


Trend di sviluppo delle pipeline

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
borghese@dsi.unimi.it

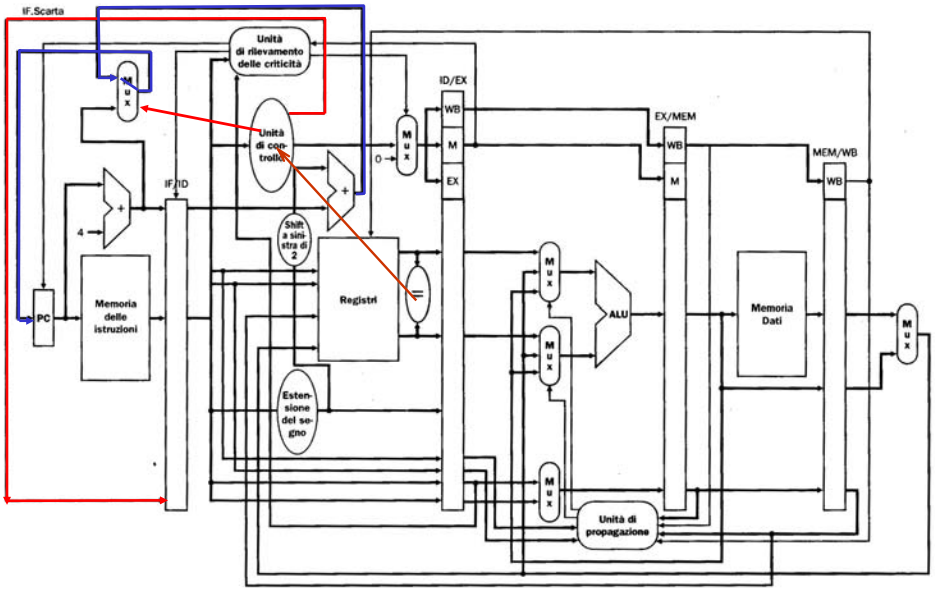
Università degli Studi di Milano

Patterson 5.9, 6.6 e 6.9


Sommario

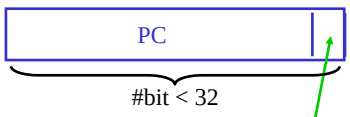
Predizione dinamica dei salti
Pipeline avanzate
Multiple-Issue Statici
Multiple-Issue Dinamici
La pipeline del Pentium IV

CPU con pipeline completa della gestione degli hazard.



Branch prediction buffer

Branch prediction ad esempio tramite: branch prediction buffer (4 kbyte nel Pentium 4).



Bit meno significativi del PC

Bit che indica se l'ultima volta il salto era stato eseguito o meno.

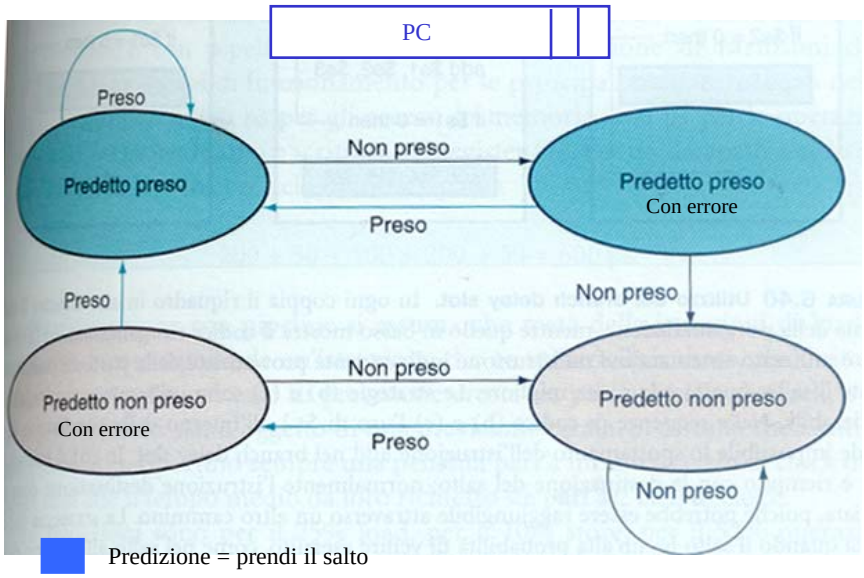
Problema:
Previsione relativa ad una beq con gli stessi bit meno significativi del PC.
E' un problema?

```
        beq $t0, $t1, SALTA
        add $s0, $s1, $s2
        sub $s3, $s4, $s5
        or  $s6 $s7, $s8
SALTA:  and $t2, $t3, $t4
```

In questo caso suppongo di non dovere saltare. Procedo in sequenza.
Se la previsione è sbagliata, devo annullare la add e saltare a SALTA.

