Il pipelining

Per potenziare un microprocessore per PC con lo scopo di renderlo più rapido, si è visto che la tecnica più semplice consiste nel fargli elaborare più velocemente le istruzioni, il che si ottiene aumentando la frequenza di clock.
Un altro metodo, denominato parallelismo, consiste invece nel fare in modo che il microprocessore sia in grado di elaborare più istruzioni contemporaneamente.
Il parallelismo ha fatto la sua comparsa con l'avvento dei primi Pentium della Intel.
Può essere implementato mediante 2 tecniche:

- il pipelining
- e l'architettura superscalare.

La pipelining permette di elaborare più istruzioni contemporaneamente perchè l'esame di una istruzione inizia prima che la precedente sia già stata completamente elaborata.
Nella tecnica con pipeline si può incominciare l'implementazione di un nuovo processo mentre l'azione in corso non è ancora terminata perchè con la suddetta tecnica si suddivide in più fasi, ovvero in più livelli, il trattamento delle istruzioni.
Nell'architettura superscalare il parallelismo è ottenuto aumentando effettivamente il numero di unità interne al microprocessore che effettuano il trattamento delle istruzioni.
Tutti i moderni microprocessori utilizzano entrambe le tecniche, per cui i processori per PC di oggi sono caratterizzati da più pipeline ognuna composta da 5 a 12 livelli.
Il numero dei livelli delle singole pipeline non può aumentare più di tanto.
I singoli livelli, infatti, vengono percorsi alla stessa identica velocità, con quest'ultima che viene calcolata sulla base dell'istruzione che richiede il tempo di implementazione più lungo.
Più cresce il numero dei livelli delle singole pipeline, più si rischia di allungare il tempo di implementazione delle istruzioni, provocando un rallentamento generale.
D'altra parte, più è corta la pipeline, più è probabile che aumentano il numero di cicli di clock durante i quali il processore rischia di trovarsi con un flusso interrotto e quindi di perdere diversi cicli di clock, ovvero di girare a vuoto nell'attesa che la memoria fornisca il codice o i dati da elaborare.

Con lo scopo di minimizzare anche l'impatto del tempo perso dal microprocessore nell'attendere le altre componenti del PC più lente, ovvero la RAM, il disco rigido, e così via, tutti i processori Intel dal Pentium II in poi, si sforzano di tenere conto di utilizzando due ulteriori tecniche:

- la previsione di salto

- e l'esecuzione speculativa. dei dati da elaborare.

Tecniche che in effetti consistono entrambe nell'anticipare il trattamento di pezzi di codice, con ciò tenendo appunto sempre piena la pipeline, che per la lentezza della RAM, del disco rigido e così via, sovente correrebbe il rischio di svuotarsi completamente facendo girare a vuoto il processore.

In effetti, si tratta di algoritmi che permettono di prevedere quali istruzioni verranno eseguite, in modo da poterle inserire nella pipeline ancor prima che il programma che sta girando sul PC vi faccia appello.

Naturalmente, prima di mandare effettivamente in esecuzione l'istruzione caricata nella pipeline in anticipo, il microprocessore verificherà che il vero comando richiesto dal programma sia esattamente quello emerso dall'implementazione anticipata.

In caso affermativo il microprocessore avrà guadagnato altro tempo, in caso contrario bisogna svuotare tutta la pipeline per ricominciare di nuovo, e questa volta su basi corrette.

Gli attuali microprocessori raggiungono una percentuale di previsioni corrette superiore al 98 %, il che equivale a dire che i moderni processori finiscono per buttar via solo il 2 % dei cicli di lavoro.

Il branch target buffer

L'uso ottimale di una pipeline è consentito dal branch target buffer: esso effettua una previsione dinamica basata sulla storia passata di un salto.
Il branch buffer consiste di una tabella con un certo numero di entry.

Ogni entry si divide in 3 campi:
-L'indirizzo di una branch instruction (istruzione ramo).
-L'indirizzo di una possibile istruzione successiva(target).
-Una voce che indica quanto è consigliata la possibile istruzione successiva.

le voci possibili del terzo campo sono: strongly taken, weakly taken, weakly not-taken e strongly not-taken.

Ogni volta che in un programma una branch instruction entra nella pipeline di esecuzione bisogna scegliere una istruzione successiva da fetchare, per farlo si guarda il terzo campo del branch buffer, dove il primo campo sia uguale alla branch instruction appena fetchata:
nel caso in cui indichi strongly taken o weakly taken viene fetchata la possibile istruzione successiva, indirizzata dal secondo campo, altrimenti in caso dica weakly not-taken e strongly not-taken viene fetchata l' istruzione successiva alla branch instruction nel programma.
Ogni volta che una strada viene scelta la terza voce della entry fa un passo verso il strongly taken.
Ogni volta che una strada viene evitata la terza voce della entry fa un passo verso il strongly not-taken.