Calcolatori Elettronici

ARCHITETTURE MULTIPLE ISSUE

11.1 ARCHITETTURE MULTIPLE-ISSUE

I processori attuali hanno tutti un a pipeline e sono secondo l'ILP architettire intenalmente MISD, ma sono nel loro complesso un processore SISD.

Dopo che tutti i processori hanno integrato la pipeline con lo scopo di ottenere un CPI=1 nei casi migliori, è stato studiato come ottenere un CPI<1 ossia un throughput ancora maggiore. Se lo scopo è di ottenere un CPI<1 (o un IPC>1) nella singola CPU, è **necessario far iniziare più istruzioni in un ciclo ossia ottenere una architettura multiple-issue.** Sono state realizzate diverse soluzioni:

1) Architettura **SUPERPIPELINE:** quando in ogni stadio molte operazioni sono eseguite in porzioni del clock differenti; in questo modo all'interno di ogni stadio il clock è il doppio (o multiplo) del clock della pipeline e più istruzioni possono partire assieme (nello stesso clock) ottenendo quindi un CPI<1. Le architetture che usano pipeline molto lunghe susano tutte porzioni di clock e lavorano in questo modo.

2) Architettura **VLIW very long instruction word** , architettura che almeno in alcune operazioni, esegue un numero fissato di istruzioni >1 , con più unita funzionali in parallelo; tali istruzioni sono formattate in una sola istruzione in modo staticamente determinato dai compilatori. La scelta di avere una architettura VLIW non è fatta a livello di microrchitettura come negli altri due casi ma è una scelta definita della ISA (che comprende istruzioni aggiuntive) e realizzato dal compilatore al tempo di complazione. Ad es. le unita' MMX e poi SSE del Pentium possono elaborare VLIW per istruzioni sui pixel. Sono operazioni vettoriali di solito, che realizzano internalmente un modello SIMD per alcune istruzioni.

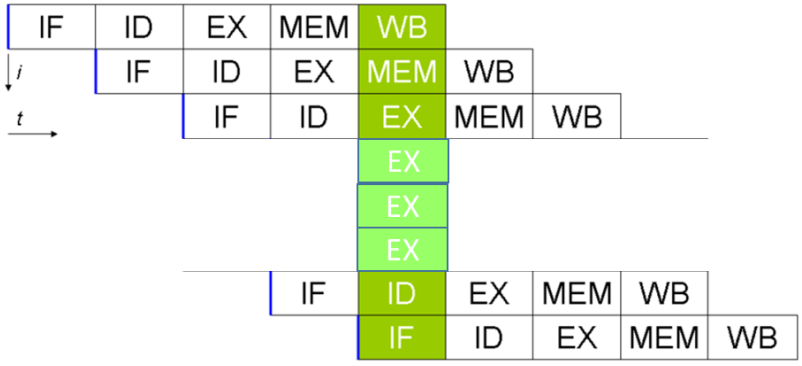


Fig. 1 VLIW

Con una sola IF ed una sola ID per alcune istruzioni speciali (istruzioni VLIW) il compilatore esegue piu' istruzioni uguali.Gli studi VLIW hanno portato allo sviluppo dei processori Itanium con Intel. I processori VLIW trovarono ampia applicazione nel mercato embedded, le famiglie TriMedia di NXP Semiconductors, SHARC DSP di Analog Devices, la famiglia C6000 di Texas Instruments e la famiglia ST200 di STMicroelectronics sono soluzioni VLIW. Ora molti processori hanno parte VLIW, i processori AMD ad esempio.

I processori VLIW sono detti a **Static Multiple Issues**, proprio perche' e' deciso staticamente dal compilatore quali operazioni fare in parallelo. Il parallelismo che si ottiene si chiama **IW Issue width** che e' il numero di istruzioni che possono essere compilate assieme in una sola istruzione in linguaggio macchina. Attenzione che un programma compilato per una architettura non puo' essere usato per una altra con un IW diverso, o almeno minore. Se e' piu' complesso il codice e' abbastanza semplice l'hardware da definire (solo replicando le unità di elaborazione) tanto che il processore ATi ha 800 unità floating point in parallelo.

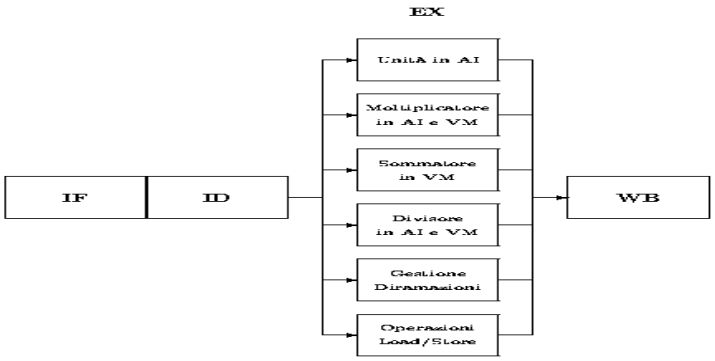


Fig.2 Architettura superscalare

3) ARCHITETTURA **SUPERSCALARE (1987)** (i processori non vettoriali si chiamavano scalari); e' l'architettura progettata per eseguire un numero variabile di istruzioni contemporaneamente in unita' funzionali diverse, decise in modo dinamico (**Dynamic multiple-issue processors**) . la scelta di quali istruzioni eseguire dipende dall'unita' di instruction fetch che acquisice piu' istruzioni assieme (infatti si parla anche di fase di pre-fetch), verifica se alcune istruzioni possono essere eseguite su unità funzionali diverse e disponibili; l'impiego efficiente è ottimizzato dai compilatori che mettono vicine istruzioni che non hanno dipendenze o anti-dimepndenze o dipendenza di uscita (come indicato nelle alee di dato). . È perciò una scelta di microarchitettura in cui

- o sono realizzate proprio pipeline separate ma sincronizzate all'inizio nella fase di fetch,

- o viene realizzata una pipeline unica in cui uno o più stadi contiene più unità in parallelo.

Nella fase di fetch e di pre-fetch l'hardware decodifica le istruzioni e verifica se è possibile eseguire più istruzioni in parallelo.

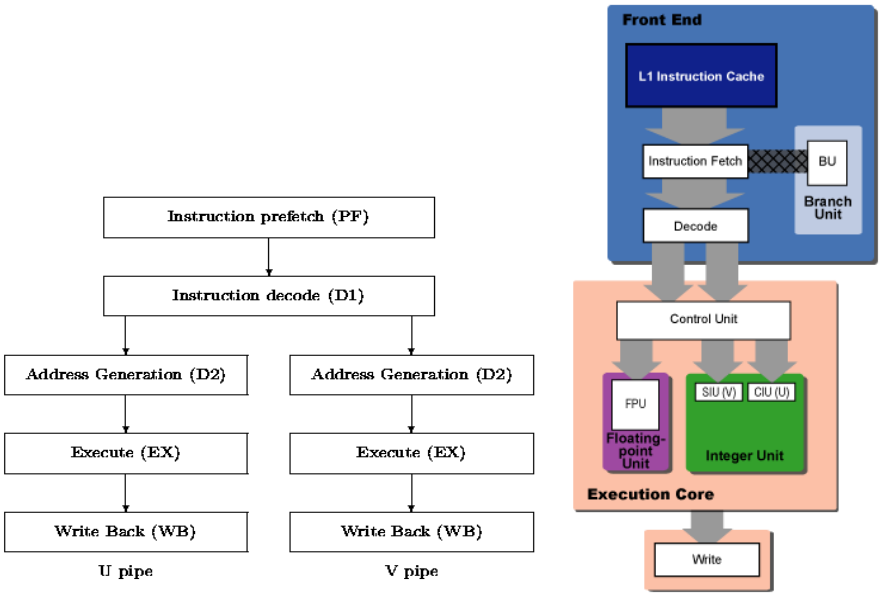


Fig.3 L' architettura del Pentium 1993 è superscalare e ha più pipeline sincronizzate

La prima architettura superscalare Intel è stata sviluppata nel Pentium del 1993 con 2 pipeline diverse:

- la pipeline U capace di eseguire tutte le istruzioni anche quelle col microcodice,

- la pipeline V solo quelle semplici.

Nel Pentium è stato provato che per sequenze di codice intere si poteva ottenere quasi uno speedup di 2, molto più basso invece con cicli ed altri istruzioni non parallelizzabili.

