro istruzioni - Instruction register (IR - 8 bit)

L'IR contiene il codice dell'istruzione che si sta attualmente eseguendo. Il IR presenta il codice direttamente all'ingresso della logica di controllo che coordina tutte le funzioni del µp in base al codice operativo stesso. L'IR può solo scrivere dati sul bus interno, ma non generarli.

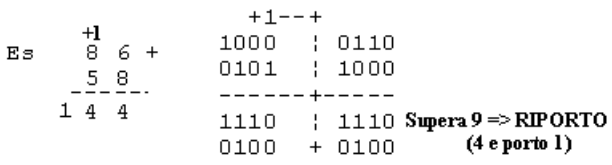
Registro di stato - Status register (SR - 8 bit)

In esso è sintetizzato il risultato dell'operazione appena eseguita (Ovvero lo stato del sistema). Analizzando (Tramite opportune istruzioni) i bit del registro di stato si possono prendere decisioni sulla sequenza delle operazioni da eseguire (e quindi effettuare dei Salti condizionati). La generica configurazione del registro di stato è la seguente:

Z	N	CY	O	P	x	I	B

- Z = Bit di "Zero". È a 1 se il risultato dell'ultima operazione effettuata dall'ALU è nullo; a 0 se il risultato è diverso da 0.
- N = Bit di segno (Negative). È a 1 quando il risultato dell'ultima operazione effettuata dall'ALU è negativo.
- CY = Bit di "Carry". È a 1 se il risultato dell'ultima operazione aritmetica effettuata dall'ALU ha generato un riporto.
- O = Bit di "Overflow". È a 1 se il risultato dell'ultima operazione effettuata dall'ALU ha superato in modulo, il massimo numero consentito.
- P = Bit di "Parità". È a 1 quando si è verificato un "errore di parità". (Viene contato il numero di 1 nel registro contenente il risultato, se è pari => P=0 viceversa P=1).
- I = Bit di "Interrupt".
- B = Bit di "Codifica B.C.D.". Esso è a 1 se c'è riporto sul 4° bit, o se questo ha superato "9". Il successivo è il bit di "Mezzo riporto" o "Half-Carry".

Tutti (o quasi) i bit del registri di stato possono essere settati a "0" o a "1" con apposite istruzioni.



Registri per i dati temporanei (TMP e ACT 8 bit). Sono due:

- TMP. Fra il bus dati interno e l'ALU.
- ACT. Fra l'accumulatore e l'ALU.

Servono per la memorizzazione temporanea dei dati operandi sui quali l'ALU deve agire. Non possono essere generalmente utilizzati direttamente in fase di programmazione, in quanto la loro funzione è solo di tamponi per l'ALU.

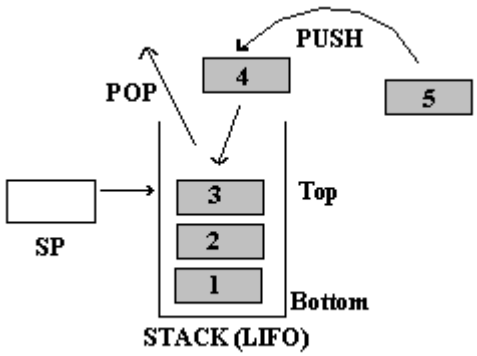
Registri di uso generale - General porpose (8 bit).

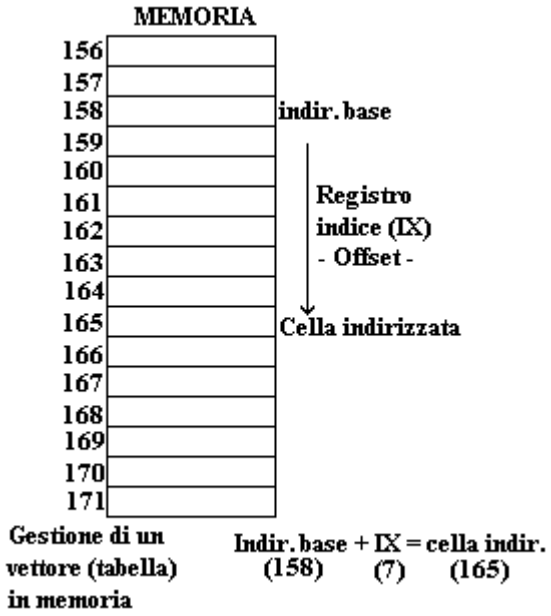
Il numero di questi registri dipende dal µp utilizzato. Alcuni di essi sono utilizzati genericamente come semplice zona di memoria; ad altri invece ci si può riferire con istruzioni particolari, e vengono utilizzati per funzioni specifiche, come lo STACK POINTER (SP) e l'INDEX REGISTER (IX o IY). Alcuni di essi possono essere accoppiati tramite certe istruzioni per formare registri a 16 bit (p.e. registro B+C, H+L o Z+W).

Puntatore allo stack - Stack pointer (SP - 16 bit).

Lo STACK è una zona di memoria organizzata a struttura L.I.F.O. (Last-In, First-Out) ovvero a pila (p.e. pila di piatti); l'ultimo ad entrare è il primo ad uscire.

- operazione d'ingresso: PUSH (o PULL)
- operazione d'uscita: POP





Lo stack pointer contiene l'indirizzo della prima cella di memoria vuota dello stack (lo SP punta sempre alla testa dello stack per rendere più veloce l'operazione di Push).

Registro indice - Index register - (IX - 8 bit).

Contiene generalmente dei valori di offset da aggiungere agli indirizzi allo scopo di mantenere costante un indirizzo base. È utile per la gestione di vettori o tabelle in modo semplice ed efficiente.

- LOGICA DI CONTROLLO (LC)

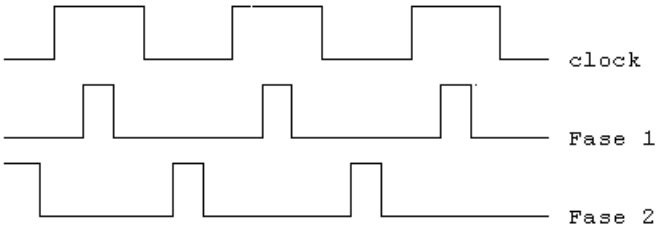
La logica di controllo è il cuore del microprocessore ed è una rete sequenziale il cui ingresso è alimentato dal codice dell'istruzione contenuta nel registro istruzioni (CODICE OPERATIVO: OPCODE). Ad ogni impulso di clock ogni uscita della LC genera una sequenza di segnali dipendenti dal particolare OPCODE presente in ingresso. Ogni uscita della LC è un comando ad un particolare dispositivo interno del µp (p.e. controllo delle operazioni ALU, scrittura in un registro ecc.) oppure esterno al µp (p.e. comando di W/R alla memoria). La LC contiene intrinsecamente un microprogramma e ad ogni impulso del clock (STATO INTERNO) viene eseguita una operazione (microistruzione) che porta il µp in un particolare stato interno. Si può quindi pensare alla LC come suddivisa in due parti ognuna delle quali genera una fase dell'esecuzione di un'istruzione.

- DECODIFICATORE:Effettua la decodifica del CODICE ISTRUZIONE, ovvero il riconoscimento della operazione da svolgere.
- ESECUTORE: Genera i segnali di comando dei dispositivi.

