Architetture a Confronto

Prestazioni dei calcolatori (1)

Gli elementi che consentono di calcolare

$$T_{esecuzione} = T_{clock} \cdot \left(\sum_{i=0}^{n} N_i \cdot CPI_i \right)$$

T_{clock} periodo di clock della macchina: dipende dalla tecnologia e dalla organizzazione del sistema.

CPI_i numero di clock per istruzione di tipo *i*. Cioè il numero di clock occorrenti affinché venga eseguita l'istruzione di tipo *i*. Se il set di istruzioni contiene istruzioni di tipo semplice, CPI assumerà valori ridotti; se invece le istruzioni eseguono operazioni più complesse si avranno CPI elevati.

N_i numero di istruzioni di tipo *i* (somme, sottrazioni, salti, ecc.). Dipende dal set di istruzioni e dalla qualità dei compilatori (per qualità dei compilatori si intende la capacità dei compilatori di ottimizzare il programma, ad es. eliminare le istruzioni non necessarie). Se il set di istruzioni contiene istruzioni di tipo semplice avremo un N_i più elevato: caso opposto per istruzioni di tipo più complesso che permetteranno di ridurre N_i. Infatti con un set di istruzioni di tipo semplice, occorreranno più istruzioni per svolgere le stesse funzioni.

Al link https://www.agner.org/optimize/
sono disponibili preziose risorse sull'ottimizzazione del software in C e Assembler, tra cui numerose tabelle che riportano le latenze (CPI) delle istruzioni ISA di molte CPU Intel e AMD

https://www.agner.org/optimize/instruction_tables.pdf

Prestazioni dei calcolatori (2)

IPS (**I**instructions **P**er **S**econd) è una misura che serve per valutare le prestazioni di un sistema; indica quante istruzioni vengono eseguite al secondo. Viene comunemente riportata come **MIPS** (**M**illion **I**nstructions **P**er **S**econd):

$$MIPS = \frac{Frequenza\ del\ clock}{10^6 \cdot CPI}$$

Questa misura è affetta da una forte approssimazione:

- Non tiene conto delle possibilità offerte dal set di istruzioni (quanto più il set di istruzioni di una certa macchina è potente tanto più è possibile ridurre la lunghezza dei programmi che vengono eseguiti su di essa),
- Non tiene conto delle percentuali delle diverse istruzioni all'interno di programmi reali (una macchina più veloce in un programma può essere più lenta in un altro).
- Non tiene conto dell'ampiezza dei bus, della presenza di cache o altre tecniche (che si vedranno nel seguito) che ottimizzano i tempi di esecuzione.

Una stima di IPS viene solitamente ottenuta attraverso il **benchmark Dhrystone**, che contiene solo operazioni in aritmetica intera. Un benchmark è un insieme di programmi di prova rappresentativi di una particolare classe di applicazioni.

FLOPS (**Flo**ating point **P**er **S**econd) È una misura che è significativa se si intende utilizzare un programma dove la maggior parte delle operazioni sono di tipo floating point. Tipicamente riportata come MFLOPS, GLOPS, o TFLOPS.

LINPACK benchmarks: Misurano il numero di FLOPS a 64 bit (principalmente moltiplicazioni e somme) per risolvere sistemi di equazioni lineari densi $n \times n$. Un'implementazione parallela chiamata HPL, scritta in C, viene utilizzata per tenere aggiornata la lista dei TOP500 calcolatori (https://www.top500.org).

MIPS a confronto

Processore	Anno	(Dhrystone) MIPS	Frequenza (MHz)
Pencil and Paper	1892	$1{,}19 \times 10^{-8}$	-
Zuse 1	1938	$4{,}24 \times 10^{-8}$	-
ENIAC	1946	0,00289	-
IBM System/370 model 158	1972	0,64	9
Intel 8080	1974	0,29	2
Motorola 68000	1979	1	8
Intel 386	1985	10	33
Intel 486	1989	41	50
PowerPC 600s (G2)	1994	35	33
Intel Pentium Pro	1995	541	200
ARM 7500FE	1996	36	40
Intel Pentium III	1999	1 354	500
AMD Athlon	2000	3 561	1 200
Pentium 4 Extreme Edition	2003	9 726	3 200
ARM Cortex A8	2005	2 000	1 000
Xbox360 IBM "Xenon" Triple Core	2005	6 400	3 200
IBM Cell All SPEs	2006	12 096	3 200
AMD Athlon FX-57	2005	12 000	2 800
AMD Athlon FX-60 (Dual Core)	2006	18 938	2 600
Intel Core 2 Extreme QX6700	2006	57 063	3 330
Intel Core i7 Extreme 965EE	2008	76 383	3 200
AMD Phenom II X6 1090T	2010	68 200	3 200
Intel Core i7 Extreme Edition 980X	2010	147 600	3 300
Intel Core i7 5960X	2014	238 310	3 000
Intel Core i7 6950X	2016	317 900	3 000
AMD Ryzen 7 1800X	2017	304 510	3 600
Intel Core i9-9900K	2018	412 090	4 700
AMD Ryzen Threadripper 3990X	2020	2 356 230	4 350