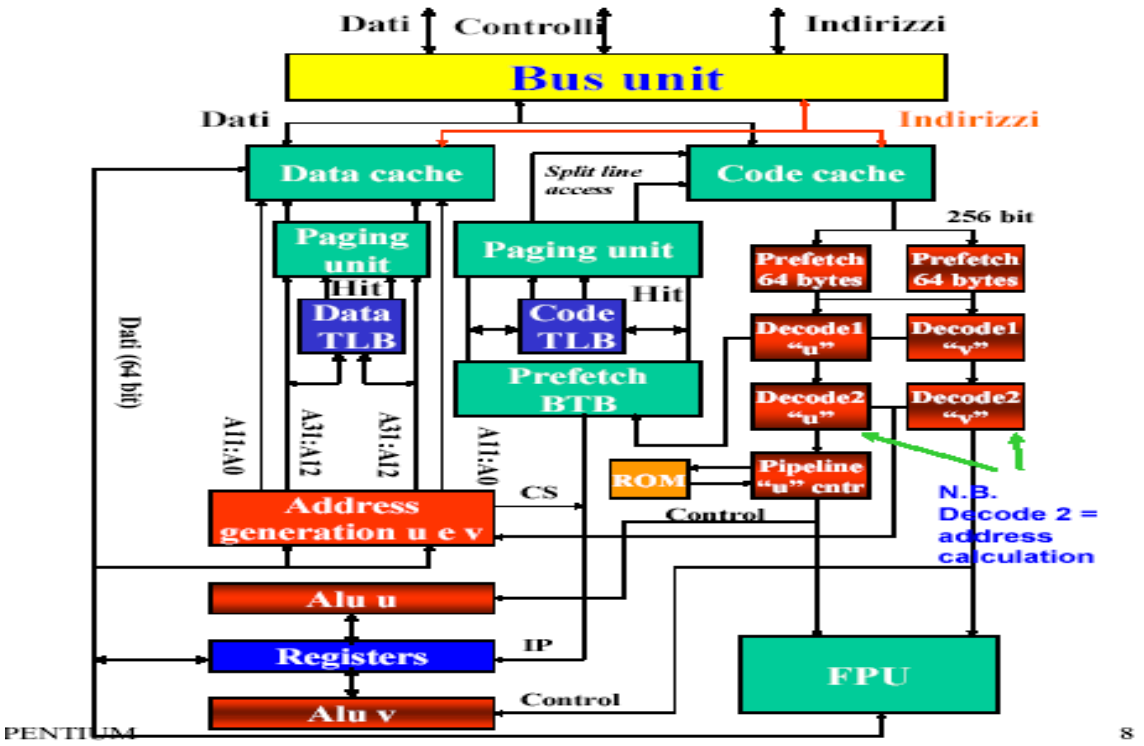


Fig.4 Pentium, architettura completa

Nella figura si vede che il Pentium esegue un doppio prefetch di 64 byte, molto lungo; in questi byte vengono decodificate le istruzioni e se si vede che le istruzioni non hanno dipendenze e possono essere messe assieme alcune passano nella pipeline U ed altre nella pipeline v ottendo (deciso a run-time) una superscalarità di 2. E quindi 2 istruzioni per clock. Le due peipleine sono diverse perche' la pipeline u decodifica anche istruzioni complesse usando il microcodice (si veda il blocco ROM in figura); le due istruzioni usano le due alu U o V o una delle due puo' usare l'unità FPU. Poi nella fase di write back entrambe le pipeline si sincronizzano sui registri che quindi sono a multipla porta per essere scritti correntemente da due pipeline (non possono esistere WAW o scritture di due istruzioni sullo stesso registro!).

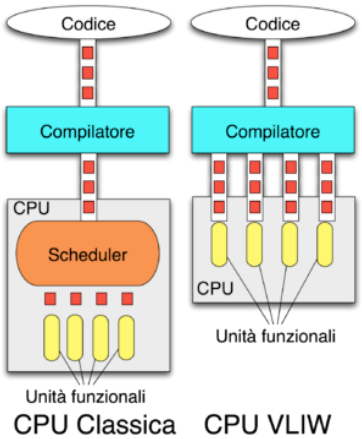


Fig. 5 superscalarità vs VILW

Si noti che le architetture VLIW che fanno una multiple issue statica sono piu' semplici da realizzare, possono avere piu' unità in parallelo ma si usano piu' rararmente; le superscaltità con piu' pipeline sono piu' complesse ma si usano in modo dipendente dai dati e dalle istruzioni. Le GPU con CUDA hanno un linguaggio che simile a quello vettoriale delle VLIW ma usano superscalarità ed hanno una unità interna, lo scheduler (che si trova nella fase di rpefetch o di execute) che capisce quando sfruttare la super-scalarità.

Le architetture superscalari sono molto piu' complesse ma piu' efficiente e ora tutti i processori general-purpose sono superscalari, magari con alcune isttruzioni VLIW; processori embedded e grafici possono usare soluzioni miste di tipo diverso ad esempio l'ATi e' VLIW e NVIDIA e' sueprscalare.

11.2 ESECUZIONE SPECULATIVA

La soluzione successiva è stata quella di non bloccare mai le pipeline se possibile, anche attraverso soluzioni di **esecuzione speculativa.**

Lo scopo primario è di non porre in stallo la pipeline cercando di evitare le alee RAW e le alee WAW e WAR in caso di multiple pipeline.

Una prima **soluzione statica è data dai compilatori ottimizzati** che prevedono un **riordino delle istruzioni** per ottimizzare l'uso delle pipeline e delle unita' superscalari (come quella non bilanciata del Pentium). Il codice viene diviso in ***blocchi (SLOT)* di codice lineare senza salti** e strutture di controllo e viene ottimizzato l'ordine. Questi blocchi sono gli elementi di base: i blocchi poi che devono essere sequenzializzati per forza compongono un **thread**.

Si dice **Esecuzione speculativa** se l'esecuzione di un blocco avviene prima di sapere se il blocco deve essere eseguito (cioè se la struttura di controllo precedente lo prevede). È necessario il supporto sia del compilatore che di hardware dedicato. I risultati vanno in registri temporanei (**scratch register)** che poi vengono copiati sui registri veri solo se il blocco doveva davvero essere eseguito. Esiste una tabella (**scoreboard**) che tiene traccia di che blocco è eseguito o è in esecuzione.

Avviene quindi internalmente una **rinomina dei registri in scratch register** (o **register alias table**)che è necessaria sia per non modificare i registri veri (prima di sapere l'istruzione era da eseguire o no) sia per evitare alee di tipo WAW e WAR quando più unità lavorano in parallelo (superscalarità) e in più pipeline.

Impiegando il paradigma speculativo, durante l'esecuzione di più istruzioni in parallelo all'interno di una pipeline superscalare alcune istruzioni non possono essere eseguite perché devono attendere anche alcuni registri utilizzati da altre istruzioni siano liberi o abbiano il risultato corretto. Per evitare di dover attendere l'esecuzione delle istruzioni in corso per la presenza di registri occupati si usa la rinomina degli stessi.

Quando il processore individua delle istruzioni che non possono essere eseguite per la presenza di registri già occupati il processore assegna all'istruzione altri registri, dei registri temporanei, da poter utilizzare fino a quando i registri di destinazione non siano liberi. Per esempio nel seguente frammento di codice:

add r1,r2,r3

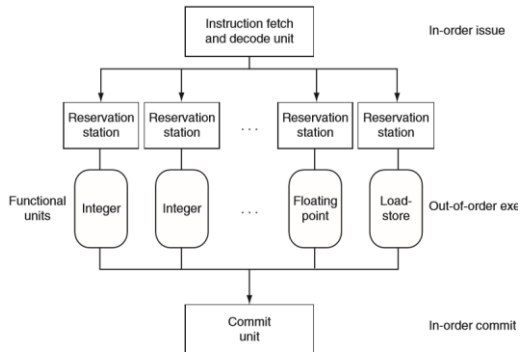
add r1,r5,56

Le due istruzioni possono provocare una alea di tipo WAW; quindi non possono essere eseguite in parallelo pur in realtà non dipendendo dagli stessi dati di partenza. Implementando la rinomina dei registri si può sostituire uno degli accessi al registro r1 con un registro temporaneo t1, in modo che la seconda diventi add t1,r5,r6. La seconda istruzione salva i dati nel registro temporaneo t1 e quindi le due istruzioni possono essere eseguite in parallelo tanto alla fine l'unità di rinominazione provvederà a memorizzare il dato contenuto in t1 nel registro r1 mantenendo la coerenza logica del programma.

Questa attività di rinomina è necessaria nelle architetture più nuove che praticano anche la **esecuzione fuori ordine**.