L'esecuzione di un'istruzione può essere quindi costituita da tre fasi:

1. FETCH (acquisizione). L'istruzione viene prelevata dalla memoria e inserita nel REGISTRO ISTRUZIONI. (IR)
2. DECODE (decodifica).
3. EXECUTE (esecuzione)

La TEMPIFICAZIONE è generata da un segnale di clock che viene fornito dall'esterno al µp. Tale segnale è scomposto internamente al µp in più segnali (dette fasi del clock) come in figura.

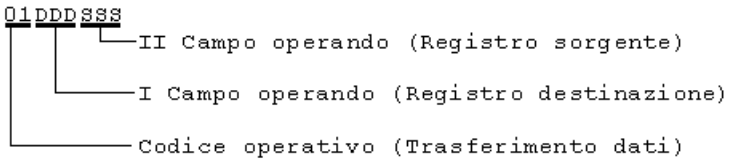


Il clock a più fasi viene generato da un oscillatore controllato in frequenza che sfrutta le caratteristiche di stabilità frequenziali di un cristallo di quarzo che fissa il valore campione (p.e. 2.5 MHz).

- ISTRUZIONI DI UN MICROPROCESSORE

La funzione che il sistema a µp deve svolgere è dettata da un programma, ovvero, una sequenza di istruzioni, ognuna delle quali permette di eseguire una particolare OPERAZIONE ELEMENTARE. La sequenza di istruzioni (programma) risiede in memoria (RAM o ROM) e per questo le istruzioni devono essere codificate in binario. Ogni sequenza di bit rappresenta una funzione specifica (istruzione) che svolge la corrispondente operazione da fare (p.e. spostamento di un carattere da una cella ad un'altra, somma fra due numeri, incremento, salto ecc.). L'insieme di tutte le possibili operazioni che il µp può compiere si dice REPERTORIO del µp.

Il formato di una istruzione è costituito da due tipi di campi:



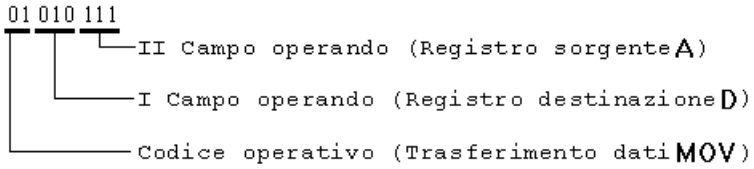
- Campo Operatore. Specifica il tipo di operazione da fare ed è detto CODICE OPERATIVO (OPCODE).
- Campi Operando. Specificano su che cosa deve agire il comando.

Il codice dei registri è il seguente:
(Vedi tabella del repertorio 8080)

Codice	Registro
000	B
001	C
010	D
011	E
100	H
101	L
110	MEMORIA[3]
111	ACC

esempio: istruzione INTEL 8080 su 8 bit (MOV r1,r2 -vedi repertorio)

Operazione realizzata: trasferisce il contenuto del registro sorgente, identificato dai bit SSS, nel registro destinazione, identificato dai bit DDD.



Ponendo come sorgente il registro ACC (SSS=111), e come destinazione il registro "D" (DDD=010), si ha il trasferimento del contenuto dell'accumulatore nel registro "D".

Note:

1. Tale operazione fa una COPIA del registro accumulatore, ovvero effettua il trasferimento del dato da ACC→D, senza distruggere il contenuto dell'ACC.
2. Qualunque valore contenga il registro D è soppresso e rimpiazzato dalla copia del dato contenuto in ACC.

[3]L'indirizzo della cella operando si trova nel registro HL dove è stato precedentemente caricato.