Algoritmi di ottimizzazione dello scheduling per predizione ottima del salto.

Branch Prediction buffer a 2 bit



Evoluzioni della branch prediction

1) **Correlating predictors.** Comportamento locale e globale dei salti. Tipicamente 2 predittori a 2 bit. Viene scelto il predittore che correla meglio con la storia del salto.

2) **Tournament predictors.** Vengono utilizzati predittori multipli a 1 o 2 bit, e per ogni branch viene selezionato il predittore migliore. Il selettore selezione quale dei due bit di informazione utilizzare, in base alla loro accuratezza di predizione. Solitamente viene utilizzato un predittore che analizza informazioni locali (di contesto), un altro che analizza informazioni globali (di contesto). Informazioni di contesto possono ad esempio essere contenuto di registri…. Il codice di selezione per il selettore viene memorizzato del branch prediction buffer.

Sommario

Predizione dinamica dei salti
Pipeline avanzate
Multiple-Issue Statici
Multiple-Issue Dinamici
La pipeline del Pentium IV

Pipeline avanzate

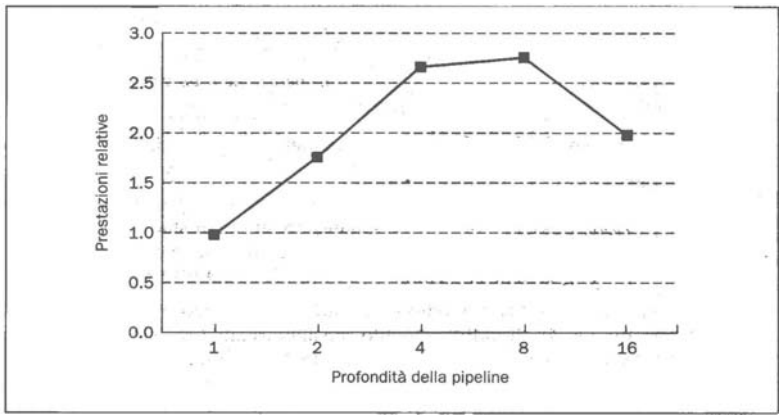
Instruction level parallelism (ILP). Come si può aumentare?

- Superpipelining (pipeline più lunga).
- Superscalar (componenti multipli) o “Multiple-issue”.

Scheduling dinamico.

Superpipeline

Pipeline più lunghe (e.g. Digital DecAlpha 21264, 5-7 stadi).
Teoricamente: guadagno in velocità proporzionale al numero di stadi.



Problemi:

- Criticità sui dati: stalli più frequenti.
- Criticità sul controllo: numero maggiore di stadi di cui annullare l'esecuzione.
- Maggior numero di registri: il clock non si riduce linearmente con il numero degli stadi.

Principi delle pipeline Superscalari - I

Duplicazione delle unità funzionali (e.g. ALU e ALUfp) ed eseguo più istruzioni in parallelo.

Aumento del numero di istruzioni per ciclo di clock (CPI < 1).

Complicazione architetturale nel raccogliere il risultato dei componenti ed organizzarlo per l'uscita.

Esempio:

Processore a 4 vie da 6Ghz potrebbe eseguire 25 miliardi di istruzioni / s. Oggi si arriva anche a 8 istruzioni / ciclo di clock.

Vincolo sul tipo di istruzioni che vengono eseguite in parallelo.

Principi delle pipe-line Superscalari - II

- **Due approcci:**
- Static multiple issues (schedulazione decisa dal compilatore)
- Dynamic multiple issues (schedulazione decisa run-time dalla CPU).

Corrisponde alla suddivisione del lavoro tra SW e HW, cioè tra il compilatore ed il processore.

Impacchettamento delle istruzioni in **issue slot**: esaminando il codice, quante e quali istruzioni possono essere impacchettate in un ciclo di clock?

Occorre gestire gli hazard nei dati. Nei sistemi statici, questi hazard vengono analizzati dal compilatore, occorre prevedere dei sistemi (e.g. forwarding), nei sistemi dinamici.

La speculazione

Ricorso massiccio alla **speculazione** (= immaginare degli scenari).

Esempio: si può speculare sul risultato di una branch, e quindi eseguire le istruzioni di conseguenza.

Esempio: si può speculare se una lw abbia come registro target uno dei registri utilizzati dalle istruzioni vicine o meno.

La speculazione può essere fatta sia dal compilatore che dal processore (mediante logica di controllo). Il compilatore può utilizzare la speculazione per riordinare le istruzioni, la CPU per riordinare l'esecuzione run-time.

E se la speculazione risulta sbagliata? Deve esistere un meccanismo di correzione (roll-back).

