PREDIZIONE DEI SALTI

Quando si trova una micro-operazione di salto condizionato (bit JAMN o JAMZ del campo JAM diversi da zero) non si ha la certezza della destinazione del salto fino a dopo l'esecuzione da parte dell'ALU. Quindi in caso di salto condizionato in MIR3 e MIR4 si forma una "bolla" (nel ciclo in cui l'istruzione di salto condizionato si trova nello stadio dell'ALU) e la nuova microistruzione sarà caricata con un ciclo di ritardo.

Per evitare di perdere sempre un ciclo (o più cicli, se la pipeline ha più stadi) si può tirare a indovinare: **BRANCH PREDICTION.**

Una volta fatta la previsione, se questa era corretta non si perde alcun ciclo, altrimenti occorre fare un'operazione di **SQUASHING** (svuotare la parte di pipeline contenente le istruzioni che non dovevano essere caricate, e ripartire dal caricamento dell'istruzione giusta).

Tecniche di **Branch Prediction**: **1-tecniche statiche:**

1)euristiche (es. salti in avanti non eseguiti, salti indietro eseguiti)

2)con indicazione compilatore (introducendo nel linguaggio macchina istruzioni di salto che contengono l'indicazione del compilatore)

2-tecniche dinamiche (mantenere una tabella che registra informazioni sul comportamento passato)

Le predizioni dinamiche avvengono durante l'esecuzione del programma.

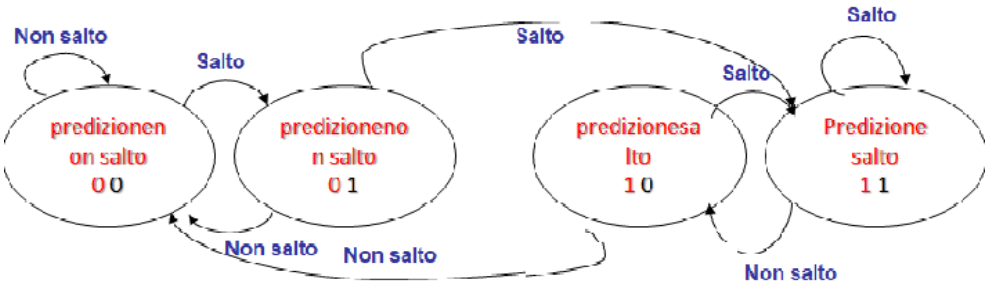
PREDIZIONE DINAMICA DEI SALTI

Il fatto che le predizioni siano accurate è molto importante, dato che in tal caso la CPU ha la possibilità di procedere alla massima velocità. Uno dei possibili approcci prevede che la CPU abbia in appositi componenti hardware una tabella della storia dei salti, nella quale tiene traccia dei salti condizionati incontrati in modo da poterli rianalizzare quando si verificano nuovamente.

Tabella della storia dei salti

La tabella della storia dei salti contiene un elemento per ogni istruzione di salto condizionale; l'elemento contiene l'indirizzo dell'istruzione di salto oltre a un bit che indica se è stata effettuata la diramazione l'ultima volta che l'istruzione è stata eseguita. Usando questo schema la predizione consiste semplicemente nell'assumere che il salto si comporterà allo stesso modo dell'occorrenza precedente. Se la predizione si rivela sbagliata, allora si modifica il bit della tabella della storia dei salti.

Quando si termina l'esecuzione di un ciclo, il salto che si trova alla fine verrà predetto in modo errato; ancor peggio, questo errore di previsione cambierà il bit nella tabella dei salti per indicare che in futuro occorrerà prevedere di non effettuare il salto. Questo problema può verificarsi spesso se il ciclo è innestato all'interno di un altro ciclo. Per eliminare questa predizione sbagliata possiamo dare una seconda chance all'elemento in tabella, ovvero modificare la predizione soltanto dopo che due previsioni consecutive si sono rivelate errate. Nella pagina seguente è disegnato l'automa a stati finiti a 2 bit per la previsione delle diramazioni.



Organizzazione delle tabelle della storia dei salti

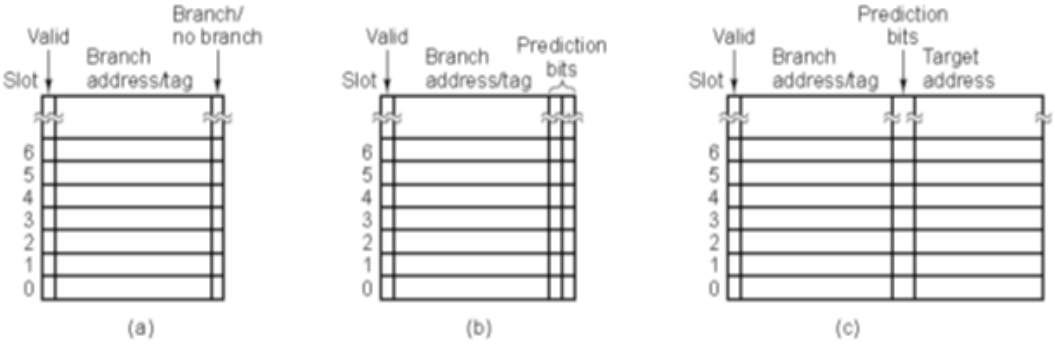
Le informazioni necessarie per la predizione dinamica dei salti vengono memorizzate in MEMORIE ASSOCIATIVE (sono memorie simili a quelle utilizzate per realizzare le cache) che vengono indirizzate in base al contenuto: è possibile chiedere alla memoria associativa di restituire il contenuto del campo.