REPERTORIO DEL MICROPROCESSORE Intel 8080

ISTRUZIONE SUCCESSIVA

CODICE MNEMONICO	CODICE OPERATIVO (OPCODE)		M1(1)					M2			M3			M4			M5						
	D7D6D5D4	D3D2D1D0	T1	T2 (2)	T3 (3)	T4	T5	T1	T2 (2)	T3	T1	T2 (2)	T3	T1	T2 (2)	T3	T1	T2 (2)	T3	T4	T5	T1	T2 (2)
ACI data	1 1 0 0	1 1 1 0	PC → MA abil. mem.	PC=PC+1	C.Op.→IR	(A)→ACT		PC → MA abil. mem. (6)	PC=PC + 1 B2→TMP		(9)	(ACT) + (TMP) + CY→A											
ADC M	1 0 0 0	1 1 1 0	" "	" "	" "	(A)→ACT		HL → MA abil. mem. (6)	DATA BUS→TMP		(9)	(ACT) + (TMP) + CY→A											
ADC r	1 0 0 0	1 S S S	" "	" "	" "	(SSS)→TMP (A)→ ACT		(9)	(ACT) + (TMP) + CY→A														
ADD M	1 0 0 0	0 1 1 0	" "	" "	" "	(A)→ACT		HL → MA abil. mem. (6)	DATA BUS→TMP		(9)	(ACT) + (TMP)→A											
ADD r	1 0 0 0	0 S S S	" "	" "	" "	(SSS)→TMP (A) →ACT		(9)	(ACT) + (TMP)→A														
ADI data	1 1 0 0	0 1 1 0	" "	" "	" "	(A)→ACT		PC → MA abil. mem. (6)	PC=PC+1 B2→TMP		(9)	(ACT) + (TMP)→A											
ANA M	1 0 1 0	0 1 1 0	" "	" "	" "	(A)→ACT		HL → MA abil. mem. (6)	DATA BUS→TMP		(9)	(ACT) + (TMP)→A											
ANA r	1 0 1 0	0 S S S	" "	" "	" "	(SSS)→TMP (A) →ACT		(9)	(ACT) + (TMP)→ A														
ANI data	1 1 1 0	0 1 1 0	" "	" "	" "	(A) →ACT		PC → MA abil. mem. (6)	PC=PC + 1 B2→TMP		(9)	(ACT) + (TMP) →A											
C cond addr(17)	1 1 C C	C 1 0 0	" "	" "	" "	VERIFICA IF TRUE, SP=SP-1	CONDIZ. SP→SP-1	PC → MA abil. mem. (6)	PC = PC + 1 B2→Z		PC → MA abil. mem. (6)	PC = PC + 1 B3→W (13)		SP → MA abil. mem. (16)	(PCH) → SP=SP-1	DATA BUS	SP → MA (16) abil. mem.[1]	(PCL) →	DATA BUS			WZ → MA (11) abil. mem.	(WZ) + 1 →PC
CALL addr	1 1 0 0	1 1 0 1	" "	" "	" "	SP=SP-1		PC → MA abil. mem. (6)	PC = PC + 1 B2→Z		PC → MA abil. mem. (6)	PC = PC + 1 B3→W		SP → MA abil. mem. (16)	(PCH) → SP=SP-1	DATA BUS	SP → MA (16) abil. mem.	(PCL) →	DATA BUS			WZ → MA (11) abil. mem.	(WZ) + 1 →PC
CMA	0 0 1 0	1 1 1 1	" "	" "	" "	(A)→A																	
CMC	0 0 1 1	1 1 1 1	" "	" "	" "	___ CY→CY																	
CMP M	1 0 1 1	1 1 1 0	" "	" "	" "	(A)→ACT		HL → MA abil. mem. (6)	DATA BUS→TMP		(9)	(ACT) - (TMP) → FLAGS											
CMP r	1 0 1 1	1 S S S	" "	" "	" "	(A)→ACT (SSS)→TMP		(9)	(ACT) - (TMP) →FLAGS														
CPI data	1 1 1 1	1 1 1 0	" "	" "	" "	(A)→ACT		PC → MA abil. mem. (6)	PC=PC + 1 B2→TMP		(9)	(ACT) - (TMP) → FLAGS											
DAA	0 0 1 0	0 1 1 1	" "	" "	" "	DAA→A, FLAGS (10)																	
DAD Rp (8)	0 0 R p	1 0 0 1	" "	" "	" "	DECOD.		(Rp_L)→ACT	(L)→TMP (ACT) + (TMP)→ALU	ALU→L, CY	(Rp_L) → ACT	(H)→TMP (ACT) + (TMP)+ CY→ALU	ALU→H, CY										
DCR M	0 0 1 1	0 1 0 1	" "	" "	" "	DECODIFICA		HL → MA abil. mem. (6)	DATA BUS→TMP (TMP) - 1→ALU		HL → MA abil. mem. (7)	ALU→DATA BUS											
DCR r	0 0 D D	D 1 0 1	" "	" "	" "	(DDD)→TMP (TMP) - 1→ALU	ALU→ DDD																
DCX Rp	0 0 R p	1 0 1 1	" "	" "	" "	(Rp) - 1→Rp																	
DI	1 1 1 1	0 0 1 1	" "	" "	" "	RESET INTE F/F																	
EI	1 1 1 1	1 0 1 1	" "	" "	" "	SET INTE F/F																	
HLT	0 1 1 1	0 1 1 0	" "	" "	" "	DECODIFICA		PC → MA abil. mem.	HALT MODE (20)														
IN port	1 1 0 1	1 0 1 1	" "	" "	" "	DECODIFICA		PC → MA abil. mem. (6)	PC=PC + 1 B2→ Z		WZ → MA abil. mem. (18)	DATA BUS→A											
INR M	0 0 1 1	0 1 0 0	" "	" "	" "	DECODIFICA		HL → MA abil. mem. (6)	DATA BUS→TMP (TMP) + 1 → ALU		HL → MA abil. mem. (7)	ALU→DATA BUS											
INR r	0 0 D D	D 1 0 0	" "	" "	" "	(DDD)→TMP (TMP)+ 1→ALU	ALU→ DDD																
INX Rp	0 0 R P	0 0 1 1	" "	" "	" "	(Rp) + 1→Rp																	
J cond addr (17)	1 1 C C	C 0 1 0	" "	" "	" "	VERIFICA	CONDIZ.	PC → MA abil. mem. (6)	PC = PC + 1 B2→Z		PC → MA abil. mem. (6)	PC = PC + 1 B3→W										WZ → MA (11) abil. mem.	(WZ) + 1 →PC
JMP addr	1 1 0 0	0 0 1 1	" "	" "	" "	DECODIFICA		PC → MA abil. mem. (6)	PC = PC + 1 B2→Z		PC → MA abil. mem. (6)	PC = PC + 1 B3→W										WZ → MA (11) abil. mem.	(WZ) + 1 →PC
LDA addr	0 0 1 1	1 0 1 0	" "	" "	" "	DECODIFICA		PC → MA abil. mem. (6)	PC=PC + 1 B2→Z		PC → MA abil. mem. (6)	PC=PC + 1 B3→W		WZ → MA abil. mem. (6)	DATA BUS→A								
LDAX Rp (4)	0 0 R P	1 0 1 0	" "	" "	" "	DECODIFICA		Rp → MA abil. mem. (6)	DATA BUS→A														
LHLD addr	0 0 1 0	1 0 1 0	" "	" "	" "	DECODIFICA		PC → MA abil. mem. (6)	PC=PC + 1 B2→Z		PC → MA abil. mem. (6)	PC=PC + 1 B3→W		WZ → MA abil. mem. (6)	DATA BUS→L WZ=WZ + 1		WZ → MA abil. mem. (6)	DATA BUS→H					
LXI Rp , data	0 0 R P	0 0 0 1	" "	" "	" "	DECODIFICA		PC → MA abil. mem. (6)	PC=PC + 1 B2→R $_{PL}$		PC → MA abil. mem. (6)	PC=PC + 1 B3→R_{PH}											
MOV M , r	0 1 1 1	0 S S S	" "	" "	" "	(SSS)→TMP		HL → MA abil. mem. (7)	(TMP)→DATA BUS														
MOV r , M	0 1 D D	D 1 1 0	" "	" "	" "	DECODIFICA		HL → MA abil. mem. (6)	DATA BUS→DDD														
MOV r1 , r2	0 1 D D	D S S S	" "	" "	" "	(SSS)→TMP	(TMP)→ DDD																
MVI M , data	0 0 1 1	0 1 1 0	" "	" "	" "	DECODIFICA		PC → MA abil. mem. (6)	B2→TMP		HL → MA abil. mem. (7)	(TMP)→DATA BUS											
MVI r , data	0 0 D D	D 1 1 0	" "	" "	" "	DECODIFICA		PC → MA abil. mem. (6)	B2→DDD														
NOP	0 0 0 0	0 0 0 0	" "	" "	" "	DECODIFICA																	
ORA M	1 0 1 1	0 1 1 0	" "	" "	" "	(A)→ACT		HL → MA abil. mem. (6)	DATA BUS→TMP		(9)	(ACT) + (TMP)→A											
ORA r	1 0 1 1	0 S S S	" "	" "	" "	(A)→ACT (SSS)→TMP		(9)	(ACT)+ (TMP)→A														
ORI data	1 1 1 1	0 1 1 0	" "	" "	" "	(A)→ACT		PC → MA abil. mem. (6)	PC = PC + 1 B2→TMP		(9)	(ACT) + (TMP) →A											
OUT port	1 1 0 1	0 0 1 1	" "	" "	" "	DECODIFICA		PC → MA abil. mem. (6)	PC=PC + 1 B2→Z		WZ → MA abil. mem. (18)	(A) →	DATA BUS										