Come viene corretta la speculazione

. La speculazione si paga in termini di meccanismi per **controllare** se la speculazione è stata corretta e di **correggerla**.

La soluzione ai problemi posti dalla speculazione è diversa nel caso statico o dinamico:

- Il compilatore inserisce delle istruzioni di controllo e di correzione a speculazioni errate, anche chiamando procedure opportune.
- La CPU mette in buffer i risultati dell'esecuzione speculativa fino a quando non si è potuto verificare la correttezza della speculazione. Nel caso di speculazione errata, la cancellazione del lavoro fatto viene ottenuta semplicemente svuotando i buffer e correggendo la sequenza di istruzioni.

Occorre speculare quando si hanno degli elementi validi, altrimenti si possono inserire problemi (vedi eccezioni "speculative") che rendono il funzionamento meno efficiente.

Sommario

Predizione dinamica dei salti

Pipeline avanzate

Multiple-Issue Statici

Multiple-Issue Dinamici

La pipeline del Pentium IV

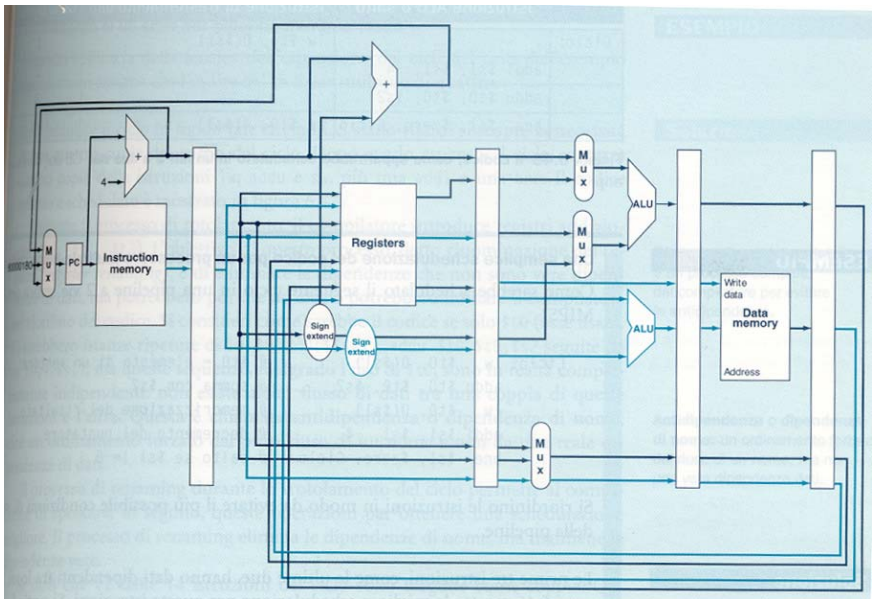
Multiple issue statici (schedulazione statica)

Issue packet è l'insieme delle istruzioni inviate alla pipeline in un ciclo di clock = 1 grande istruzioni con tante operazioni e tanti operandi.

Basato su compilazione.

L'approccio ILP di fatto vede le istruzioni come una **VLIW** (Very Long Instruction Word).

Schedulazione statica: il MIPS64



Schedulazione delle istruzioni in MIPS a 2 Vie

Tipo di istruzioni	Stadi di pipeline							
ALU o salto	IF	ID	EX	MEM	WB			
Load o store	IF	ID	EX	MEM	WB			
ALU o salto		IF	ID	EX	MEM	WB		
Load o store		IF	ID	EX	MEM	WB		
ALU o salto			IF	ID	EX	MEM	WB	
Load o store			IF	ID	EX	MEM	WB	
ALU o salto				IF	ID	EX	MEM	WB
Load o store				IF	ID	EX	MEM	WB

Esempio

```
Ciclo: lw $t0, 0($s1)
       addu $t0, $t0, $t2
       sw $t0, 4($s1)
       addi $s3, $s1, -4
       bne $s1, $zero, Ciclo
```

Address	ALU o salto	Trasferimento dati	Ciclo clock
Ciclo:		lw $t0, 0($s1)	1
	addi $s1, $s1, -4		2
	addu $t0, $t0, $s2		3
	bne $s1, $zero, Ciclo	sw $t0, 4($s1)	4

Pipeline delle Architetture IA-64

•L'architettura INTEL IA-64, utilizza questo approccio che ha battezzato **EPIC** (Explicitly Parallel Instruction Computer): **Itanium**, progetto MERCED "venduto" a Digital (2000), **Itanium-2** (2002). **Pentium D** → Estensione della capacità di indirizzamento a 64 bit (throughput maggiore verso la memoria). http://www.intel.com/products/processor/pentium_d/

- Architettura Load/Store su 64 bit
- Ricca dotazione di registri: 128 interi, 128fp, 8 per i salti e 64 per i flag (cf. Architetture Intel classiche).
- Organizzazione delle istruzioni in **bundle** con formato fissa ed identificazione delle dipendenze.
- Riorganizzazione del codice per eliminazione dei salti ed istruzioni speciali per l'esecuzione speculativa.

