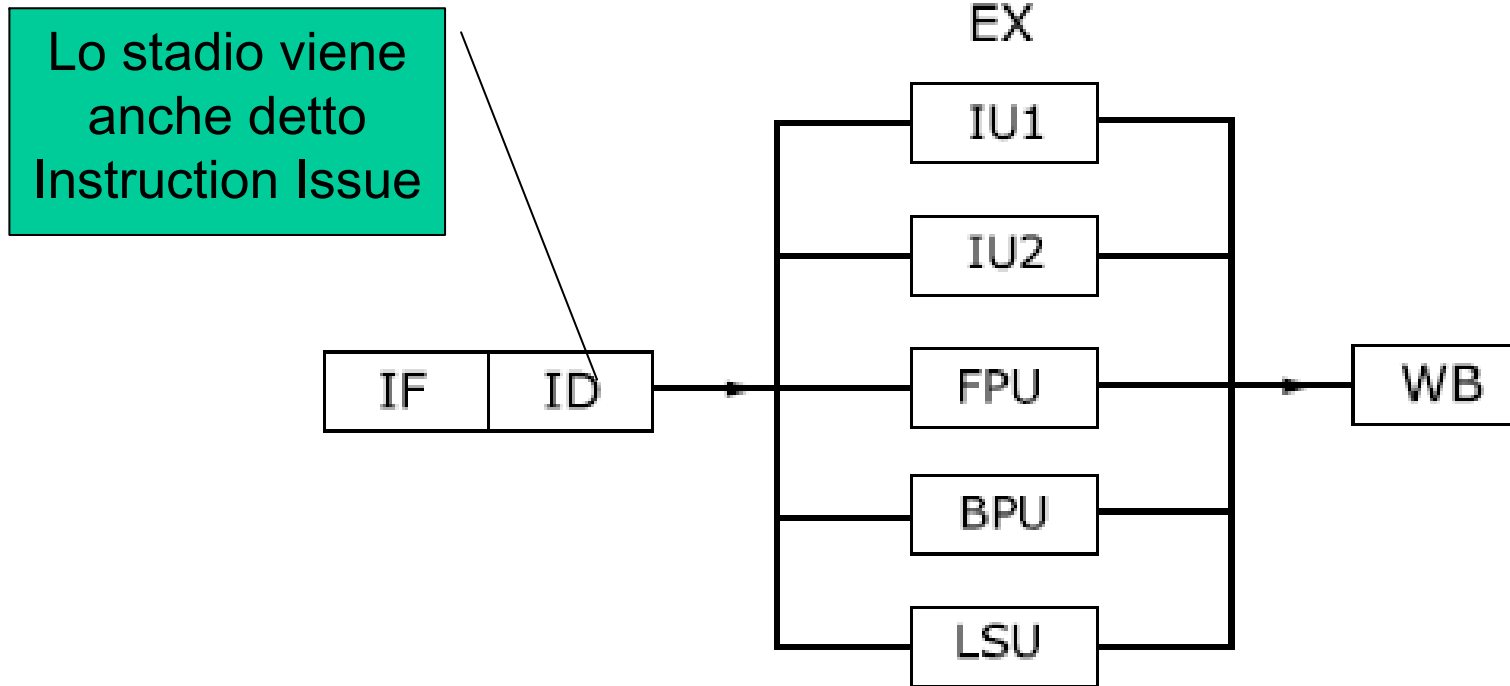


Sistemi superscalari

- Contenuto della lezione
 - Unità multifunzionali parallele
 - Architetture superscalari
 - Esecuzione fuori ordine

Architettura superscalare



Ipotesi sui CPI delle unità

Classe	UF	C_{PI}
• add/sub	UI1	1
• mul	UI2	2
• addf/mulf	FPU	4

- Ipotizzeremo che venga **emessa** verso lo stadio EX una istruzione alla volta
 - ➔ potrà **al massimo** essere ritirata una istruzione alla volta
 - ➔ per ora di superscalare c'è solo il parallelismo tra le UF; non c'è l'esecuzione di più di un'istruzione alla volta!!

Emissione in ordine

- Determina il completamento fuori ordine. Esempio:

1	(100)	mul	f1,f7,f8	4 cicli in FPU
2	(104)	add	r2,r4,r5	1 ciclo in IU1 
3	(108)	mul	r3,r10,r11	2 cicli in IU2
4	(112)	addf	f4, f6,f6	4 cicli in FPU

- Profilo di esecuzione

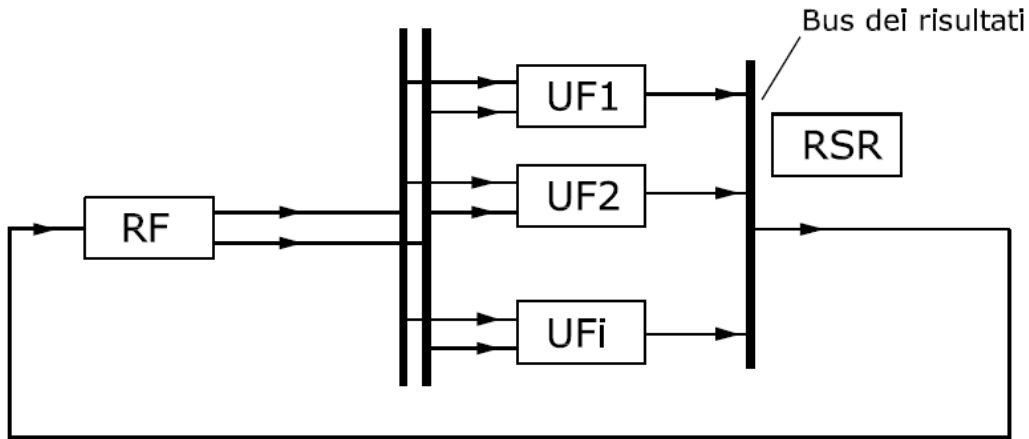


Tre metodi

- Lo stato di macchina deve corrispondere all'ordine del programma
- Completamento in ordine:
si forzano le istruzioni ad essere completate seguendo l'ordine architetturale
- Buffer di Riordinamento:
le istruzioni sono completate fuori ordine ma sono forzate a modificare lo stato della macchina seguendo l'ordine architetturale
- History Buffer:
le istruzioni possono aggiornare lo stato in qualsiasi ordine, conservando però la possibilità di ripristinare uno stato coerente in presenza di conflitti.

Modello

RSR: Reservation Shift Register



Formato

mul entra qui

	UF	Rd	V	PC
1				
2	mul	r3	1	108
3				
4				

Uso campi RSR

UF	Rd	V	PC
----	----	---	----

V: Bit di validità; dice se la posizione contiene informazioni o se è da ritenersi vuota.

PC: PC dell'istruzione; necessario per ripristinare uno stato coerente in caso di predizione di salto errata.

UF: al completamento di una istruzione (uscita da RSR) serve a individuare da quale UF deve essere preso il risultato da trasmettere a Rd.

Rd: individua il registro di destinazione del risultato

➔ Non c'è bisogno di tenere traccia di Rd nelle (pipeline delle) UF.

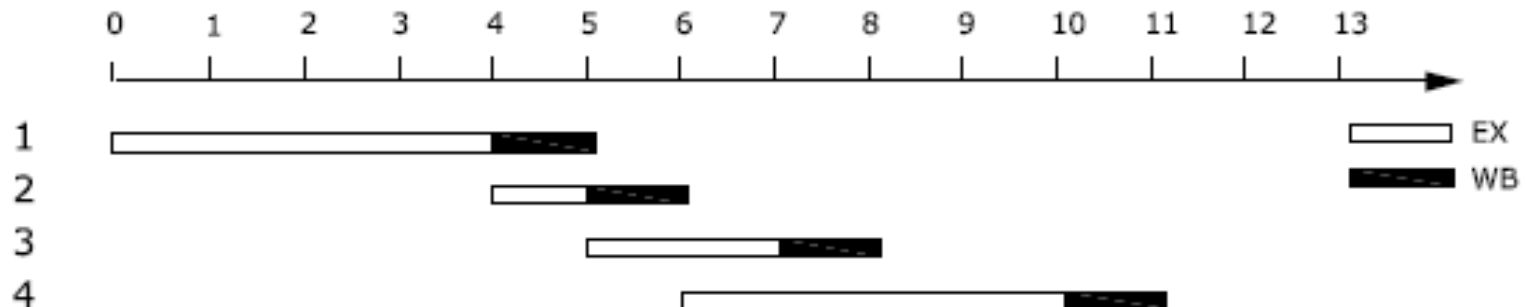
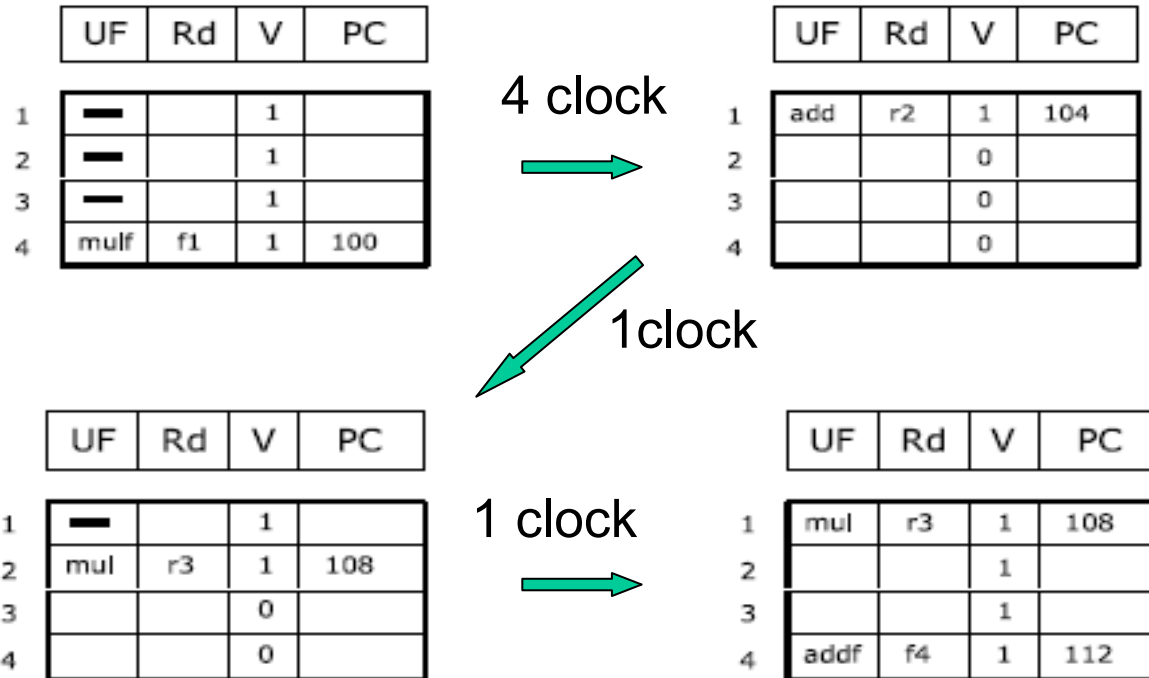
Completamento in ordine

- Si ottiene attraverso la prenotazione di RSR in modo forzare la conclusione secondo l'ordine del programma.
- Due aspetti:
 - Completamento in ordine rispetto ai registri
 - Completamento in ordine rispetto alla memoria
- Nel completamento in ordine rispetto ai registri si tratta di prenotare il bus dei risultati in modo che le istruzioni *non passino avanti*

Rispetto ai registri

- L'istruzione che occupa la posizione j in RSR occupa anche tutte le precedenti
 - Per occupare le posizioni si usa V
 - Nelle posizioni occupate che non contengono una effettiva istruzione si usa *dummy* per in campo UF.
- L'istruzione che trova la posizione occupata attende che si liberi (ovvero che le istruzioni che la precedevano abbiano superato la posizione)

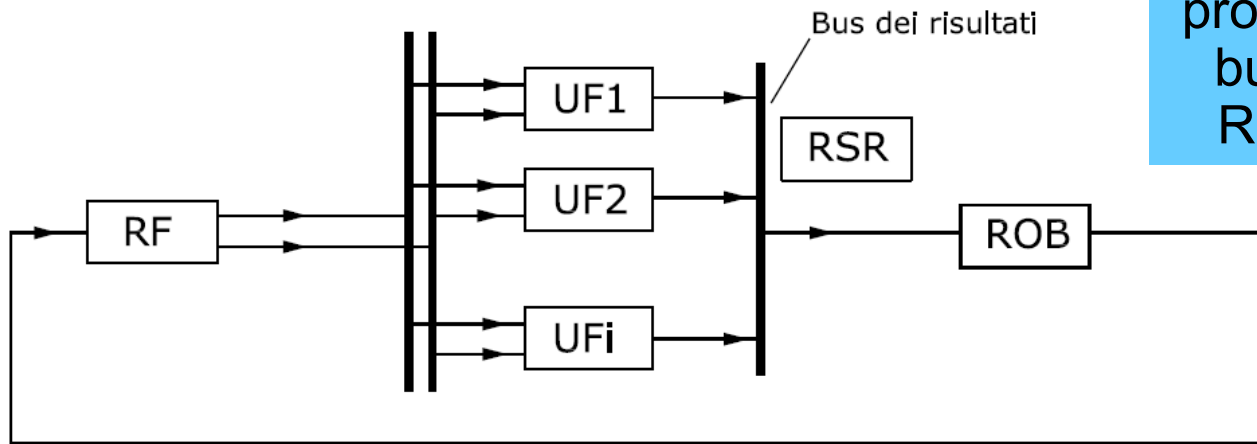
Sequenza e profilo di esecuzione



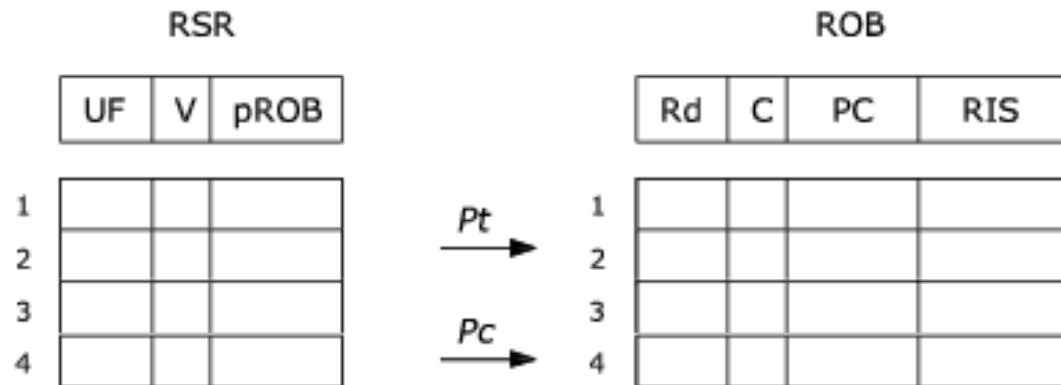
...Rispetto alla memoria (Store)

- Due metodi (equivalenti):
 - non viene emesso alcun comando di memorizzazione (store) prima che le istruzioni precedentemente emesse (tra cui eventuali istruzioni di diramazione) siano state completate in modo corretto: l'istruzione ST resta nel registro di emissione fino a quando RSR non è vuoto.
 - La memoria è considerata come un'unità funzionale. Analogamente a ogni altra istruzione, store occupa una posizione in RSR in modo che questa raggiunga la cima quando tutte le istruzioni emesse in precedenza sono state completate. Al raggiungimento della cima di RSR, l'unità load/store viene comandata all'esecuzione della memorizzazione richiesta.