REPERTORIO DEL MICROPROCESSORE Intel 8080

ISTRUZIONE SUCCESSIVA

CODICE MNEMONICO	CODICE OPERATIVO (OPCODE)		M1(1)					M2			M3			M4			M5						
	D7D6D5D4	D3D2D1D0	T1	T2 (2)	T3 (3)	T4	T5	T1	T2 (2)	T3	T1	T2 (2)	T3	T1	T2 (2)	T3	T1	T2 (2)	T3	T4	T5	T1	T2 (2)
PCHL	1 1 1 0	1 0 0 1	PC → MA abil. mem.	PC=PC+1	C.Op.→IR	(HL)→PC																	
POP PSW	1 1 1 1	0 0 0 1	" "	" "	" "	DECODIFICA		SP → MA abil. mem. (15)	SP=SP + 1 DATA BUS→FLAGS		SP → MA abil. mem. (15)	SP = SP + 1 DATA BUS→A											
POP Rp	1 1 R P	0 0 0 1	" "	" "	" "	DECODIFICA		SP → MA abil. mem. (15)	SP=SP + 1 DATA BUS→Rp_L		SP → MA abil. mem. (15)	SP = SP + 1 DATA BUS→Rp_H											
PUSH PSW	1 1 1 1	0 1 0 1	" "	" "	" "	SP=SP - 1		SP → MA abil. mem. (16)	SP=SP - 1 (A) → DATA BUS		SP → MA abil. mem. (16)	FLAGS→ DATA BUS											
PUSHRp	1 1 R p	0 1 0 1	" "	" "	" "	SP=SP - 1		SP → MA abil. mem. (16)	SP=SP - 1 (Rp_H)→ DATA BUS		SP → MA abil. mem. (16)	(Rp_L) → DATA BUS											
R cond addr (17)	1 1 C C	C 0 0 0	" "	" "	" "	VERIFICA CONDIZ. (14)		SP → MA abil. mem. (15)	SP= SP + 1 DATA BUS→Z		SP → MA abil. mem. (15)	SP = SP +1 DATA→W										WZ → MA (11) abil. mem.	(WZ) + 1 →PC
RAL	0 0 0 1	0 1 1 1	" "	" "	" "	(A) CY→ALU ROTATE		(9)	ALU→(A),CY														
RAR	0 0 0 1	1 1 1 1	" "	" "	" "	(A) CY→ALU ROTATE		(9)	ALU→(A),CY														
RET	1 1 0 0	1 0 0 1	" "	" "	" "	DECODIFICA		SP → MA abil. mem. (15)	SP = SP + 1 DATA BUS→Z		SP → MA abil. mem. (15)	SP = SP +1 DATA BUS→W										WZ → MA (11) abil. mem.	(WZ) + 1 →PC
RLC	0 0 0 0	0 1 1 1	" "	" "	" "	(A) →ALU ROTATE		(9)	ALU→(A),CY														
RRC	0 0 0 0	1 1 1 1	" "	" "	C.Op.→TMP/IR	(A) →ALU ROTATE		(9)	ALU→(A),CY														
RST n	1 1 N N	N 1 1 1	" "	" "	0→W INS→TMP/IR	SP = SP - 1		SP → MA abil. mem. (16)	SP = SP - 1 (PCH)→ DATA BUS		SP → MA abil. mem. (16)	(TMP=00NNN000) →Z (PCL) → DATA BUS										WZ → MA (11) abil. mem.	(WZ) + 1 →PC
SBB M	1 0 0 1	1 1 1 0	" "	" "	C.Op.→TMP/IR	(A)→ACT		HL → MA abil. mem. (6)	DATA BUS→ TMP		(9)	(ACT) - (TMP) - CY→A											
SBB r	1 0 0 1	1 S S S	" "	" "	" "	(SSS)→TMP (A) → ACT		(9)	(ACT) - (TMP) - CY →A														
SBI data	1 1 0 1	1 1 1 0	" "	" "	" "	(A) → ACT		PC → MA abil. mem. (6)	PC=PC + 1 B2→ TMP		(9)	(ACT) - (TMP) - CY→A											
SHLD addr	0 0 1 0	0 0 1 0	" "	" "	" "	DECODIFICA		PC → MA abil. mem. (6)	PC=PC + 1 B2→Z		PC → MA abil. mem. (6)	PC=PC + 1 B3→W		WZ → MA abil. mem. (7)	(L)→DATA BUS WZ=WZ + 1		WZ → MA abil. mem.(7)	(H)→DATA BUS					
SPHL	1 1 1 1	1 0 0 1	" "	" "	" "	(HL)→SP																	
STA addr	0 0 1 1	0 0 1 0	" "	" "	" "	DECODIFICA		PC → MA abil. mem. (6)	PC=PC + 1 B2→Z			PC=PC + 1 B3→W		WZ → MA abil. mem. (7)	(A)→DATA BUS								
STAX Rp (4)	0 0 R P	0 0 1 0	" "	" "	" "	DECODIFICA		Rp → MA abil. mem. (7)	(A)→ DATA BUS														
STC	0 0 1 1	0 1 1 1	" "	" "	" "	1→CY																	
SUB M	1 0 0 1	0 1 1 0	" "	" "	" "	(A)→ACT		HL → MA abil. mem. (6)	DATA BUS→ TMP		(9)	(ACT) - (TMP)→A											
SUB r	1 0 0 1	0 S S S	" "	" "	" "	(SSS)→TMP (A)→ACT		(9)	(ACT) - (TMP)→A														
SUI data	1 1 0 1	0 1 1 0	" "	" "	" "	(A)→ACT		PC → MA abil. mem. (6)	PC=PC + 1 B2→ TMP		(9)	(ACT) - (TMP)→A											
XCHG	1 1 1 0	1 0 1 1	" "	" "	" "	(HL)←→(DE)																	
XRA M	1 0 1 0	1 1 1 0	" "	" "	" "	(A) →ACT		HL → MA abil. mem. (6)	DATA BUS→ TMP		(9)	(ACT) + (TMP) →A											
XRA r	1 0 1 0	1 S S S	" "	" "	" "	(A) →ACT (SSS) →TMP		(9)	ACT) + (TMP)→A														
XRI data	1 1 1 0	1 1 1 0	" "	" "	" "	(A) →ACT		PC → MA abil. mem. (6)	PC=PC + 1 B2→ TMP		(9)	(ACT) + (TMP) →A											
XTHL	1 1 1 0	0 0 1 1	" "	" "	" "	DECODIFICA		SP → MA abil. mem. (15)	SP=SP + 1 DATA BUS→Z		SP → MA abil. mem. (15)	DATA BUS→W		SP → MA abil. mem. (16)	(H)→DATA BUS		SP → MA (16) abil. mem.	(L) →DATA BUS		WZ →HL		WZ → MA (11) abil. mem.	(WZ) + 1 →PC

NOTE :

1) Il primo ciclo di memoria (M1) è sempre un'istruzione eseguibile ; durante questo ciclo viene eseguito il primo ed unico byte contenente il codice operativo.
2) Se l'ingresso READY dalla memoria non è alto durante T2 di ogni ciclo di memoria, il processore entrerà in uno stato di attesa.
3) Gli stati T4 e T5 sono presenti, come richiesto, per i funzionamenti completamente interni alla CPU. I contenuti del bus interno durante T4 e T5 sono disponibili sul bus dati ; questo è previsto solamente per scopi di testing. Lo stato è presente ma è impiegato soltanto per funzionamenti interni come la decodifica dell'istruzione.
4) Possono essere specificate solo le coppie di registri Rp = B (registro B e C) oppure Rp = D (registri D e E).
5) Questi stati vengono saltati.
6) La memoria legge sottocicli : verrà letta un'istruzione oppure una parola dati.
7) La memoria scrive sottocicli.
8) Il segnale READY non è richiesto durante il secondo e terzo sottociclo (M2 ed M3). Il segnale HOLD è accettato durante M2 ed M3. Il segnale SYNC non è generato durante M2 ed M3. Durante l'esecuzione di DAD sono richiesti M2 ed M3 per una somma di coppia di registri interni ; non c'è riferimento alla memoria.
9) I risultati di queste operazioni aritmetiche, logiche o di rotazione non vengono mossi nell'accumulatore A fino allo stato T2 del successivo stato di istruzione. Cioè A viene caricato mentre si sta eseguendo l'istruzione successiva : questa sovrapposizione consente un'elaborazione più veloce.
10) Se il valore dei 4 byte meno significativi dell'accumulatore è maggiore di 9, oppure è uguale a 1 il bit di carry ausiliario, viene sommato 6 all'accumulatore. Se ora il valore dei 4 bit più significativi dell'accumulatore è maggiore di 9, oppure è uguale ad un bit di carry, viene sommato 6 ai 4 bit più significativi dell'accumulatore.
11) Questo rappresenta il primo sottociclo (l'esecuzione dell'istruzione) del successivo ciclo di istruzione.
12) Se la condizione è soddisfatta i contenuti della coppia di registri WZ vengono fatti uscire sulle linee di indirizzo (A_0 - A_{15}) invece del contenuto del Program Counter (PC).
13) Se la condizione non è soddisfatta, vengono saltati i sottocicli M4 ed M5 ; il processore procederà immediatamente all'esecuzione di istruzione (M1) del successivo ciclo d'istruzione.
14) Se la condizione non è soddisfatta, vengono saltati i sottocicli M2 ed M3 ; il processore procederà immediatamente all'esecuzione d'istruzione (M1) del successivo ciclo d'istruzione.
15) Sottociclo di lettura dello stack.
16) Sottociclo di scrittura dello stack.
17) CONDIZIONE

