


Trend di sviluppo delle pipeline

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

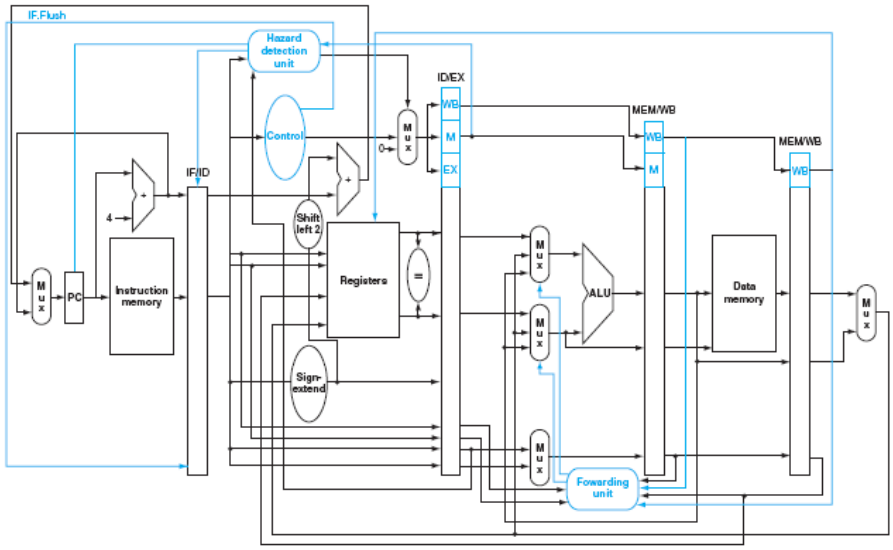
Università degli Studi di Milano

Patterson 4.10, 4.11


Sommario

Superpipeline
Multiple-Issue Statici
Multiple-Issue Dinamici
Alcune pipeline

CPU con pipeline completa della gestione degli hazard



Pipeline avanzate: esecuzione parallela

Instruction level parallelism (ILP), parallelismo implicito.

Come si può aumentare?

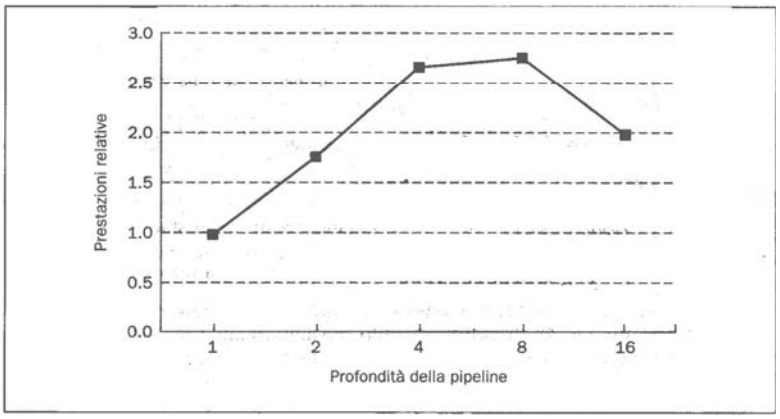
- Superpipelining (pipeline più lunga).
- "Multiple-issue" (esecuzione parallela di un issue packet).
  Static multiple issues (schedulazione decisa dal compilatore)
  Dynamic multiple issues (schedulazione decisa run-time dalla CPU).

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

Speculazione e 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.

