

ESERCITAZIONE 9

Sommario

- Operandi
- Registri dedicati
- Gestione di sottoprogrammi
- Meccanismi di interruzione

1. Operandi

Le prestazioni di una CPU possono variare considerevolmente a seconda del numero di operandi di cui dispone la singola istruzione macchina. Per analizzare i possibili diversi comportamenti di architetture aventi diverso numero di operandi si mostra un esempio di soluzione di una particolare operazione.

Si supponga ad esempio di voler svolgere la seguente operazione:

$$X = A*B + C*C$$

1.1. Architettura ad 1 operando

LOAD	A	;	$AC \leftarrow M(A)$
MUL	B	;	$AC \leftarrow AC * M(B)$
STORE	T	;	$M(T) \leftarrow AC$
LOAD	C	;	$AC \leftarrow M(C)$
MUL	C	;	$AC \leftarrow AC * M(C)$
ADD	T	;	$AC \leftarrow AC + M(T)$
STORE	X	;	$M(X) \leftarrow AC$

1.2. Architettura a 2 operandi

MOV	T, A	;	$M(T) \leftarrow M(A)$
MUL	T, B	;	$M(T) \leftarrow M(T) * M(B)$
MOV	X, C	;	$M(X) \leftarrow M(C)$
MUL	X, C	;	$M(X) \leftarrow M(X) * M(C)$
ADD	X, T	;	$M(X) \leftarrow M(X) + M(T)$

1.3. Architettura a 3 operandi

MUL	T, A, B;	$M(T) \leftarrow M(A) * M(B)$
MUL	X, C, C;	$M(X) \leftarrow M(C) * M(C)$
ADD	X, X, T;	$M(X) \leftarrow M(X) + M(T)$

1.4. Pregi e difetti legati al numero di operandi utilizzati

Architetture che utilizzano 1 solo operando permettono istruzioni più semplici, il processore esegue velocemente le operazioni più frequentemente utilizzate. Minore è il numero di operandi, più corte sono le istruzioni. Limitando il numero di operandi si limitano i tipi di operazioni che possono essere effettuate.

Architetture a più operandi permettono di effettuare operazioni complesse, ma richiedono un alto costo in termini di unità di controllo e di numero di byte per istruzione.

2. Funzioni specializzate

La gestione efficace di operazioni particolari svolte frequentemente richiede la presenza di funzioni specializzate:

- registri dedicati;
- gestioni sottoprogramma;
- interruzione.

2.1. Registri dedicati

2.1.1. Registro base e registro indice

Estendendo il concetto di indirizzo relativo si utilizzano 2 registri:

- registro base
- registro indice.

Il registro base contiene l'indirizzo di inizio di un'area di memoria (tabella o vettore), il registro indice contiene la posizione all'interno del blocco di memoria.

Attraverso tale meccanismo è possibile per il processore rilocalizzare gli accessi ad una zona di memoria cambiando solamente il contenuto del registro base.

2.1.2. Registro di stato

Il registro di stato memorizza lo stato del processore. Particolari bit assumono valore 0 oppure 1 in base al risultato delle operazioni svolte dall'unità centrale.

Nel processore Intel 8086 il *registro di stato*, detto anche *Processor Status Word (PSW)*, è un registro di 16 bit contenente 9 flag, suddivisi in *flag di stato* e *flag di controllo*.

I flag di stato indicano le condizioni prodotte come risultato dell'esecuzione di istruzioni aritmetiche o logiche. Essi sono:

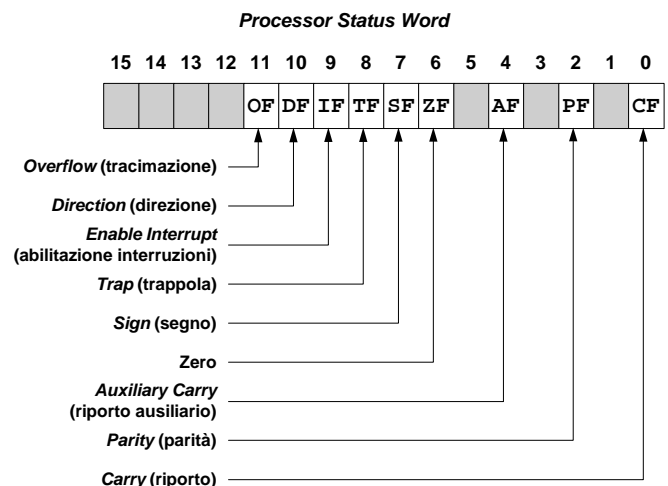
1. flag di carry (**CF**): viene forzato a 1 se un'operazione genera un riporto (*carry*) od un prestito (*borrow*) dall'operando destinazione;
2. flag di parità (**PF**): viene forzato a 1 se è pari il numero di bit a 1 nel byte meno significativo del risultato;

3. flag di carry ausiliario (**AF**): viene forzato a 1 se un'operazione genera un carry od un borrow dal bit 3; è usato nelle operazioni aritmetiche tra numeri rappresentati secondo la notazione BCD (si veda Cap. 10);
4. flag di zero (**ZF**): viene forzato a 1 se il risultato dell'operazione è 0;
5. flag di segno (**SF**): coincide con il bit più significativo del risultato di un'operazione;
6. flag di overflow (**OF**): viene forzato ad 1 se nell'esecuzione di un'operazione si è verificata una condizione di tracimazione (*overflow*).