La rinomina dei registri è fondamentale nell'architettura X86 dato che questa tipologia di processori è dotata di soli 8 registri per i numeri interi e 8 registri per i numeri in virgola mobile. In un'architettura con così pochi registri la probabilità che due istruzioni utilizzino gli stessi registri è molto elevata difatti fin dalla microarchitettura P6 (architettura del Pentium Pro e successori fino la Pentium III) i microprocessori Intel implementavano questa tecnica.

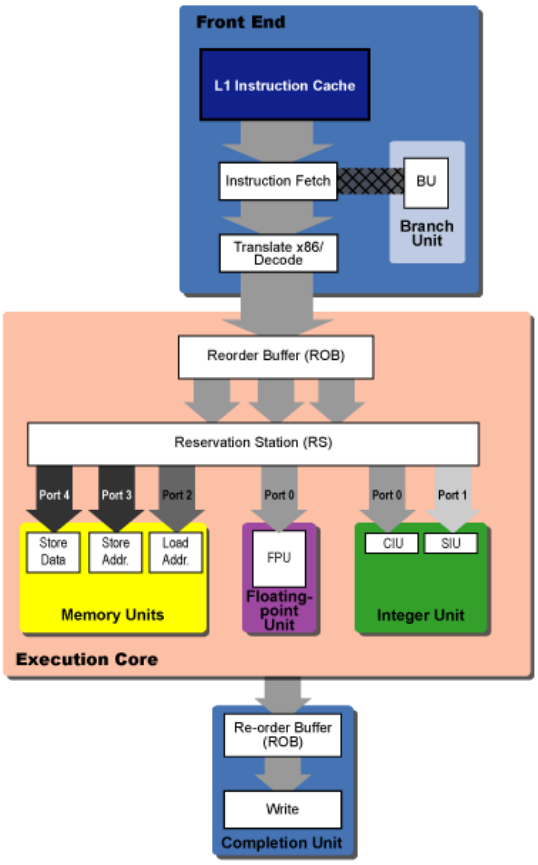
La soluzione seguente (dal Pentium II) è appunto quella di permettere una **Esecuzione fuori ordine:** ogni blocco può essere eseguito anche non in ordine ossia non nell'esatto ordine deciso dal programmatore e dal compilatore, con i soli vincoli che a) i dati vanno prelevati in order (in-order issue) e b) riscritti in ordine alla fine (in-order commit).



Come avviene l'esecuzione fuori ordine?

La rilevazione delle dipendenza tra le varie istruzioni è un compito complesso dato che le istruzioni possono essere vincolate in vari modi, con alee WAW e WAR. Il problema può essere risolto replicando delle unità funzionali oppure eseguendo in parallelo le istruzioni che non hanno dipendenze

. In generale, le istruzioni sono prelevate in un blocco, vengono decodificate per valutare le dipendenze, eventualmente rinominando i registri; e istruzioni che non hanno vincoli (o che hanno vincoli che sono stati risolti) entrano in un buffer (**reservation station**) che alimenta le pipeline e utilizza le unità disponibili



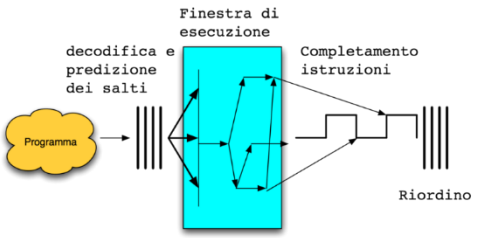


Fig.6 Pentium II

LA reservation station puo' essere una unica o in generale una per ogni pipeline se l'architettura e' anche superscalare.

Come si vede dallo schema del Pentium II (ora mantenuta nelle architetture successive), la pipeline interna e' fatta da piu' stadi divisi in tre insieme:

- **la parte iniziale (Front End)** dove avviene la parte di fetch e decode. In figura si vede la presenza del Fetch, della predizione dei salti, della traduzione di istruzioni complesse x86

in microistruzioni piu' semplici di tipo RISC e di decodifica. In questa fase si verifica anche se le microperazioni semplici, divise in slot possono essere eseguite in parallelo o no.

- la parte centrale o **Execution core** contiene gli skratch register e lo scorebord per tenere traccia delle esecuzioni. L'Intel ha in questa parte la RAT (register alias table) e mette le istruzioni in un ROB (reorder buffer) . Questa e' una tabella circolare che tiene traccia di un ID di ogni istruzione (o microistruzione che sia), dei registri ALIAS usati e dello stato della istruzione se in attesa o in fase di esecuzione o conclusa. Le istruzioni entrano nella reservation station che fa da scheduler e manda in esecuzione l'istruzione sulla unità libera disponibile. In questo modo si usa anche un paradigma di "cache non bloccanti"; se una istruzione ha una miss e deve attendere, rimane in ttesa nel ROB e se una altra istruzione puo' essere eseguita viene posta in esecuzione.

-Al termine ci sono altri stadi della pipeline nella **Commit Unit o Completion Unit**. In questa unità alla fine delle pipeline vi è un accesso al **ROB (reorder buffe**r) che e' un registro circolare a doppia porta, le istruzioni entrano all'inizio e ritornano, ossia cambiano di stato quando sono completate; utilizzando le etichette collegate alle istruzioni ricostruisce l'ordine cronologico delle istruzioni facendo uscire dal processore i risultati delle istruzioni con l'ordine cronologico impostato dal programma. Questa unità inoltre provvede ad eliminare le eventuali operazioni eseguite erroneamente dal processore, ossia quelle speculative che ad esempio a causa di una errata predizione dei salti o di una non predizione sono state eseguite per sbaglio.

Si ricorda che la presenza di unità di predizione dei salti implica che il processore spesso esegua delle istruzioni presupponendo che il processore esegua (o non esegua) un certo salto. Ma, se la previsione fornita dall'unità di predizione dei salti si dovesse dimostrare non corretta le istruzioni eseguite erroneamente vanno eliminate per preservare il corretto funzionamento del programma.

AL termine viene eseguito il Write back sui registri originari.

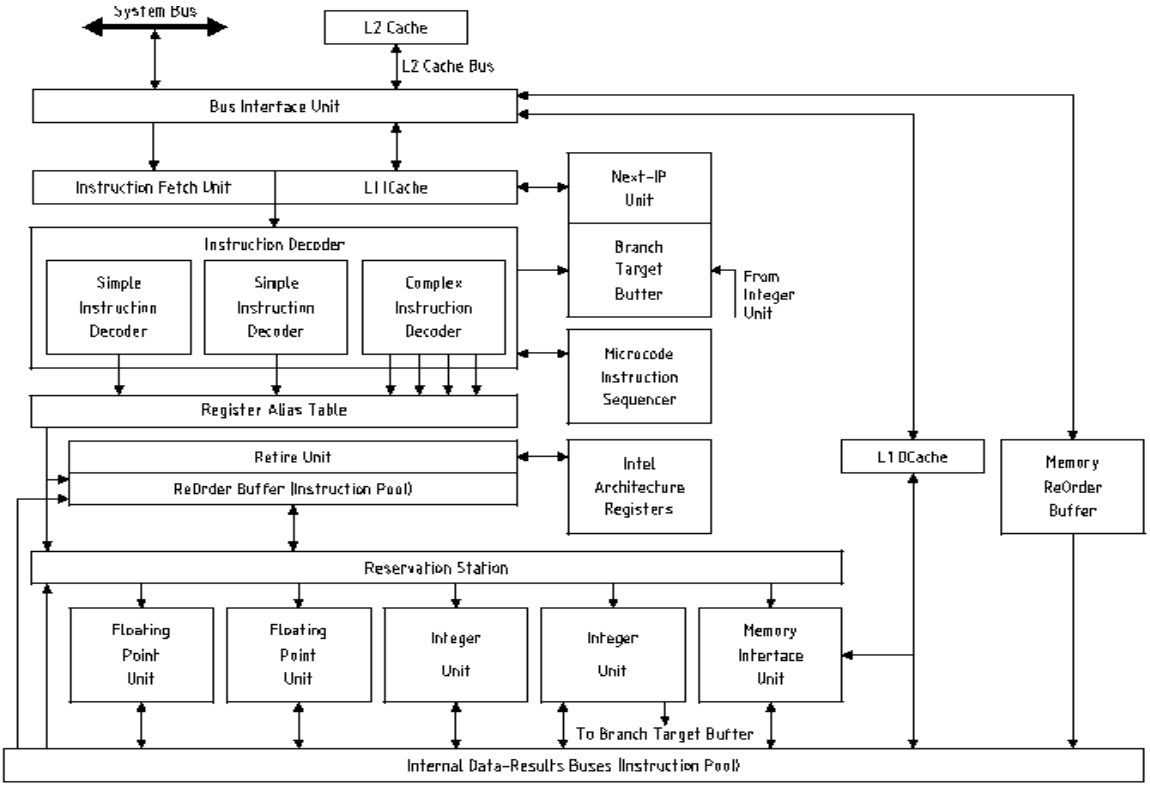


Fig7 Il Pentium II

Attenzione: dal Pentium II la microarchitettrua interna e' stata fortemente modificata per riprogettare il processore in modo RISC pur mantenendo l'ISA CISC.

