

PIÙ USO MICROCODICE

giovedì 24 novembre 2022 21:41

Ci sono alcuni pezzi dei miei microprogrammi che si ripetono;

Se io dovesse scrivere tanti microprogrammi in cui ci sono delle parti che si ripetono, la mia ROM sarebbe molto grande, perché le stesse microoperazioni vengono copiate ed incollate in punti differenti.

È come quando copiamo e incolliamo nel codice il contenuto della funzione, piuttosto che usarla: la quantità di codice crece.

Quello è un problema perché ho una ROM molto grande e ho tanti transistor.

Vediamo come si può organizzare il microcodice per riutilizzarlo in più parti nei microprogrammi.

IL NOSTRO formato istruzione rende parametrica l'esecuzione del microprogramma.

LA NOSTRA CU Leggendo dall'IR può capire se l'istruzione utilizza un REGISTRO BASE, indice, se c'è uno spiazzamento, etc...

Può capire quali campi sono presenti;

Significa che noi possiamo realizzare un microprogramma parametrico che fa sì che tutte le istruzioni in prima istanza che accedono in memoria, indipendentemente da quali operandi in memoria stiamo utilizzando, possono far girare lo stesso microprogramma.

Questo sottoprogramma deve essere "chiamabile", noi dobbiamo permettere al (ad esempio) microprogramma dell'istruzione MOV, di saltare al microprogramma de calcolare il mio indirizzamento;

IL MICROPROGRAMMA CHE POSSIAMO ANDARE A REALIZZARE È UN MICROPROGRAMMA PARAMETRICO!

↳ modifichiamo la nostra CU per far sì che possa effettuare internamente ad un microprogramma dei salti!

LA ROM È IN grado di sovrascrivere l'offset.

Se una qualche variabile di condizione è vera/falsa, il nostro microprogramma può decidere se saltare o meno da una parte all'altra del mio microcodice;

Possiamo implementare dei salti all'interno del "primo", come se fosse del codice: verifichiamo il valore di una variabile di condizione, saltiamo da un'altra parte o andiamo all'istruzione successiva;

A finezione possiamo realizzare un sotto programma di questo tipo:

Cerchiamo di verificare quali variabili di condizione provengono dall'IR valgono 1, per andare a scrivere un mio microprogramma che calcoli un indirizzo di memoria;

Il campo D di IR mi dice se sto utilizzando uno spiazzamento, e mi interessa.

```

1 if D == 1 and Bp == 0 and Ip == 0
2   MAR ← IR[0:31]
3 else if D == 0 and Bp == 1 and Ip == 0
4   MAR ← B
5 else if Ip == 1
6   TEMP2 ← I
7   MAR ← SHIFTER_OUT[SHL, T]
8   TEMP1 ← MAR
9   if D == 1
10    TEMP2 ← IR[0:31]
11    MAR ← ALU_OUT[ADD]
12    TEMP1 ← MAR
13  endif
14  if Bp == 1
15    TEMP2 ← B
16    MAR ← ALU_OUT[ADD]
17  endif
18 endif

```

IL BIT D È UN FLAG CHE MI PERMETTE DI SPECIFICARE SE ESISTE UNO SPIAZZAMENTO; B_p mi dice se è presente un registro di base o se ho usato un registro indice I_p; al seconda delle variabili di condizione che entrambe mille ROM, noi possiamo effettuare dei salti all'interno del firmware!

Se ho uno spiazzamento, non ho né Reg. Base e né indice, la mia istruzione sarà del tipo

→ Lo spiazzamento si trova nei 32 bit meno significativi dell'IR;

MOV %eax, disp

qui ciò significa che posso andare nel registro MAR e prendere i 32 bit meno significativi dell'IR.

E li che ho la codifica binaria del mio spiazzamento;

SE NON STO UTILIZZANDO alcun spiazzamento, ma sto utilizzando una base e non sto utilizzando un indice, in questo caso le mie modalità di indirizzamento saranno di questo tipo:

MOV %eax, (%ebx) HO SOLO UN REGISTRO BASE

Il contenuto del registro base è l'indirizzo che ci interessa

Quindi il codice del registro base lo prendiamo dall'IR, e questo contenuto, per fare un accesso in memoria lo copio MAR.