In seguito all'esecuzione di un'istruzione, il valore di ciascun flag di stato può risultare o modificato o inalterato o indefinito.

Mentre i flag di stato danno informazioni circa le istruzioni che sono state appena eseguite, i flag di controllo modificano il comportamento delle istruzioni che verranno eseguite in seguito. I flag di controllo dell'8086 sono:

1. flag di direzione (**DF**): il valore del flag determina la direzione con cui sono eseguite le operazioni su stringhe; se è forzato ad 1 le istruzioni per la manipolazione delle stringhe decrementano l'indirizzo che punta agli operandi e in questo modo l'elaborazione delle stringhe procede da indirizzi alti verso indirizzi bassi; se è forzato a 0 le stesse istruzioni eseguono l'incremento dell'indirizzo, e l'elaborazione procede verso indirizzi crescenti;
2. flag di abilitazione all'interruzione (**IF**): gli interrupt mascherabili vengono riconosciuti e serviti se e solo se questo flag è forzato ad 1; viene azzerato per disabilitare gli interrupt mascherabili;
3. flag di trap (**TF**): se è forzato ad 1 il processore opera in modo *single-step*, generando una *trap* (ossia la chiamata ad un'apposita procedura di interruzione) al termine di ogni istruzione; questo tipo di operazione è utile per eseguire il *debug* del codice.



2.1.3. Stack e registro Stack Pointer

Lo *stack* è una struttura che permette di memorizzare i dati e di richiamarli secondo una strategia *Last-In-First-Out (LIFO)*.

All'inizio di un programma lo stack non contiene dati; le operazioni di caricamento e di prelievo dei dati dallo stack sono chiamate rispettivamente operazioni di *push* e di *pop*.

La locazione di memoria nella quale è contenuto l'ultimo dato inserito è detta *cima dello stack* (*top of the stack*); la locazione di memoria che contiene il primo dato inserito è detta *fondo dello stack* (*bottom of the stack*).

Normalmente, lo stack viene utilizzato nell'ambito del meccanismo di chiamata a procedura (per memorizzarne l'indirizzo di ritorno, per il salvataggio dei registri, per l'eventuale passaggio di parametri) e per il salvataggio di variabili temporanee.

Il registro Stack Pointer (SP) punta alla cima dello stack (ultima parola scritta nello stack).

Lo stack cresce da locazioni di memoria con indirizzo maggiore verso quelle con indirizzo minore. Ogni operazione di *push* decrementa di 2 unità il contenuto di SP e trasferisce una word nella locazione puntata da SP; ogni operazione di *pop* estrae una word dalla locazione puntata da SP ed incrementa di 2 unità il contenuto di SP.

Le operazioni di push e di pop in alcuni processori sono svolte da particolari istruzioni ed in altri sono svolti da particolari modi di indirizzamento.

Nel processore Intel 8086 le operazioni di push e di pop sono effettuate attraverso le seguenti istruzioni:

- PUSH BX ; decrementa SP e copia nello stack il contenuto del registro BX
- POP AX ; copia la locazione puntata da SP nel registro AX ed incrementa SP.

Nel processore Motorola 68000 le operazioni di push e pop sono effettuate attraverso particolari modi di indirizzamento:

- MOVE.W D0, -(SP) ; predecrementa SP e copia nello stack il contenuto del registro D0
- MOVE.W (SP)+, D0 ; copia la locazione puntata da SP nel registro D0 e post-incrementa SP.

2.2. Gestione dei sottoprogrammi

Una *procedura*, detta anche *subroutine* o *sottoprogramma*, è una parte di programma costituita da un gruppo di istruzioni che eseguono un compito specifico; ogni procedura, memorizzata in memoria una volta sola, può essere eseguita un numero qualsiasi di volte.

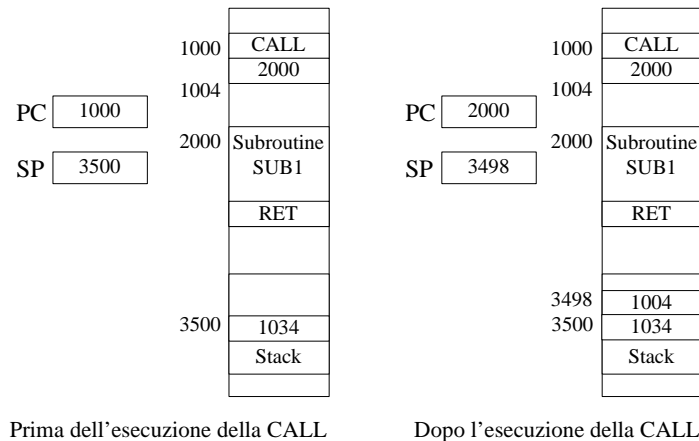
L'utilizzo delle procedure permette di risparmiare spazio in memoria e di rendere più modulare lo sviluppo di programmi. Il principale svantaggio risiede nel tempo di elaborazione necessario per le operazioni di chiamata e di ritorno tra la procedura chiamante e quella chiamata.