Il repertorio di istruzioni dell'architettura Intel non si presta bene all'esecuzione fuori ordine perchè nate per essere CISC. Le istruzioni dell' IA-32 vengono allora tradotte in sequenze di **micro operazioni** (uop) in fase di decodifica: ogni uop è in forma di *tripla* (ha due sorgenti e una destinazione logiche) come nelle macchine RISC, durante la fase di fetch (collegata alla L1 cache di istruzioni).

Osservando ad esempio la architettura del Pentium II, dopo la fase di IF**, nella fase di instruction decoder, superscalare a 3 vie può decodificare fino a tre uop due semplici ed una complessa col microcodice**, e nel frattempo interrogare la BTB.

Lo stadio RAT/allocator ha capienza per 3 uop (ne può incamerare/rilasciare 3 operazioni a ciclo di clock). Il RAT, register alias table, esegue il register renaming e di fatto mette in corrispondenza i registri EAX, EBX etc con registri interni ortogonali r0, r1… etc. Le uop vanno nell ReOrder Buffer (ROB) che può contenere fino a 40 uop. I registri fisici sono nel ROB e sono 40.

La funzione dello stadio RAT/allocator è di 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. Il RAT/allocator è l'ultimo stadio *in ordine*.

Le istruzioni anche in fuori ordine entrano nella reservation station e appena possibile usano una delle unità in parallelo e quando eseguite tornano nella *retire unit*, che ritira le istruzioni nell'ordine originario rimettendo i risultati nei registri intel originari.

Il Pentium 4 è stato l'ultimo esempio di processore progettato da Intel usando tutti i concetti di ILP con una microarchitettura chiamata **NetBurst**, architettura nata originariamente per spingere il processore fino a frequenze di 10 Gigahertz (in realtà il più veloce Pentium 4 non è arrivato ai 4 GHz) e difatti dotata di pipeline molto lunghe che possono arrivare fi**no a 31 stadi** con alcuni stadi che servono solo per trasferire le informazioni tra diverse unità.

Pipeline così lunghe subiscono penalizzazioni elevatissime in caso di salti non predetti correttamente o in caso di istruzioni devono stallare per la mancanza di qualche risorsa.

Per ridurre al minimo il problema il Pentium 4 implementa praticamente tutte le tecniche disponibili per ridurre le condizioni di stallo delle pipeline e implementa internamente più pipeline per sfruttare il parallelismo del codice.

11.3 ARCHITETTURE N-CORE CORE I7

Nei calcolatori di nuova generazione la velocità non è tutto, ma è importante sempre di più il **consumo** .

Aumentando la frequenza di clock si è assistito a un proporzionale aumento della potenza dissipata tanto che se il Pentium nel 1993 con una frequenza di 66Mhz aveva una potenza dissipata dell'ordine dei 10Watts, i Pentium 4 Willamett nel 2001 arrivavano a 2Ghz e una potenza di 75W. Il processore Pentium 4 Prescott del 2004 a 3.6 Ghz con una potenza dissipata di 103 Watt ha segnato la fine del progetto Pentium 4 e l'inizio della produzione del processore CORE.

Il Core del 2007 a una frequenza di 2.6Ghz con una potenza dissipata di 95Watt. Dal 2006 nella produzione di sistemi desktop l'Intel ha scelto di cambiare l'obiettivo dal diminuire il tempo di risposta a singolo processore verso quello di offrire più capacità di calcolo inserendo più processori (ora chiamati **core**) nel singolo chip di microprocessore (multicore microprocessor).

Ora la legge di **Moore è solo parzialmente soddisfatta dato che l'incremento di prestazioni nell'ambito multicore non è più solo dovuta all'hardware ma anche al software parallelo e multithread capace di usare questo parallelismo**. I sistemisti del futuro dovranno sempre maggiormente conoscere come sfruttare il parallelismo, le gerarchie di memorie, le nuove soluzioni architetturali (ad esempio nel 2009 si parla spesso di GP-GPU general purpose graphic processing unit) molto più che che nel recente passato.

L'Intel ha chiamato proprio CORE la sua nuova architettura, di fatto b asata sulla replicazione di piu' Pentium. L'architettura "Core" segue le idee del Pentium "Centrino", che è stato il primo

progetto in cui Intel si è preoccupata di ottimizzare il consumo energetico pur mantenendo un elevato livello di prestazioni. La prima versione della piattaforma "Core" era basata su processori dual core costruiti a 65 nm.

Architettura Core (Core duo)

La architettura "Core" di fatto e' molto simile alla architettura logica Netburst ma ha visto l'introduzione di alcune innovazioni rispetto al passato. Innanzitutto la **wide dynamic execution**:. *Ogni core può ora completare 4 istruzioni contemporaneamente, contro le 3 consentite da NetBurst*.

La lunghezza della pipeline si è accorciata molto rispetto a quella impiegata precedentemente, Nel Pentium 4 si era evoluta da 20 fino a 31 nell'ultima evoluzione del Pentium 4. L'architettura "Core" invece è **a 14 stadi a**nche se non ha clock elevati è più bilanciata. La pipeline più corta è meno vulnerabile ai salti. .

Si è esteso il concetto di VLIW con una tecnologia chiamata "**Macro-Fusion"** che consente, di unire tra loro alcune istruzioni per ottenere un'elaborazione più veloce con aumento di prestazioni di circa il 10%.

Cosi' come sono aumentate le **istruzioni SIMD** per il Multimedia anche grazie alla presenza di ben 3 ALU (Arithmetical Logical Unit).

Le memorie sono molto migliorate (Smart cache) con cache L2 condivise da ogni core. Questo poi nel Core i5 e i7 e' diventata la cache di Livello L3 condivisa.

Nella architettura dual CORE, i core avevano ancora una parte condivisa: l' architettura integra 8 unità prefetcher: più precisamente, si tratta di 2 data prefetcher e 1 istruction prefetcher per ciascun core e altri 2 prefetcher come parte della cache L2 condivisa. I prefetchers di memoria monitorizzano costantemente i modelli d'accesso di memoria, tentando di prevedere se qualche dato possa essere inserito nella cache L2, nel caso in cui questi dati possano essere richiesti successivamente. Questo meccanismo e' stato spostato su L3 per i processori seguenti.

Infine nell'architettura Dual-Core si passa dalla IA-32 alla estensione al 64 bit conosciuta come **EM64T** .

Dal **Core 2 si è passati al Core 4, al Core-i7** , con tutte le architetture a partire dalla Nahlem e alle versioni successive.

Core i7

L'Architettura Nahlem introdotta nel 2007 per il processore Core-i7 (il Core i5 e' molto simile) ha 4 core interni e per la prima volta integrata il memory controller ossia la parte alta del chipset.
Ha integrate anche le cache di terzo livello.

Attenzione che Intel con il Core i7 ha deciso di dare nome in codice diverse alla architettura del processore in vendita e alle microarchitetture interne, ma di fatto mantenere sempre il nome Core i7.

La microarchitettura interna si e' chiamata nel tempo

- **Nahlem,**
- **Webstmere,**
- **Sandy Bridge,**
- **IvyBridge,**
- **Haswell,**
- **Broadwell,**
- **Skylake**.

Tutte con un numero di core variabili da 2 a 10 e di thread epr core da 2 a 20.

I nomi dei processori in commercio sono diversi, inzialmente Bloomfield del 2008 con 4 core e d architettura Nahlem,. Nel 2015 la 12° generazione di Corei7 si e' chiamata Skylake, con 4 varianti di microarchitettura interna.

L'ultima generazione attuale si **chiama nel 2017 Caffe Lake con 6 core** (sia nel core i5 che core i7 ma con un numero di thread diversi in parallelo) .