Classificazione dei calcolatori

(parametri dimensionali al 2018)

Computer usa e getta

Cartoline d'auguri musicali, RFID

centesimi €, alcuni MIPS

Sistemi embedded

in SmartCard, Orologi, Automobili, Elettrodomestici, Hi-Fi, Lettori audio/video, giocattoli, apparati medicali

1..50€, 1..1000 MIPS

SmartPhone e Tablet

Applicazioni mobili di ogni genere

100..1000€, 1..2000 GFLOPS (CPU)

Classificazione dei calcolatori (2)

Consolle da gioco

Videogiochi *100..500€, 250..10000 GFLOPS (GPU)*

Personal Computer (PC)

Desktop o Portatili

500..2000€, 50..5000 GFLOPS (CPU)

Server (e Workstation)

Gestione archivi, centralizzazione servizi, multiutenza, HPC (High Performance Computing), Deep Learning.

5..100K€, 1..5000 TFLOPS (CPU + GPU)

Classificazione dei calcolatori (3)

Cluster (raggruppamento) di più Server

Per scalare verso l'alto prestazioni (in genere il livello di concorrenza) di singoli Server.

10K€..1M€, 1..10000 TFLOPS

Supercalcolatore

Elevata potenza di calcolo per applicazioni scientifiche

50M€..250M€, 1 PFLOPS..1 EFLOPS

Confronto Potenze di Calcolo

https://wordlesstech.com/processing-power-compared
(valido al 18/01/2023)

Come misurare il miglioramento delle prestazioni

Poter valutare le prestazioni di un calcolatore permette di calcolare:

- La bontà di un investimento
- L'adeguatezza del calcolatore al tipo di applicazione

$$Prestazione\ P = \frac{1}{T_{\text{esecuzione}}}$$

$$Speed\ up = \frac{P_A - P_B}{P_B} = \frac{T_B - T_A}{T_A} \quad \text{(del sistema A rispetto a B)}$$

La **legge di Amdhal** afferma che il miglioramento delle prestazioni che si ottiene in un sistema di elaborazione accelerando un qualsiasi sottoinsieme del calcolatore è proporzionale alla percentuale di tempo per cui quel sottoinsieme è utilizzato.

$$T_{es.\,finale} = \frac{p \cdot T_{es.iniziale}}{a} + (1 - p) \cdot T_{es.iniziale}$$

Dove *p* è la percentuale di tempo utilizzata dal sottoinsieme migliorato, e *a* è il fattore di accelerazione.

Esempio: Modificando l'unità che esegue le operazioni in virgola mobile (unità floating point) si riduce a 1/10 il tempo necessario per eseguire le stesse. Si supponga che del tempo di esecuzione solo il 40% venga utilizzato per eseguire operazioni in virgola mobile.

$$T_{es.\,finale} = \frac{0.4 \cdot T_{es.iniziale}}{10} + (1 - 0.4) \cdot T_{es.iniziale} = 0.64 \cdot T_{es.iniziale}$$

$$Speed\ up = \frac{T_{es.iniziale} - 0.64 \cdot T_{es.iniziale}}{0.64 \cdot T_{es.iniziale}} = 0.56 = 56\%$$

RISC VS CISC

La velocità di esecuzione di un'istruzione all'interno della CPU determina in larga misura la velocità della CPU ed è da sempre oggetto di discussione tra due correnti di pensiero.

CISC (Complex Instruction Set): i sostenitori di questa idea ritengono che l'instruction set di un calcolatore debba contenere quante più istruzioni possibili, anche se ognuna di queste richiede più cicli di data path, poiché ciò permette di creare macchine più potenti (es. CPU x86).

RISC (Reduced Instruction Set): i sostenitori di questa idea ritengono che ogni istruzione dell'Instruction Set debba essere eseguita in un solo ciclo (o comunque pochi cicli) di data path: saranno necessarie più istruzioni RISC per ottenere lo stesso risultato di una istruzione CISC, ma il sistema risulterà comunque più veloce poiché non sarà più necessario interpretare le istruzioni (es. ARM).

Le macchine CISC hanno dominato il mercato negli anni '70 e '80 mentre attualmente la tecnologia è fortemente orientata verso soluzioni RISC.

La superiorità di una soluzione rispetto all'altra è comunque un fatto relativo che dipende da fattori di mercato e fattori tecnologici: un radicale innovamento tecnologico potrebbe muovere nuovamente l'ago della bilancia a favore delle macchine CISC.

Principi di progettazione RISC

Tutte le istruzioni del livello ISA vengono eseguite direttamente dall'hardware: in particolare l'unità di controllo all'interno della CPU è **cablata** (realizzata con un circuito digitale sequenziale ovvero una macchina a stati) e non **microprogrammata** (istruzioni associate a microprogrammi composti da micro-istruzioni). Eliminando il livello di interpretazione aumenta la velocità della maggior parte delle istruzioni.