Esempio:

```
...
CALL SUB1
NEXT:  ...
...
SUB1:  ...
...
RET
```

Con l'esecuzione dell'istruzione CALL viene caricato nello stack l'indirizzo di ritorno corrispondente all'indirizzo dell'istruzione successiva e viene caricato nel *program counter* l'indirizzo di inizio della procedura.

Con l'esecuzione dell'istruzione RET viene eseguito il pop dallo stack dell'indirizzo di ritorno e tale valore viene caricato nel *program counter*.



2.2.1. Annidamento dei sottoprogrammi

Utilizzando il meccanismo di salvataggio nello stack dell'indirizzo di ritorno è possibile gestire chiamate annidate di sottoprogramma, ossia una chiamata di un sottoprogramma all'interno di un sottoprogramma.

2.3. Interruzione

L'interruzione è un evento asincrono rispetto al normale flusso di esecuzione del programma. Evento infrequente ed eccezionale che si verifica nel corso delle attività svolte da un elaboratore.

Un'interruzione può essere generata internamente oppure esternamente al processore; causa il trasferimento del controllo del programma corrente ad un programma specifico di servizio dell'evento.

Lo svolgimento del normale flusso di programma deve essere trasparente rispetto al servizio di un'interruzione: il programma deve riprendere nello stato in cui si trovava nel momento dell'interruzione.

2.3.1. Interruzioni interne

La CPU interrompe il proprio normale flusso di operazione nel caso in cui avvengano determinati eventi interni:

1. eventi catastrofici (caduta di tensione);
2. malfunzionamenti hardware (errori di trasferimento dei dati sul bus);
3. errori aritmetici (divisione per zero/overflow);
4. interruzioni software (*trap*).

2.3.2. Gestione dei dispositivi periferici

Esistono due tecniche per la gestione dei dispositivi periferici:

- interrogazione (*polling*);
- interruzione (*interrupt*).

Nel caso di gestione in *polling* il processore interroga ciclicamente i vari dispositivi e legge lo stato di ciascun dispositivo di I/O, nel caso in cui il dispositivo sia pronto a ricevere (o ad inviare) un dato, il processore esegue la procedura di lettura (o scrittura) del dispositivo. La CPU deve dunque verificare con continuità lo stato dei dispositivi di I/O. Tale meccanismo di interruzione è estremamente semplice dal punto di vista software ed hardware (non richiede logica aggiuntiva), ma causa un notevole dispendio di risorse di calcolo in quanto la CPU è principalmente dedicata ad interrogare i vari dispositivi periferici.

Nel caso di gestione in interrupt la CPU esegue il normale flusso di programma ignorando i dispositivi periferici e viene interrotta da eventi esterni attraverso un segnale proveniente dall'esterno (o da un segnale interno nel caso di interruzioni interne). L'elemento che genera l'interruzione ha il compito di segnalare la richiesta di servizio e si preoccupa di segnalare alla CPU la propria identità in modo da permettere al processore di saltare all'opportuna procedura di servizio dell'interruzione.

Il segnale di interruzione viene testato dalla CPU alla fine di ogni ciclo di istruzione. Se esiste una richiesta di interruzione la CPU deve interrompere il normale flusso di programma per eseguire la procedura di servizio dell'interruzione. Per permettere il ripristino dell'esecuzione del programma interrotto occorre salvare il *contesto* del programma in corso di esecuzione. Il salvataggio del contesto corrisponde al salvataggio nello stack del registro di stato e dell'indirizzo della prossima istruzione in sequenza.

2.4. Flusso di servizio di un'interruzione

1. La CPU esegue un'istruzione;
2. la CPU al termine dell'istruzione corrente verifica se vi è una richiesta di interruzione (interna od esterna);
3. se non vi è richiesta di interruzione il flusso torna al punto 2;
4. salvataggio del contesto nello stack;
5. copia in PC dell'indirizzo di inizio della procedura di gestione dell'interruzione;
6. esecuzione delle procedura di gestione dell'interruzione;
7. terminazione delle procedura di gestione dell'interruzione;
8. ritorno al programma interrotto attraverso il ripristino del contesto: prelevamento dallo stack del registro di stato e caricamento di PC con l'indirizzo della prossima istruzione da eseguire;
9. ritorno al punto 1 del flusso.

2.5. Interruzioni annidate

È possibile che una procedura di gestione di un'evento di interruzione sia a sua volta interrotto da un evento di interruzione.

La corretta gestione delle varie procedure è garantita dal meccanismo di salvataggio degli specifici contesti nello stack.

Per impedire che eventi molto critici siano interrotti da eventi di importanza inferiore si associa ad ogni evento un livello di priorità. Una procedura di gestione delle interruzioni può essere interrotta solo da eventi aventi priorità maggiore.

[Torna al Sommario](#)