Buffer di riordinamento (ROB)



Se non ci fosse il problema dell'accesso al bus basterebbe solo il ROB a tenere l'ordine



Sequenza

RSR

UF	V	pROB
----	---	------

1		0	
2		0	
3		0	
4	mulf	1	3

a)



ROB

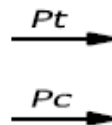
Rd	C	PC	RIS
----	---	----	-----

1			
2			
3	f1	0	100
4			
5			

UF	V	pROB
----	---	------

1	add	1	4
2		0	
3	mulf	1	3
4		0	

b)



Rd	C	PC	RIS
----	---	----	-----

1			
2			
3	f1	0	100
4	r2	0	104
5			

...Sequenza

c)

	UF	V	pROB
1		0	
2	mulr	1	3
3		0	
4		0	

P_t →

P_C →

	Rd	C	PC	RIS
1				
2				
3	r1	0	100	
4	r2	1	104	2222
5				

d)

	UF	V	pROB
1	mulr	1	3
2	mul	1	5
3		0	
4		0	

P_C →

P_t →

	Rd	C	PC	RIS
1				
2				
3	r1	0	100	
4	r2	1	104	2222
5	r3	0	108	

Sequenza

e)

	UF	V	pROB
1	mul	1	5
2		0	
3		0	
4	addf	1	1

$\xrightarrow{Pc}$
 $\xrightarrow{Pt}$

	Rd	C	PC	RIS
1	f4	0	112	
2				
3	f1	1	100	111.1
4	r2	1	104	2222
5	r3	0	108	

f)

	UF	V	pROB
1		0	
2		0	
3	addf	1	1
4		0	

$\xrightarrow{Pc}$
 $\xrightarrow{Pt}$

	Rd	C	PC	RIS
1	f4	0	112	
2				
3				
4	r2	1	104	2222
5	r3	1	108	3333

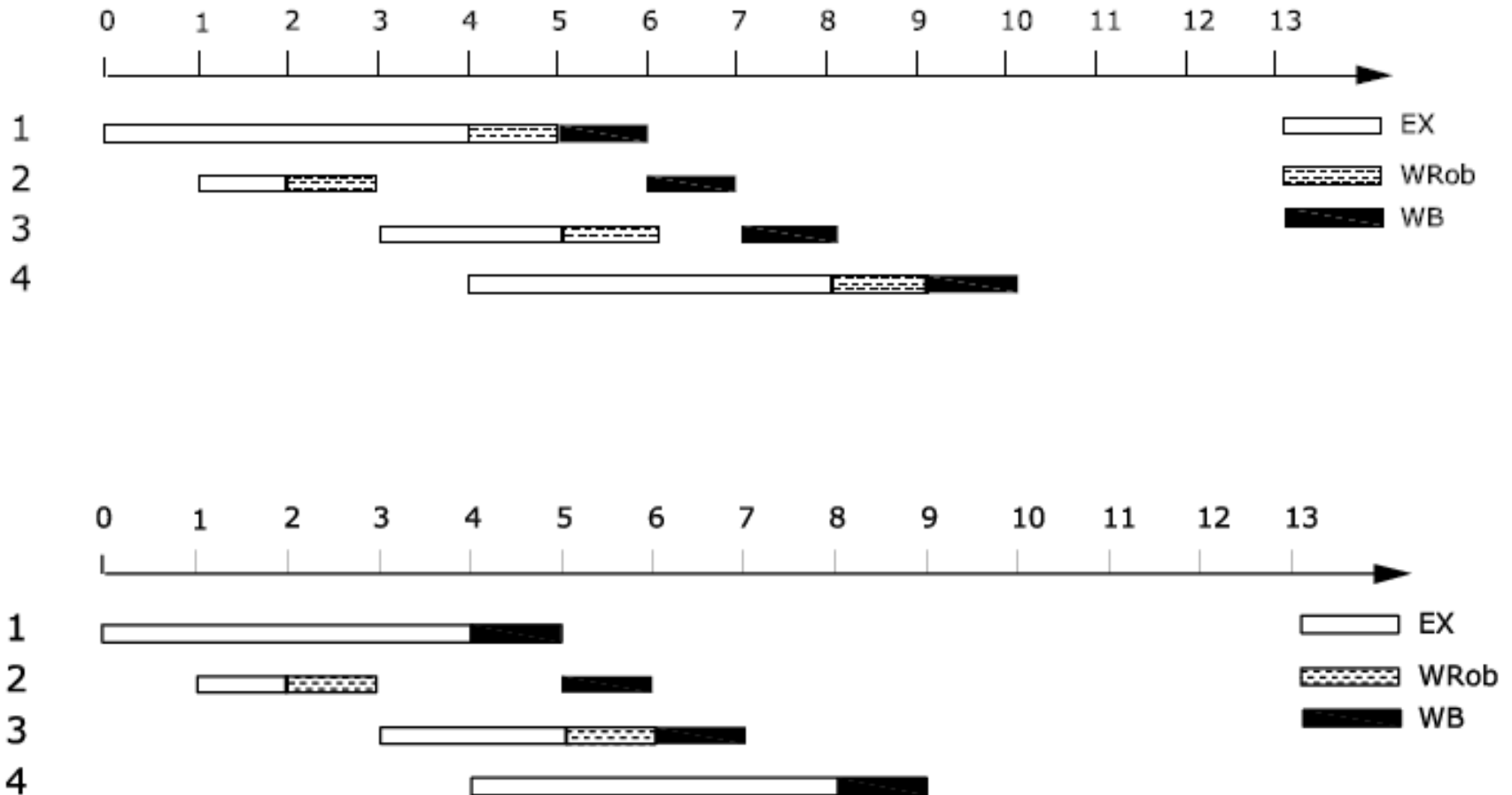
g)

	UF	V	pROB
1		0	
2	addf	1	1
3		0	
4		0	

$\xrightarrow{Pc}$
 $\xrightarrow{Pt}$

	Rd	C	PC	RIS
1	f4	0	112	
2				
3				
4				
5	r3	1	108	3333

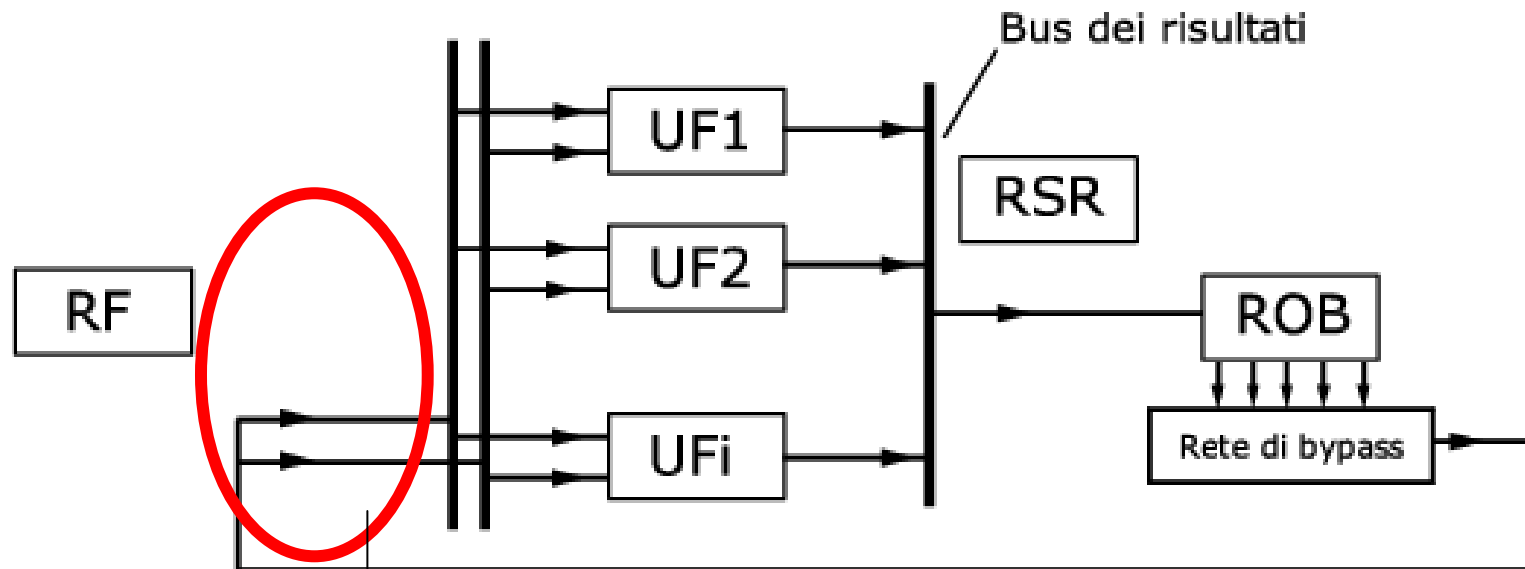
Profili di esecuzione con il ROB



Nel libro c'è un errore nella caption

Bypass

Per anticipare i risultati ancora non consegnati



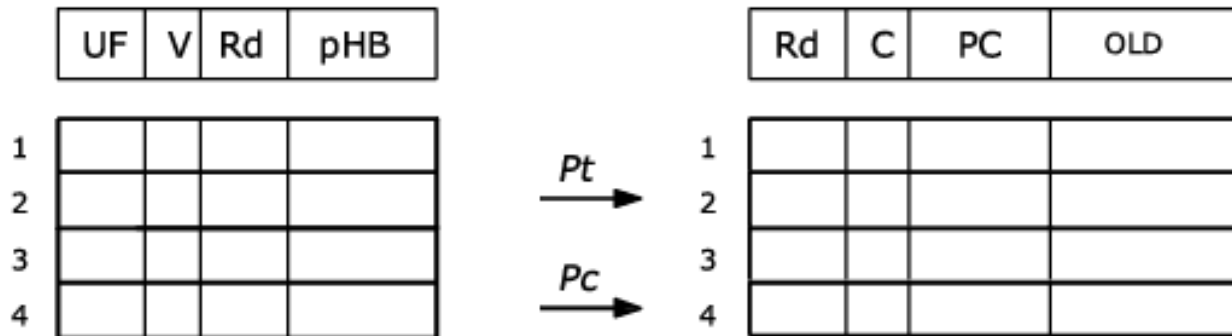
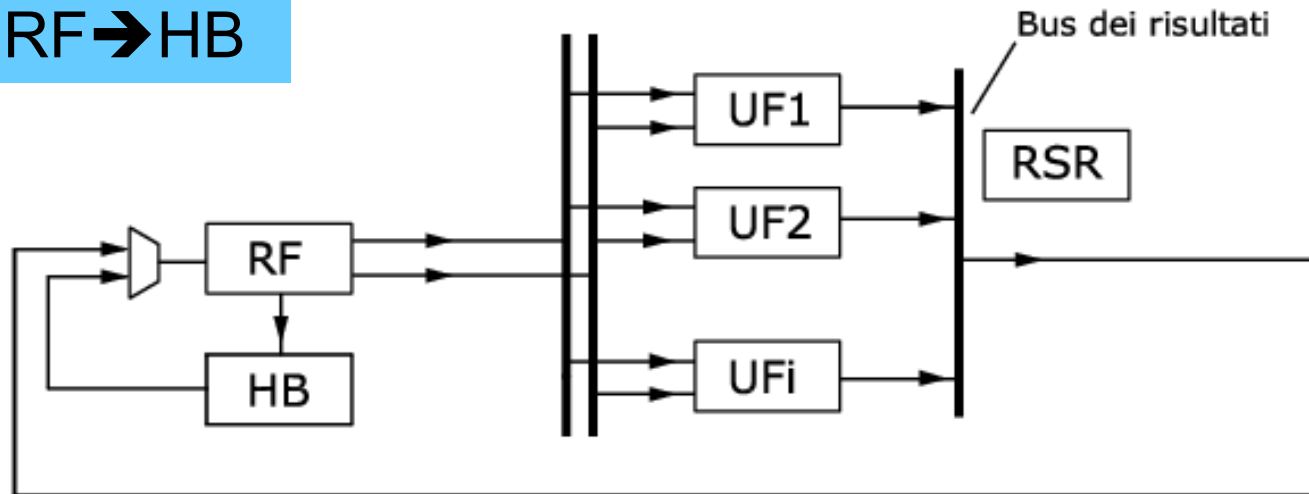
Ovviamente qui ci
va un MUX

History buffer (HB)

- L'istruzione completata dalla sua UF scrive immediatamente i risultati in RF, ma il vecchio valore contenuto in RF resta in HB fino a che l'istruzione non emerge
 - L'ordine di emissione deve garantire coerenza rispetto alle dipendenze dati
 - I motivi per ripristinare lo stato corrente sono legati alle interruzioni e alle previsioni di salto errate

History Buffer (HB)

Ci vuole un
percorso RF → HB



Gestione conflitti di controllo

- Predizione errata con esecuzione fuori ordine:
 - istruzioni provenienti dal percorso errato possono essere state completate prima della soluzione della diramazione.
 - In tal caso, lo stato della macchina è modificato ed occorre ripristinare lo stato che si aveva prima dell'istruzione di diramazione
- Nota:
 - Nel caso della pipeline lineare bastava svuotarla delle istruzioni prese dal percorso errato.

Ripristino stato coerente (con ROB)

- Si associa alle istruzioni il bit EPR (errata predizione, che ha significato per le sole diramazioni)
- Quando l'istruzione emerge dalla testa del ROB se EPR è asserito si svuota la pipeline (il ROB) di tutte le istruzioni sul percorso errato (che vengono dietro nel ROB) e si inizia a ricaricare dalla effettiva destinazione.
 - Equivale a non far scrivere nei registri e in memoria alle istruzioni che risultano sul cammino mal predetto

Ripristino stato coerente (con HB)

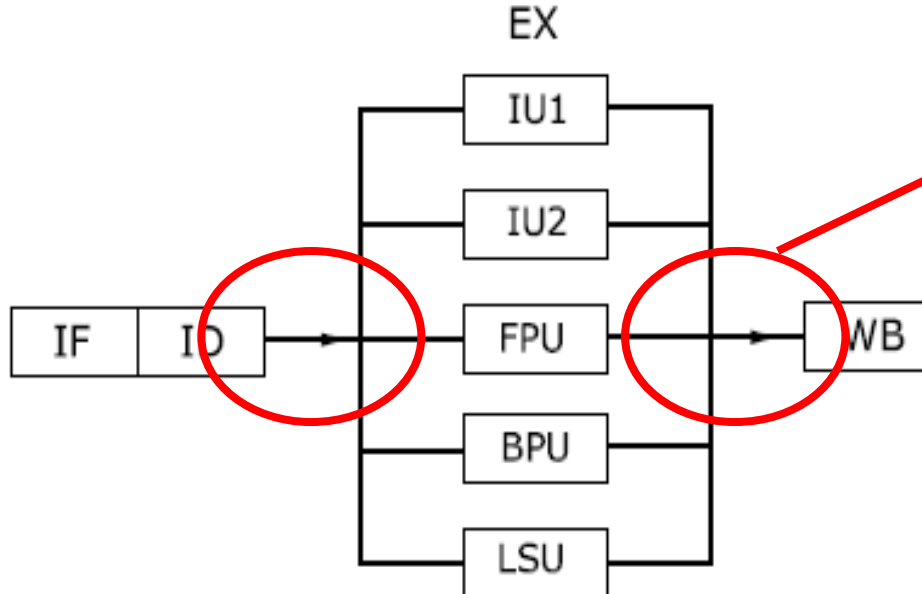
- Si aggiunge EPR. Quando un'istruzione di diramazione arriva in testa ad HB, se EPR è asserito:
 - Viene bloccato HB e viene bloccata l'emissione di nuove istruzioni da parte dell'Issue Register
 - Si attende che le operazioni ancora attive vengano completate (cioè che si svuoti la pipeline)
 - Partendo dalla coda verso la testa, vengono cancellati gli elementi in HB e, contemporaneamente, vengono riscritti nei registri di destinazione i vecchi valori, contenuti nel buffer, fino ad arrivare alla prima istruzione sul percorso sbagliato
 - L'esecuzione del programma riprende prelevando l'istruzione proveniente dal percorso corretto.

Superscalari?

- Abbiamo assunto l'emissione e il ritiro di una istruzione alla volta
- Se viene emessa una sola istruzione alla volta non accadrà mai che vengano eseguite più di una istruzione per clock
 - Decodificare ed emettere più istruzioni in parallelo
 - Ritirare più istruzioni in parallelo

Come fare con il ROB

- Le macchine correnti leggono dalla cache linee che possono contenere più parole
 - Per esempio linee di 4 parole (128 bit)