Se ho soltanto un Registro indice, si avranno situazioni:

- Base, indice, scala $\text{Movl \%Eax}, (\%EBX, \%ECX, 1);$
- Indice, scala $\text{Movl \%Eax}, (,%ECX, 1);$

SE HO SOLO IL REGISTRO INDICE E LA SCALA, Vado a calcolare indice + scala.

Mi copio il contenuto del registro indice in tempo, e calcolo uno shift di un numero di posizioni codificate dalla base (T sta per base)

Se ho indice ed è presente anche uno spiazzamento, indice + scala lo copio dentro tempi, (con una base)

SE HO UNO SPIAZZAMENTO lo dovrò sommare a questo calcolo, lo spiazzamento mi lo prendo dall'IR, lascio la somma con la ALU, e se è presente un reg di base mi serve ancora la ALU, SE QUESTO È VERO, COPIO IN TEMPO IL CONTENUTO DEL REGISTRO DI BASE, E SI SCRIO IN MAR IL RISULTATO DELLA MIA SOMMA.

QUESTO È UN MICROPROGRAMMA CHE VALUTA TUTTE LE POSSIBILI COMBINAZIONI DELLA MIA MODALITÀ DI INDIRIZZAMENTO;

MA SE LO COPIO PER TUTTE LE VOLTE DA DEVO IMPLEMENTARE UN INDIRIZZAMENTO, NON È PROPRIO OTTIMALE.

COSA POSSO FARE? MODIFICO LA MIA CU!

PERCHÉ? PER SUPPORTARE LA CHIAMATA A SOTOPROGRAMMA FIRMIARCA!

IL MIO MICROCODE INTERAMENTE È IN GRADO DI SALVARE DA UNA PARTE ALL'ALTRA DELLE SUA ORGANIZZAZIONI.

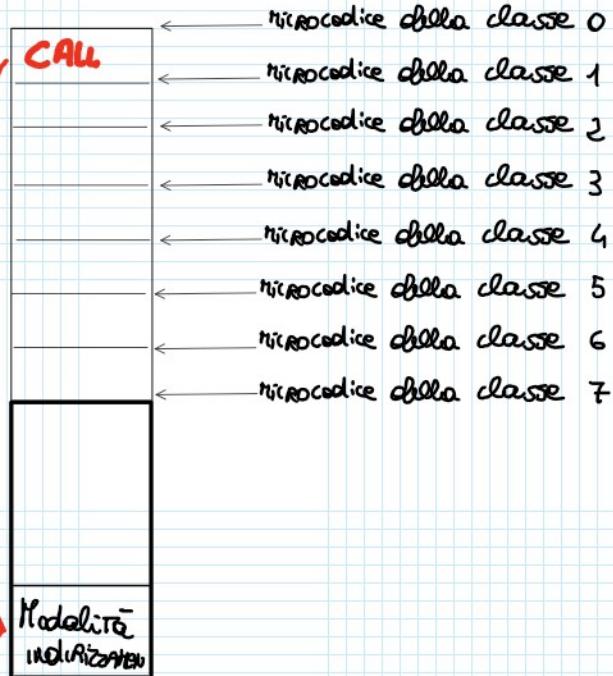
Se andiamo ad osservare la ROM PRIMA DI PASINARLA, IL NOSTRO MICROCODE È:

DENTRO IL CAMPO "opcode" ci sono 4 bit per la classe, e con questi 4 bit posso inserire 16 microprogrammi, MA ho 8 classi! Quindi, in realtà la ROM è piena per metà.

INSOLO CI POSSO METTERE DEI SOTOPROGRAMMI DI SERVIZIO.