	CONDIZIONE	CCC
NZ	non zero (Z = 0)	000
Z	zero (Z = 1)	001
NC	non riporto (CY =0)	010
C	riporto (CY = 1)	011
PO	parità dispari (P = 0)	100
PE	parità pari (P = 1)	101
P	più (S = 0)	110
M	Meno (S = 1)	111

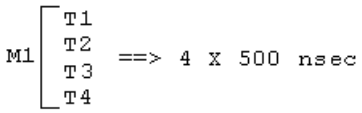
18) Sottociclo di I/O : il codice di selezione ad 8 bit della porta I/O è duplicata sulle linee d'indirizzo 0 -7(A0 - A7) ed 8 - 15 (A8 - A15).
19) Sottociclo d'uscita.
20) Il processore permarrà nello stato di hall finchè non si accetta un interrupt, un reset oppure un hold. Quando si accetta una richiesta di hold la CPU entra nel modo hold. Al termine del modo hold il processore ritorna nello stato di hall. Dopo che è stato accettato un reset il processore esegue l'istruzione forzata sul bus dati (normalmente un'istruzione restart).

SSS o DDD	Valore	Rp	Valore
A	111	BC	00
B	000	DE	01
C	001	HL	10
D	010	SP	11
E	011		
H	100		
L	101		
MEMORIA	110		

NB: SSS e DDD rappresentano un registro interno a 8 bit (SSS: sorgente; DDD: destinazione). Rp è un registro doppio a 16 bit

- ESECUZIONE D'UNA ISTRUZIONE AD UN BYTE

L'esecuzione di un'istruzione, distinta nelle tre fasi (Fetch, Decodifica e Esecuzione), si può considerare costituita da uno o più cicli macchina (in base alla sua lunghezza). Ad ognuno di questi corrisponde un ACCESSO AL BUS DATI. Ogni ciclo macchina (Mi) è costituito da più Stati interni o microistruzioni (Tj), ognuno dei quali corrisponde un impulso di Clock.



P.e. Con Clock a 2MHz, 1 stato interno ⇔ 500 nsec. Un'istruzione di 1 ciclo macchina (M1) di 4 stati interni ha una durata di 2 µsec.

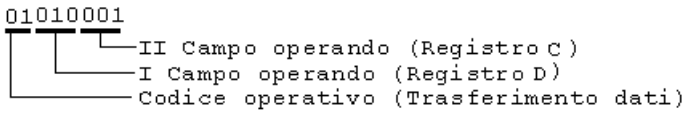
Nota: la lunghezza di un'istruzione può essere più lunga di 1 byte. In questo caso l'istruzione è distribuita in più celle di memoria e pertanto l'acquisizione (Fetch) necessita di più accessi alla memoria. Si ha quindi più di un ciclo macchina.

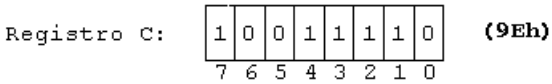
- TRASFERIMENTO DATI FRA REGISTRI INTERNI (1 BYTE)

Supponiamo che l'istruzione da eseguire sia il trasferimento dei dati (MOV r1,r2) e che tale istruzione sia Memorizzata nella cella di indirizzo "3":

INDIRIZZO (0003h) =	0000000b	000000011b
	PC(H)	PC(L)

Da Registro C (Sorgente:SSS=001) ------ > a Registro D (Destinazione:DDD=010)

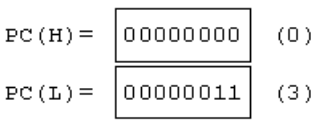




Supponiamo inoltre che nel registro C sia memorizzato il valore 158 (ovvero 9Eh base 16)

FASE DI FETCH

È la prima fase (Primi 3 stati del 1° Ciclo Macchina "M1"). La fase di fetch è uguale per tutte le istruzioni, in quanto si va a prendere in memoria la parte relativa al codice OPERATIVO.



Al momento dell'esecuzione, nel'PROGRAM COUNTER si trova l'indirizzo della cella che contiene l'istruzione da eseguire (p.e. valore 3):

CICLO MACCHINA "M1"

STATO INTERNO T1

La logica di controllo effettua due operazioni (quasi contemporanee):

1) Mette in comunicazione il PC con il registro indirizzi di memoria (MA) e il contenuto del PC viene copiato nel MA in due volte (Dato che il bus interno è a 8 bit):

PC(H) → MA(H) / PC(L) → MA(L)

Pertanto dopo quest'operazione sul Bus Indirizzi (EXT) c'è l'indirizzo della locazione di memoria dove risiede l'istruzione.

2) Contemporaneamente mette sul Bus Controlli (EXT) il comando di lettura (abilitazione uscita Memoria, (CS=0). La Memoria reagirà riversando sul Bus Dati il codice istruzione contenuta nella cella indirizzata.

NOTA: questo succede solo dopo che è trascorso un certo tempo d'accesso. Nel frattempo viene effettuata la 2° microistruzione che non utilizza il Bus Dati (incremento del PC).

STATO INTERNO T2 (Incremento del PC)

Il PC viene incrementato AUTOMATICAMENTE dopo ogni accesso alla memoria.

PC = PC + 1 (PC contiene ora il valore 4)

STATO INTERNO T3

La Logica di Controllo effettua anche, contemporaneamente all'indirizzamento della cella, l'abilitazione a ricevere il codice dell'istruzione, ovvero attiva il comando di caricamento del REGISTRO ISTRUZIONI (IR). Pertanto durante gli stati interni T2 e T3 il codice istruzione fluisce dalla cella di memoria sul BUS DATI e così all'interno della CPU fino a raggiungere l'IR.

STATI INTERNI T4 E T5

Fino ad ora l'operato della Logica di Controllo è stato indipendente dall'istruzione da eseguire, in quanto questa non era nota perché doveva essere recuperata. Pertanto questi primi 3 passi sono AUTOMATICI e uguali per ogni istruzione.

Adesso in ingresso alla Logica di Controllo si trova il codice dell'istruzione, che influenza l'emissione dei comandi in uscita.

Il codice binario dell'istruzione (codice operativo) viene ora DECODIFICATO ed eseguito, ovvero:

$$T4: \quad C \rightarrow TMP$$

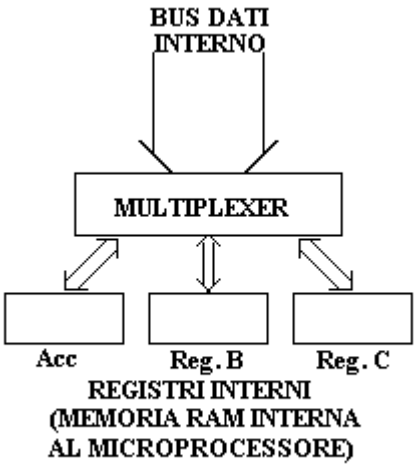
La LC mette in comunicazione il registro C con il registro TMP in ingresso all'ALU e abilita il registro TMP; i dati contenuti in C vengono COPIATI in TMP.

$$T5: \quad TMP \rightarrow D$$

Con lo stesso sistema i dati contenuti in TMP (e quindi in C) sono copiati in D.