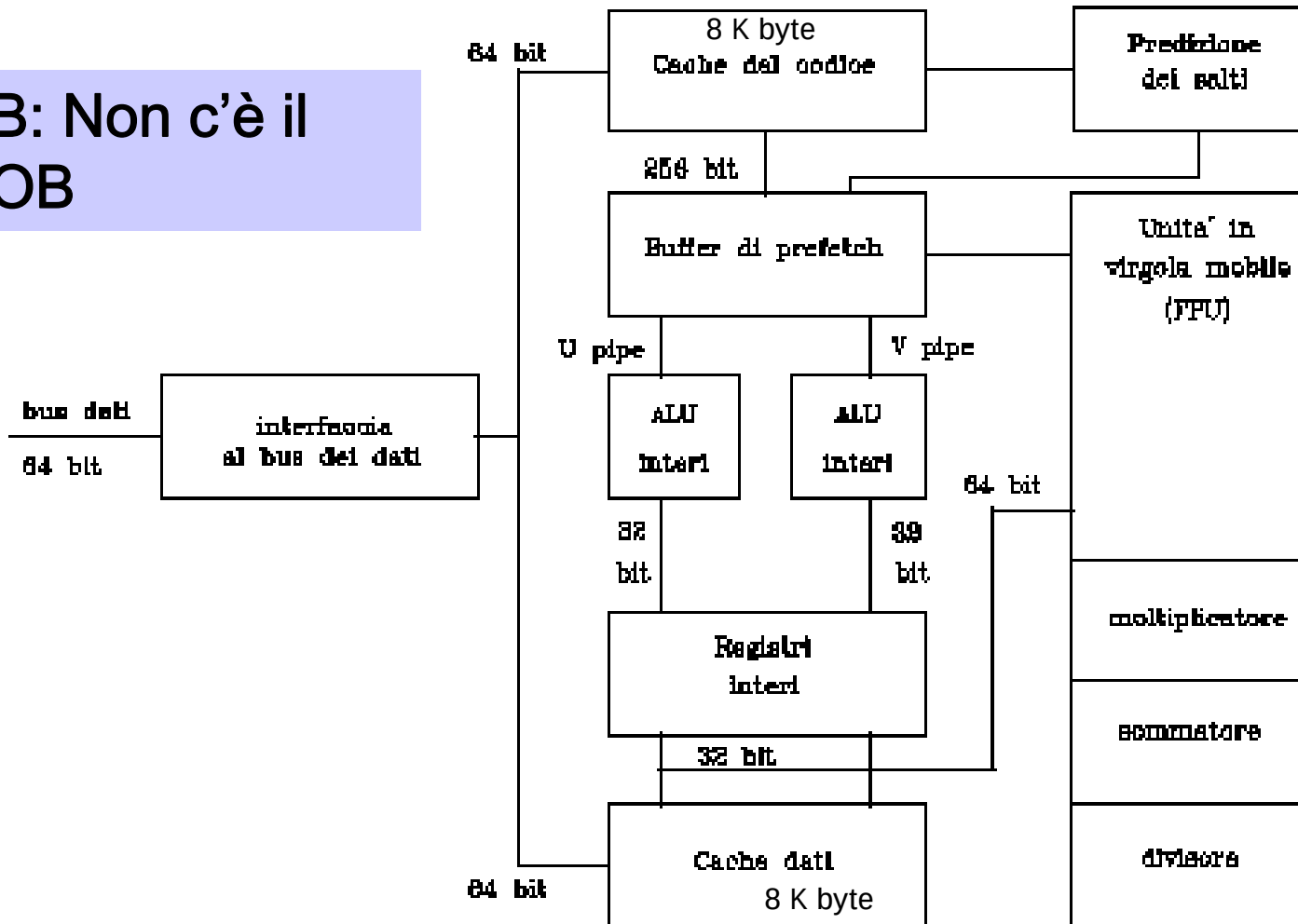


Qui c'è il ROB

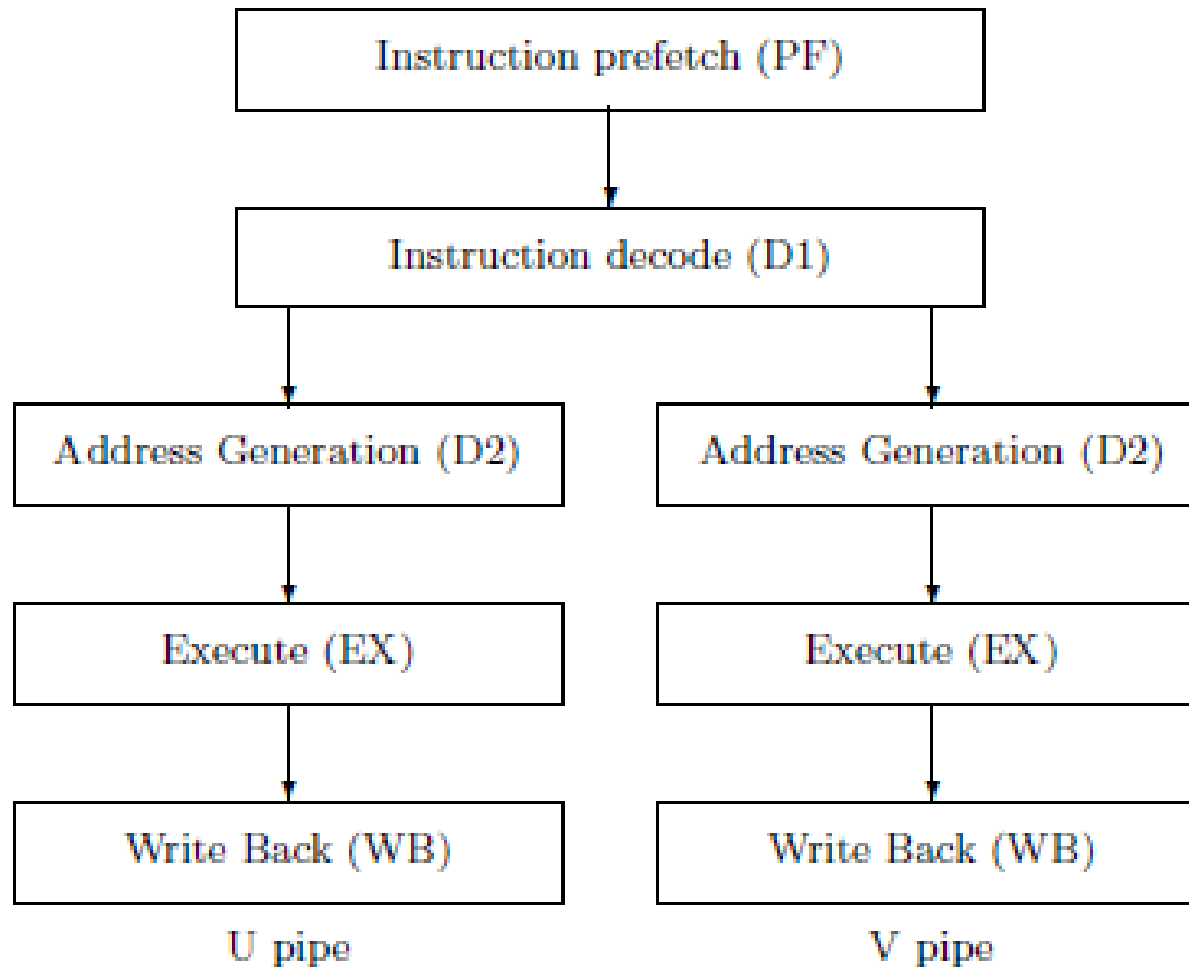
Esempio: il Pentium

Due pipe in parallelo

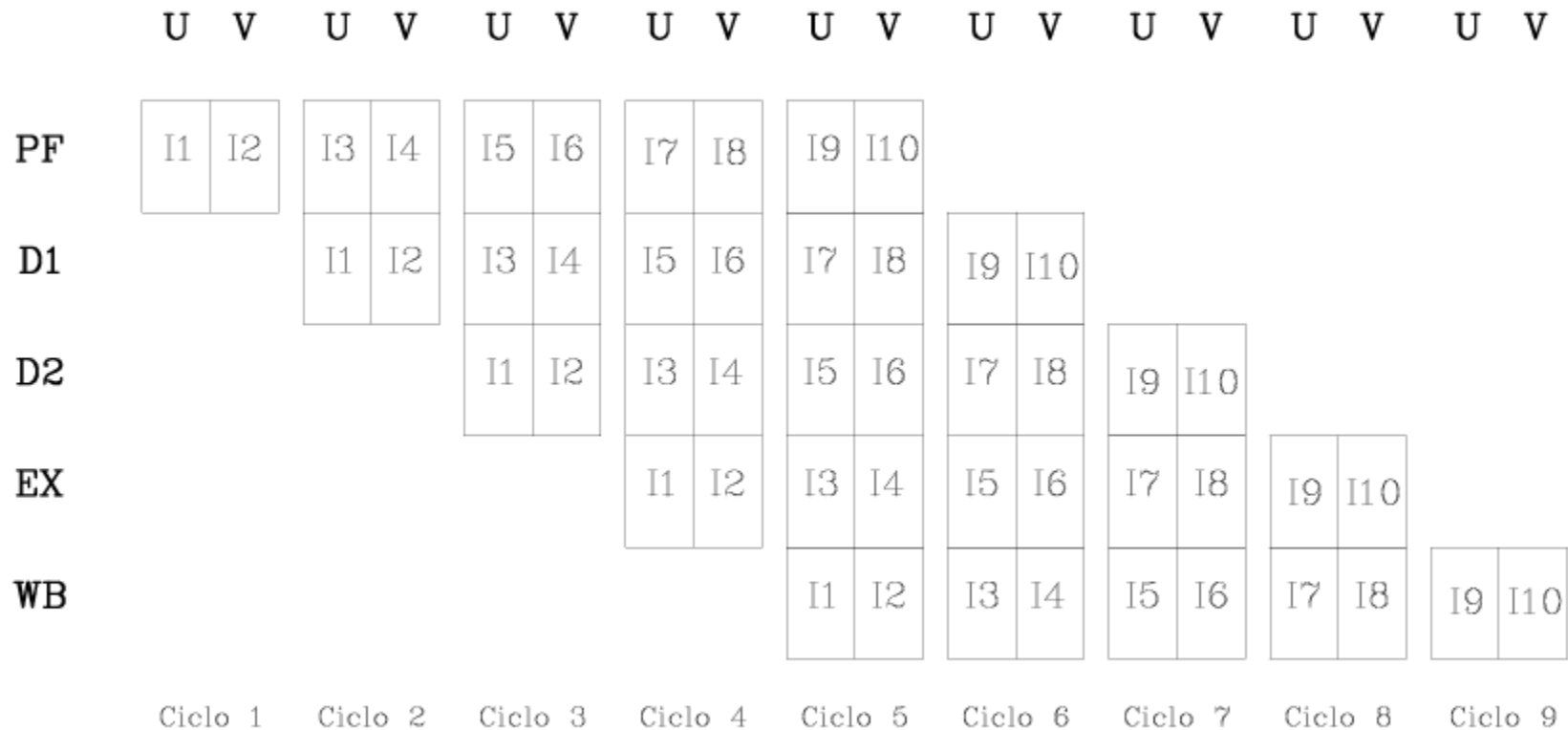
NB: Non c'è il
ROB



La pipeline del Pentium



Pipeline Pentium

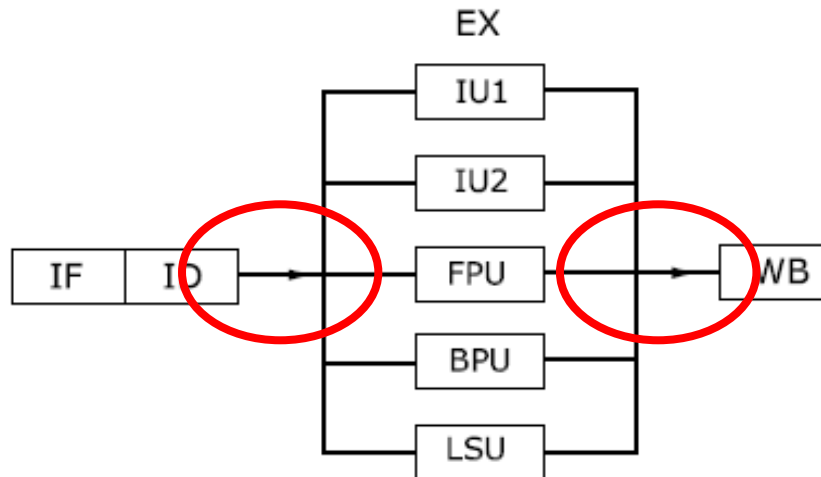


Pentium

- Le due pipe non erano perfettamente uguali; la pipe U aveva una sorta di predominanza
- Normalmente venivano emesse due istruzioni in parallelo, ma questo (a seconda delle istruzioni) non sempre era possibile
- Al termine venivano consegnate due istruzioni in parallelo (se era possibile) sempre conservando l'ordine del programma

In generale

- Se si emettono e si ritirano più istruzioni in parallelo vuol dire che ci sono più percorsi verso le unità di esecuzione e più percorsi (bus) verso il ROB utilizzabili in parallelo
- Si complica la gestione dei bus dei risultati: occorre garantire che se un'istruzione viene emessa essa, al completamento, troverà un percorso verso il ROB



Emissione e ritiro

- Il ROB fa da buffer e deve essere riempito secondo l'ordine del programma. Resta quindi il problema della prenotazione delle posizioni nel ROB
- L'emissione deve avvenire in modo che ad un generico clock completi un numero di istruzioni non superiore al numero di percorsi verso il ROB (altrimenti occorre stallare alcune FU, con ripercussione sull'emissione)
- Non è detto che si possano sempre emettere tante istruzioni quanti sono i percorsi alle FU

Esempio

- Supponiamo che ci sia la possibilità di emettere e di ritirare fino a 3 istruzioni in parallelo (la pipeline è quella del trasp. n. 3)

1. mul	r2, r6, r1	; 2 cicli	}	Possono essere emesse in parallelo
2. mul	f12, f8, f10	; 4 cicli		
3. add	r4, r5, r6	; 1 ciclo		
4. mul	r3, r7, r7	; 2 cicli	}	Emesse queste 2
5. sub	r8, r9, r8	; 1 ciclo		
6. add	r20, r21, r22	; 1 ciclo	}	Emesse queste 2
7. mul	r3, r4, 45	; 2 ciclo		
8. mul	r2, r2, r2	; 2 cicli	}	
9. mul	f2, f3, f4	; 4 cicli		
10. add	r5, r6, r7	; 1 ciclo		

Primi due clock

3

1

2

5

4 1

2

ROB

3 2 1

ROB

5 4 3 2 1

3° e 4° clock

6

7 4

2

ROB

7 6 5 4 3 2 1

ROB

10

8 7

9 2

10 9 8 7 6 5 4 3 2

Al clock successivo

- (ipotesi: il ROB è lungo quanto serve)

(3 emesse)



- Successivamente possono essere prelevate 2, 3 e 4
- Al clock di poi verrebbero prelevate 5, 6 e 7

Emissione/Ritiro

- Nell'ipotesi di 3 percorsi verso il ROB e 3 FU non ci possono essere conflitti in uscita alle FU: l'emissione può avvenire appena possibile
- Se ci sono meno percorsi che FU c'è conflitto tutte le volte che sullo stesso clock completa un numero di istruzioni superiore al numero dei percorsi.
 - Richiede RSR (più complesso)
- Il ROB è limitato: è possibile che si debba sospendere l'emissione per evitare il trabocco

Dipendenze...

- Nell'esempio precedente non c'erano dipendenze tra le istruzioni (solo vincoli legati al numero di FU e al numero di percorsi)

```
1.  ld      r1,v1(r0)      ; 3 cicli
2.  mul     r1,r1,r1        ; 2 cicli
3.  mul     f2,f3,f4        ; 4 cicli
4.  add     r4,r5,r6        ; 1 ciclo
5.  st      x(r8),r7        ; 2 cicli
```

- 2 non può essere emessa deve aspettare che 1 abbia scritto il registro r1: si blocca l'emissione

...Dipendenze

ld	r1,v1(r0)	;3	c
mul	r1,r1,r1	;2	c

- Quanto aspetta **mul** prima di essere emessa?
 - 4 clock se il risultato viene comunque parcheggiato in ROB
 - 3 clock se non c'è il passaggio da ROB per le istruzioni che completano e non hanno da aspettare il completamento delle precedenti
- Bisogna migliorare lo schema precedente in modo da poter emettere le istruzioni seguenti che non hanno dipendenze (esecuzione fuori ordine)

Soluzione 1

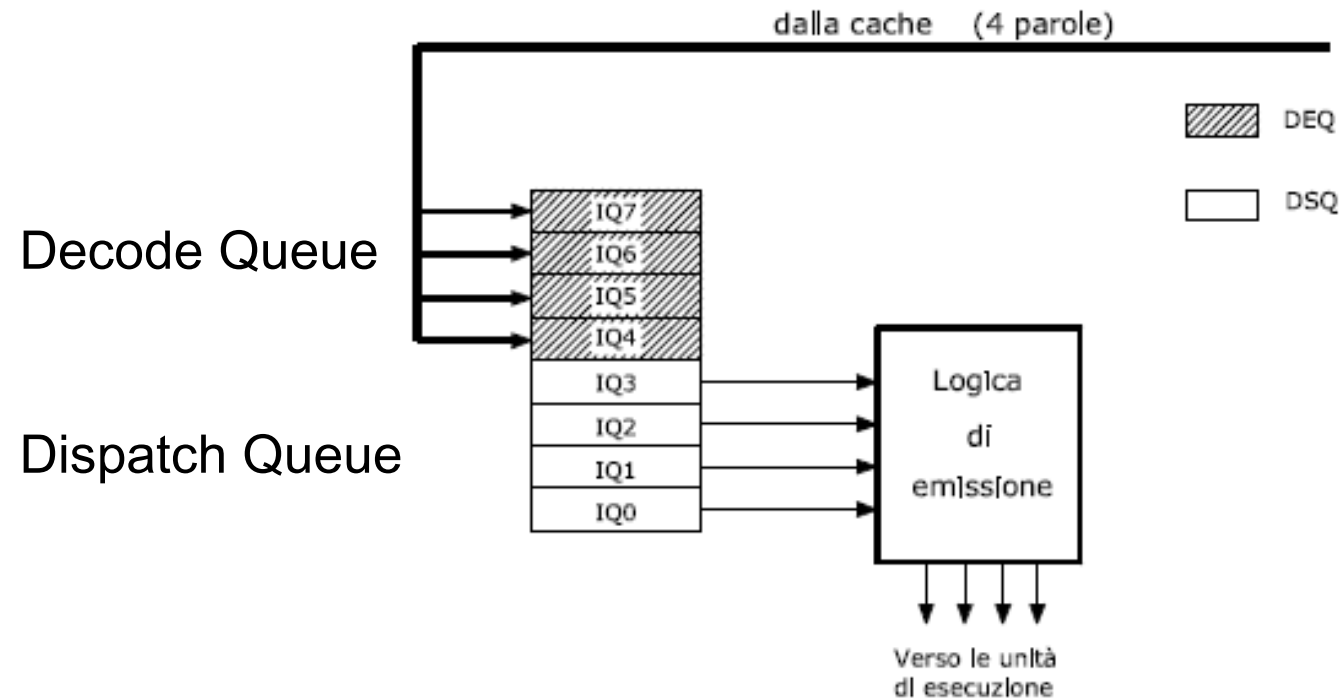
- Prevedere uno (o più) slot di attesa per ogni FU, in modo che un'istruzione come la **mul** possa essere comunque emessa consentendo di avanzare con l'emissione delle istruzioni che seguono

ld	r1, v1(r0)	}	Emesse assieme: mul va ad attendere nello slot di IU2
mul	r1, r1, r1		
mulf	f2, f3, f4		
add	r4, r5, r6	}	Emesse assieme: mulf va ad attendere nello slot di FPU
st	x(r8), r7		
mulf	f1, f2, f5		