Sommario

Superpipeline
Multiple-Issue Statici
Multiple-Issue Dinamici
Alcune pipeline

Principi delle pipeline Multiple issues

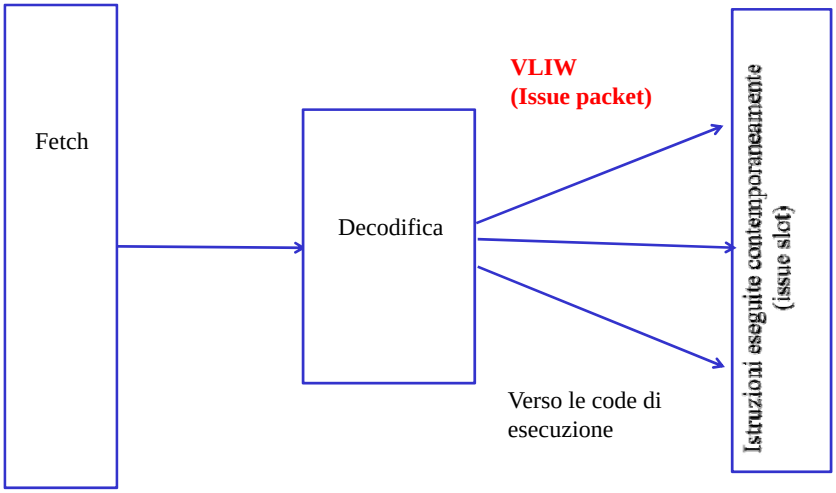
Duplicazione delle unità funzionali (e.g. ALU e ALUfp) ed eseguo più istruzioni in parallelo.

Aumento del numero di istruzioni per ciclo di clock (CPI < 1) => issue packet
Esempio:

Processore a 4 vie da 6Ghz potrebbe eseguire 24 miliardi di istruzioni / s. Oggi si arriva anche a 8 istruzioni / ciclo di clock (CPI_{ideale} = 0,125).

Esecuzione fuori-ordine e riordinamento dei risultati forniti dalla Pipeline.

Struttura Pipeline Multiple Issue statico



Esistono dei vincoli su quali istruzioni inserire nello stesso issue packet (VLIW)
L'ordine di esecuzione viene deciso dal compilatore.
Primi progetti: Itanium (2000), Itanium-2 (2002) by Intel.

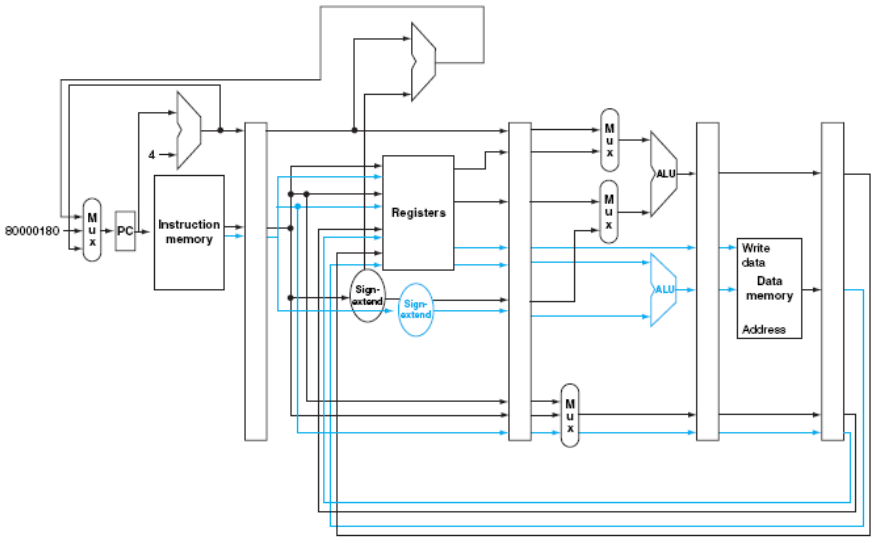
Schedulazione delle istruzioni in MIPS a 2 Vie (MIPS64)

La prima via è riservata alle operazioni aritmetico-logiche o di branch
La seconda via è riservata alle operazioni di load / store
(load / store architecture)

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

CPI = 0,5 – 2 istruzioni per ogni ciclo di clock. E' vero?

Multiple-issue statici: il MIPS64



Problemi nella creazione dello issue packet

Hazard sul controllo. Due istruzioni vengono eliminate dalla pipeline.