Gruppo di istruzioni.

Istruzioni che non hanno dipendenze nei registri destinazione.

Possono essere eseguite in parallelo.

Vengono messe in sequenza, separate da un separatore esplicito (stop).

Bundle nell'architettura IA-64

- 5 cammini diversi per la fase di esecuzione:
  - ALU intera
  - ALU non intera (shift, multimedia)
  - ALUfp
  - Memoria
  - Salti.

Bundle: istruzione VLIW di 128 bit: template (5 bit) + 3 istruzioni (41 bit)

template: stop + tipo di unità richiesta dalle 3 istruzioni.

Ampio utilizzo di esecuzione speculativa e predicativa.

Esecuzione predicativa

Srotolo il codice in modo da evitare i salti.
Utilizzo solo condizini:
if (p) procedura1.
if (!p) procedura2

Sostituisco procedura_i con nop nel caso in cui la condizione non sia vera.

Sommario

Predizione dinamica dei salti
Pipeline avanzate
Multiple-Issue Statici
Multiple-Issue Dinamici
La pipeline del Pentium IV

Dynamic issues (schedulazione dinamica)

Questi processori sono detti anche **superscalari**.

La scelta di quali istruzioni inviare alla pipe-line viene eseguita durante l'esecuzione stessa. Dipende dalla compatibilità tra le varie istruzioni e da eventuali hazard su dati e controllo.

Nella versione più semplice, le istruzioni sono processate in sequenza ed il processore decide se elaborarne nessuna (stallo), una o più di una a seconda delle criticità riscontrate.

L'ottimizzazione del codice da parte del compilatore è comunque richiesta.

Confronto tra Multiple-issue statici e dinamici

Confronto tra superscalare (sequenziale) ed architetture basate su VLIW.

L'hardware di una pipe-line superscalare garantisce la correttezza del codice.

Il codice verrà eseguito correttamente qualunque sia la CPU sul quale viene fatto girare (purchè contenga l'ISA su cui il codice è basato!).

Nella schedulazione statica, spesso occorre ricompilare passando da una CPU ad un'altra oppure il codice può girare con prestazioni molto scadenti.

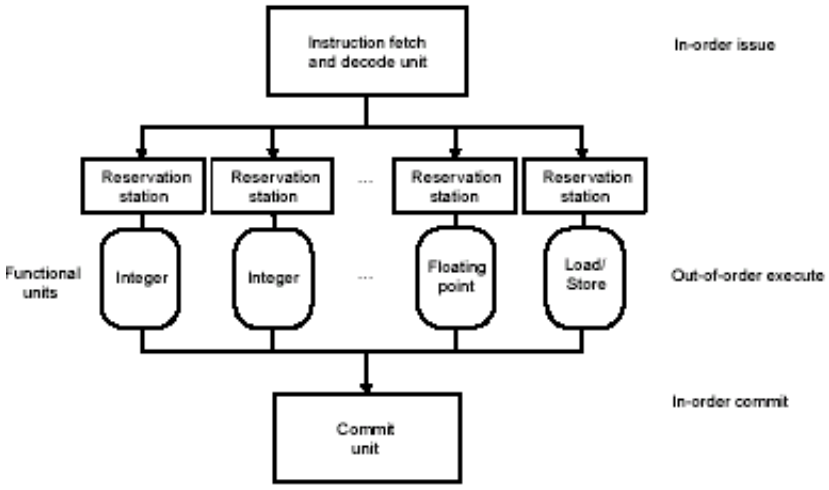
Schedulazione dinamica

Consente di riordinare il codice in modo da decidere quale insieme di istruzioni eseguire in un certo ciclo di clock in modo da evitare hazard su dati e controllo.

```
lw     $t0, 20($s2)
addu   $t1, $t0, $t2
sub    $s4, $s4, $t3
slti   $t5, $s4, 20
```

Posso eseguire la sub, senza dove aspettare il completamento della lw e della addu. Decido cosa schedulare prima nel momento in cui ho un gruppo di istruzioni (nella trace cache).

Pipeline con schedulazione dinamica



Esistono diversi cammini paralleli (per la fase di esecuzione e memoria) dell'istruzione, vengono scelti dinamicamente, run-time.

Principi della schedulazione dinamica

Obbiettivo: mettere in esecuzione istruzioni che non presentino criticità.

Le istruzioni vengono bufferizzate dalla **reservation station**, la quale gestisce la coda delle istruzioni che hanno bisogno della stessa unità funzionale.