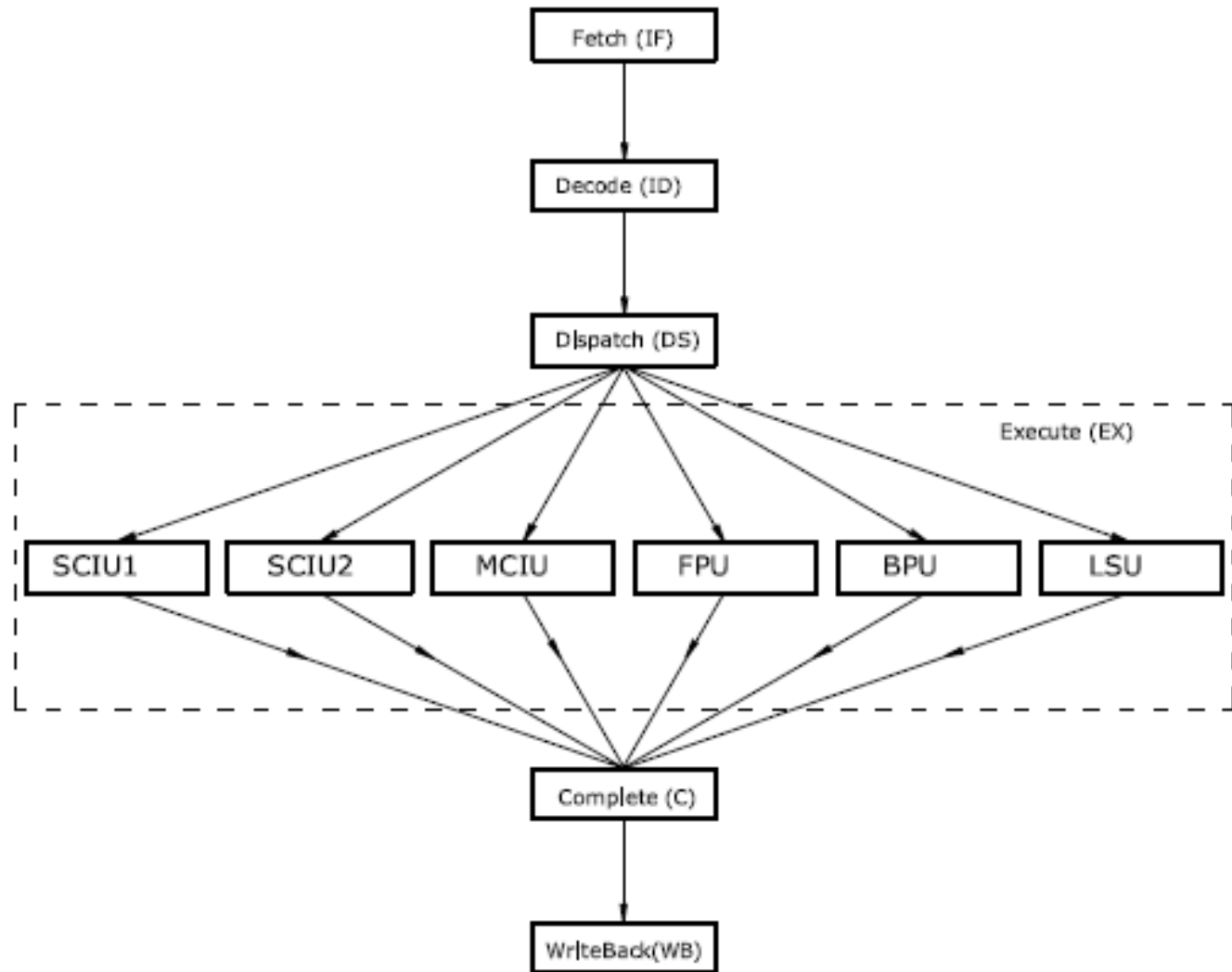
Tecnica usata dal PowerPC

PowerPC

- Emissione

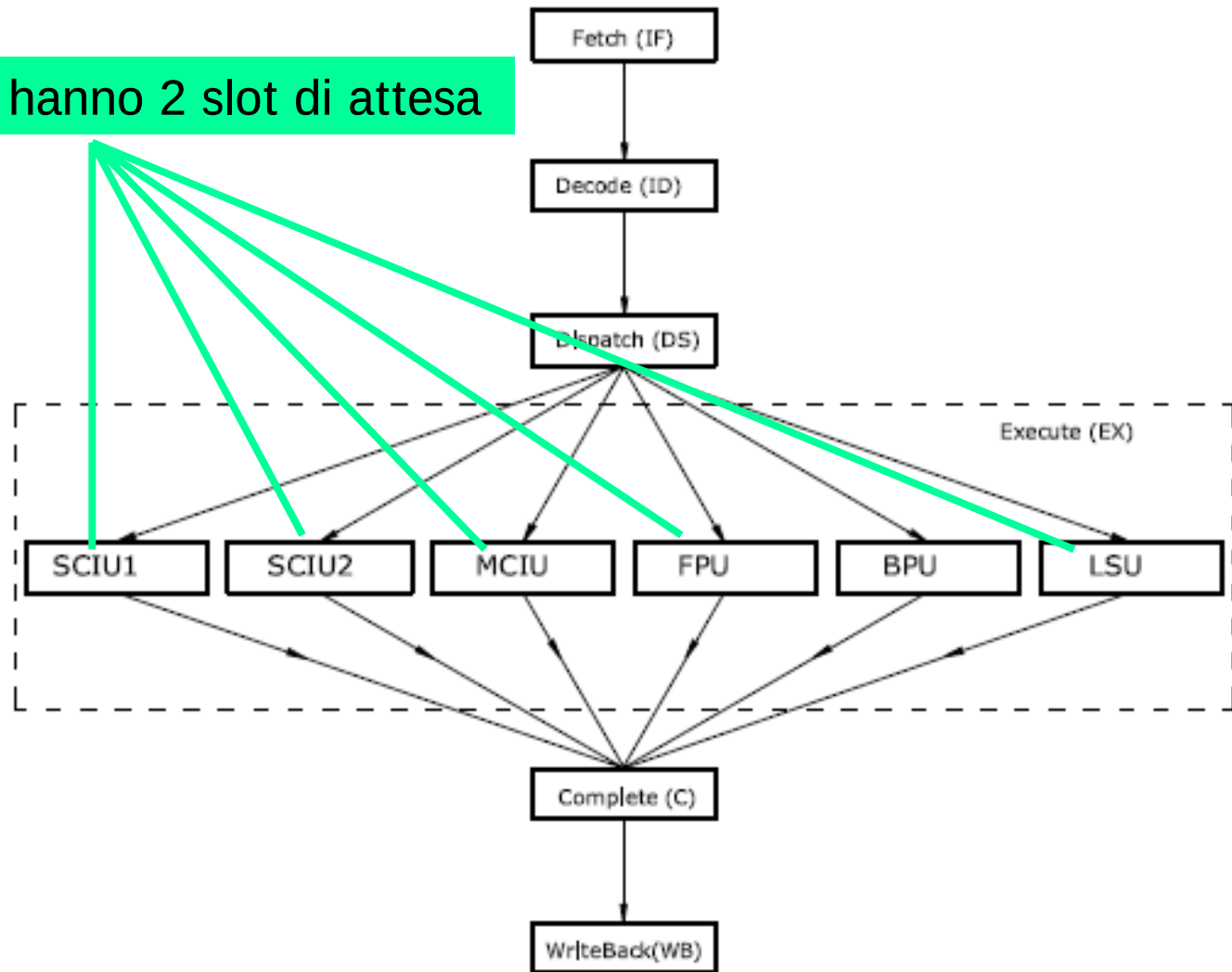


PowerPC pipeline



PowerPC pipeline

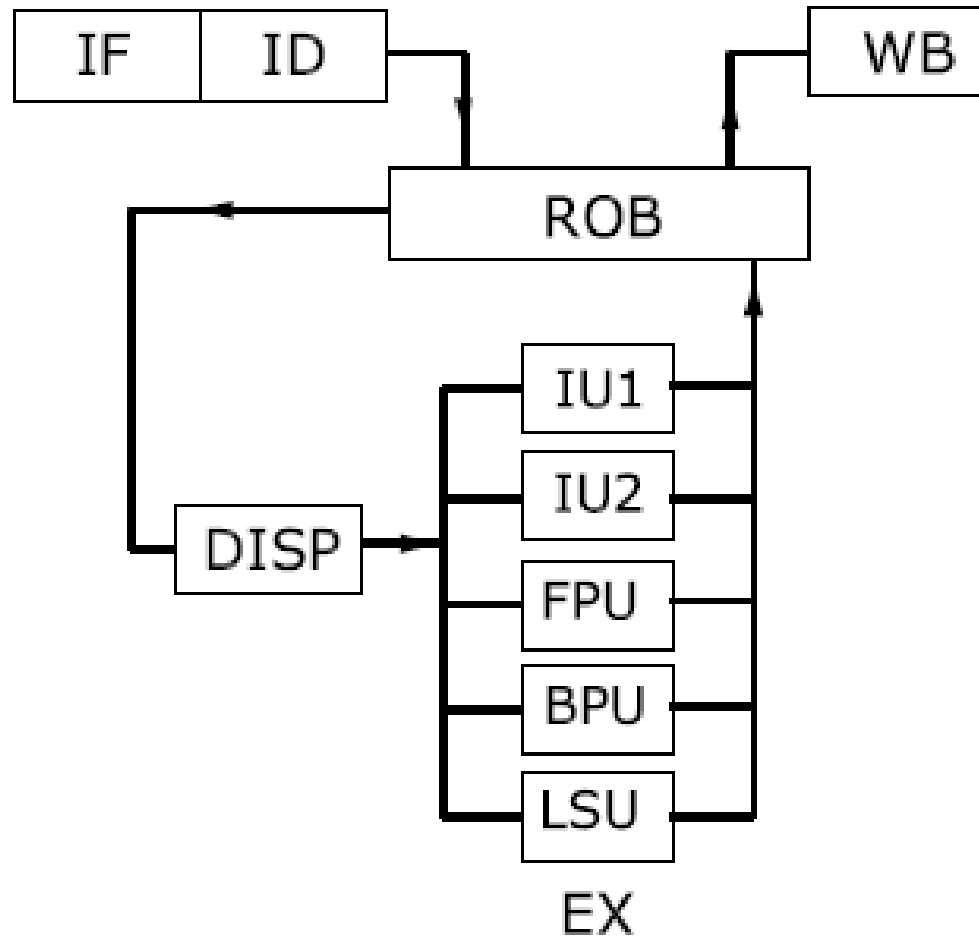
Queste hanno 2 slot di attesa



Soluzione 2....

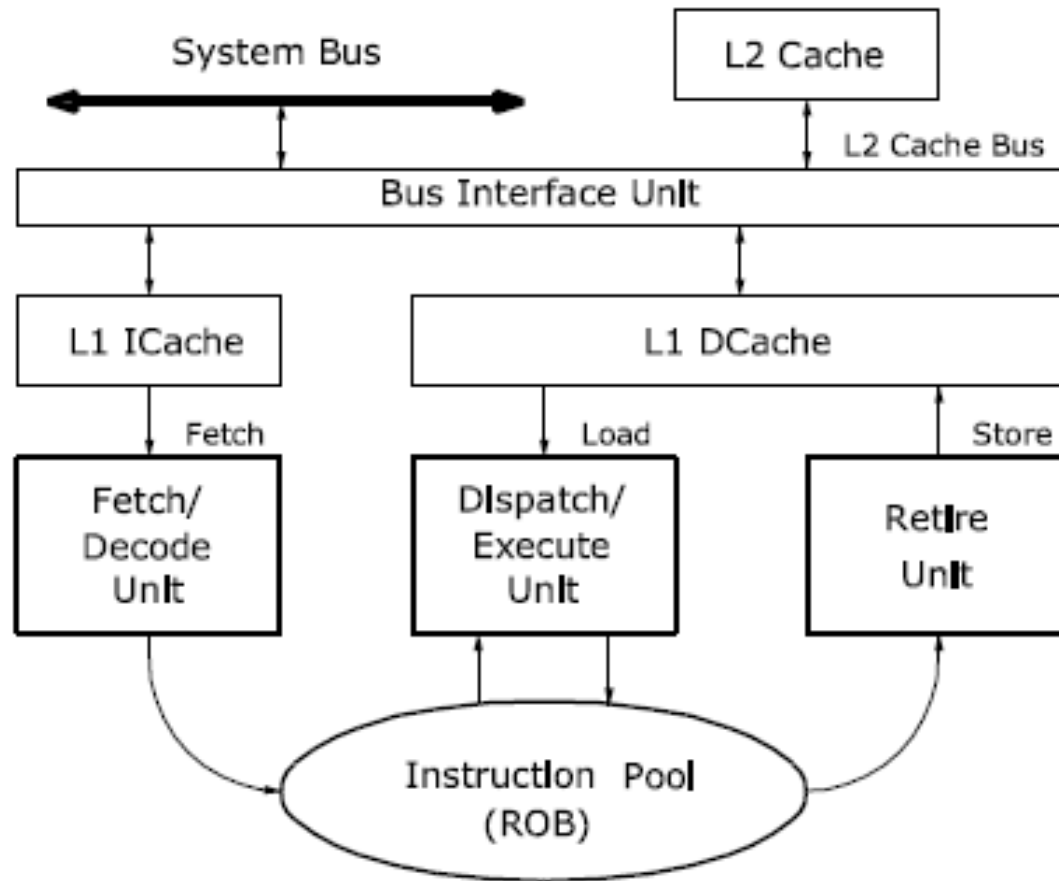
1. Posizionare diversamente il ROB
 - Lo stadio di emissione emette direttamente in ROB
 - Emette quando nel ROB c'è posto (non serve RSR)
2. Le unità di esecuzione prelevano dal ROB e riscrivono i risultati nel ROB
3. Lo stadio di WB preleva dal ROB

...Soluzione 2



...Soluzione 2

- E' lo schema del Pentium Pro (e seguenti)

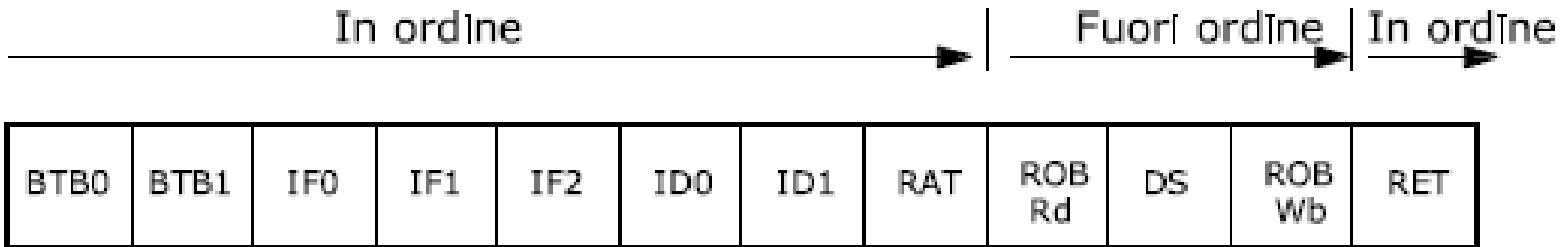


Il Pentium Pro (Novembre 1995)

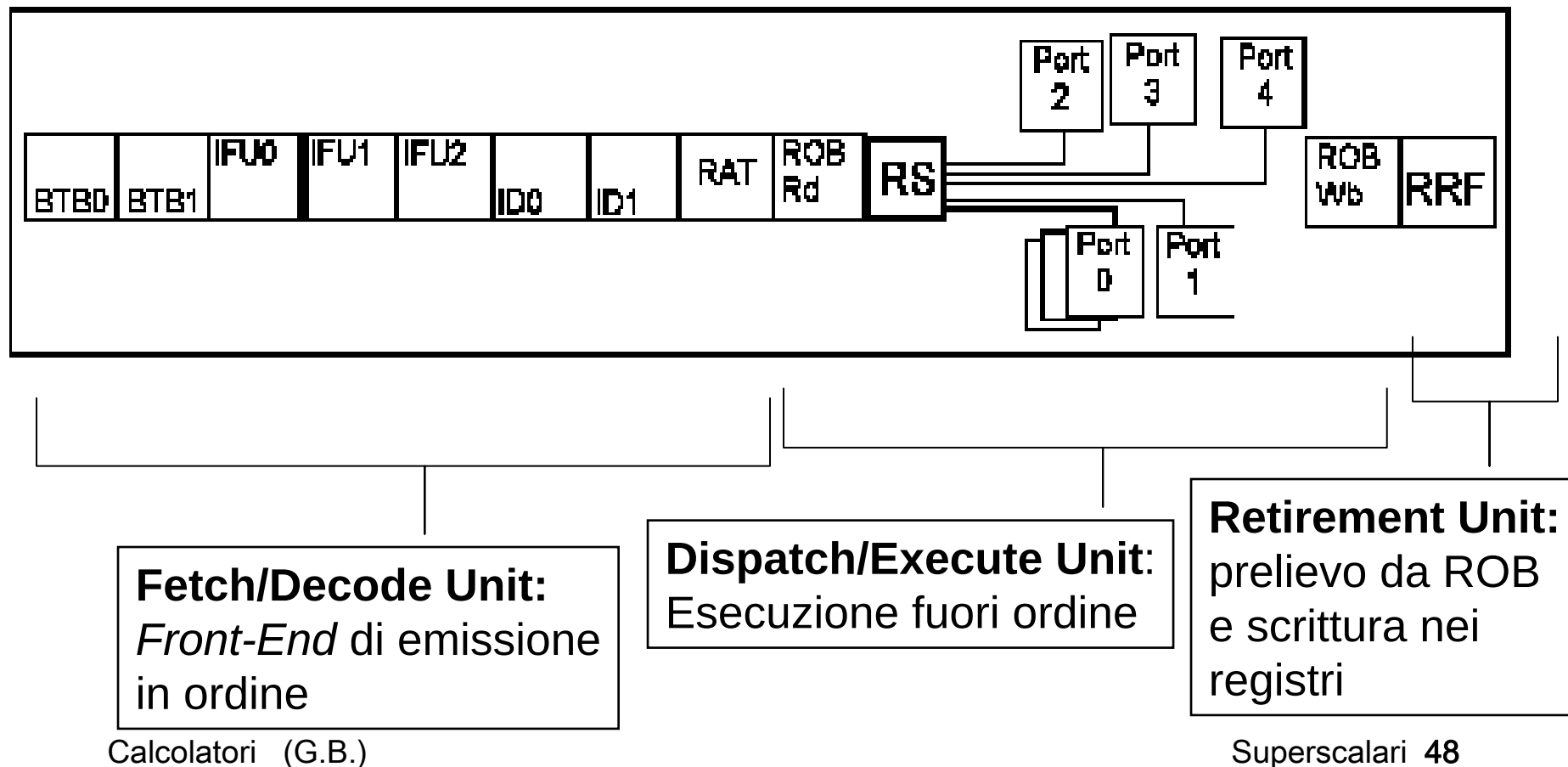
Frequenza di CPU	da 150 a 200 MHz
Frequenza Bus	60 MHz
Bus Dati	64-bit
Bus Indirizzi	36-bit
Cache L1 integrata	8 kbyte dati, 8 kbyte istruzioni
Cache L2 da 256 a 1024 kbyte	
Pipeline	12 stadi
Superscalare	3-vie
Unità di Esecuzione	5
Branch Prediction	BTB 4-vie 512-entrate
	4-bit di storia, predittore a 2-livelli
Modello di Esecuzione	Fuori ordine
Esecuzione Speculativa	Sì

