Soluzioni del Capitolo 8

Questo documento contiene le soluzioni ad un numero selezionato di esercizi del Capitolo 8 del libro "Calcolatori Elettronici - Architettura e organizzazione", Mc-Graw Hill 2017.
Coloro che avessero sviluppato soluzioni alternative a quelle qui proposte, o soluzioni a esercizi non compresi tra quelli qui trattati, sono invitati a trasmetterle all'indirizzo sotto riportato. Serviranno a migliorare e tenere aggiornati i contenuti di questo sito.
L'autore sarà grato nei confronti di coloro che segnaleranno errori di qualunque genere, sia nella parte che segue sia nel libro menzionato.
giacomo.bucci@unifi.it

Aggiornato il 19 aprile 2017

8.2 I due registri servono a disaccoppiare la CPU rispetto all'esterno e a disaccoppiare le parte interne della CPU stessa. Nel caso della CPU vista in questo capitolo, nelle fasi di fetch, il contenuto di PC viene direttamente trasmesso alla memoria; nella CPU di Figura 2.11 del testo, avente un solo bus interno, il registro MAR serve ad appoggiare il contenuto di PC in modo che durante la lettura della memoria la ALU e il bus interno vengano usati per incrementare PC.

Inoltre, nell'ipotesi che, per esempio, il bus dati sia a 8 bit e le istruzioni occupino 16 bit, la lettura dell'istruzione richiede due cicli di memoria. In tal caso, dopo la lettura del primo byte, il secondo viene letto incrementando direttamente MAR, senza passare ad un nuovo caricamento del PC. In quest'ultimo caso il registro MDTR (detto anche DTR), serve a memorizzare temporaneamente la metà dell'istruzione letta, in attesa della seconda metà con la quale viene ricompattata per essere copiata in IR.

8.3 Le istruzioni di salto condizionato calcolano la condizione durante la fase EX. Se la condizione è verificata, PC deve essere aggiornato con l'indirizzo di destinazione, che a sua volta è calcolato come scostamento dal PC corrente. Dunque è necessario che in fase EX l'indirizzo di destinazione sia già disponibile, in quanto la ALU viene usata per il calcolo della condizione. Ne consegue che il calcolo dell'indirizzo di destinazione può essere effettuato in ID, fase durante la quale ALU non verrebbe altrimenti usata, salvando il risultato (l'indirizzo di destinazione) appunto in DEST.

Alternativamente si potrebbe immaginare di: (1) effettuare il calcolo della condizione in fase EX, salvando il risultato; (2) al clock successivo usare la ALU per calcolare l'indirizzo di destinazione (con conseguente aggiornamento di PC). Ma questa soluzione allungherebbe di 1 periodo di clock l'esecuzione delle istruzioni di salto condizionato.

8.4 PC1 serve a memorizzare l'indirizzo di ritorno in caso di chiamata di sottoprogramma (istruzione JAL), in modo da poter salvare tale indirizzo nel registro R31. Si ricordi che il registro R31 viene scritto in fase WB, quando PC è già stato aggiornato con l'indirizzo di destinazione (indirizzo del sottoprogramma chiamato da JAL) in fase ID. Si veda la Figura 8.14.

PC1 può essere eliminato se l'aggiornamento di PC e la scrittura di R31 avvengono sullo stesso clock (fronte finale). Questo è quanto avviene con lo schema ottimizzato di Figura 8.16. In tal caso PC1 può essere eliminato e su T2, ma, sul ramo JAL, si deve avere PCSorg=1.

8.5 Con riferimento alle Figure 8.12 e 8.14, si osservi che asserire il comando OUT su T3 non ha alcuna controindicazione. Al contrario se si asserisse M_Write potrebbe generarsi un problema di natura non deterministica: M_Write verrebbe asserito presumibilmente prima che la ALU abbia prodotto l'indirizzo e ciò determinerebbe la (possibile) scrittura alla posizione di memoria individuata dall'indirizzo corrente sul bus indirizzi (che poi la scrittura avvenga effettivamente non si può dire, in quanto i tempi non sono noti).

Rinviando l'asserzione di M_Write a T4 è garantito che quando la memoria riceve il comando di scrittura l'indirizzo è corretto (e stabile).

Si potrebbe pensare di asserire M_Write in T3 e prevedere un ritardo, in modo che M_Write arrivi alla memoria solo dopo che l'indirizzo prodotto dalla ALU è stabile. Ma questa soluzione è impraticabile perché porterebbe ad estendere la durata del clock in modo tale da comprendere l'accesso alla memoria e l'operazione di ALU.

Del resto i ritardi sono comunque da evitare in quanto determinano una dipendenza slegata rispetto al sincronismo scandito dal clock; infatti, a meno di escogitare qualche complicato artificio che renda il ritardo dipendente dalla frequenza del clock, se la frequenza del clock viene cambiata il ritardo resta lo stesso.

8.6 Il segnale è sincrono perché, a parte gli inevitabili ritardi introdotti dalle porte, le sue variazioni sono esclusivamente determinate dall'avanzamento del clock.