Hazard sui dati oltre a quelli insiti nel codice sequenziale, si generano **hazard dovuti alla parallelizzazione del codice**.

add $t0, $t1, $t2 — Non provoca stallo su una pipeline single issue
lw $s0, 40($t0) — stallo di 1 istruzione nella pipeline a due vie:

	Q1	Q2
è errata:	add $t0, $t1, $t2	lw $s0, 40($t0)
è corretta:	add $t0, $t1, $t2	nop
		lw $s0, 40($t0)

Hazard sui dati (stall on load).

lw $s0, 40($t0) — Non provoca stallo su una pipeline single issue
add $t0, $t1, $s0 — stallo nella pipeline a due vie:

	Q1	Q2
è errata:	add $t0, $t1, $s0	lw $s0, 40($t0)
è corretta:	nop	lw $s0, 40($t0)
	nop	
	add $t0, $t1, $s0	

Esempio

```
Ciclo: lw   $t0, 0($s1)
       addu $t0, $t0, $t2
       sw   $t0, 4($t0)
       addi $s1, $s1, -4
       bne  $s1, $zero, Ciclo
       or   $s6, $s7, $s5
```

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
	or $s6, $s7, $s5		5

CPI = ?

La speculazione

Ricorso massiccio alla **speculazione** (= immaginare degli scenari). La speculazione richiede **l'analisi del valore assunto run-time dai registri**.

La speculazione consente di spostare le istruzioni all'interno del codice per rimuovere l'hazard.

Esempio: si può speculare sul risultato di una branch, e quindi inserire nel branch delay slot l'istruzione opportuna.

Esempio: si può speculare che una sw non abbia come indirizzo di memoria lo stesso della lw successiva. In questo caso, si possono scambiare le due istruzioni.

sw $t0, 4($s0)

lw $t1, 16($t0)

Solo se gli indirizzi di memoria sono indipendenti, posso scambiare l'ordine di lettura e scrittura dalla memoria (questo anche se i due registri base sono diversi).

Come viene corretta la speculazione nelle multiple-issue statiche

La speculazione può essere fatta sia dal compilatore che dal processore (mediante logica di controllo).

E se la speculazione risulta sbagliata? Deve esistere un meccanismo di correzione (roll-back). La speculazione si paga in termini di meccanismi per **controllare** se la speculazione è stata corretta e di **correggerla**.

Nelle multiple-issue statiche, il compilatore inserisce delle istruzioni di controllo e di correzione a speculazioni errate, anche chiamando procedure opportune che correggono quanto fatto (e.g. Procedure che eseguono le operazioni inverse erroneamente eseguite).

Occorre speculare quando si hanno degli elementi validi, altrimenti si possono inserire problemi (vedi eccezioni generate dall'esecuzione di un'istruzione sbagliata o con dati sbagliati, eccezioni "speculative") che rendono il funzionamento meno efficiente.

Sommario

Superpipeline
Multiple-Issue Statici
Multiple-Issue Dinamici
Alcune pipeline

Dynamic multiple issues

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.

E' la CPU che garantisce la correttezza dell'esecuzione.

Confronto tra Multiple-issue statici e dinamici

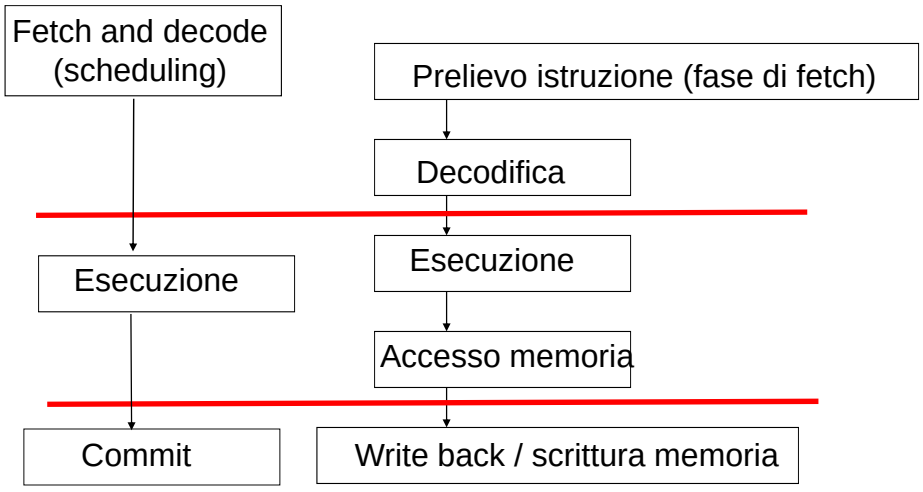
L'hardware di una pipe-line superscalare garantisce la correttezza del codice.