NOTA 1: con clock a 2 MHz, l'istruzione è stata eseguita in 5 impulsi di clock (5 stati interni); pertanto il tempo di esecuzione dell'istruzione è :

$$Tesec. = 5x500 \text{ nsec.} = 2.5 \ \mu sec$$



NOTA 2: non è stato possibile trasferire direttamente da C→ D il dato (in un solo stato interno T4) perché sia D che C appartengono alla stessa RAM INTERNA. Pertanto può essere indirizzato solo un registro di tipo generale per volta.

NOTA 3: nel caso che uno dei due codici Sorgente (SSS) o Destinazione (DDD) fosse stato

110 → MEMORIA

ovvero, il dato doveva essere copiato da un registro interno in una cella di memoria o viceversa. Sarebbe stato allora necessario conoscere l'indirizzo della cella dove riporre o prendere il dato.

Tale indirizzo doveva trovarsi nel Registro Doppio HL (formato dai due registri H e L di 8 bit ciascuno), ovvero:

H: BYTE ALTO INDIRIZZO / L: BYTE BASSO INDIRIZZO

In questo caso la sequenza delle microistruzioni è sempre la solita fino allo stato interno T4

P.e. TRASFERIMENTO da D → MEM

T5: non utilizzato

CICLO MACCHINA M2

STATO INTERNO T1

H → MA(H) / L → MA(L)

Sul Bus indirizzi è presente l'indirizzo della cella dove riporre il dato (p.e. 10).

STATI INTERNI T2 e T3

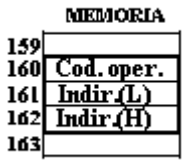
T2 e T3: TMP → MEM.

Avviene quindi l'abilitazione del TMP e riversamento del suo contenuto sul BUS DATI. Vengono poi inviati i comandi di scrittura in memoria.

Totale N. 7. STATI INTERNI (in 2 cicli macchina) : M1(T1,T2,T3,T4) + M2(T1,T2,T3). Si hanno quindi 7 impulsi di clock (Clock 2 Mhz → 3.5 µsec.)

ESECUZIONE DI UNA ISTRUZIONE DI 3 BYTE

(Trasferimento di un dato a 8 bit da una cella di memoria all'accumulatore del µP)



Il codice operativo a 8 bit dell'istruzione è (per l'INTEL 8080) 00111010 (LDA addr: 3Ah). L'istruzione prevede il caricamento di un dato contenuto nella cella di memoria indirizzata (addr), nell'accumulatore. Per fare questo l'istruzione deve contenere, oltre al codice operativo (1 byte) anche l'indirizzo della cella contenente il dato da trasferire (16 bit). L'indirizzo è contenuto nelle due celle consecutive a quella contenente il codice operativo e l'istruzione pertanto occupa 3 byte.

Lo stato di partenza è definito dall'indirizzo della cella che ospita il codice operativo dell'istruzione (primo byte dell'istruzione). Tale indirizzo è depositato nel PC.

CICLO MACCHINA M1

STATO INTERNO T1, T2 e T3

Questi primi tre stati interni sono identici per tutte le istruzioni (vedi istruzione ad un byte spiegata precedentemente). PC → MA /ABIL. MEM., PC=PC+1 e C.Op.→IR.

STATO INTERNO T4: usato per scopi interni (decodifica)

CICLO MACCHINA M2

L'istruzione è stata adesso decodificata, quindi è noto il processo da seguire. Tale processo consiste nell'andare a prendere i due byte successivi dell'istruzione. I due byte formano l'indirizzo (16 bit) della cella dove si trova il dato da recuperare. Questo ciclo macchina provvede a trasferire il primo byte dell'indirizzo (parte bassa B2) nel registro Z.

STATO INTERNO T1, T2 (PC → MA /ABIL. MEM., PC=PC+1)

Questi stati interni servono ad indirizzare la cella del primo byte da recuperare e incrementare il PC

STATO INTERNO T3 (B2→Z)

Questo stato interno serve per trasferire il contenuto depositato sul bus dati nel registro Z.

CICLO MACCHINA M3

Questo ciclo macchina provvede a completare il trasferimento dell'indirizzo. È quindi simile al precedente; trasferisce il secondo byte dell'indirizzo (parte alta B3) nel registro W. Alla fine l'indirizzo si trova nel registro doppio WZ.

STATO INTERNO T1, T2 (PC → MA /ABIL. MEM., PC=PC+1)

Questi stati interni indirizzano la cella del secondo byte da recuperare e incrementare il PC

STATO INTERNO T3 (B3→W)

Questo stato interno serve per trasferire il contenuto depositato sul bus dati nel registro W.

CICLO MACCHINA M4

Questo ciclo macchina effettua il trasferimento del dato indirizzato (l'indirizzo è nel registro WZ) nell'accumulatore.

STATO INTERNO T1 (WZ → MA /ABIL. MEM.)

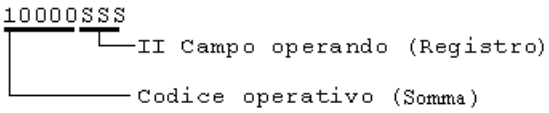
S'indirizza la cella interessata.

STATO INTERNO T2 e T3 (DATA BUS→ACC)

Il dato impostato sul bus dati viene depositato nell'accumulatore

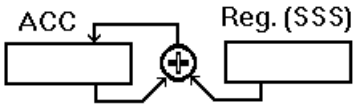
NOTA: sono stati necessari 4 cicli macchina e 13 stati interni, ovvero 13 impulsi di clock. Pertanto, supponendo che la freq. del clock sia di 2MHz, l'istruzione è durata (13 x 500 nsec) 6,5 µsec.

OVERLAP (PIPELINE)



Consideriamo l'istruzione 1000 0SSS ("ADD r" istruzione ad 1 Byte). Essa somma il contenuto del registro sorgente, identificato dai bit SSS (p.e. 010: registro "r" = D), al contenuto del registro accumulatore e deposita il risultato nell'accumulatore stesso.

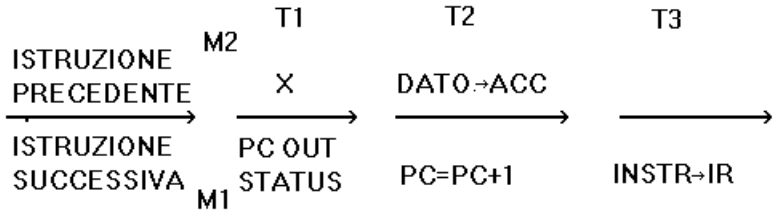
Gli stati interni T1, T2, T3 (M1) rimangono invariati. Nello stato interno T4 il contenuto del registro SSS (p.e. Reg. D) viene copiato nel registro tampone TMP (D→TMP) e contemporaneamente il contenuto dell'accumulatore viene depositato nell'altro registro ACT (i due registri tampone si trovano davanti agli ingressi dell'ALU). Immediatamente l'ALU forma il risultato della somma che è disponibile all'uscita di essa.



N.B. T5(M1) non è utilizzato.