Al termine dell'esecuzione la **reorder station**, provvede ad ordinare le istruzioni nella sequenza con la quale devono essere restituite.

Per eseguire un'operazione è sufficiente che il dato sia già pronto nel reorder buffer, senza che sia necessariamente scritto nel register file.

Un'operazione viene lanciata, quando i dati sono pronti. Se un dato non è pronto viene inserita un'etichetta che associa (traccia) il dato al cammino che lo deve produrre. Quando il dato viene eseguito, tramite etichetta si libera il blocco all'esecuzione dell'istruzione.

NB Le istruzioni non sono eseguite sequenzialmente.

Sommario

Predizione dinamica dei salti
Pipeline avanzate
Multiple-Issue Statici
Multiple-Issue Dinamici
La pipeline del Pentium IV

Le pipeline Intel

Ultimo processore Intel senza pipeline: 386, 1985.

Esecuzione multi-ciclo:
- Durata diversa per istruzioni molto diverse (cf. `MOVS`).
- Riutilizzo di unità funzionali in diversi passi di esecuzione.

UC cablata per le istruzioni semplici e microprogrammata per le istruzioni più complesse.

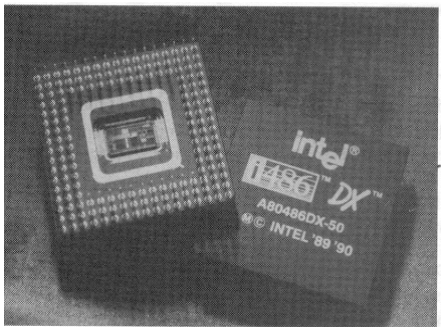
Pentium 4: Superpipeline superscalare. Fino a 3 istruzioni per ciclo di clock.

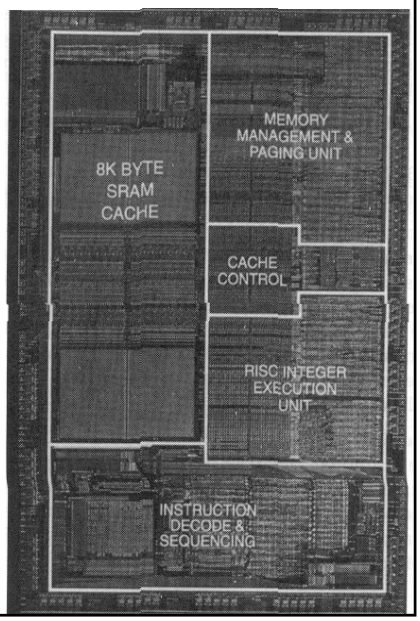
All'interno della pipeline, abbiamo microistruzioni di ampiezza pari a 70 bit (fissa).
A partire dal codice operativo, vengono generati i segnali di controllo: 120 per le ALU intere, 275 per le unità ALUfp e 400 per le istruzioni SS2.

Trasformo le istruzioni dell'ISA Intel in microistruzioni di lunghezza uguale.

L'Intel 80486

Integrazione in un solo chip di CPU, Coprocessore e Cache controller.





L'Intel 80486 – Struttura interna

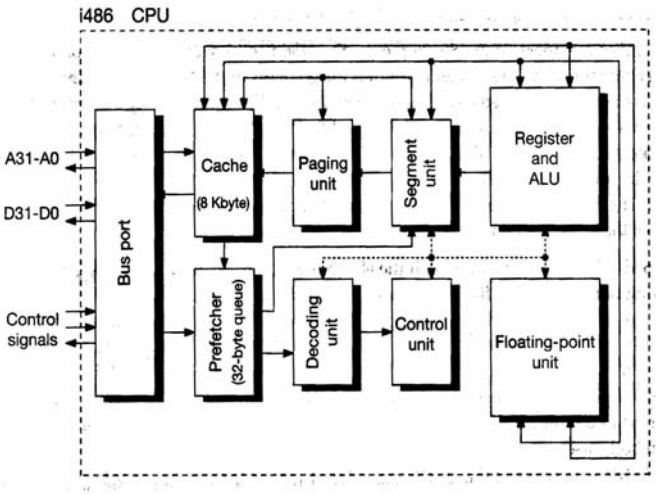
Interfacciamento con il resto della macchina tramite la porta verso il bus. Sul bus vengono inviati i dati, gli indirizzi ed i segnali di controllo.

Pipeline a 5 stadi.

Bus interno a 64bit.

Cache a 8kbyte. Modalità di scrittura: write-through con buffer.

Control unit micro-programmata



Da IA-32 alle microistruzioni della pipeline: fetch

486, PentiumPro, I, II, III – conversione di istruzioni IA-32 in triplette di microistruzioni da inserire nella stessa parola di istruzione, VLIW.