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

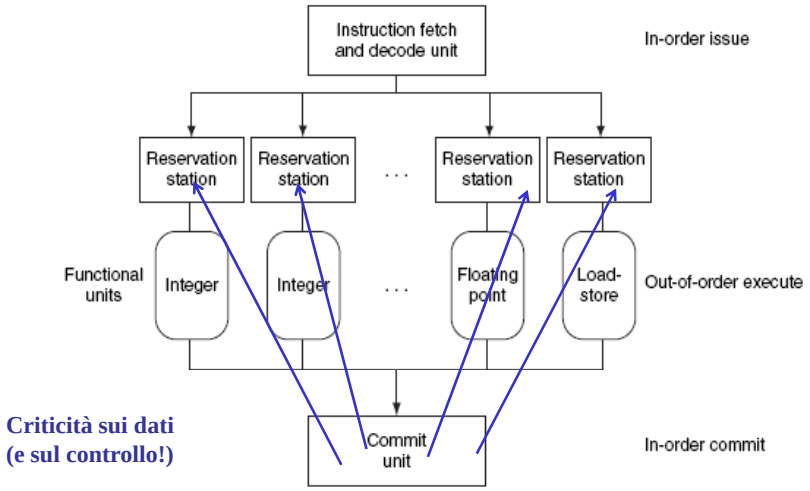
Nelle multiple-issue statiche, spesso occorre ricompilare passando da una CPU ad un'altra oppure il codice può girare con prestazioni molto scadenti. Nelle multiple-issue dinamiche, ciò non è necessario.

La speculazione può essere fatta sia dal compilatore (multiple-issue statici) che dal processore (multiple-issue dinamici).

Ciclo di esecuzione di un'istruzione MIPS



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.

Register renaming e roll-back

Il compilatore può utilizzare la speculazione per riordinare le istruzioni, decidere il nome dei registri della pipeline da associare ai registri del register file (visibili al programmatore).

Register renaming. Posso associare più registri interni ad uno stesso registro dell'architettura in modo da evitare hazard. Esempio:

```
lw $t0, 20($t1)
add $s0, $t0, $t1        hazard che si crea con la parallelizzazione del codice
lw $t0, 24($t1)
add $s1, $t0, $t2
```

Questa dipendenza non ci sarebbe se allocassi alla seconda coppia di istruzioni un registro diverso.

Codice all'interno della pipeline dopo il renaming:

```
lw $t01, 20($t1)
add $s0, $t01, $t1
lw $t02, 24($t1)
add $s1, $t02, $t2
```


Renaming e hazard

- La CPU mette in buffer i risultati dell'esecuzione fino a quando non si è potuto verificare la correttezza della speculazione (esecuzione condizionata).
- Nel caso di speculazione errata, la cancellazione del lavoro fatto viene ottenuta semplicemente svuotando i buffer e correggendo la sequenza di istruzioni (meccanismo di **roll-back**).
- Nel caso in cui l'esecuzione sia corretta, il risultato viene copiato in memoria dati e/o nel register file. Nell'esempio precedente: $t0 = $t02, viene copiato il valore più recente di $t0.
- Il register renaming può essere utilizzato anche per la gestione degli hazard => invece di correggere il register file è sufficiente cancellare l'associazione registro interno – registro del register file.

Sommario

Superpipeline
Multiple-Issue Statici
Multiple-Issue Dinamici
Alcune pipeline

Tendenze di sviluppo delle pipeline

- Difficilmente si ha una issue packet di più di 2 istruzioni
  - Le dipendenze sono difficili da eliminare.
  - Alimentazione da parte del sistema di memoria.
  - Il costo della speculazione è cresciuto fino alla barriera dell'energia => multi-core.