8.8 Lo schema di Figura 8.9 deve prevedere un ulteriore percorso verso PC, con l'aggiornamento di PC in fase EX. Tale nuovo percorso deve determinare, in fase EX, la somma di PC con il campo [6-30] e la scrittura del risultato in PC.

8.9 Il calcolo si effettua esattamente (stessi percorsi) come nel caso del calcolo dell'indirizzo nelle istruzioni LD/ST, ma qui il risultato è da prendere come dato finale da scrivere nel registro di destinazione. Seguendo lo schema di Figura 8.14 la durata resta 5 cicli di clock. Ottimizzando secondo lo schema di Figura 8.16, il calcolo può essere effettuato in T2, mentre l'aggiornamento può avvenire in T3, per un totale di 3 cicli.

8.10 L'esercizio è banale, trattiamo il solo caso dell'eliminazione del multiplexer comandato da JAL. Si tratta semplicemente di portare l'uscita di PC1 e della ALU al multiplexer in ingresso a RF.D, prevedendo, ovviamente, un ingresso in più e i relativi valori del selettore.

8.11 Anzitutto occorre stabilire che ci sono istruzioni che determinano l'aggiornamento della parola di stato e altre che non hanno effetto sulla medesima. L'istruzione SUB (come la ADD e tutte le aritmetiche in generale) appartiene alla prima categoria; le istruzioni di salto/diramazione appartengono alla seconda. Le istruzioni della prima categoria scrivono in PSW le condizioni di stato che esse determinano (bit di riporto, trabocco, zero, eccetera). L'istruzione successiva trova le condizioni in PSW, dunque la modifica di PC può essere effettuata prima possibile, compatibilmente con il calcolo dell'indirizzo di destinazione: nel nostro modello nello stadio ID (dove si calcola l'indirizzo di destinazione).

8.14 Per quanto si riferisce alla costruzione della parte dei bit di comando (parte b della parola di controllo) si faccia riferimento alla soluzione dell'esercizio 2.9.

L'esercizio richiede però che la prima parte della parola di controllo (parte a) contenga l'indirizzo della prossima istruzione. Supponiamo che i primi tre passi – quelli relativi

al fetch – siano codificati agli indirizzi 0, 1 e 2 della memoria di controllo. Facendo riferimento alla soluzione dell'esercizio 2.9, per queste tre posizioni di memoria si avrebbero questi contenuti (vengono riporti solo i bit di interesse nella fase di fetch (Paragrafo 2.6.1 del testo):

IND	Next μPC	PC out	PC in	MAR in	SEL4	ADD ALU	T0 in	T0 out	M read	DTR in	DTR out	IR in
0	1	1	0	1	1	1	1	0	0	0	0	0
1	2	0	1	0	0	0	0	1	1	1	0	0
2	–	0	0	0	0	0	0	0	0	0	1	1

Il μPC, ad ogni passo viene aggiornato con il contenuto del campo "Next μPC". Si noti che in posizione 2, quella corrispondente all'ultimo passo del fetch, contiene "−". Ciò perché il prossimo passo dipende dal codice di istruzione. Ne consegue che su T3, il μPC deve essere aggiornato in modo diverso. A tale scopo la logica deve essere prevista di un meccanismo in base al quale al codice di istruzione è associato un numero corrispondente alla posizione nella memoria di controllo da cui inizia il microcodice relativo alla fase di esecuzione dell'istruzione. Nell'ultima parola di tale microcodice il campo "Next μPC" conterrà sempre 0, in modo che possa riprendere il fetch della prossima istruzione.

8.15 Se la CPU legge il tag del registro da un campo dell'istruzione, è sempre possibile che un programmatore (mattacchione o disonesto) possa confezionare l'istruzione INT ponendo in quel campo un numero diverso da 30, con ciò determinando il salvataggio in un registro diverso da R30. Se la logica di CPU, di fronte all'istruzione INT assume che il registro di salvataggio è R30, il programmatore (né il compilatore) può farci nulla. Ma ciò richiede un minimo di logica addizionale. Infatti, con riferimento alla Figura8.12, è richiesta la generazione del numero 30 e la presenza del relativo percorso verso l'ingresso RF.RW (si veda anche l'Esercizio 8.16).

8.16 All'esercizio 8.15 abbiamo notato che lasciare implicito il registro R30 è la soluzione più sicura, ma richiede un minimo di logica addizionale. Se invece la logica di macchina preleva il numero 30 dal campo Rsd, non c'è da aggiungere nessuna logica aggiuntiva.

Ovviamente ci si aspetta che il compilatore generi codice corretto, ma resta il problema del programmatore di cui si è parlato all'esercizio 8.15.

8.17 L'istruzione RFI deve prelevare il contenuto del registro R30 e copiarlo in PC. Se R30 non è codificato nell'istruzione occorre prevedere la logica aggiuntiva per presentare il numero 30 su RF.RR2, in modo da leggere l'indirizzo di ritorno sulla porta B di RF. Si veda quanto detto per gli esercizi 8.15 e 8.16.

Ovviamente, se il numero 30 è codificato nel campo Rsd dell'istruzione, occorre solo la logica per dirottare l'uscita della porta B di RF verso PC (come per l'istruzione JR).