1)Branch/no branch(a),

2)prediction bits (b),

3)Target address(c),

In base al contenuto del campo «Branch address» (cioè in base all'indirizzo dell'istruzione di salto condizionato su cui dobbiamo effettuare la predizione).



Finora abbiamo assunto che la destinazione di un salto condizionale fosse nota, generalmente come indirizzo esplicito al quale andare (contenuto all'interno dell'istruzione stessa). Questa situazione è valida nella maggior parte dei casi, tuttavia esistono alcune istruzioni di salti condizionali che calcolano l'indirizzo di destinazione eseguendo delle operazioni aritmetiche sui registri. L'automa a stati finiti illustrato sopra predice in modo accurato se eseguire i salti oppure no, ma queste previsioni sono inutili se non si conosce l'indirizzo di destinazione.

Un modo per gestire questa situazione consiste nel memorizzare nella tabella della storia l'effettivo indirizzo verso il quale è stato effettuato il salto l'ultima volta. Seguendo questo metodo, se la tabella indica che l'ultima volta il salto all'indirizzo 516 è stato effettuato e ha portato all'indirizzo 4000, allora, nel caso in cui la predizione dica di effettuare la diramazione, l'esecuzione continuerà nuovamente a partire dall'indirizzo 4000.

Uno dei vantaggi delle tecniche dinamiche è che si adattano al comportamento corrente del programma. Lo svantaggio è che richiedono un costoso hardware specializzato e una grande complessità del chip.

PREDIZIONE STATICA DELLE DIRAMAZIONI

Una tecnica che consideriamo per effettuare la predizione dei salti si basa sul **profiling.** Questa tecnica è di tipo statico, ma invece di delegare al compilatore il compito di prevedere quali diramazioni verranno oppure no effettuate, si manda in esecuzione il programma di solito su un simulatore per catturare il comportamento delle diramazioni. Questa informazione viene poi fornita al compilatore, che la utilizza per impostare le istruzioni speciali di salto condizionale mediante le quali può specificare all'hardware in che modo deve comportarsi.

ESECUZIONE FUORI SEQUENZA

La maggior parte delle architetture moderne funziona sia a pipeline sia in modo superscalare. In generale ciò comporta la presenza di un'unita di fetch che preleva le istruzioni dalla memoria prima che siano necessarie, in modo da alimentare un'unità di decodifica. L'unità di decodifica attribuisce l'istruzione decodificata all'unità funzionale appropriata perché sia eseguita.

L'architettura più semplice prevede che tutte le istruzioni siano eseguite nell'ordine in cui sono prelevate dalla memoria (ipotizzando che la predizione non sbagli mai). Tuttavia l'esecuzione in ordine non sempre fornisce prestazioni ottimali a causa delle dipendenze tra istruzioni. Nel tentativo di aggirare questi problemi e ottenere prestazioni migliori, alcune CPU permettono di saltare le istruzioni dipendenti per raggiungere le istruzioni successive che non lo sono (concetto di riordino delle istruzioni, allo scopo di migliorare le prestazioni). Dobbiamo però garantire che l'esecuzione produca lo stesso risultato che produrrebbe l'esecuzione ordinata.

Ora vediamo l'esecuzione ordinata (guarda tabella pagina 328)

Le istruzioni devono essere sia avviate sia concluse in ordine e rispettando sia le dipendenze.

-di tipo RAW (una istruzione usa come operando il risultato di una istruzione precedente) .

-e sia le dipendenze di tipo WAR e WAW (Scrittura di un risultato in un registro destinazione usato come operando da istruzioni precedenti ancora in corso o scrittura di un risultato in un registro destinazione usato come destinazione anche da altre istruzioni in corso).

Ora vediamo l'esecuzione non ordinata (guarda tabella pagina 331)

Le istruzioni possono essere sia avviate sia concluse in disordine

-sempre rispettando le dipendenze di tipo RAW (una istruzione usa come operando il risultato di una istruzione precedente)

-e gestendo in modo efficiente le dipendenze di tipo WAR e WAW (Scrittura di un risultato in un registro destinazione usato come operando da istruzioni precedenti ancora in corso o scrittura di un risultato in un registro destinazione usato come destinazione anche da altre istruzioni in corso).

Mentre iniziare le istruzioni in modo non ordinato non è un problema (eseguire il calcolo non ha conseguenze, finche' i risultati non sono resi "ufficiali"):1) concludere la loro esecuzione in modo non ordinato, anche quando non vi sono dipendenze, può produrre un problema in caso di interruzioni.2) Interruzioni non precise: la conclusione "disordinata" non permette di caratterizzare

in modo semplice il punto a cui era arrivata l'esecuzione (valore dei PC e di tutti gli altri registri) nel momento in cui è intervenuta l'interruzione.

Pentium: possono solo essere iniziate le microistruzioni in modo disordinato, ma devono essere concluse in modo ordinato.

UltraSPARC II: ammette sia l'inizio sia la conclusione in modo disordinato.

ESECUZIONE SPECULATIVA

L'**Esecuzione speculativa** è una tecnica di ottimizzazione che consiste nel fare eseguire al computer operazioni che potrebbero essere necessarie solo in un secondo tempo. Elaborando i dati prima di sapere se è davvero necessario farlo può ridurre i ritardi che si avrebbero facendo il lavoro solo dopo aver saputo se è davvero necessario o no. Se ad un certo momento del flusso di esecuzione il lavoro svolto anticipatamente si dimostra inutile, allora i risultati ottenuti verranno semplicemente ignorati.