- **Fase di fetch**: lettura dell'istruzione mediante la coppia:
  - CS:IP dal segmento codice + instruction counter (PC + offset)
  - Quindi: IP ← IP + #byte_istruzione

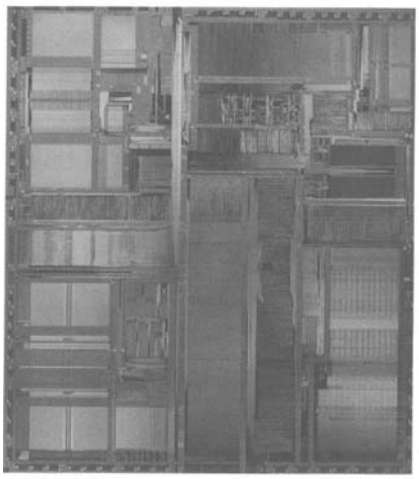
- **Pre-fetch Queue** (Coda di pre-fetch). Streaming dal segmento codice di RAM in un buffer fino al riempimento. L'UC legge il primo byte dell'istruzione dalla coda di pre-fetch e trasferisce un certo numero di byte nell'IR.

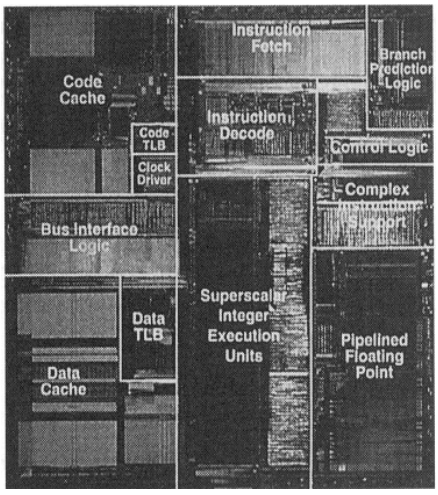
Da IA-32 alle microistruzioni della pipeline: decode

Possibilità di traduzione di un'istruzione IA-32:

1) La traduzione è all'interno della trace cache. Fino a 3 microistruzioni vengono inviate in output dalla trace cache a seconda dell'istruzione IA-32.
2) La traduzione è all'interno della trace cache, ma lo sviluppo della traduzione è costituito da più di 3 microistruzioni. In questo caso si utilizza una ROM di appoggio che contiene il microprogramma di traduzione che può arrivare anche a 8,000 microistruzioni per i task più complessi quali task switch tramite gate che viene operata in modalità protetta.
3) La traduzione nonè all'interno della trace cache. In questo caso il decodificatore dell'architettura IA-32 viene utilizzato per decoficare. Se la traduzione consiste in fino a 3 microistruzioni, queste vengono inserite nella trace cache, altrimenti viene trasferito il controllo alla ROM.

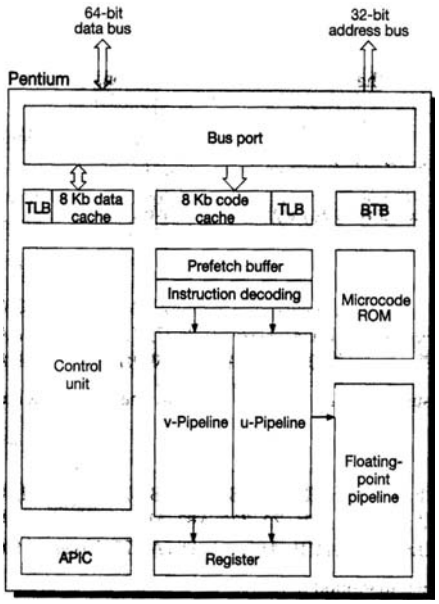
Il pentium





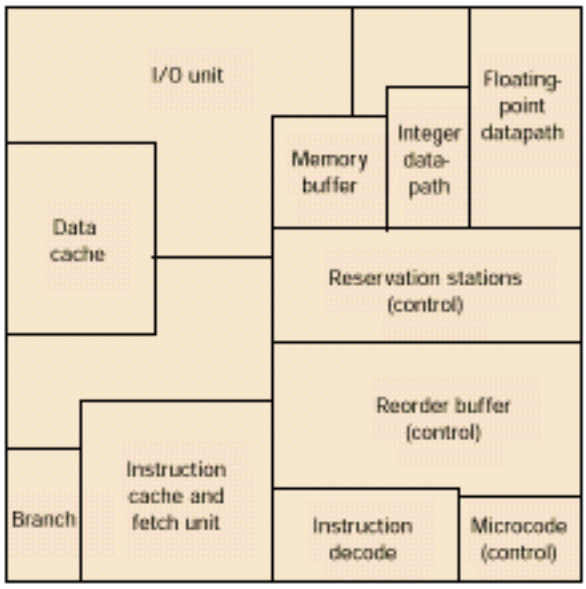
Pentium – struttura interna

- 2 pre-fetch queues: la pipeline u si interfaccia con la pipeline floating point.
- Advanced Programmable Interrupt Controller (è attivo per la doppia CPU).
- Microcodice per l'esecuzione del controllo.
- Cache interfacciate sul bus del processore.