CICLO MACCHINA M2

Lo stato interno T1 viene utilizzato per la fase di FETCH dell'istruzione successiva (il bus dati multiplexato è utilizzato per il trasporto dei comandi ABIL. MEM.). Nello stato interno T2 avviene il trasferimento del risultato della somma, già presente in uscita dell'ALU, nell'accumulatore.



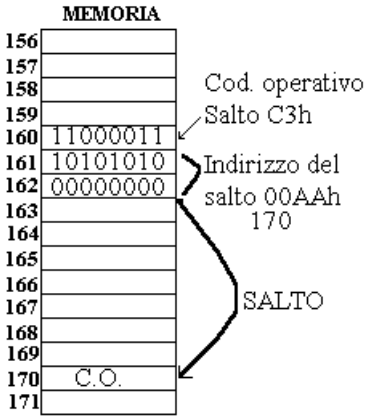
NOTA: il trasferimento del risultato nell'accumulatore viene ritardato al secondo stato interno per poterlo sovrapporre all'incremento del PC dell'istruzione successiva. Tale sovrapposizione (OVERLAP o PIPELINE) è possibile in quanto l'incremento del PC non richiede l'utilizzo del bus. In questo modo si risparmia uno stato interno eseguendo in parallelo parte di due istruzioni consecutive. L'istruzione che doveva durare 5 stati interni ne dura in effetti solo 4.

- ISTRUZIONE DI SALTO INCONDIZIONATO (JMP ADDR)

Questa istruzione permette di variare il normale flusso di esecuzione di un programma effettuando un SALTO senza condizioni ad una cella qualsiasi della memoria. È una istruzione a 3 Byte: il primo è il codice operativo del salto (11000011b , C3h), mentre i due byte successivi formano l'indirizzo (16 bit) della cella alla quale saltare (prima il byte basso B2 e poi quello alto B3). Tale operazione consiste mettere l'indirizzo al quale saltare nel PC. L'istruzione è composta da 3 cicli macchina, ognuno corrispondente al recupero di uno dei byte che compone l'istruzione.

CICLO MACCHINA "M1"

STATO INTERNO T1,T2,T3, T4 recupero del codice operativo del salto e decodifica



CICLO MACCHINA "M2"

STATO INTERNO T1,T2,T3: recupero del byte (B2→Z) corrispondente alla parte bassa dell'indirizzo che viene riposto nel registro Z.

CICLO MACCHINA "M3"

STATO INTERNO T1,T2,T3: recupero del byte (B3→W) corrispondente alla parte alta dell'indirizzo che viene riposto nel registro W.

A questo punto l'indirizzo da riporre nel PC è nel registro WZ. In effetti, il PC non viene interessato in questa fase e l'istruzione è finita. Infatti per effettuare il salto è sufficiente iniziare l'istruzione successiva ponendo sul bus indirizzi non il contenuto del PC, bensì quello del registro WZ. Tale operazione è svolta dal primo stato interno del primo ciclo macchina dell'istruzione successiva:

CICLO MACCHINA "M1" (ISTRUZIONE SUCCESSIVA)

STATO INTERNO T1 (WZ → MA /ABIL. MEM.)

STATO INTERNO T2 (PC=WZ + 1)
questo secondo stato interno serve per aggiornare il PC in modo da ripartire al prossimo ciclo macchina correttamente. Segue l'iter ordinario per l'istruzione successiva:

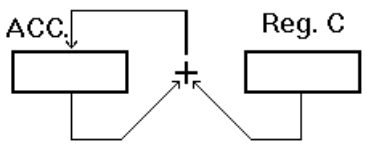
STATO INTERNO T3 (C.Op.→TMP/IR)

- TECNICHE DI INDIRIZZAMENTO

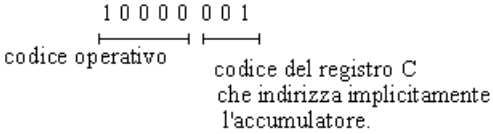
La tecnica di indirizzamento di una istruzione è il modo con il quale viene specificato il recupero dell'operando in esso contenuta.

Nota: spesso i microprocessori non hanno tutte le possibilità di indirizzamento, ma solo parte di esse.

- INDIRIZZAMENTO IMPLICITO



Il codice del registro da indirizzare (p.e. accumulatore) è implicito nel codice operativo.
p.e. ADD C (Somma il contenuto dell'ACC con quello del registro C e deposita il risultato nell'accumulatore. Il codice macchina si presenta come in figura.

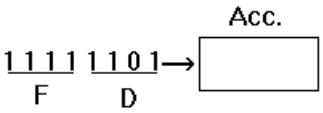


Permette lo svolgimento più veloce delle istruzioni e viene utilizzato generalmente per trasferimento dati fra registri interni al µP in modo da poter sfruttare il fatto che il codice macchina può essere lungo solo 1 byte.

- INDIRIZZAMENTO IMMEDIATO

L'operando (dato) segue immediatamente il codice operativo.

P.e. LD A,FDh carica nell'accumulatore il valore binario corrispondente al numero esadecimale FDh.

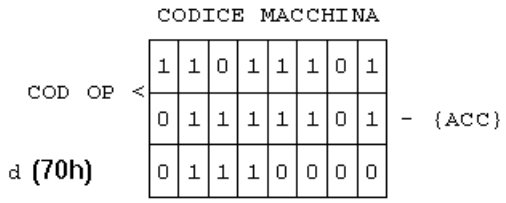


Spesso è usato il simbolo assembler "#" d'indirizzamento immediato.

L'istruzione può essere lunga 2 o 3 byte in dipendenza dal fatto che il dato immediato sia un valore di 8 bit o un indirizzo a 16 bit.

Nota: nel codice macchina l'indirizzamento dipende dalla

Cod. Oper. (8bit) Dato (8bit)
0 0 1 1 1 1 1 0 + 1 1 1 1 1 1 0 1 (2 byte)



scelta di alcuni bit del codice operativo.

- *INDIRIZZAMENTO DIRETTO*

Il codice operativo (su 8 bit) è seguito da un indirizzo a 8 bit pertanto l'istruzione è a 2 byte. Si dice per questo che il modo di indirizzamento è in pagina 0, cioè nella prima parte della memoria. Il dato si trova nelle prime 255 celle di memoria. P.e. LDA Temp (Etichetta) dove alla etichetta "Temp" è associato tramite la tabella dei simboli uno dei primi 255 indirizzi in modo da poterlo contenere in 8 bit. L'istruzione carica nell'accumulatore il contenuto della cella con indirizzo associato a "Temp".

Codice macchina

Cod. Oper. Ind. 8 bit
0 0 1 X X X 1 0 + 0 0 0 1 0 0 0 1

XXX=010=> ind. diretto

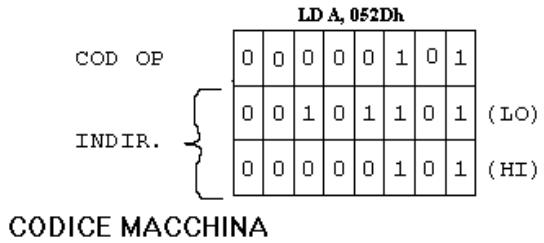
Nel MP 8080 le istruzioni che utilizzano una tecnica di indirizzamento indiretto sono quelle che accedono alle porte di I/O ovvero IN e OUT.

Questo tipo di indirizzamento permette di utilizzare istruzioni di 2 byte al max e quindi accelerare la velocità di esecuzione.