Ci Posso mettere delle implementazioni di

altri sottoprogrammi che possono essere richiamati dal microcodice delle nostre classi.



implementazione di

Indirizzamento

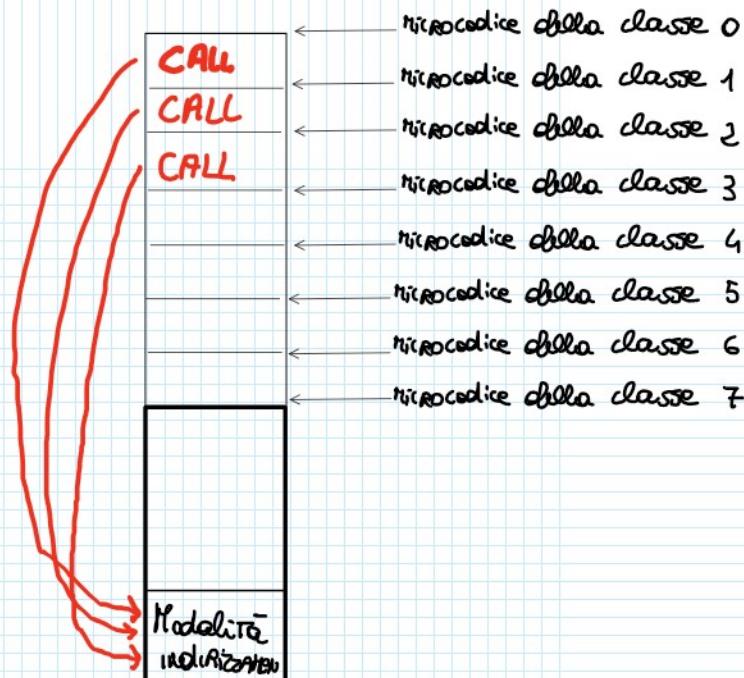
altri sottoprogrammi che possono essere richiamati dal microcodice delle nostre classi;

POSSO CHIAMARE IL SOTTOPROGRAMMA CHE MI CALCOLA LA MIA MODALITÀ DI INDIRIZZAMENTO (QUELLO CHE ABBIAMO VISTO SOPRA).

SE PIÙ DI UNA CLASSE CALCOLA LA MIA MODALITÀ DI INDIRIZZAMENTO, POSSO METTERE IN PIÙ PUNTI DELLE SOTTOCHIAMATE A MICROPROGRAMMI DI SERVIZIO CHE PERMETTANO DI RIDURRE LA DUPLICAZIONE DI MICROOPERAZIONI CHE SONO CONTENUTE NELLE VARIE PAGINE:

Se lo eseguo questo microprogramma di servizio, quando questo termina devo RITORNARE AL CHIAMANTE!

↳ Devo tenere traccia di chi è il chiamante;



IN UN PROGRAMMA C NOI MENTIONIAMO UN INDIRIZZO DI ATTORE ALLA FUNZIONE CHIAMANTE!

QUELLO CHE DOBBIAMO FAR E' INSERIRE ALL'INTERNO DELLA NOSTRA C (O UN REGISTRO, CHE PUÒ ESSERE UN REGISTRO DI RITORNO).

LA MIA ROM QUANDO STA ESEGUENDO LA CALL A SOTTOPROGRAMMA, PER POTER MODIFICARE IL SOTTOPROGRAMMA IN ESECUZIONE, DOVRÀ MODIFICARE SIA LA BASE CHE L'OFFSET.

MA PRIMA DI SOVRASCRIVERE IL CONTENUTO DI QUESTO REGISTRO SI DOVRÀ FAR ESSERE UNA COPIA, DEL REGISTRO DI RITORNO, DEL PUNTO NELL'MICROPROGRAMMA IN CUI E' ASSOCIAVATO.

Devo caricare il registro di ritorno con il valore del registro offset e base che identifica le microoperazioni successive da eseguire all'interno della ROM;

Quando il nostro microprogramma di servizio terminerà, dovrà RESTITUIRE IL CONTROLLO ALL'INDIRIZZO CHIAMANTE.

Quando il nostro microprogramma di servizio terminerà, dovrà restituire il controllo all'indirizzo chiamante.

Quindi dobbiamo modificare l'architettura perché così la base possa essere risposta all'interno della base e perché l'offset possa essere risposto all'interno dell'offset.

aggiungiamo delle copie di Fuller Three state che ad ogni colpo del clock dicono: "Mi il carico di questo puntatore e microistruzione deve avvenire da IR, oppure dal Registro di ritorno?