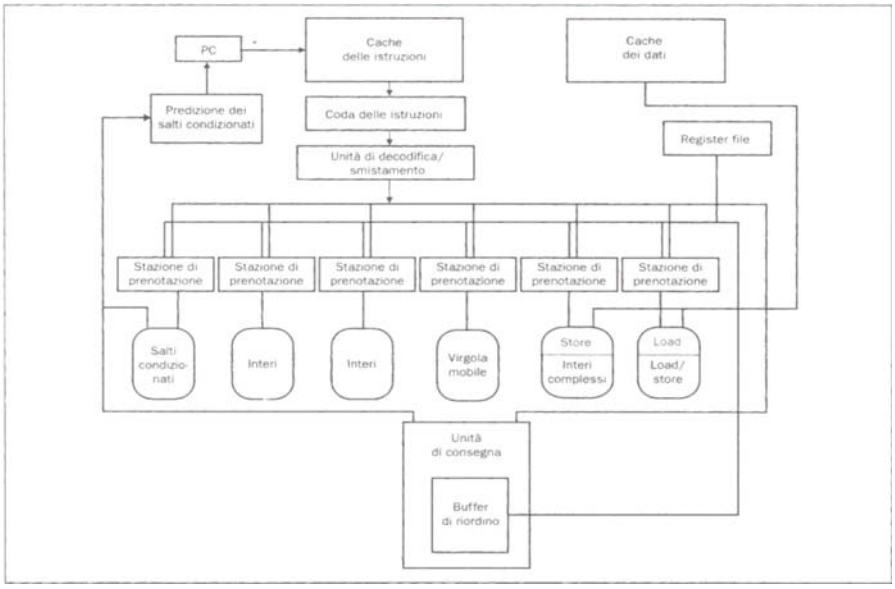


Il processore Pentium Pro

Esecuzione "speculativa": scheduling dinamico + predizione dei salti (e.g. Intel 80x86 dal Pentium).



La pipeline del PentiumPro



Il funzionamento della pipeline del PentiumPro

Fetch: Viene prelevato un blocco di memoria istruzioni di 16 byte ed inviato alla coda istruzioni.

Decodifica:

1. Le istruzioni dell'ISA INTEL vengono convertite in microistruzioni di lunghezza fissa pari a 72 bit. **Le microistruzioni sono estremamente simili all'ISA del MIPS.**
2. In questa fase vengono valutate le branch (su un certo numero di istruzioni) attraverso una ***Branch Prediction Table*** con 512 elementi**.**
3. Recupero degli operandi ed assegnazione loro dei registri replicati (rename buffer).

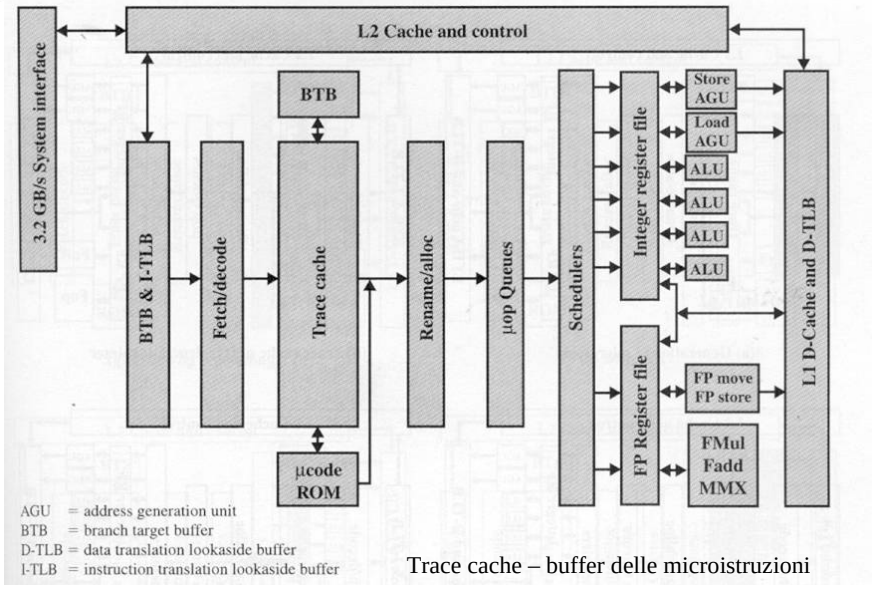
Esecuzione:

1. Smistamento. Le istruzioni vengono alla stazione di prenotazione corrispondente all'unità funzionale richiesta.
2. Un riferimento all'istruzione viene inserito nel buffer di riordino.
3. Esecuzione vera e propria (calcolo)
4. Riconsegna. **L'unità di consegna** tiene tracia di tutte le istruzioni pendenti all'interno del buffer di riordino. Essa **riordina le istruzioni** in modo che termino in modo sequenziale. Può fornire in uscita più itruzioni in un ciclo di clock.