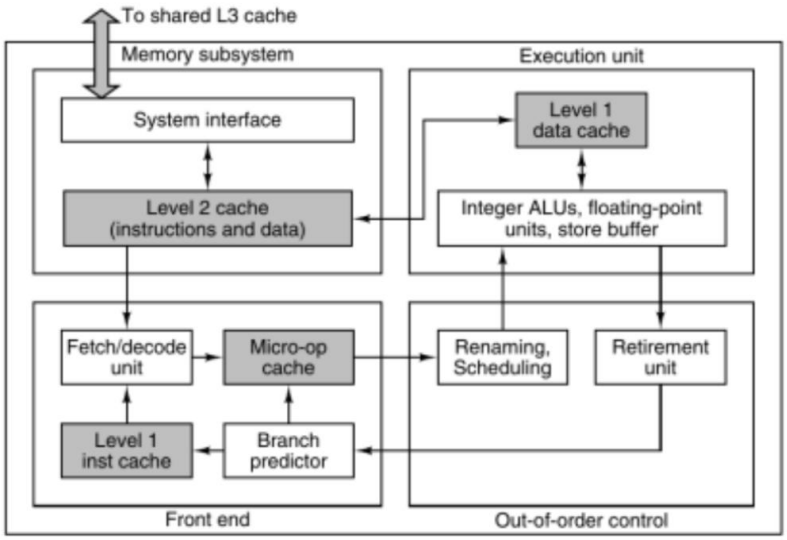


Fig. 8 Architettura semplificata di un core del Core i7 Sandy Bridge.

Ogni core è costituito da 4 parti principali: Il **Memory subsystem**, il sottosistema di memoria locale del core con l'interfaccia verso l'esterno ; e poi il **front end**, **l'out of order uni**t e la **execution unit.** La architettura e' molto simile a quella precedente

Osservando bene la pipeline come nella figura che segue si puo' osservare partendo dalla parte **Front-end:**

- **1. Instruction Fetch:** comprende un **Multi-level BTB** che ha un ritardo di miss-prediction di 15 clock; se la predizione e' corretta vengono caricate dalla **Instruction cache** una linea di istruzioni (cache L1 a 32KB), lunga 16 byte. Nel blocco c'e' anche il TLB che come la Cache L1 e' a 4-vie associativa. LA cache e' quindi ad indirizzo fisico ed indicizzazione virtuale. Nello stadio il **pre-code instruction buffer** prende 16 byte di istruzioni e le divide nelle istruzioni x86 che possono essere di lunghezza diversa che vengono inserite in una coda di 18-entry **instruction queue**, con una piccola cache di **micro-code** chiamata **cache di livello 0 L0.**
- **2. Micro-op decode**. Le istruzioni x86 sono trasformate in micro-op di tipo RISC. Le semplici, (in cui la traduzione e' 1 a 1) possono entrare in uno dei 3 decodificatori semplici, mentre le istruzioni piu' complesse entrano in una parte micro-programmata (unità **Micro-code**). Quindi di fatto decodifica assieme 4 micro-operazioni. Esiste in questa parte anche un **Loop stream detector**, che se vede alcune istruzioni in loop (un numero basso fino a 28 istruzioni) le manda in esecuzione senza dovere di nuovo fare fetch e decodifica.
- **3**. La parte centrale e' la **unità fuori ordine** per fare la **dynamic instruction issue**: Le istruzioni passano per il RAT (**Register alias table**)che rinomina i registri, e per il **reorder buffer** con 128 entry che tiene conto come **scoreboard** delle istruzioni che dovranno poi essere ritirate in ordine nel **Retirement register file (come fase di commit finale)**. Dal ROB poi entrano nella **reservation station con 36 entry.** A questo livello diventa una **superscalarità di 6-vie** mettendo in esecuzione in parallelo al massimo 6 istruzioni (6 micro-op), r con le alu e 3 per le memorie per load o store. Le 3 operazioni di ALU che possono andare in parallelo se non ci sono dipendenze possono anche sfruttare la parte SIMD SSE o la parte floating point.. Le ALU sono alimentate dal register file che invece di avere gli 8 registri dell'ISA (EAX…etc) **ha 2 banchi di 128 registri uno per le operazioni intere ed una per le operazioni FP.** Le operazioni di memoria che vanno in parallelo 2 load ed 1 store fanno dentro ad un buffer di priorità per collegarsi alla Cache.
- **Le cache di dato sono una cache L1 di 32KB con proprio TLB**; una cache L2 unificata (che come si vede e' collegata anche alla cache L1 di Istruzioni) ed una cache L3 unificata tra tutti i core.

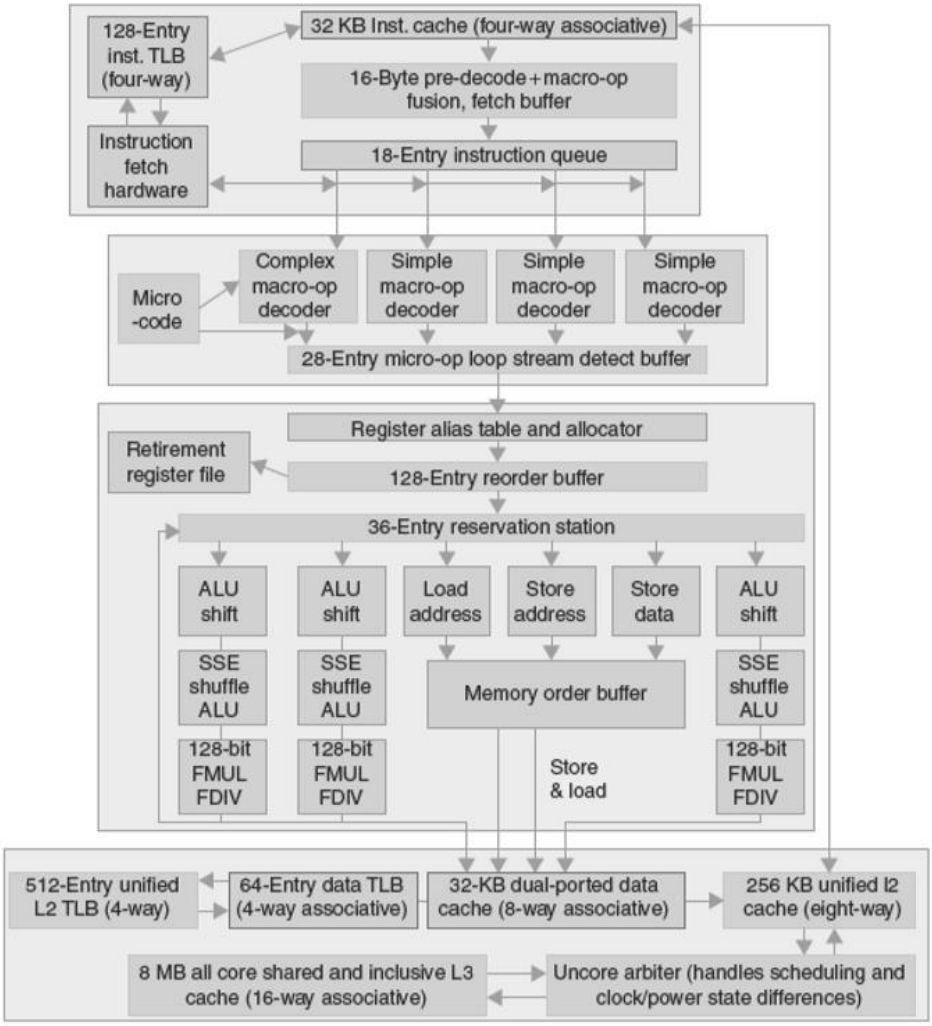


Fig. 9 Pipeline del core i7

L'obiettivo dell'ILP fuori ordine è di avere l'occupazione al 100% delle risorse con questa ottimizzazione fatta in hardware in ogni blocco di istruzioni, **riuscendo quindi a gestire più di 150 istruzioni assieme.** Le alee WAW e WAR sono evitate rinominando i registri dell'ISA. Se ci sono dipendenze vere tra i dati, come RAW o attese di memoria, le micro-op rimangono nel ROB fino a che le risorse non sono disponibili.

Con questa architettura, 6 unità in parallelo possono concludere 1 istruzione per clock. Nel Core i7 a 3Ghz, il throughput massimo sarebbe di 18GFLOPS (18 miliardi istruzioni FP al secondo), che non viene mai raggiunto. Ad esempio le memorie anche cache hanno bisogno di 3 o più TCK per essere lette in caso di hit. La branch prediction non sempre predice correttamente.. etc.