Pipeline Pentium Pro

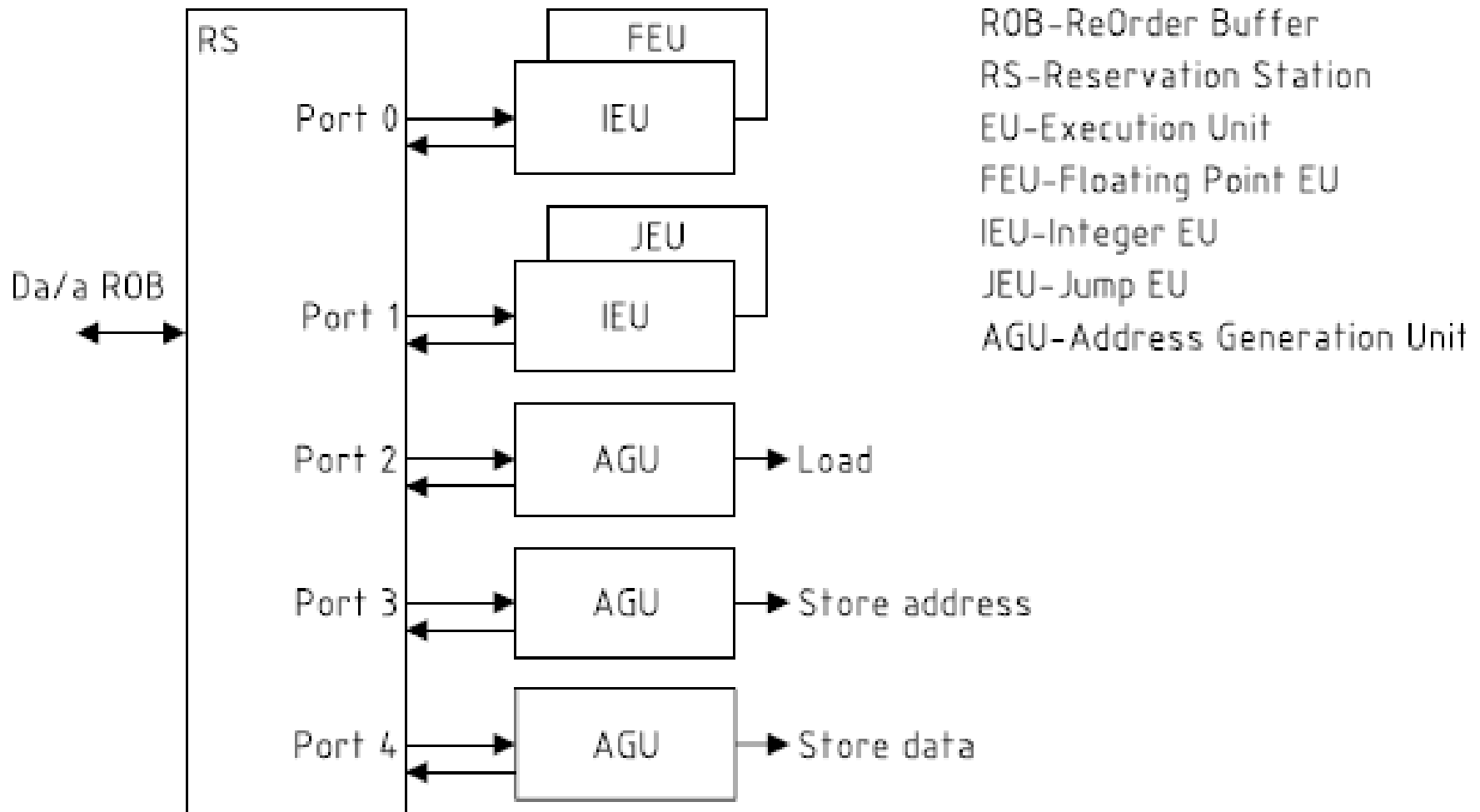
- 12 stadi
 - Fetch/Decode primi 8 stadi
 - Dispatch/Execute 3 (di più, di più !) stadi intermedi
 - Retirement stadio finale



Schema Intel (oggi introvabile)



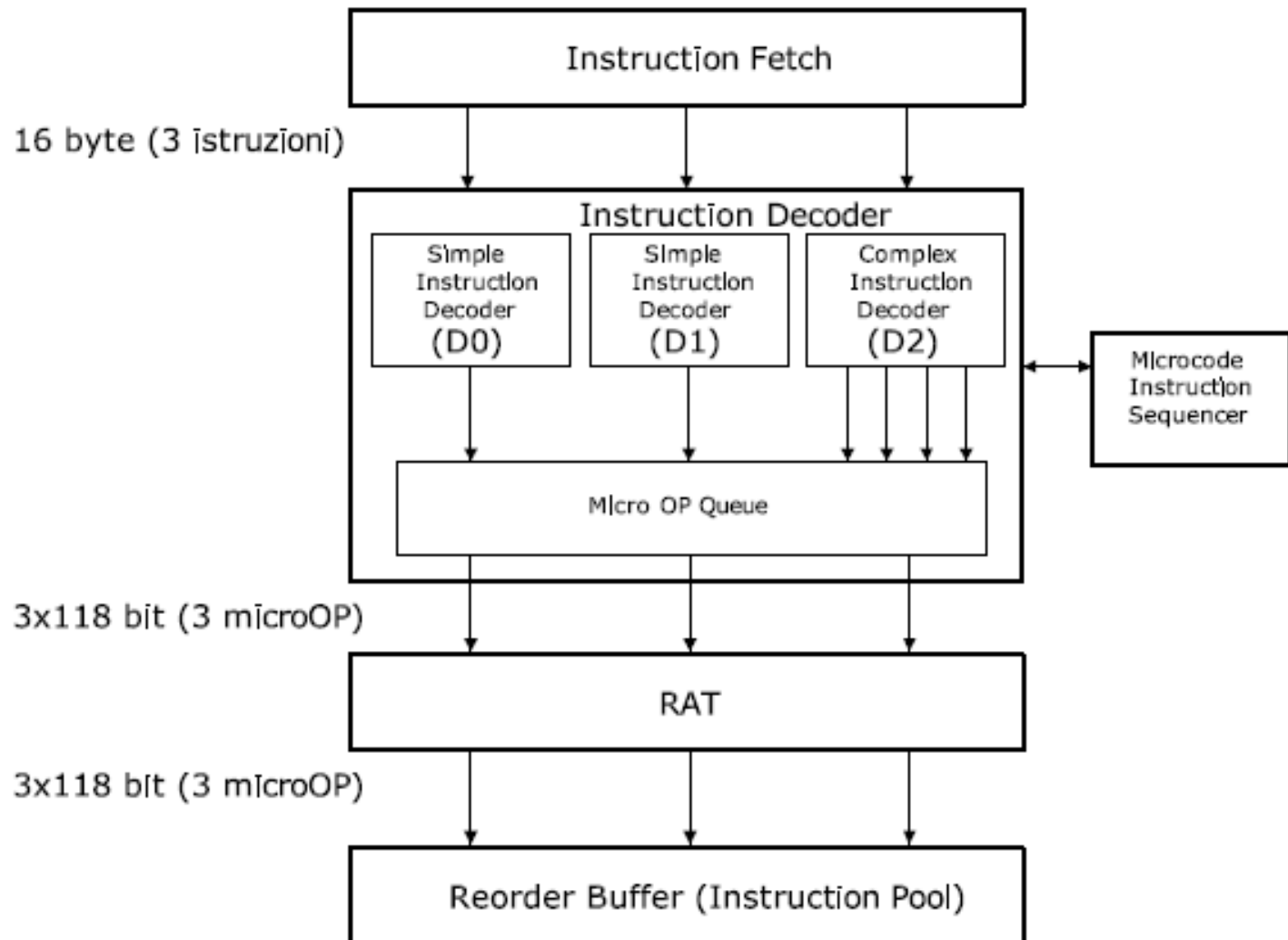
Dispatch/execution unit



Un “*piccolo*” dettaglio

- L'architettura Intel ha un repertorio CISC
 - Istruzioni di diverso formato (da 1 a 8 e più byte)
 - Repertorio esteso
 - Pochi registri di uso generale
- Per rendere efficiente la pipeline:
 - Istruzioni architetturali trasformate in “microoperazioni” (uOP) dallo stadio di decodifica
 - Ridenominazione dei registri
- Una uOP è una *tripla* (due sorgenti e una destinazione)

Trattamento microOP



Dettagli

- La stazione di decodifica contiene 3 decodificatori (stadio ID0)
- Ad ogni clock possono essere decodificate fino a 3 istruzioni IA e possono uscire dal decodificatore fino a 6 uop [a valle del decodificatore c'è un buffer (stadio ID1) di 6 posizioni]
- Le istruzioni IA che generano più di una uop devono essere necessariamente decodificate dal decodificatore D2 (microcode).

... Dettagli

- Lo stadio RAT/allocator ha capienza per 3 uop (ne può incamerare/rilasciare 3 a ciclo di clock).
- Il ROB può contenere fino a 40 uop. I registri fisici sono nel ROB e sono 40
- Funzione dello stadio RAT/allocator:
 - Assegnare un numero d'ordine progressivo alle uop all'atto dell'immissione (in sequenza) nel ROB
 - Mappare gli otto registri dell'architettura nei 40 registri fisici
- RAT/allocator è l'ultimo stadio *in ordine*

Esecuzione fuori Ordine

- Nel ROB le uop vengono eseguite in base al loro stato (dipendenza dati e disponibilità operandi) e disponibilità delle risorse necessarie alla loro esecuzione.
 - La ricerca può far avanzare anche di 20 o 30 istruzioni all'interno del ROB
- Le micro-op che soddisfano ai requisiti vengono prelevate dal ROB e passate alle effettive unità di esecuzione.
- Il processo di selezione si svolge nello stadio DS (RS) che dispone di 5 porte di comunicazione con le unità deputate all'esecuzione delle uop.

Esecuzione fuori ordine

- Se viene predetta una diramazione: si possono emettere le istruzioni dalla nuova posizione, ma se la predizione è errata le istruzioni sulla via mal predetta non devono avere alcun effetto.
- Si parla di esecuzione **Speculativa**.
- Le interruzioni costituiscono una complicazione orrenda: l'interruzione può arrivare in qualunque istante. Occorre salvare uno stato congruente (alcune istruzioni potrebbero essere già completate, ma vanno annullate e rifatte quando tornerà il controllo)

... nel ROB

- Per ogni ciclo di clock possono essere avviate fino a 3 uop (non 5).
- I risultati (incerti) dell'esecuzione ***speculativa*** non possono essere passati come risultati effettivi: vengono immagazzinati nuovamente nel ROB.
- Se una predizione di salto condizionato (da parte del BTB) si rivela errata, lo stato di tutte le micro-op eseguite in maniera speculativa in seguito al salto vengono rimosse dal ROB;
 - la vera destinazione del salto condizionato viene passata al BTB, che provvede ad aggiornare i bit di predizione.

..Prestazioni

Cicli di clock richiesti per l'esecuzione:

- Addizioni intere: 1
- Addizioni floating: 3
 - con tasso (massimo) di uscita di una per ciclo di clock
- Moltiplicazioni floating: 5
 - con tasso di uscita di una ogni 2 cicli di clock
- Moltiplicazioni intere: 4
 - con tasso di uscita di una per ciclo di clock.
- La divisione floating **non** è in pipeline e richiede 17 cicli in singola precisione, 32 in doppia precisione, 37 in precisione estesa.

Register Renaming

- IA prevede solo 8 registri di uso generale

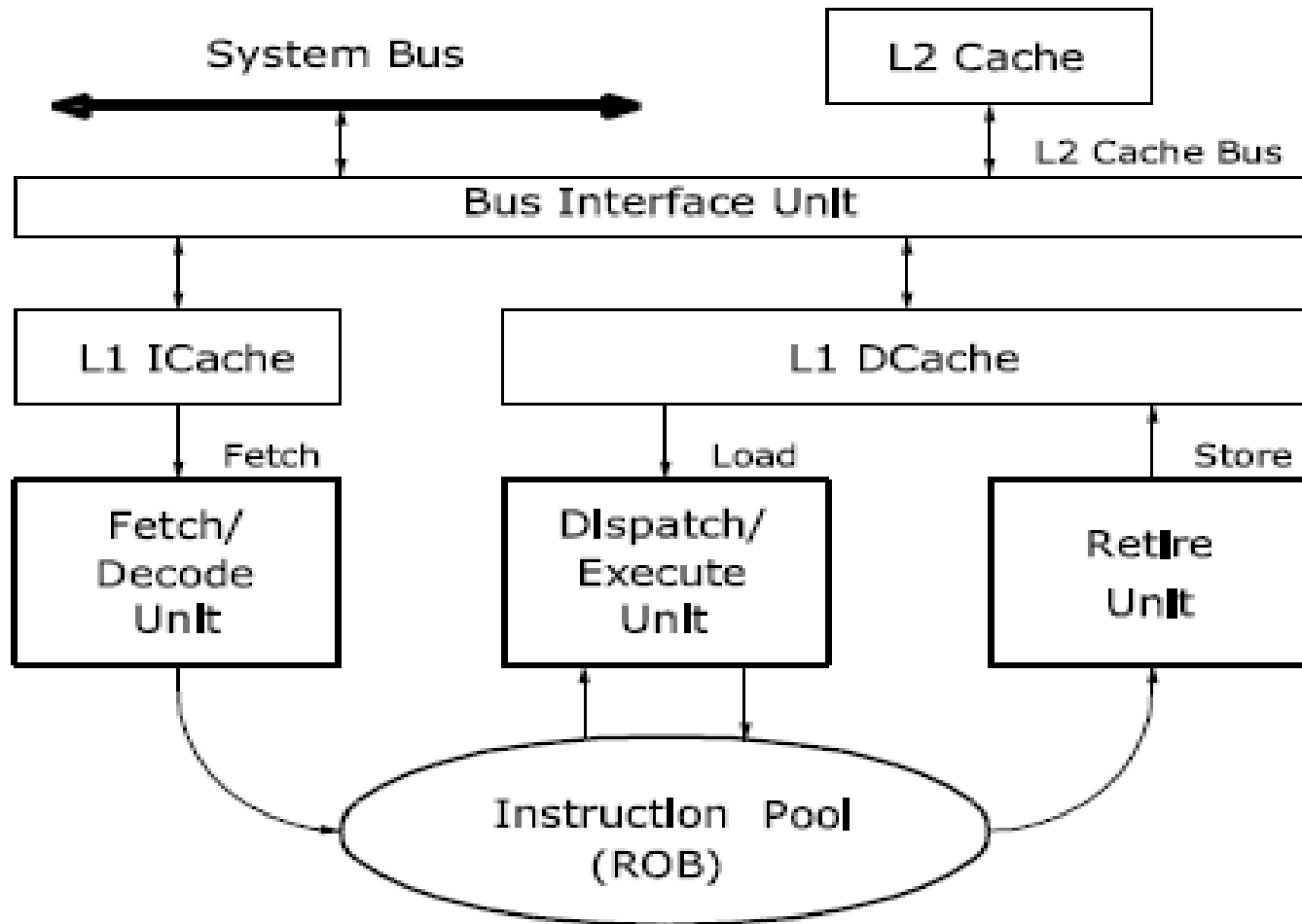
```
MOV    EAX, 50
ADD    MEM1, EAX
MOV    EAX, 77
ADD    MEM2, EAX
```

Due tratti di codice
indipendenti se non
usassero lo stesso
registro

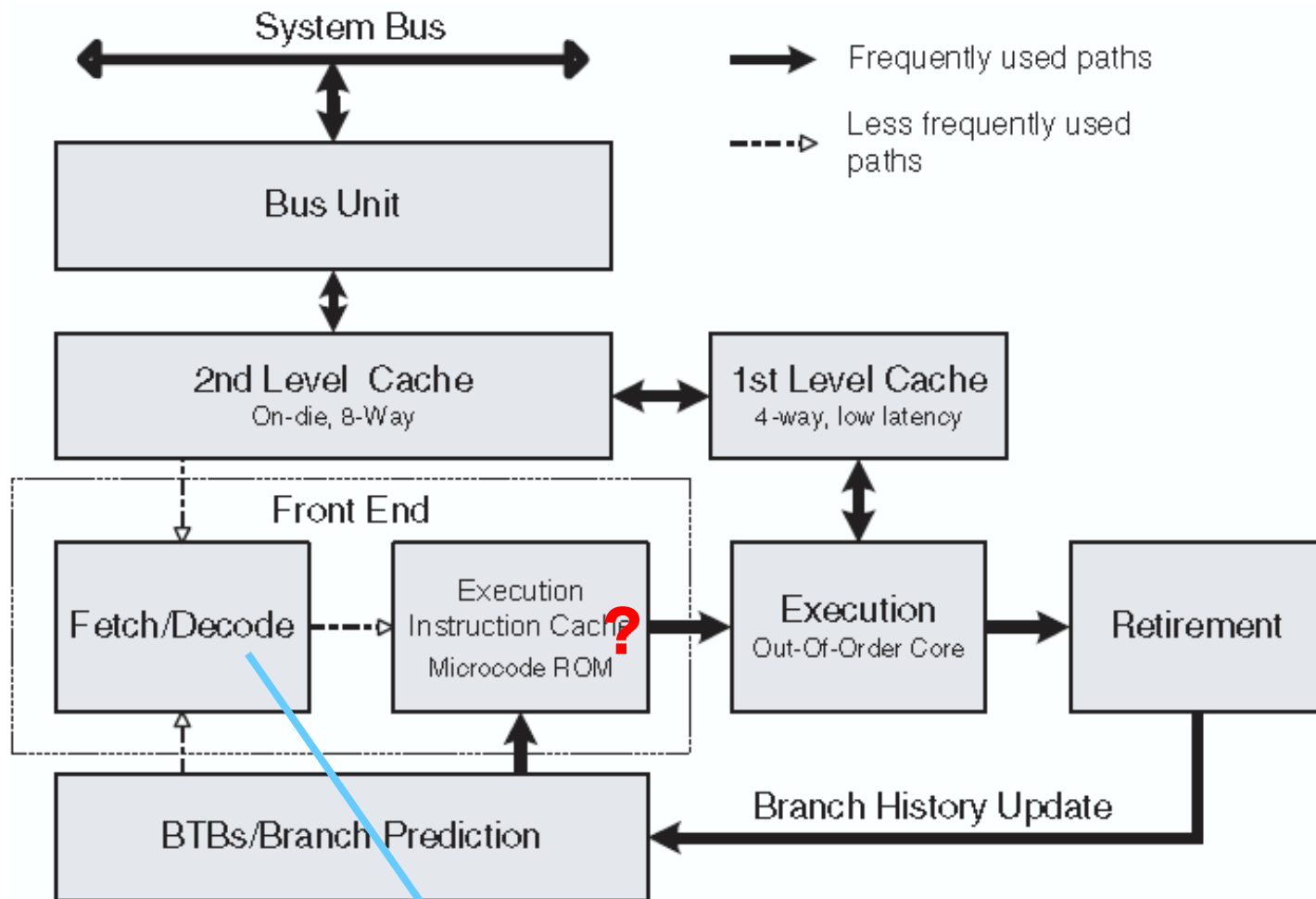
- EAX viene mappato su due registri fisici differenti
 - le istruzioni possono essere avviate in parallelo
 - comporta l'analisi del flusso dati e delle interdipendenze.
- Al sito Intel si trovano tutorial sull'architettura PentiumPro e sull'esecuzione fuori ordine