Osservazioni sulla fase di esecuzione

- **Per considerare un'istruzione occorre che ci sia spazio nel Buffer di Riordino e nella stazione di prenotazione relativa alla pipeline di esecuzione richiesta.**
- La CPU contiene duplicazione dei registri in cui scrivere il risultato. Con cosi' tante istruzioni in esecuzione, è possibile avere il registro di destinazione occupato e occorre scrivere quindi su un registro ausiliario (**rename registers or rename buffers**). In questi registri vengono memorizzati i risultati in attesa che l'unità di consegna **dia il permesso di scrivere i registri effettivi della CPU**.
- L'istruzione viene eseguita quando il corrispondente elemento all'interno della stazione di prenotazione possiede tutti gli operandi, e la relativa unità funzionale è libera.
- Un'istruzione non viene terminata (riconsegnata) fino a quando non lo decide l'unità di consegna. Se la predizione di un salto è scorretta, vengono scartate le istruzioni sbagliate dalle stazioni di prenotazione e dai buffer di riordino e liberati i registri interni.

LA CPU del Pentium 4



Trace cache – buffer delle microistruzioni

Funzionamento della Pipeline del Pentium 4

Fetch delle istruzioni della Memoria (Cache)

Ciascuna istruzione viene tradotta in una o più microistruzioni di lunghezza fissa.

Il processore esegue le micro-istruzioni in una pipeline superscalare a schedulazione dinamica.

Il processore restituisce il risultato (finale) dell'esecuzione delle micro-istruzioni relative alla stessa istruzione ai registri nell'ordine nel quale le istruzioni sono state inviate in esecuzione.

Il Pentium 4, come i suoi predecessori, trasforma un'ISA CISC in un'ISA interno RISC costituito dalle micro-operazioni.

Esecuzione parallela di istruzioni, con parallelismo esplicito: si può specificare in assembly se istruzioni possono essere eseguite contemporaneamente o sequenzialmente (estensione Processor del compilatore Microsoft Visual C++) o librerie dell'Intel.

Stadi della pipeline del Pentium 4: Front end

Le istruzioni passano dalla cache primaria al buffer L2 (64 byte). A questo stadio vengono gestite le miss ed i trasferimenti da cache.

In parallelo i dati vengono caricati da cache nel buffer dati L1.

Nel trasferimento da L1 ed L2 agiscono Branch Target Buffer e lookaside buffer che gestiscono le predizioni sulle branch.

A questo punto inizia la fase Fetch e Decodifica che legge il umero di byte pari alla lunghezza dell'istruzione.

Il decoder traduce l'istruzione in da 1 a 4 micro-operazioni: 118bit, appartenenti ad un'ISA RISC.

Stadi della pipeline del Pentium 4: Trace cache

Riordina localmente il codice in modo da evitare le criticità: la pipeline ha 20 stadi.

Invia alla fase di esecuzione (Allocate e Renaming).

Per istruzioni complesse (> 4 microistruzioni) vengono create in questa fase con una macchina a stati finiti implementata con una ROM + memoria.

Stadi della pipeline del Pentium 4: Esecuzione

Elemento centrale è il **Reorder Buffer** – **ROB** (Buffer circolare che può contenere fino a 126 micro-istruzioni).

Contiene le micro-istruzioni con il loro stato, la presenza o assenza dei dati, per tutta la durata dell'esecuzione.

Alloca i registri (traducendo i registri simbolici dell'Assembly in uno dei 128 registri di pipeline).

Avvia alla coda di esecuzione l'istruzione (coda per le lw/sw e coda per le altre istruzioni).

Lo scheduling avviene prendendo la prima istruzione che può essere eseguita nella coda. Dagli scheduler si può passare alle ALU in modi diversi.

La fase di esecuzione scrive poi i risultati nei registri o nella cache L1.

C'è una parte dedicata ai flag che vengono poi inviati alla BTB per predirre correttamente i salti.

Sommario

Predizione dinamica dei salti
Pipeline avanzate
Multiple-Issue Statici
Multiple-Issue Dinamici
La pipeline del Pentium IV