Le operazioni di memorie vengono passate ad un memory buffer. Per andare nella parte di Memoria di dati che come si vede dalla figura comprende

- **Cache L1 dual ported** di **32KB per i dati 8 via associativa a blocchi di 64byte**, una TLB a 64 entry sempre 4 vie- associative ; La cache dati L1 è una copia della cache L2; lavora in write back rispetto a L2. La cache riesce ad eseguire 2 letture ed 1 scrittura in ogni ciclo di clock se sono fatte in banchi diversi. La cache L1 riesce inoltre a fare fino a 10 richieste pendenti alla cache L2.
- **La cache L1 di istruzioni** come visto prima ha stessa dimensione ma solo4 vie associative)
- **Una cache L2 unificata, dati** con quella delle istruzioni a 256KB a 8 vie associativa con linee a 64 byte e un TLB per la cache L2 a 512 entry.
- **La cache L3** che è memoria condivisa su cui si applica una evoluzione del MESI poi a sua volta collegata alla memoria centrale.
  La L3 dipende dalle realizzazioni ed è da 1 a 20Mbyte a 16 vie con linee a 64byte. Il numero di bit di indirizzi è 46 per gli indirizzi fisici (ma usando la memoria virtuale si può arrivare a 48). .

Il Sandy Bridge ha inoltre migliorato la parte vettoriale SIMD chiamata **AVX (advanced vector extension)** che migliora la SSE e SSE2 delle architetture precedenti. **Il parallelismo interno del processore (le ALU) è a 128 bit** ma sono possibili operazioni vettoriali a 256 bit unendo 2 ALU assieme.

Quando le operazioni sono terminate entrano nella Retirement unit che le mette in ordine. Le scritture sulla cache L1 devono essere per forza in ordine. Fino a quando le micro-op non sono tutte nella unità di ritiro il blocco non viene considerato concluso, tanto *che se per caso arrivasse una interruzione in questo periodo, tutto il blocco verrebbe rieseguito*. Esistono poi meccanismi sofisticati, come uno chiamato store-to-load e la disambiguazione per evitare che i dati sulla cache L1 non siano corretti a causa della esecuzione fuori ordine. In conclusione però i dati nelle cache L1 ed L2 sono sempre riscritti in ordine.

Sandy bridge: Riassunto

- L0 cache micro-op
- L1: 32 kB data + 32 kB instruction L1 data cache (3 clocks) 8-way associative, block of 64byte, L2 cahce 3 way associative;
- L2: 1 cache 256 kB L2 cache (8 clocks) per core 8-way, block 64byte
- Shared L3 cache – 3-8-20 MB (25 clocks)

I core del corei7 sono collegati tra loro in shared memory sulla memoria L3 mediante una **rete ad anello (topologia ring)** per cui tutte le cache L2 sono collegate alla cache L3 (dette anche **LLC Last level cache**) anche essa unificata dati ed istruzioni. Nella rete ad anello al richiesta passa da una unità all'altra fino a che non giunge a destinazione. La rete ad anello

viene impiegata anche per mantenere la coerenza sulla cache con un protocollo simile al protocollo MESI.

Alla rete ad anello viene anche collegata l'unità grafica, ossia al GPU integrata.

In totale la **architettura e' una pipeline a 14 stadi**.

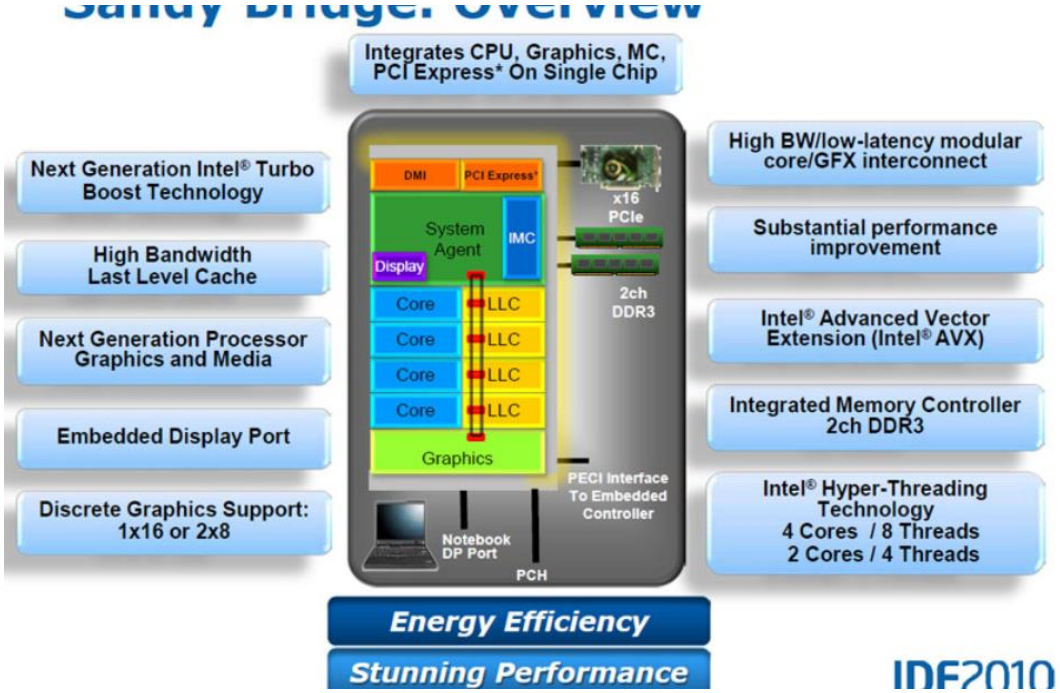


Fig. 10 core i7

Con la architettura del Core i7 si ottengono risultati in prestazioni molto buoni, che dipendono anche dello specifico Core i7 in termini di microarchitettura e di clock.

Si veda in figura i benchmark SPEC200'0 del Corei7 920

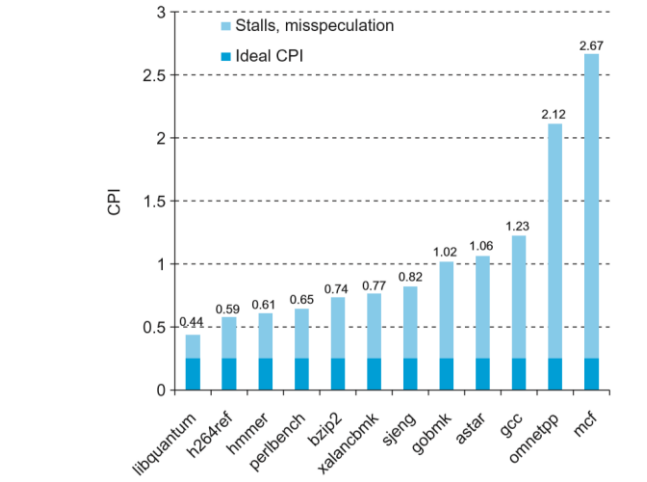


FIGURE 4.78 CPI of Intel Core i7 920 running SPEC2006 integer benchmarks.

Fig. 11 core i7 benchmark

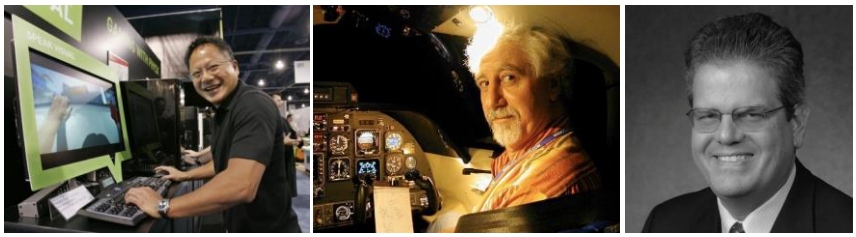
11.4 ARCHITETTURE GRAFICHE

Le architetture che si stanno affermando sono anche differenti da quelle CISC o RISC come quelle della famiglia Intel o di famiglie simili (come ad esempio le CORTEX A9).

Uno spazio particolare è ora occupato dalle **architetture SIMD che hanno all'interno molte unità di elaborazione e sfruttano il parallelismo dei dati**. La casa costruttrice che ora è la più importante al mondo per architetture grafiche è l'NVIDIA che fornisce processori di dimensioni e funzioni diverse.

Nvidia è entrata sul mercato della grafica nel 2001 con la GeForce 3 orientata alla grafica per realizzate in fretta algoritmi di rendering 2D e 3D.

L'azienda è stata fondata nell'aprile del 1993 da **Jensen Huang** (attuale presidente e CEO), trentenne originario di Taiwan con esperienze presso LSI Logic e AMD, Chris Malachowsky (HP e Sun Microsystems) e Curtis Priem (IBM e Sun Microsystems).