8.18 L'istruzione RFI ha formato analogo all'istruzione JR, ma con il campo Rsd forzato a contenere 30. Il suo trattamento non presenta particolari difficoltà: essa compie le stesse azioni di JR, con la differenza che essa deve anche riabilitare il sistema di interruzione. Con riferimento alla Figura 8.14, ciò si riduce ad aggiungere il corrispondente ramo di esecuzione, identico a quello di JR, cioè costituito da un solo stato su T2, ma comprendente anche il comando SIE.

8.19 Per capire cosa cambierebbe se non ci fosse l'AND tra IRQ e T2 conviene seguire l'evoluzione degli eventi a partire dalla "condizione di riposo", quella in cui IINTR è disasserito, RIS è disasserito e lo stato di IS è basso.

A partire dalla condizione di riposo, fintantoché IINTR non risulta asserito niente accade, in quanto l'ingresso "00" a IS mantiene il Flip-Flop nello stato in cui si trova. L'uscita dalla condizione di riposo si ha solo a seguito del passaggio di IINTR allo stato di asserito, il comando RIS viene asserito solo su T2 sul ramo di servizio all'interruzione, cioè quando IRQ è asserito – ovvero a seguito di IINTR).

Si hanno queste due possibilità

1. IINTR passa allo stato alto dopo il fronte finale[1] del periodo T2 e prima del fronte finale di RSRT; ovvero IINTR viene asserito in T3, T4 o T5.
   Questo caso corrisponde esattamente a quanto abbiamo visto in precedenza. Sul fronte di RSRT viene asserito IRQ e ciò determina il percorso di destra di Figura 8.23 (ramo IRQ). Su T2 viene asserito RIS, per cui il fronte finale del periodo fa commutare IS riportandolo allo stato baso (l'ingresso "11" fa cambiare stato al FF JK). Su T3 viene disabilitato il sistema di interruzione e con ciò il sistema è nella condizione di riposo, mentre il controllo è passato alla routine di servizio. Da questo momento, fino a quando il Flip-Flop IE non viene riportato nello stato "1", IINTR non potrà mai essere asserito. In conclusione, per ora togliere l'AND tra T2 e IRQ non ha alcun impatto.
2. IINTR passa allo stato alto dopo il fronte finale di RSRT e prima del fronte finale del periodo T2; ovvero IINTR viene asserito in T1 o T2, cioè quando il controllo percorre il ramo di sinistra di Figura 8.23 (ramo $\overline{\text{IRQ}}$).
   Poiché IINTR viene campionato sul fronte finale di T2, i due stadi S1 ed S2 vengono percorsi normalmente e il prossimo stato S3 sarebbe quello relativo all'istruzione in corso di esecuzione. Si noti che questo stato verrebbe percorso con IRQ asserito, mentre in Figura 8.23 è indicato che IRQ è disasserito. Poiché tuttavia per tutti gli stati da S3 a S5, indipendentemente dalla specifica istruzione, IRQ è sempre disasserito nelle espressioni di comandi e selettori esso non interviene e dunque il fatto che esso possa essere asserito è indifferente. In altre parole, se non entra nelle espressioni di comandi e selettori negli stati da S3 a S5, IRQ può indifferentemente essere asserito. Si noti anche che al termine dell'esecuzione dell'istruzione corrente risulta ancora IINTR asserito e quindi il fronte finale di RSRT non farà che confermare IRQ.
   Terminata l'istruzione corrente verrà percorso il ramo di destra, dando luogo al servizio dell'interruzione in modo del tutto analogo a quanto visto in precedenza.

In conclusione, fare a meno dell'AND tra IRQ e T2 non ha nessun effetto negativo. L'unica cosa osservabile è che gli stati dopo S2 possano essere attraversati con IRQ asserito.

Si noti infine che il ragionamento precedente porta a eliminare il RSRT come segnale di campionamento di IINTR, in quanto la funzione di campionamento può essere svolta dal solo T2 al quale è anche affidata la funzione di reset di IS. I altre parole, prendendo come clock di IS il solo T2, il fronte finale di questo periodo avrebbe la funzione di campionare IINTR dello stato di riposo e di resettare IS nel corso del servizio dell'interruzione.

8.20 Se PSW viene salvata in un registro di CPU le modifiche da apportare allo schema del testo si riducono a poca cosa: prevedere il registro in questione e un percorso da PSW a tale registro: tale percorso verrà attivato assieme a quello usato per salvare PC.

[1] Ricordiamo che per assunzione i FF commutano sul fronte finale

Se PSW viene salvata sullo stack le cose si complicano. Ovviamente, in questo caso, pure PC sarà salvato sullo stack. La macchina dovrà essere dotata della logica per la gestione dello stack (almeno il registro SP). Bisognerà mettere in conto almeno un passo per salvare PC e uno per salvare PSW. In ambedue i casi si tratta di scritture in memoria, all'indirizzo dato da SP. Ovviamente SP dovrà essere aggiornato ad ogni scrittura (come pure ad ogni lettura nell'operazione RFI).