Un'occhiata agli sviluppi successivi

Pentium Pro (meglio questa della successiva)

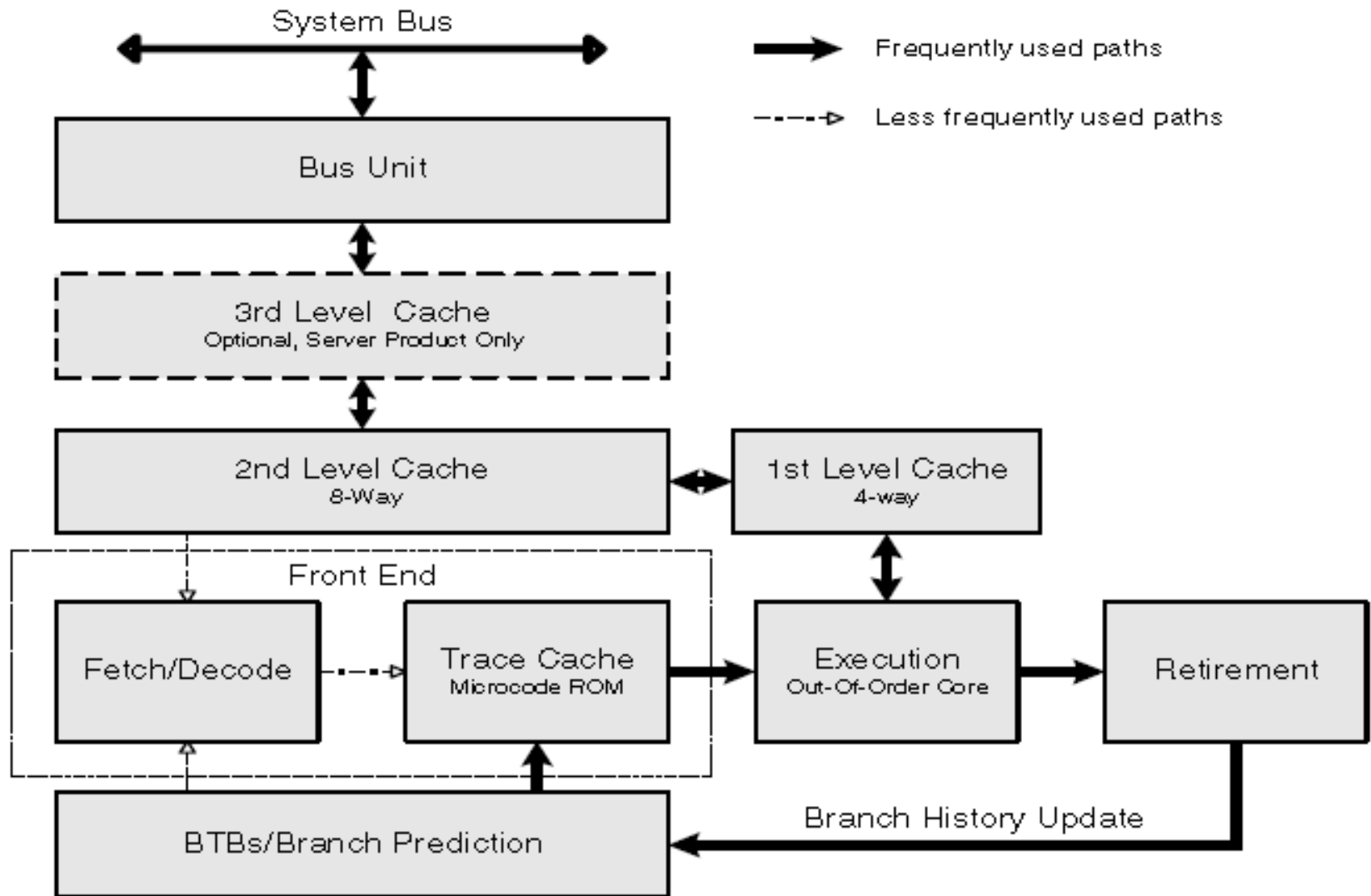


P6 (Pentium 6)

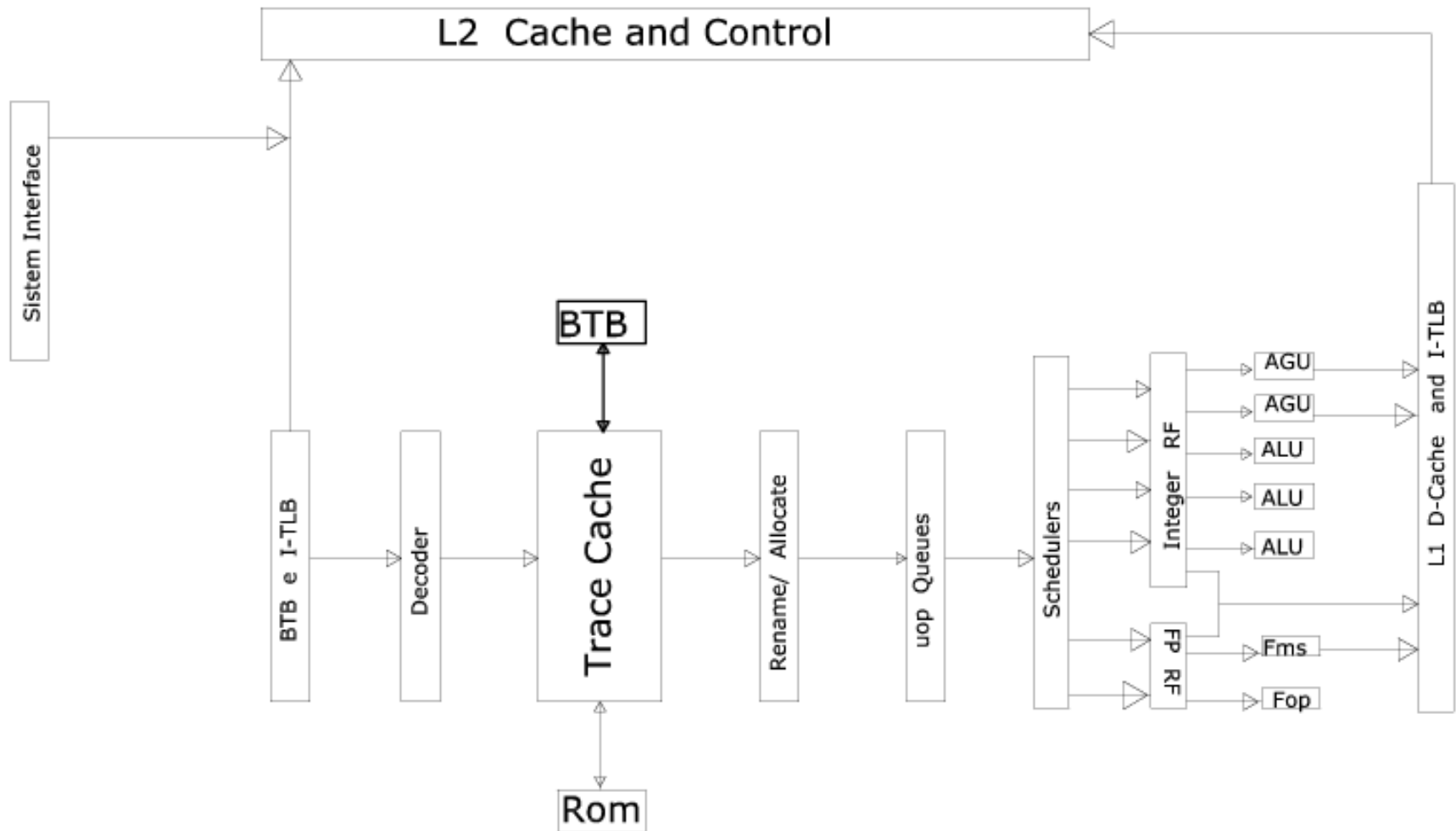


In realtà Fetch/decode si fa da L1 cache

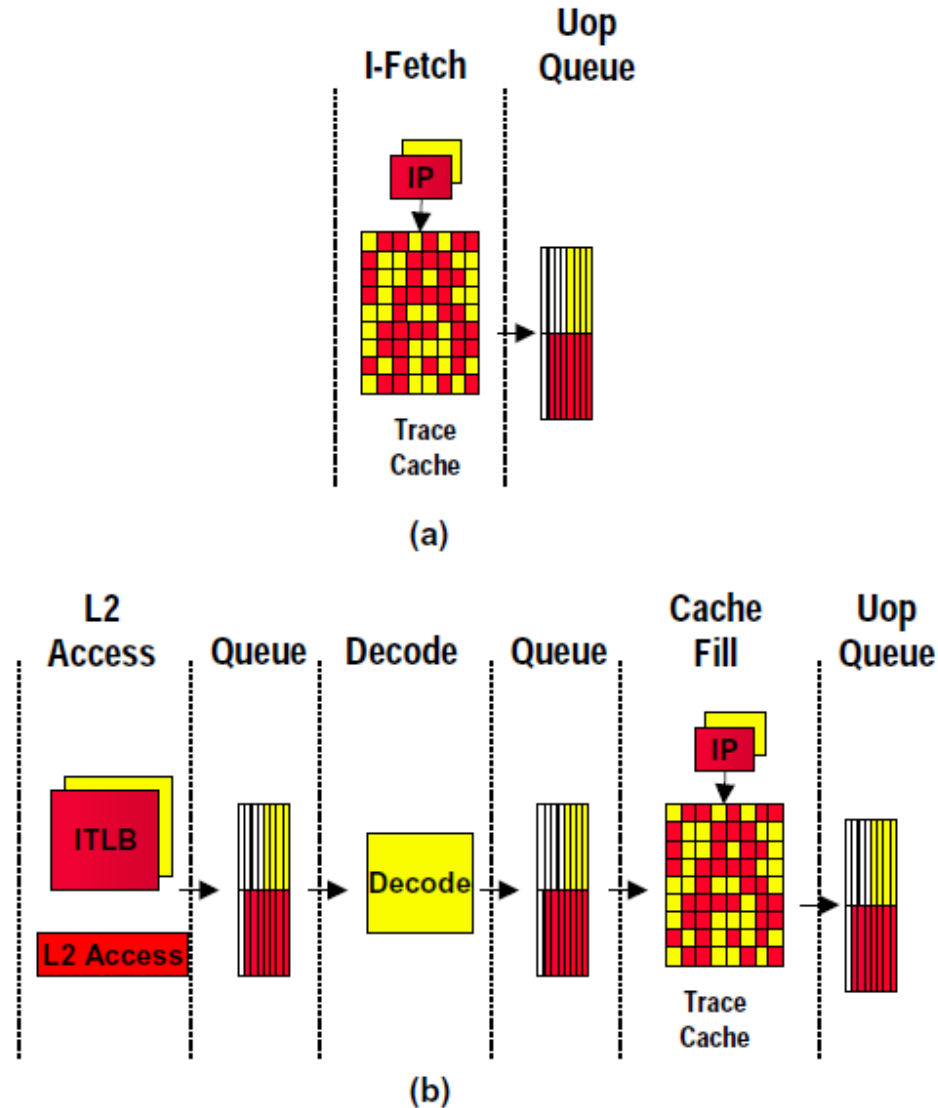
Pentium 4



II Pentium 4 (2002)



Trace Cache



Pentium 4

- Trace cache
 - Tiene le microop, in modo da evitare la decodifica tutte le volte
 - La trace cache **non** è presente in tutti i modelli successivi
- Pipeline a 20 stadi
- Bus “Quad pumped” (??? !!!!!)
- Multithreading (Hyper-Threading)
- Dicevano di portarlo a 10 GHz si sono fermati in zona 4 GHz.

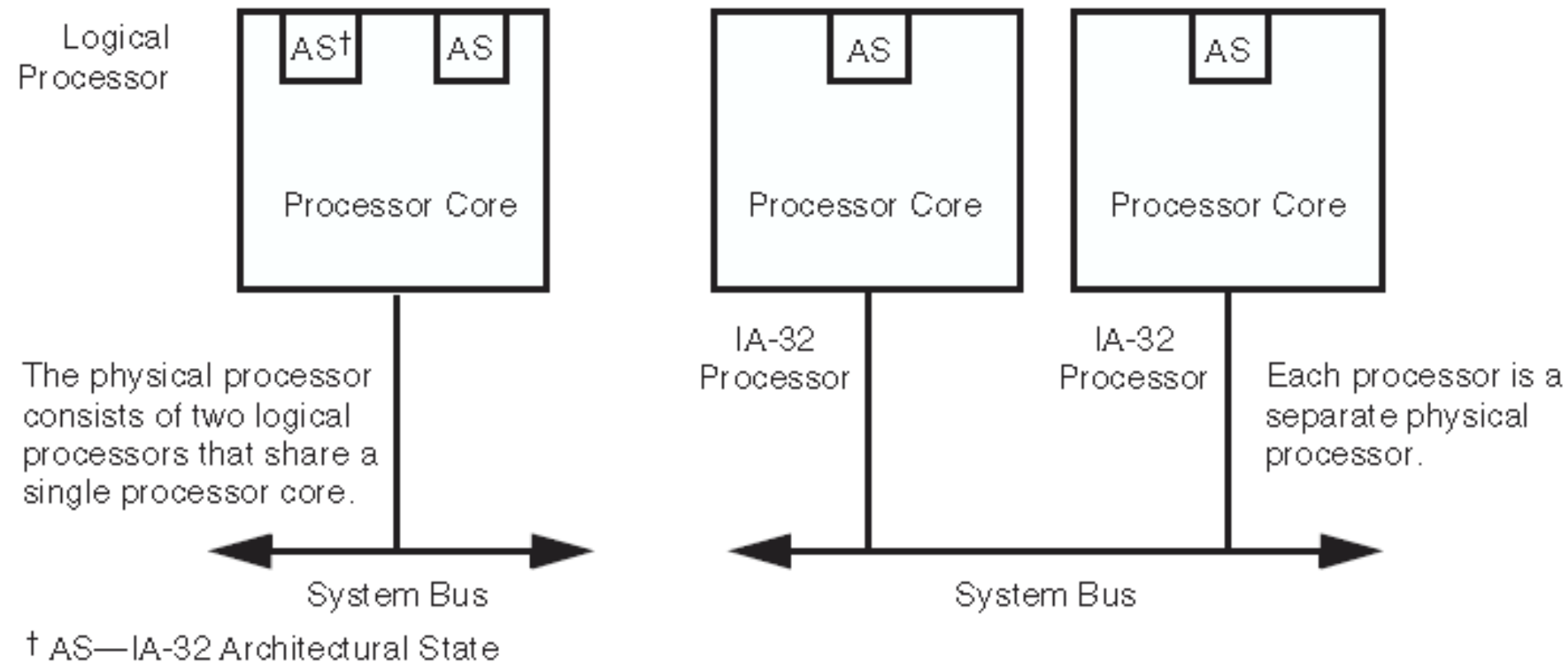
Quad pumped

- Viene usato il fronte di salita e quello di discesa
- Dal clock si derivano 2 due segnali sfasati di 90° in modo da avere 4 fronti
- Ne consegue che la frequenza del trasferimento sul bus risulta 4 volte quella del clock.
 - In pratica un bus dichiarato a 800 MHz si basa su un clock a 200 MHz