- I fondatori: Huang, Malachowsky e Priem

Nel 2008 è entrata sul mercato la **GeForce 8800 con 128 processori in parallelo divisi in 8 core**. Con questa architettura è stato introdotto anche il CUDA (**CUDA** era per **Compute unified device architecture**) come ambiente di sviluppo per GPU basato sul C.

Le GPU sono ormai usate come processori general-purpose (da cui il nome **GP-GPU**), ma sono ancora diverse dalle CPU per molti motivi

a) Di solito le GPU sono acceleratori addizionale alla CPU; in questo modo non devono eseguire tutti i task delle CPU come ad esempio il supporto al sistema operativo;
b) LE GPU nascono per lavorare bene con problemi di dimensioni pari alle immagini o a video streams (gigabytes max) perche' poi hanno memorie limitate; per problemi piu' piccoli non conviene e per piu' grossi la memoria non e' gestita con memoria virtuale e con i meccanismi delle CPU.
c) LE GPU di solito non hanno una architettura di gerarchia a multilivello, ma cercando di avere un forte parallelismo a livello di thread (anche centinaia di thread indipendenti); quindi le memorie sono piu' orientate alla bandwidth (throughput) che alla latenza , le memorie sono di solito minori anche se piu' veloci, proprio perche' non ci sono tanti livelli di gerarchia;
d) Il modo di lavorare delle GPU e' MIMD a livello di thread all'interno di un programma che nasce SIMD all'interno: e' quindi un processore **Multi-thread SIMD. Sono tanti thread diversi ma che contengono solo istruzioni SIMD**

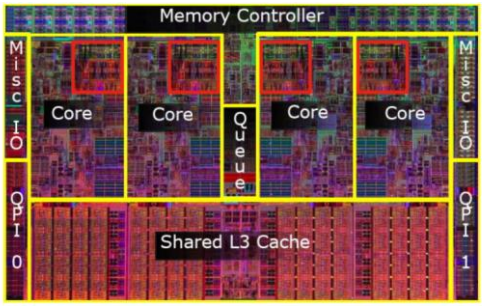
Poi sono entrati sul mercato altri prodotti oltre come il TESLA per il mercato HPC (high performance architecture). Un processore come Tesla è un vero e proprio processore general purpose che può essere considerato come un multicore, i cui core al suo interno sono SIMD. O processori per il mercato Mobile.

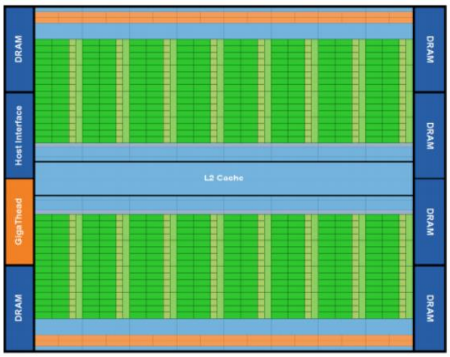
	NVIDIA Tegra 2	NVIDIA Fermi GTX 480
Market	Mobile client	Desktop, server
System processor	Dual-Core ARM Cortex-A9	Not applicable
System interface	Not applicable	PCI Express 2.0 × 16
System interface bandwidth	Not applicable	6 GBytes/sec (each direction), 12 GBytes/sec (total)
Clock rate	Up to 1 GHz	1.4 GHz
SIMD multiprocessors	Unavailable	15
SIMD lanes/SIMD multiprocessor	Unavailable	32
Memory interface	32-bit LP-DDR2/DDR2	384-bit GDDR5
Memory bandwidth	2.7 GBytes/sec	177 GBytes/sec
Memory capacity	1 GByte	1.5 GBytes
Transistors	242 M	3030 M
Process	40 nm TSMC process G	40 nm TSMC process G
Die area	57 mm^2	520 mm^2
Power	1.5 watts	167 watts

Una delle ultimi generazioni di GPU NVIDIA è il **FERMI** che ha da 7 a 15 processori multithread SIMD. Dopo la FERMI e' stata fatta la **KEPLER** ed ora la **MAXWELL**.

Nella tabella si vede un confronto tra un processore Tegra e un processore FERMI. Il FERMI e' realizzato tutto in NVIDIA ed essendo general purpose ha una interfaccia OCI, lavora in modo SIMD e interfacce di memorie veloce . IL Tegra e' molto piu' piccolo in termini di transistor , di area e di potenza dissipata e contiene un processore Dual-core Arm Cortex A9.

Si veda http://www.nvidia.com/content/pdf/fermi_white_papers/p.glaskowsky_nvidia's_fermi-the_first_complete_gpu_architecture.pdf





core i7 vs. Fermi

Il FERMI SM ha

16 processori paralleli chiamati SM **Streaming Multiprocessor** ,

16 unità load/store, 4 SFU (special function units), un register file di 32K word,una memoria interna di 64Kbyte RAM.

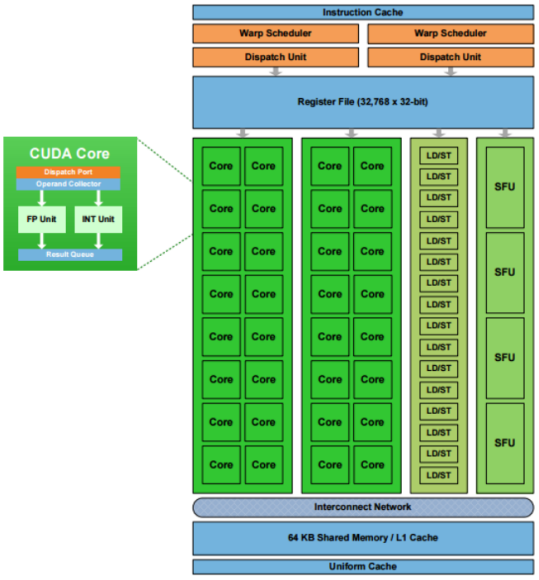
Ogni SM ha 32 core CUDA con la propria L1 privata, quindi in totale ci sono 32 x 16 ossia con 512 core in parallelo.

Sui 16 processori (o 16 thread SIMD lanes) possono essere allocati 32 threads profondi. Come i processori vettoriali, per gestire tutti i thread sono a disposizione all'interno 32.000 registri a 32 **bit, ossia 2048 registri per ognuna delle 16 SIMD Lanes**: ogni thread puo' vedere fino a 64 registri. Anche la architettura delle memorie e' diversa, **ma solo l'ultima versione del Fermi ha la cache**

I processori SIMD erano impiegati inizialmente solo per l'image processing per il parallelismo dei dati ma ora sono molto impiegati per tutte le operazioni paralleli come le reti neurali, i sistemi di pattern recognition, le operazioni sui grafi, parlando ora di GP-GPU (general purpose graphic processing unit).

I processori con SIMD e le GPU sono abbastanza diverse come filosofie ed utilizzo.

Feature	Multicore with SIMD	GPU
SIMD processors	4 to 8	8 to 16
SIMD lanes/processor	2 to 4	8 to 16
Multithreading hardware support for SIMD threads	2 to 4	16 to 32
Largest cache size	8 MiB	0.75 MiB
Size of memory address	64-bit	64-bit
Size of main memory	8 GiB to 256 GiB	4 GiB to 6 GiB
Memory protection at level of page	Yes	Yes
Demand paging	Yes	No
Cache coherent	Yes	No



Nella tabella che segue (del 2010 prodotta da Intel) si mettono a confronto il processore quad-core Intel i7 con estensioni multimediali SIMD e la GPU NVIDIA Tesla GTX 280. Il Core i7 è realizzato da Intel con processo costruttivo a 45 nm, mentre la GPU è realizzata da TSMC, come gli altri prodotti NVIDIA citati in precedenza, ma a 65 nm.

