CLASSE 5/RET/JMP/CALL

martedì 22 novembre 2022 15:06

CONTROLLO DEL FLUSSO DEL PROGRAMMA

IL MIO PROGRAMMA PASSA da UN' ISTRUZIONE alla SUCCESSIVA IN MANIERA CONTIGUA E lineare. IN UN COSTRUTTO COME IF, ELSE ABBIAMO LA NECESSITÀ DI SALTARE DA UN'ISTRUZIONE ALL'ALTRA, AL VERIFICARSI DI UNA DETERMINATA CONDIZIONE.

OPPURE abbiamo NECESSITÀ di implementare dei cicli, e i cicli RIPETONO PER PIU VOLTE la STESSA ISTRUZIONE.

Quindi voglio poter alterare IL flusso di esecuzione!

Devo anche CONSENTIRE al PROCESSORE di INVOCARE una determinata funzione;

JMP E CALL PERMETTONO DI MODIFICARE IL VALORE DELL' IP

JMP: Salto incondizionato, ossia quando IL PROCESSORE INTERPRETA ed esegue questa ISTRUZIONE, automaticamente SOVRASCRIVERÀ IL CONTENUTO dell' IP! Mi sposto verso una specifica locazione di memoria, in cui c'è la mia ISTRUZIONE che è IL BERSAGLIO di questa ISTRUZIONE di salto;

Esistono delle varianti di questa ISTRUZIONE:

1. L'operando è di TIPO M, in questo caso parliamo di **salto relativo**!
   M è uno spiazzamento a valle della fase di fetch.

   > ● M, l'operando è una locazione di memoria, codificata come uno spiazzamento a partire dal contenuto del REGISTRO RIP dopo l'esecuzione della fase di fetch

   È un offset che devo sommare al contenuto dell' IP dopo la fase di fetch, per raggiungere la mia ISTRUZIONE;

2. L'operando è di TIPO *G, in questo caso parliamo di **salto assoluto**!
   L'operando è un REGISTRO di uso generale.
   Carichiamo all'INTERNO di un REGISTRO un INDIRIZZO a 64 BIT, dopodiché chiediamo al PROCESSORE di saltare all' INDIRIZZO CONTENUTO NEL REGISTRO.
   Così esplicitiamo in maniera completa (E SENZA SPIAZZAMENTO) IL BERSAGLIO di UN salto.

Questi salti, ALTERANO IL FLUSSO DI CONTROLLO SENZA DOVER TORNARE INDIETRO;

MA QUANDO IMPLEMENTIAMO UNA FUNZIONE, IO CHIAMO UNA FUNZIONE, E SUCCESSIVAMENTE MI DEVE RESTITUIRE IL CONTROLLO AL CHIAMANTE!

Per questo utilizzo una funzione che si chiama "call", ossia chiamo una funzione;

CALL: gli operandi sono gli stessi. O esprimo l'indirizzo iniziale della mia funzione come spiazzamento relativo rispetto al valore dell'IP (call M), oppure devo esplicitare l'indirizzo iniziale della mia funzione all'interno di un registro e chiamerò la funzione il cui indirizzo è contenuto nel registro (call *G).

Tipo	Mnemonico	Operandi	O S Z P C	Descrizione
0	jmp	M	- - - - -	Esegue un salto relativo
1	jmp	*G	- - - - -	Esegui un salto assoluto
2	call	M	- - - - -	Esegue una chiamata a subroutine relativa
3	call	*G	- - - - -	Esegue una chiamata a subroutine assoluta
4	ret	–	- - - - -	Ritorna da una subroutine
5	iret	–	⇕ ⇕ ⇕ ⇕ ⇕	Ritorna dal gestore di una interruzione

DA qualche parte io devo poter memorizzare chi è il chiamante della mia funzione!

E QUANDO LA FUNZIONE TERMINA DEVO RESTITUIRE IL CONTROLLO AL CHIAMANTE!

Questa informazione di collegamento viene memorizzata sullo stack (da dove arrivo).

La chiamata, e di conseguenza l'attivazione di una funzione, scriverà sullo stack l'indirizzo del chiamante, per permettere al processore di restituire il controllo nel punto del codice di dove è stata effettuata la chiamata.

Per restituire il controllo al chiamante, devo leggere l'indirizzo dell'istruzione che ha causato l'invocazione, dallo stack, e per fare questo devo necessariamente utilizzare un'istruzione assembly dedicata!

RET: Permette di restituire il controllo al chiamante.

Se voglio attivare una funzione utilizzerò una CALL per dare il controllo alla funzione chiamata.

Il mio processore memorizzerà sullo stack l'informazione che mi permette di dire chi è il chiamante di questa funzione.

L'istruzione RET all'interno della mia funzione restituirà il controllo al chiamante.

IRET: È equivalente alla RET ma chiude l'esecuzione di un driver;