Microprocessor	Year	Clock Rate	Pipeline Stages	Issue Width	Out-of-Order/ Speculation	Cores/ Chip	Power
Intel 486	1989	25 MHz	5	1	No	1	5 W
Intel Pentium	1993	66 MHz	5	2	No	1	10 W
Intel Pentium Pro	1997	200 MHz	10	3	Yes	1	29 W
Intel Pentium 4 Willamette	2001	2000 MHz	22	3	Yes	1	75 W
Intel Pentium 4 Prescott	2004	3600 MHz	31	3	Yes	1	103 W
Intel Core	2006	2930 MHz	14	4	Yes	2	75 W
Sun UltraSPARC III	2003	1950 MHz	14	4	No	1	90 W
Sun UltraSPARC T1 (Niagara)	2005	1200 MHz	6	1	No	8	70 W

Altre pipeline

Questi processori sono detti anche **superscalari**.

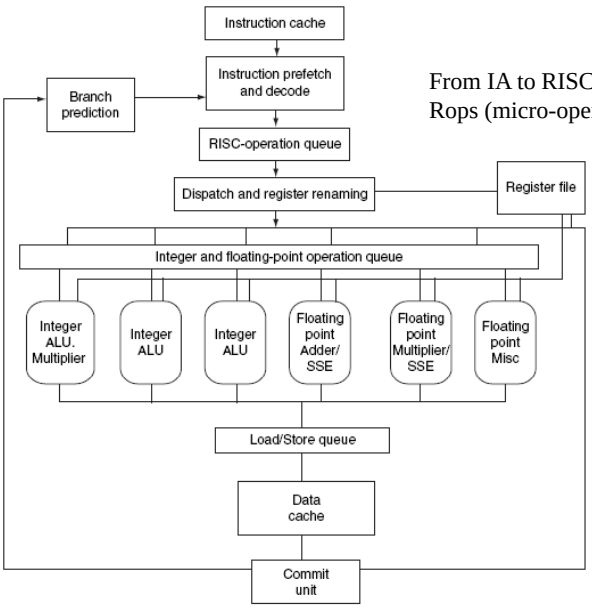
La scelta di quali istruzioni inviare alla pipe-line viene eseguita durante l'esecuzione stessa. Dipende dalla compatibilità tra le varie istruzioni e da eventuali hazard su dati e controllo.

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.

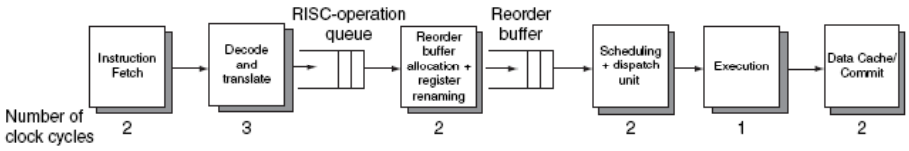
E' la CPU che garantisce la correttezza dell'esecuzione.

La pipeline AMD Opteron X4 (Barcelona)



From IA to RISC –
Rops (micro-operazioni)

Dettagli sulla pipeline



3 istruzioni per ciclo di clock
Microarchitettura (memoria, register file, issue packet, controllo)
Register renaming

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.

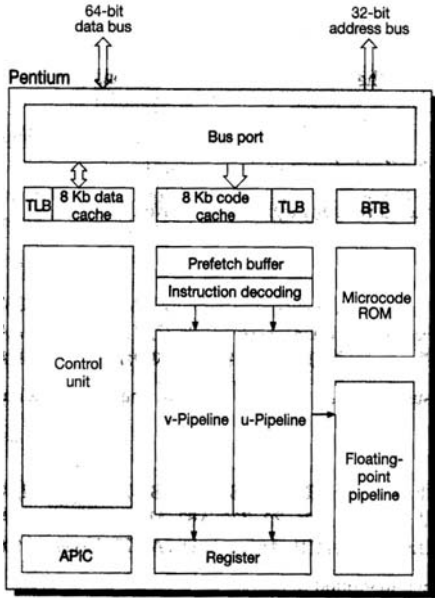
All'interno della pipeline, abbiamo microistruzioni di ampiezza pari a 70 bit (fissa), RISC.

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.

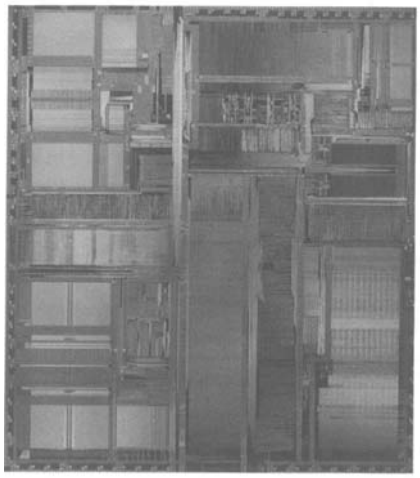
Trasformo le istruzioni dell'ISA Intel in microistruzioni di lunghezza uguale.