	Core i7-960	GTX 280	GTX 480	Ratio 280/i7	Ratio 480/i7
Number of processing elements (cores or SMs)	4	30	15	7.5	3.8
Clock frequency (GHz)	3.2	1.3	1.4	0.41	0.44
Die size	263	576	520	2.2	2.0
Technology	Intel 45 nm	TSMC 65 nm	TSMC 40 nm	1.6	1.0
Power (chip, not module)	130	130	167	1.0	1.3
Transistors	700 M	1400 M	3030 M	2.0	4.4
Memory bandwidth (GBytes/sec)	32	141	177	4.4	5.5
Single-precision SIMD width	4	8	32	2.0	8.0
Double-precision SIMD width	2	1	16	0.5	8.0
Peak single-precision scalar FLOPS (GFLOP/Sec)	26	117	63	4.6	2.5
Peak single-precision SIMD FLOPS (GFLOP/Sec)	102	311 to 933	515 or 1344	3.0–9.1	6.6–13.1
(SP 1 add or multiply)	N.A.	(311)	(515)	(3.0)	(6.6)
(SP 1 instruction fused multiply-adds)	N.A.	(622)	(1344)	(6.1)	(13.1)
(Rare SP dual issue fused multiply-add and multiply)	N.A.	(933)	N.A.	(9.1)	--
Peak double-precision SIMD FLOPS (GFLOP/sec)	51	78	515	1.5	10.1

- *Confronto tra CPU e GPU*

Sono nate infine architetture speciali o **architetture ibride**:

l'intel ha creato una architettura ibrida (APU) unendo una multicore con una GPU integrata con memoria condivisa; cosi' come NVIDIA ha fatto nascere architetture particolari come ad esempio la architettura Maxwell M40.

Il nuovo acceleratore Tesla M40 composto dalla nuova microarchitettura Maxwell ha le seguenti caratteristiche

TESLA M40 FEATURES AND SPECS:

- NVIDIA Maxwell™ architecture
- Up to 7 Teraflops of single-precision performance with NVIDIA GPU Boost™
- 3072 NVIDIA CUDA® cores
- 24 GB of GDDR5 memory
- 288 GB/sec memory bandwidth
- Qualified to deliver maximum uptime in the datacenter

- See more at: http://www.nvidia.com/object/tesla-m40.html#sthash.5evpJGh8.dpuf

Table 1. A Comparison of Maxwell GM107 to Kepler GK107

GPU	GK107 (Kepler)	GM107 (Maxwell)
CUDA Cores	384	640
Base Clock	1058 MHz	1020 MHz
GPU Boost Clock	N/A	1085 MHz
GFLOP/s	812.5	1305.6
Compute Capability	3.0	5.0
Shared Memory / SM	16KB / 48 KB	64 KB
Register File Size / SM	256 KB	256 KB
Active Blocks / SM	16	32
Memory Clock	5000 MHz	5400 MHz
Memory Bandwidth	80 GB/s	86.4 GB/s
L2 Cache Size	256 KB	2048 KB

Ecco un riassunto finale delle architetture NVIDIA come GP-GPU, mobile ed embedded:

- 2009 **GPU FERMI:** usata nella grafica di film con effetti speciali (Avatar, Harry Potter, Star Treck) e nel supercomuter piu' veloce delle top500 del periodo il Tianhe-1A. Nello stesso anno esce **Tegra-2** per mobile dual-core.
- 2012 **GPU Kepler:** usata per il supercomputer Titan.
- 2013 esce Tegra-4 mobile quad-core
- 2014 **GPU Maxwell** e anche Tegra K1 orientata al Deep Learning
- 2015 esce la Jetson Tx1
- 2016 esce la **GPU Pascal**, il DGX-1 supercomputer per Deep Learning e la Tx2 per l'automotive e per l'AI integrato.
- 2017 la **GPU Volta**.
-

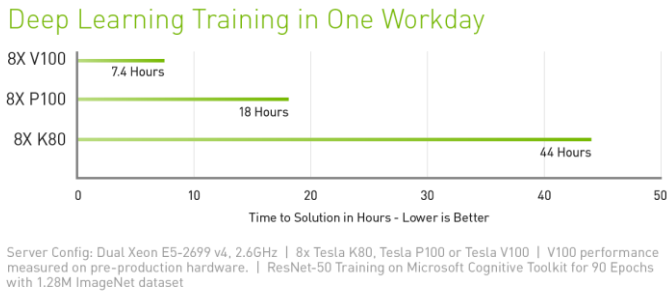
  Ogni architettura (Fermi, Kepler, Maxwell, Pascal Volta) hanno 4 specializzazioni con serie epr i diversi ambiti
- GeForce (PC e gaming)
- Quadro (workstation)
- Tesla (server HPC)
- Tegra (mobile)

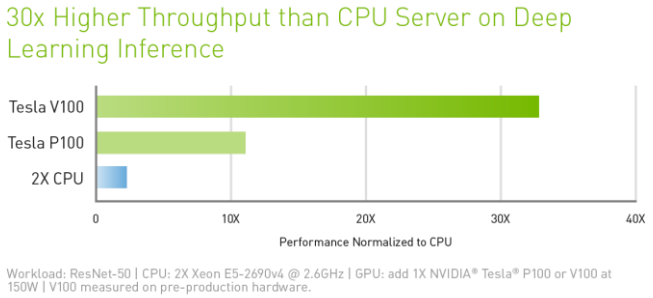
Ora il mercato GPU e' diviso tra piu' produttori i cui piu' famosi sono AMD e NVIDIA

GPU Supplier	Market share this quarter	Market share last quarter	Market share last year.
AMD	29.4%	27.5%	30.8%
Nvidia	70.6%	72.5%	69.1%
Total	100%	100%	100%

Table 1: Market share changes quarter-to-quarter, and year-to-year

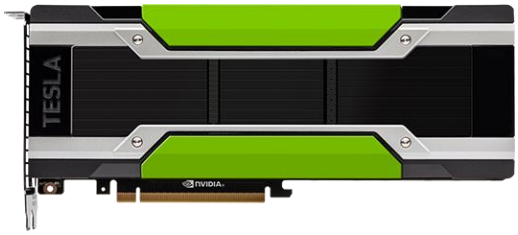
Questa tabella indica i dati di vendita GPU per le soluzioni embedded nel secondo Quartile 2017.
Le prestazioni sono in crescita esponenziale come si vede dai seguenti grafici.





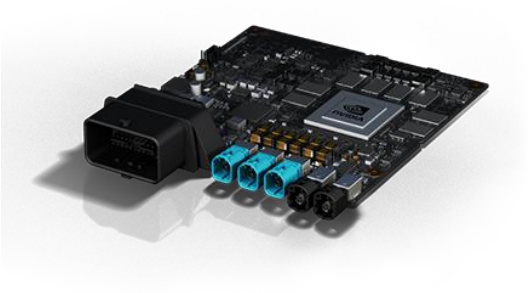
Questi grafici sono epr le prestazioni ottenute per computazioni speciali di Deep Learning ma mostrano come le prestazioni cambiano di architettura in architettura.
Anche le dimensioni, i pesi, il consumo cambia da architettura e da famiglia a famiglia

Nvidia Jetson Tx2



Nvidia Tesla P40

Nvidia Volta V100



Nvidia Drive PX2 settore Automotive

La nuova architettura V100 integra direttamente dentro al chip una parte specifica per il Deep Learning con i tensor core.

Nella tabella che segue sono indicati alcuni aspetti architetturali.

VOLTA SM UNIT

Tesla Product	Tesla K40	Tesla M40	Tesla P100	Tesla V100
GPU	GK180 (Kepler)	GM200 (Maxwell)	GP100 (Pascal)	GV100 (Volta)
SMs	15	24	56	80
TPCs	15	24	28	40
FP32 Cores / SM	192	128	64	64
FP32 Cores / GPU	2880	3072	3584	5120
FP64 Cores / SM	64	4	32	32
FP64 Cores / GPU	960	96	1792	2560
Tensor Cores / SM	NA	NA	NA	8
Tensor Cores / GPU	NA	NA	NA	640
GPU Boost Clock	810/875 MHz	1114 MHz	1480 MHz	1530 MHz
Peak FP32 TFLOPS[1]	5	6.8	10.6	15.7
Peak FP64 TFLOPS[1]	1.7	.21	5.3	7.8
Peak Tensor TFLOPS[1]	NA	NA	NA	125
Texture Units	240	192	224	320
Memory Interface	384-bit GDDR5	384-bit GDDR5	4096-bit HBM2	4096-bit HBM2
Memory Size	Up to 12 GB	Up to 24 GB	16 GB	16 GB
L2 Cache Size	1536 KB	3072 KB	4096 KB	6144 KB
Shared Memory Size / SM	16 KB/32 KB/48 KB	96 KB	64 KB	Configurable up to 96 KB
Register File Size / SM	256 KB	256 KB	256 KB	256KB
Register File Size / GPU	3840 KB	6144 KB	14336 KB	20480 KB
TDP	235 Watts	250 Watts	300 Watts	300 Watts
Transistors	7.1 billion	8 billion	15.3 billion	21.1 billion
GPU Die Size	551 mm²	601 mm²	610 mm²	815 mm²
Manufacturing Process	28 nm	28 nm	16 nm FinFET+	12 nm FFN

[1] Peak TFLOPS rates are based on GPU Boost Clock