- *INDIRIZZAMENTO ESTESO O NORMALE*

È il più comune tipo di indirizzamento, in quanto è il normale tipo di accesso in memoria. L'istruzione è di tre byte (1 byte di cod. Operativo e 2 di indirizzo).

P.e. LD A, 052Dh oppure LD A,LABEL pone nell'accumulatore il contenuto della cella di indirizzo specificato o associato alla etichetta "LABEL" qualunque esso sia.

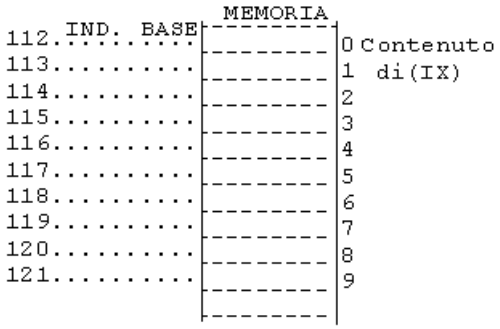


CODICE MACCHINA

(LABEL: 0000 0101 0010 1101 = 052Dh.)

- *INDIRIZZAMENTO INDICIZZATO*

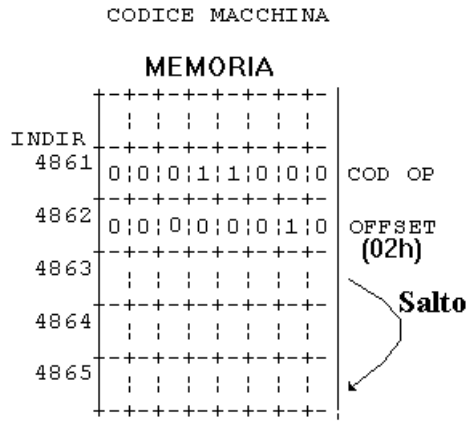
Una istruzione con indirizzamento indicizzato contiene un campo di spostamento che verrà sommato automaticamente al contenuto del Registro indice (IX); p.e. LD A,(IX+d) carica nell'accumulatore il contenuto della cella di memoria di indirizzo IND = Contenuto REG IX+valore d (spiazzamento). Può essere a 2 o 3 byte a seconda della lunghezza dell'indirizzo associato all'etichetta.
Questo tipo di indirizzamento viene utilizzato generalmente per la



gestione di tabelle in memoria. La tabella inizia all'indirizzo base ed è consultata variando il contenuto del registro indice o viceversa.

- ***INDIRIZZAMENTO RELATIVO***

Viene utilizzato per fare dei salti brevi ad istruzioni che non distano (in avanti o indietro) più di 127 locazioni di memoria.



P.e. JR LABEL oppure JR 02H

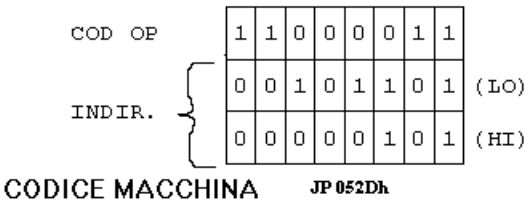
Salta all'indirizzo associato all'etichetta "LABEL", PC+e-2→PC e=spiazzamento fra la cella che ospita il codice operativo e quella alla quale si salta. L'etichetta "LABEL" deve essere associata ad un indirizzo che differisce da quello della cella che ospita l'istruzione meno di 127 locazioni di memoria. L'assemblatore conta il numero di celle di memoria che intercorre e scrive nel byte successivo a quello del codice operativo JR il valore del salto.

Dato che al momento del salto il PC è già stato incrementato due volte allora il valore effettivo dello spiazzamento è "e-2".

P.e. LABEL=4865 4865-4861+(-2)=2

NOTA: l'istruzione aggiunge al PC il valore 2 (02h). L'operazione che viene quindi svolta è di incrementare il PC di 2 (PC=PC+02h). Dato però che il PC punta già all'istruzione successiva, ovvero 2 celle più avanti, il salto effettivo è di 4 celle rispetto alla posizione del codice operativo (e quindi all'indirizzo 4865). Se l'offset è negativo (p.e. -3 ovvero FDh) si salta all'indietro.

- ***INDIRIZZAMENTO ASSOLUTO***



È l'indirizzamento specifico per i salti assoluti e normalmente di 3 byte (1 byte di cod. Operativo e 2 di indirizzo)

P.e. JP 052Dh oppure JP LABEL: salto alla cella di indirizzo specificato o associato all'etichetta "LABEL" qualunque esso sia.

LABEL: 0000 0101 0010 1101 = 052Dh. Salta alla cella d'indirizzo 052Dh.

- ***INDIRIZZAMENTO INDIRETTO***

Il dato da recuperare si trova all'indirizzo specificato in una cella di memoria.

COD OP [indir 1]→indir 1 [indir 2]→indir 2 [dato]

Le istruzioni che utilizzano questo tipo di indirizzamento sono a 3 byte. Questo tipo di indirizzamento viene utilizzato soprattutto per il passaggio dei parametri nei sottoprogrammi (equivale a nascondere la chiave della porta sotto lo zerbino: è nota la posizione della chiave, ma non la chiave stessa).

1 Dire qual è la differenza fra logica CABLATA e PROGRAMMATA. Definire pregi e difetti. Definire il concetto di "PROGRAMMA" e "ISTRUZIONE".

MP8

2 Dire quali sono le componenti di un elaboratore digitale e le loro funzioni.

MP8

3 Tracciare lo schema di un sistema a microprocessore e definire la finzione di ogni parte.

MP8

4 Definire le caratteristiche e le funzioni di ogni tipo di BUS in un sistema a MP tenendo conto anche del numero di linee.

MP8

5 Dire quali sono le parti che formano un microprocessore e le loro funzioni.

MP8

6 Descrivere il funzionamento dell'ALU in un MP

MP8

7 Descrivere quali sono i registri interni di uso ordinario che si trovano nel MP e le loro funzioni.

MP8

8 Dire quali registri SPECIALI che sono utilizzati per il funzionamento del MP.

MP8

9 Definire la funzione del PC (program Counter) e le relative caratteristiche.

MP8

10 Descrivere le funzioni del REGISTRO DI STATO in un MP.

MP8

11 Definire la funzione dello STACK nel MP e come vien utilizzato.

MP8

12 Descrivere il funzionamento della LOGICA DI CONTROLLO in un MP e la relativa funzione del CLOCK.

MP8

13 Dire come avviene l'esecuzione di una istruzione da parte della logica di controllo.

MP8

14 Spiegare che cosa è il CODICE OPERATIVO e il suo scopo.

MP8

15 Spiegare l'evoluzione dell'istruzione (repertorio Intel 8080): **ACI 6Ah**

MP8

16 Spiegare l'evoluzione dell'istruzione (repertorio Intel 8080): **ADD C**

MP8

17 Spiegare l'evoluzione dell'istruzione (repertorio Intel 8080): **JMP 23ACh**

MP8

18 Spiegare l'evoluzione dell'istruzione (repertorio Intel 8080): **MOV A,C**

MP8

19 Spiegare l'evoluzione dell'istruzione (repertorio Intel 8080): **MOV A, M**

MP8

20 Spiegare l'evoluzione dell'istruzione (repertorio Intel 8080): **STA BB89h**

MP8

21 Spiegare l'evoluzione dell'istruzione (repertorio Intel 8080): **LDA DABAh**

MP8