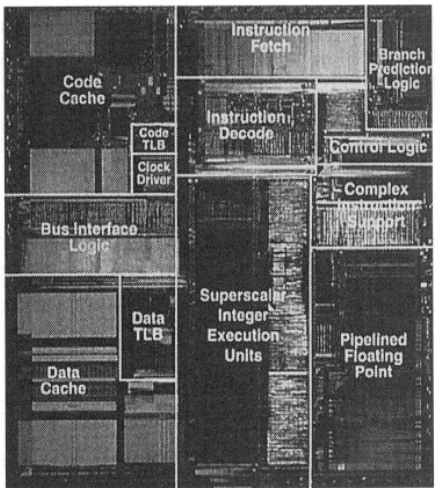
Pentium – struttura interna

- 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.



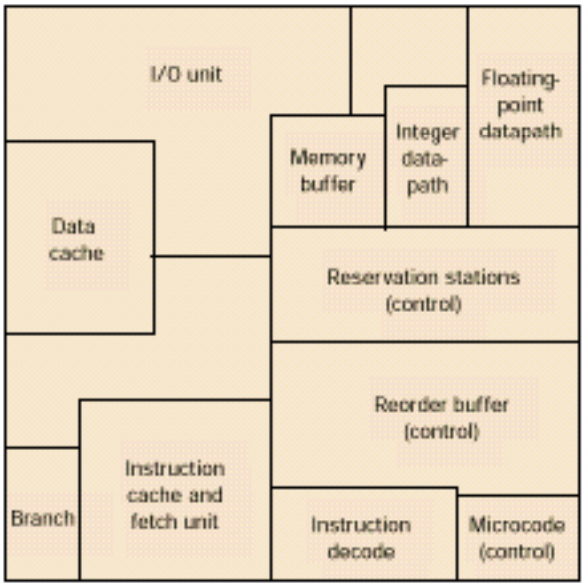
Il pentium



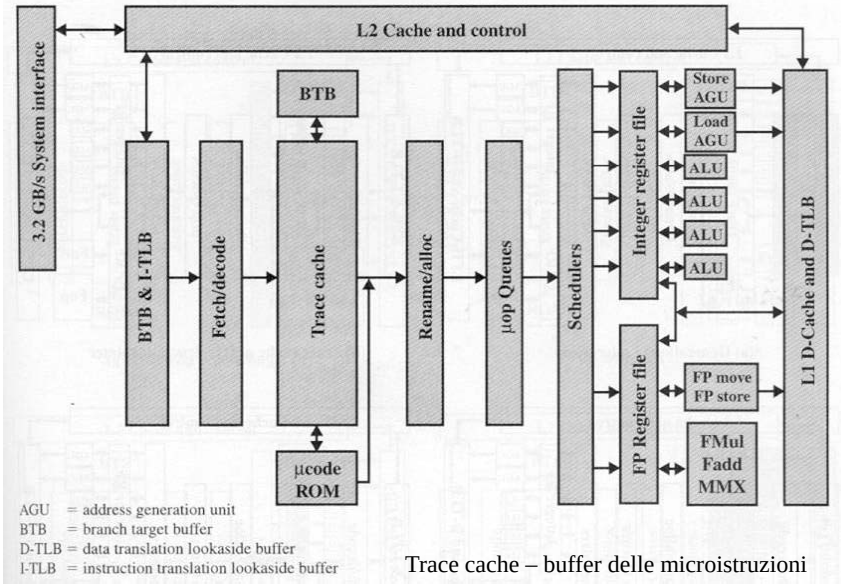


Il processore Pentium Pro

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



LA CPU del Pentium 4



Trace cache – buffer delle microistruzioni

Sommario

Superpipeline
Speculazione
Multiple-Issue Statici
Multiple-Issue Dinamici
Alcune pipeline