Hyper-Threading

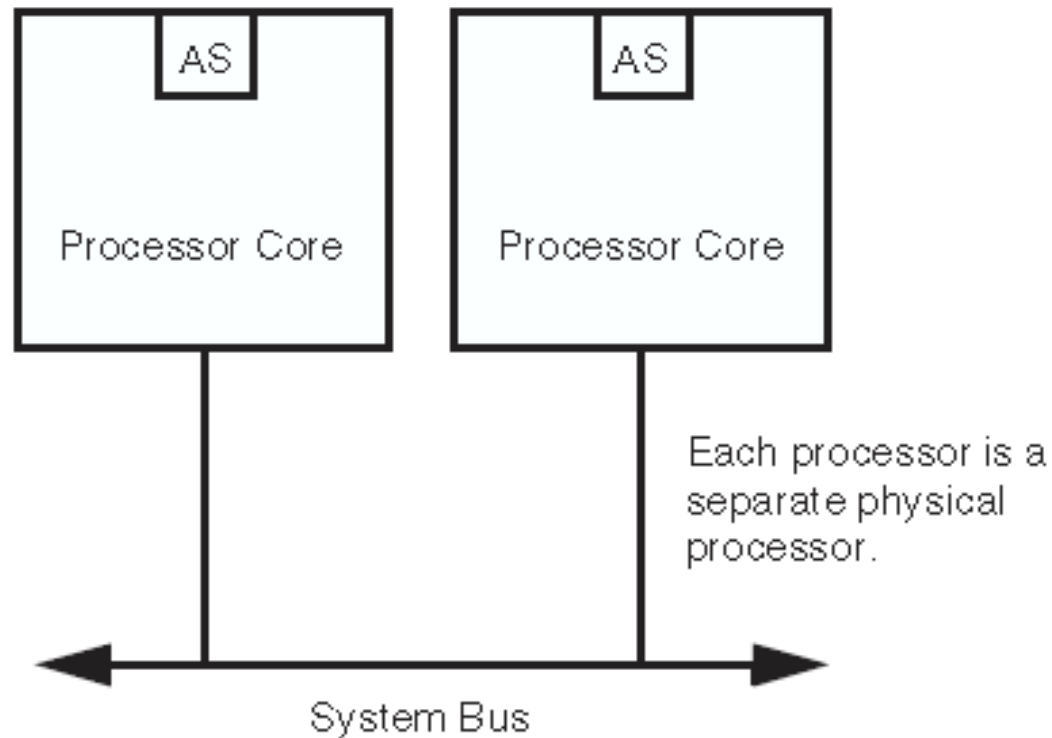
IA-32 Processor With
Hyper-Threading Technology

Traditional Multiple Processor (MP) System



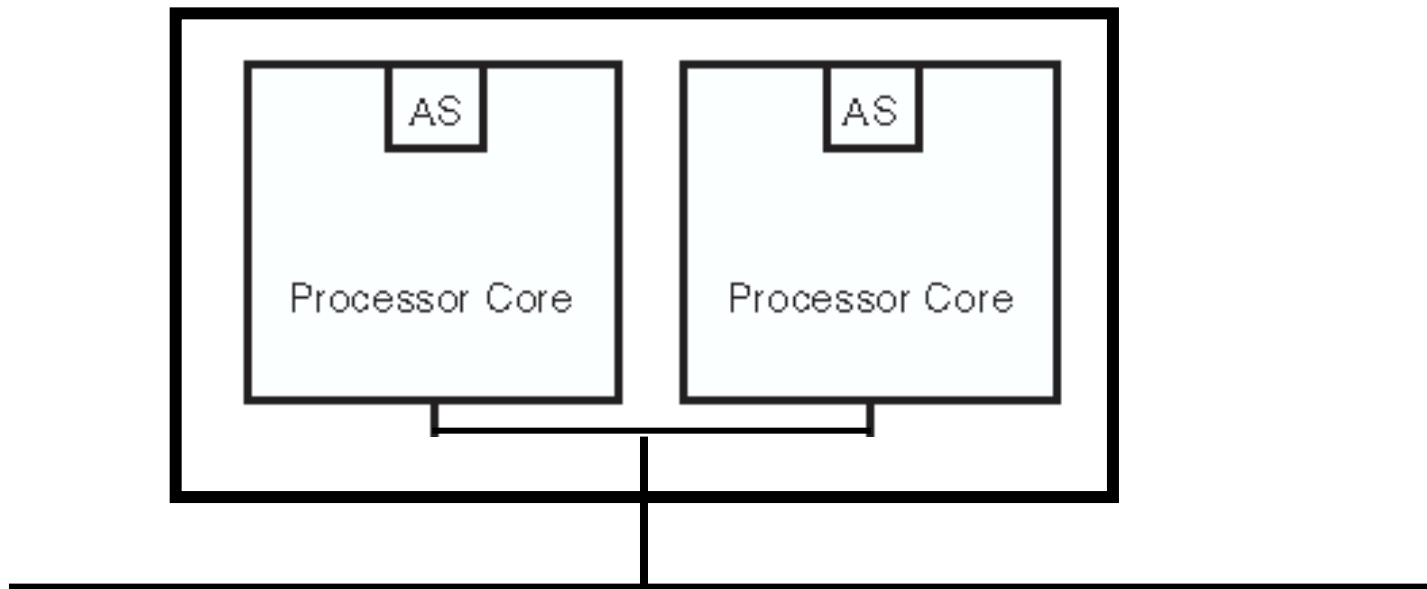
Multi Processor System tradizionale

Traditional Multiple Processor (MP) System



Multi Core

- Racchiudere nello stesso chip più processori
 - Si contendono il bus esterno
 - Per il resto sono indipendenti



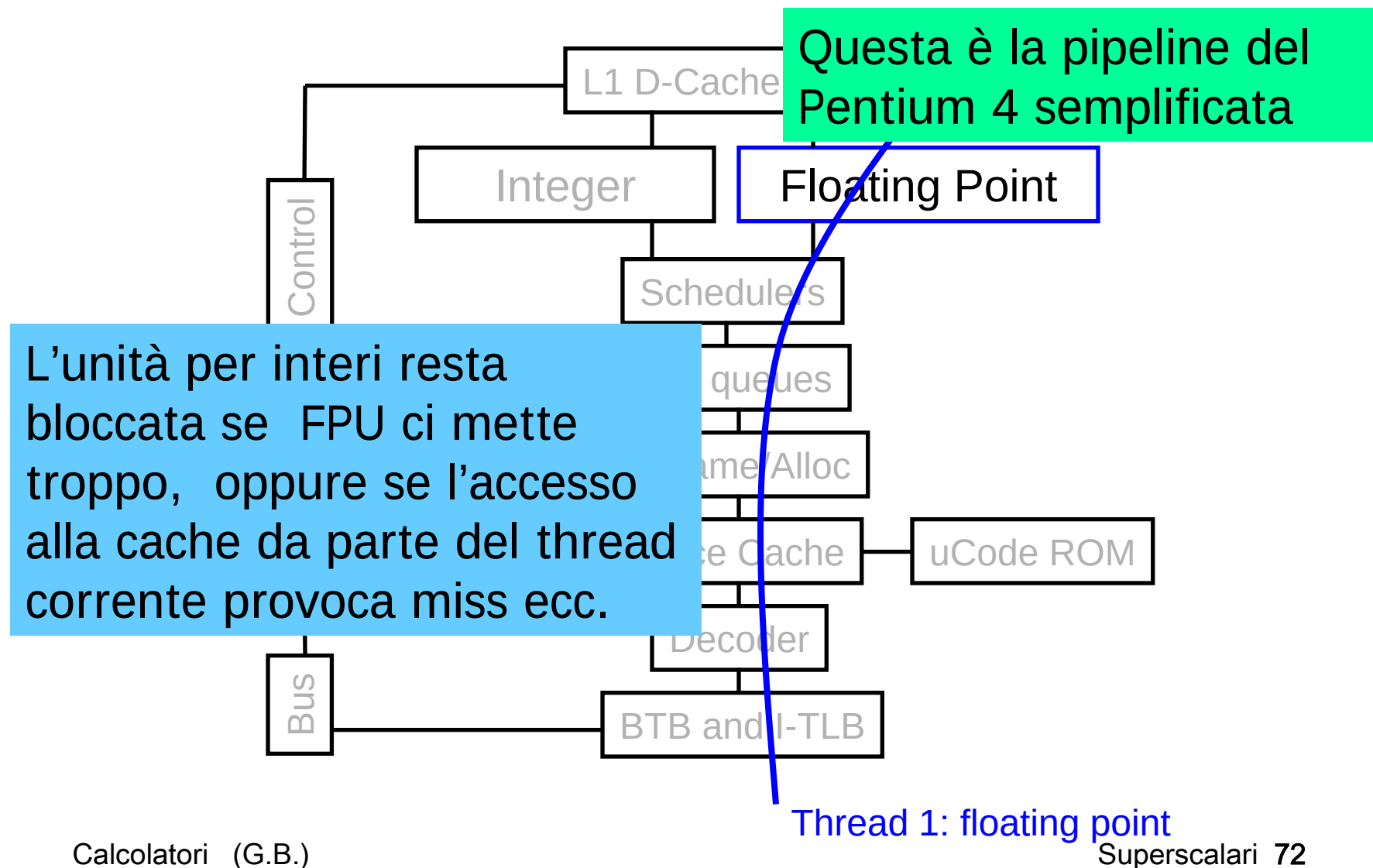
Perché multi core

- Minori consumi di due distinti processori
- Minor occupazione sulla scheda
- E' un modo per accrescere le prestazioni attraverso il parallelismo
- Come se fossero due processori distinti, con l'arbitraggio del bus all'interno del chip
 - Parallelismo effettivo
 - Se non ci fosse il problema del bus: potenziale raddoppio delle prestazioni (se si riuscisse a farli lavorare al 100% in parallelo)
 - Nella pratica se si ottiene un miglioramento dell'80% è grassa
 - Sistemi operativi e applicazioni conformi

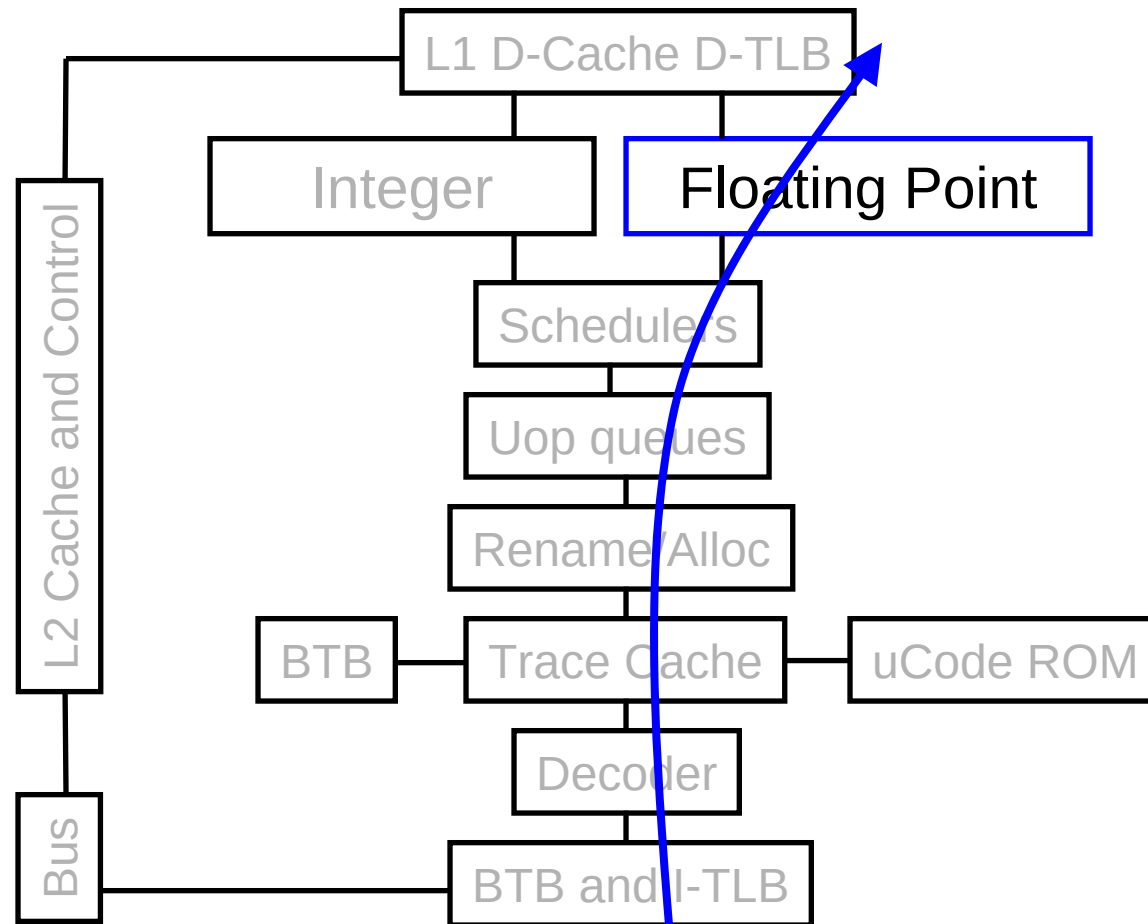
Threading

- Multithread : Threads ottenuti attraverso l'allocazione del processore (time-slice)
- SMT (Simultaneous multithreading) : Threads eseguiti simultaneamente sullo stesso processore , ovvero senza time-slicing
 - E' la logica del core che permette l'esecuzione parallela ottimizzando l'uso delle risorse
 - Esempio: Se un thread è in attesa che la FPU completi, un altro può usare la IU
 - Hyper-Threading è il nome di Intel per SMT (Intel la chiama HTT, HyperThreading Technology)

Senza SMT :

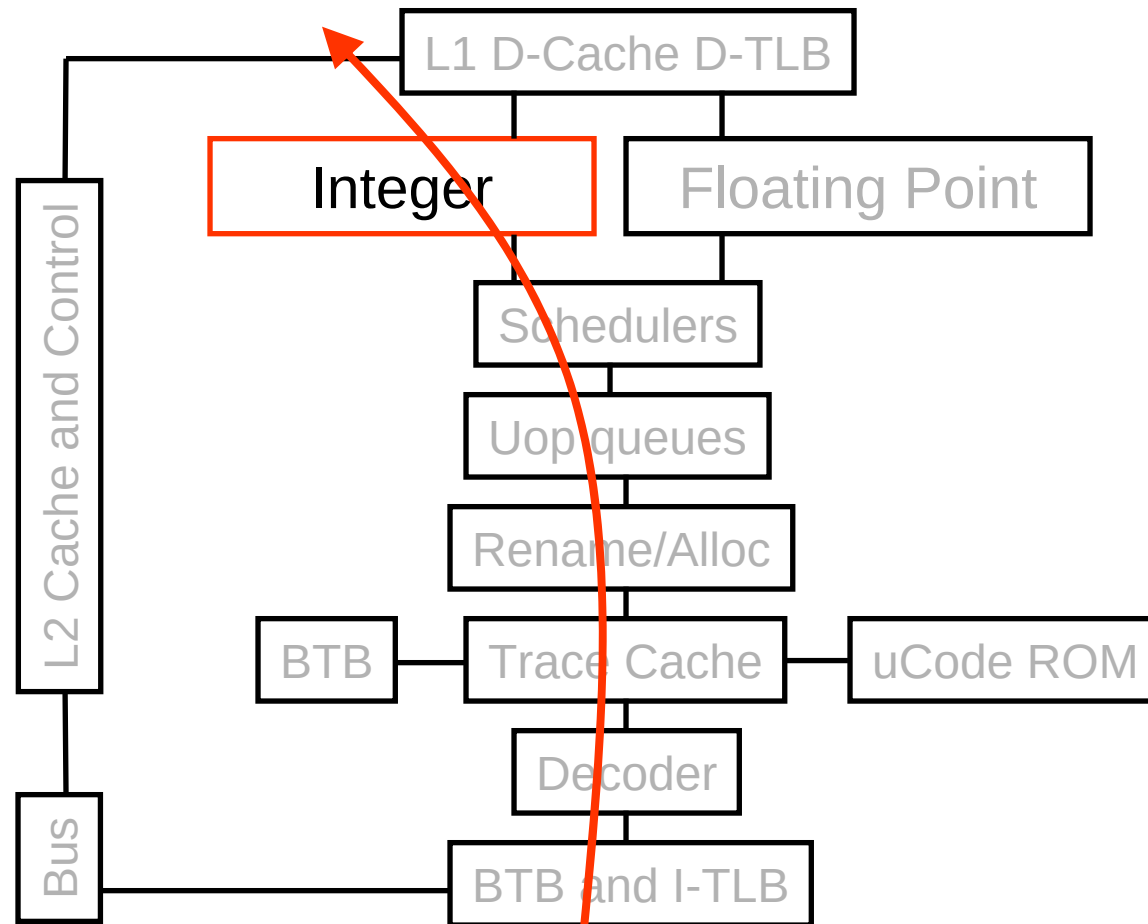


Senza SMT : un solo thread (che usa FPU)



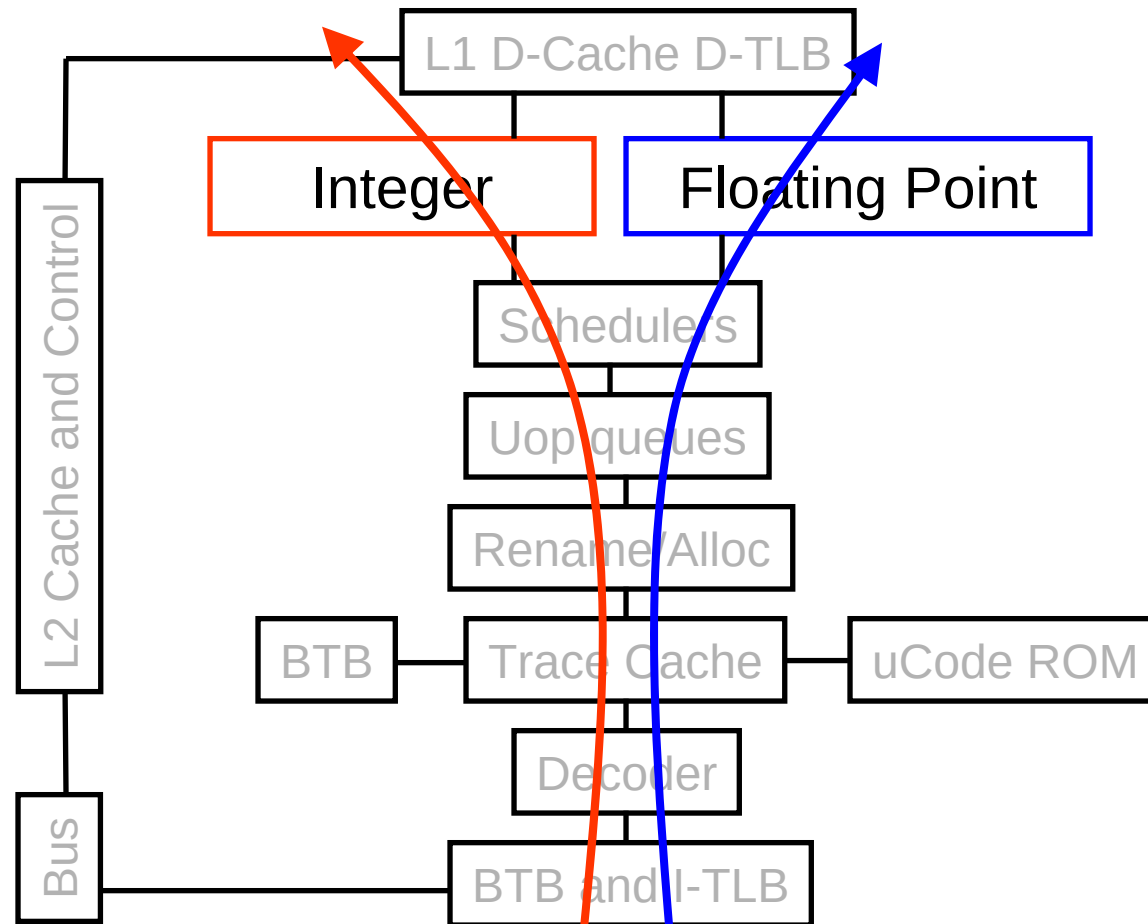
Thread 1: floating point

Senza SMT : un solo thread (che usa IU)

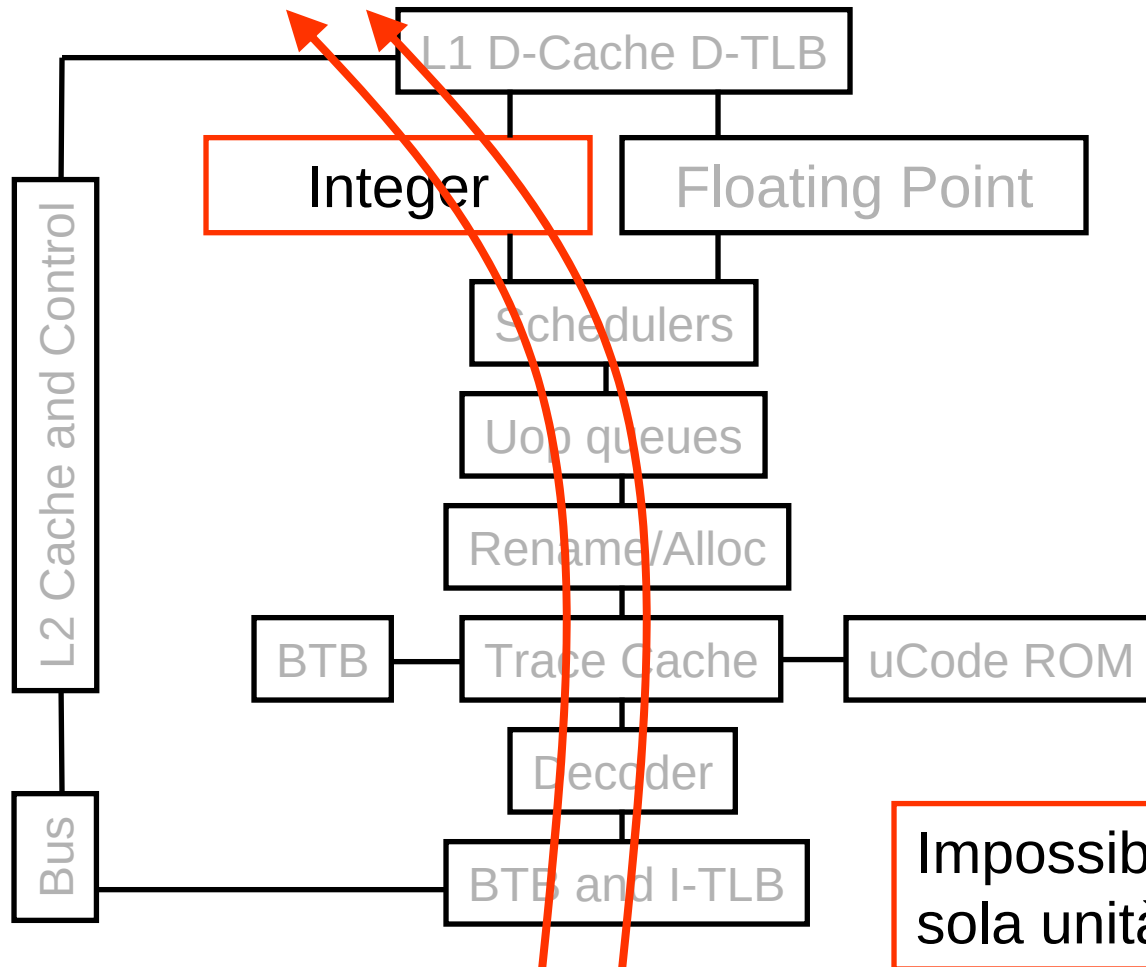


Thread 2:
integer operation

Con SMT : I due corrono in parallelo



Impossibile

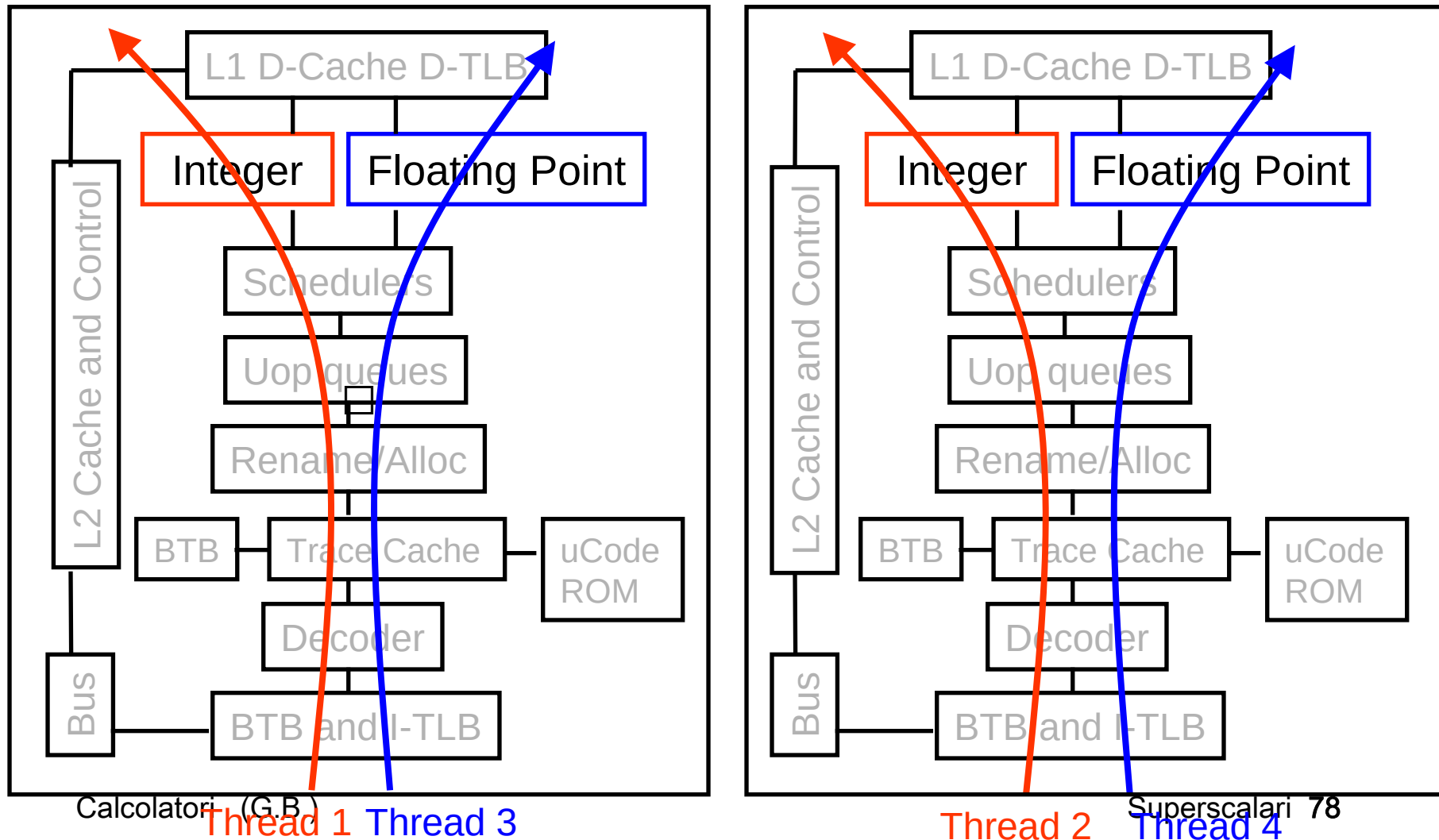


Thread 1 Thread 2
IMPOSSIBLE

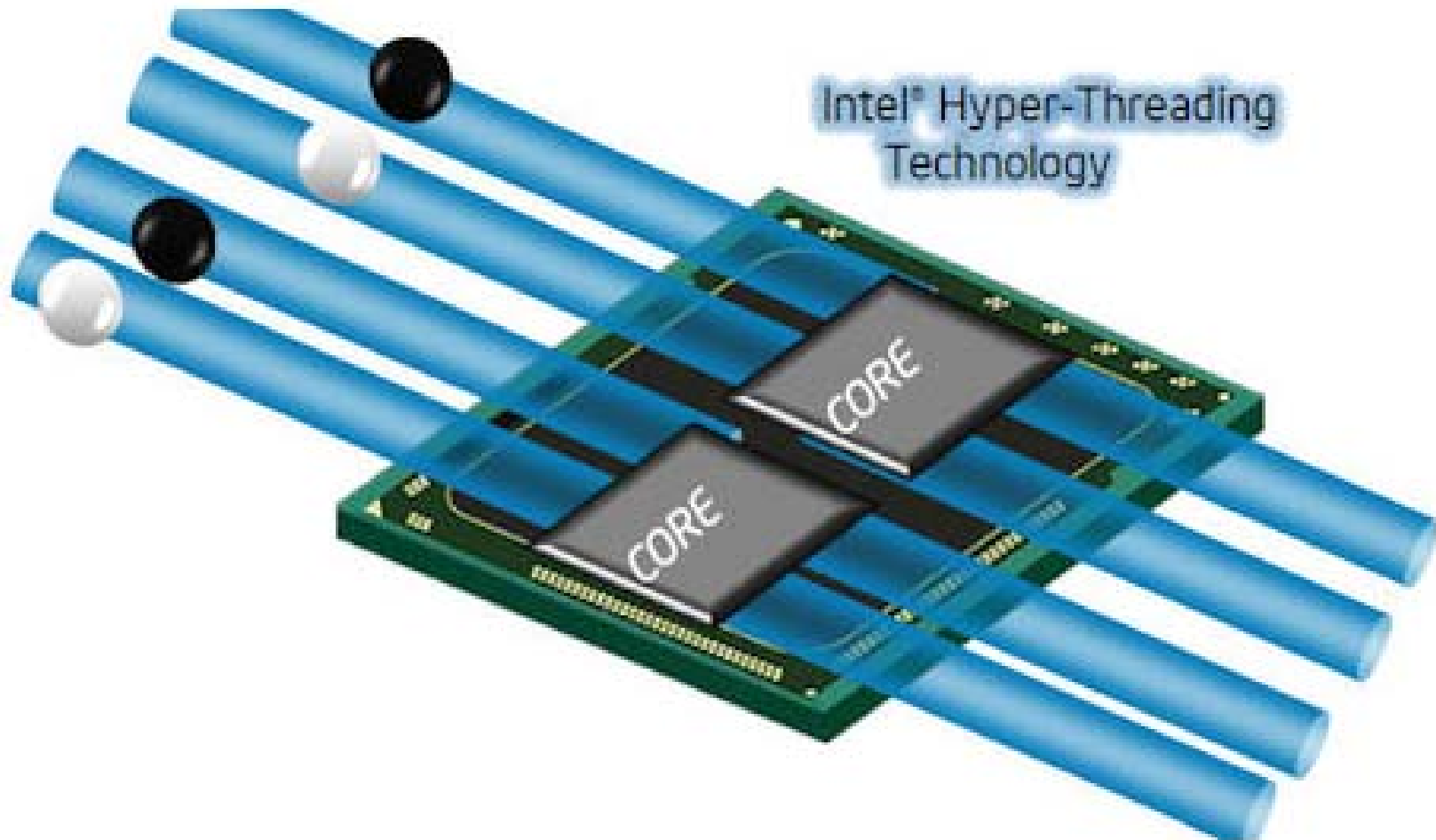
SMT non è parallelismo vero !

- OS e applicazioni vedono il singolo thread come se fosse un processore virtuale
- Le risorse (cache, decoder, ...) sono a comune
 - Il chip ha una sola copia di ogni risorsa eccetto quelle che danno lo stato del processore virtuale
- Migliora il threading (del 30% se va bene)
- Il passo successivo è: multicore/multithread

SMT Dual-core : 4 thr (parallelismo effettivo)



Lo stato delle cose (multicore-hyperthread)

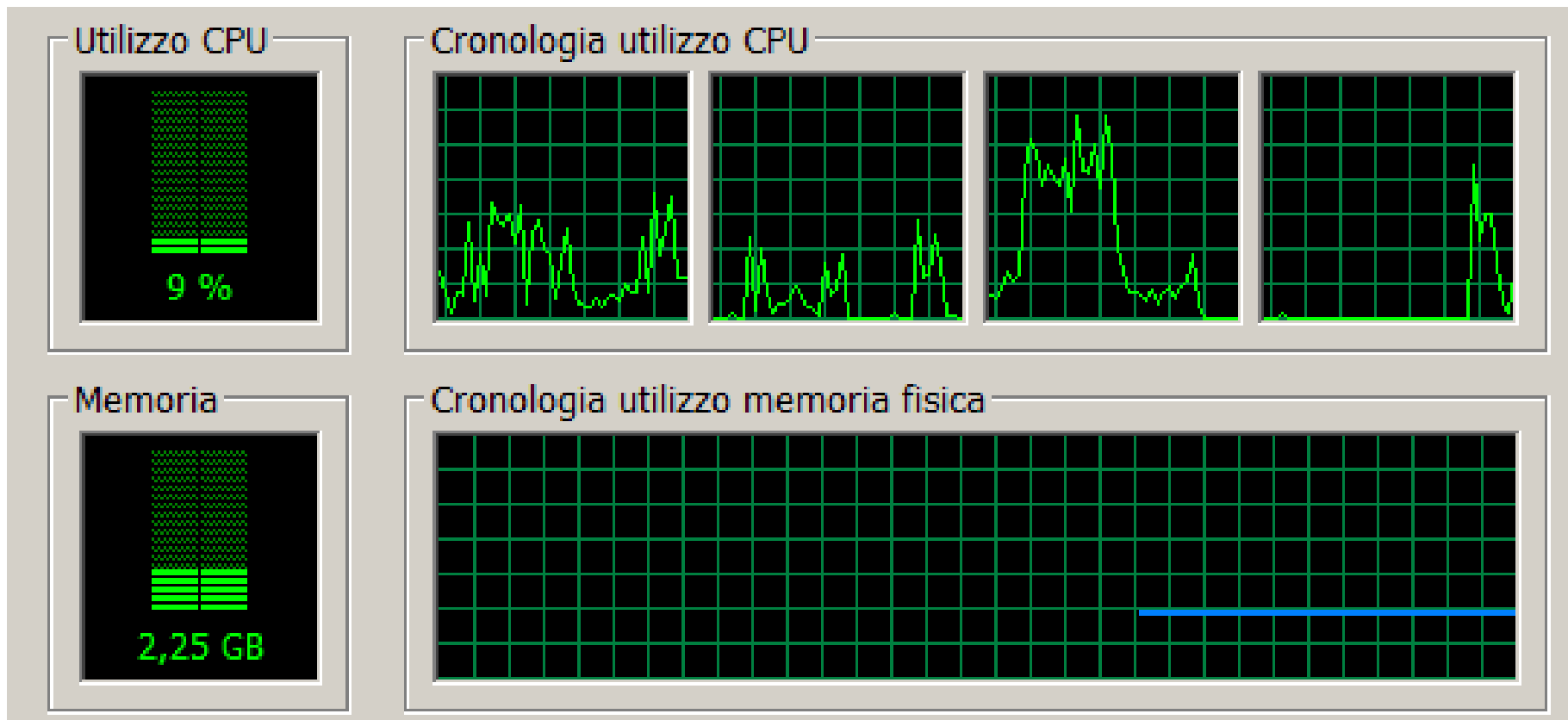


Questo computer portatile

Intel® Core™ i7-2620M Processor

- 2 core
- 4 thread (2 per core)
 - Come se ci fossero 4 processori logici
- 2,7 GHz
- 4 MB cache
- (il modello i7-2630 ha 4 core e 8 thread)

Gestione attività Windows



La legge di Moore

If transistors were people

If the transistors in a microprocessor were represented by people, the following timeline gives an idea of the pace of Moore's Law.



2,300

Average music hall capacity



134,000

Large stadium capacity



32 Million

Population of Tokyo



1.3 Billion

Population of China



Now imagine that those 1.3 billion people could fit onstage in the original music hall. That's the scale of Moore's Law.