Ottimizzare la velocità con la quale le istruzioni vengono mandate al primo stadio di esecuzione: anche in presenza di operazioni complesse la velocità del processore è determinata dal numero di istruzioni "iniziate" per secondo (**MIPS**: **M**ilions of **I**nstructions **P**er **S**econd). Nella maggior parte dei processori moderni si fa ampio uso di pipelining e parallelismo.

Le istruzioni dovrebbero essere facilmente decodificabili: la velocità di esecuzione delle istruzioni dipende anche dal tempo necessario a identificare le risorse necessarie a eseguirle. Questo processo può essere velocizzato utilizzando istruzioni con **struttura regolare** e **lunghezza fissa**.

Solo le istruzioni Load e Store dovrebbero contenere indirizzi di memoria: in questo modo le rimanenti istruzioni utilizzeranno solo operandi contenuti nei registri eliminando i tempi morti dovuti ai ritardi nella lettura dei dati dalla memoria. Inoltre le operazioni di Load e Store potranno essere parallelizzate in quanto rappresenteranno operazioni a se stanti.

Disporre di molti registri: i registri sono una risorsa fondamentale per ridurre i tempi di accesso alla memoria.

Come aumentare le prestazioni

I progressi tecnologici (aumento del numero di porte per unità di superficie, maggiori velocità di commutazione delle porte, minor consumo di corrente e quindi dissipazione di calore, ...) costituiscono una fonte primaria per il miglioramento delle prestazioni delle CPU.

In ogni caso, indipendentemente dal cambiamento di tecnologia, esistono diversi altri modi per migliorare le prestazioni di una microarchitettura, e ognuno di questi deve essere valutato considerando il **trade-off** prestazioni/costi:

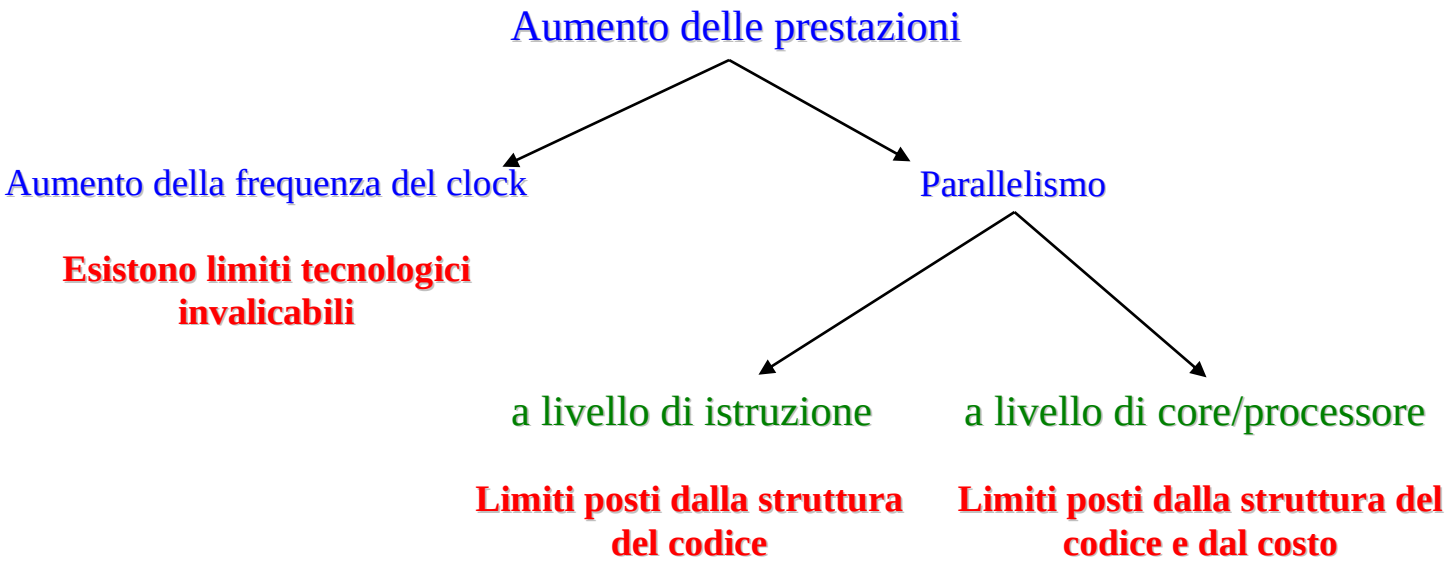
- **Ridurre il numero di cicli** (path di esecuzione) necessari per implementare le istruzioni ISA (Livello della microarchitettura).
- Semplificare l'organizzazione per consentire di **aumentare la frequenza di clock**. Limiti fisici.
- **Parallelismo:** sovrapporre l'esecuzione delle istruzioni (architetture superscalari, pipelining e CPU in parallelo)

Analizzeremo inoltre alcune tecniche utilizzate dalle CPU di recente fabbricazione per migliorare ulteriormente le prestazioni:

- **Predizione di salto**
- **Esecuzione fuori ordine**
- **Esecuzione speculativa**

Parallelismo

Per migliorare le prestazioni di un calcolatore si possono seguire più strade:



Nel parallelismo a livello di istruzione più istruzioni vengono eseguite contemporaneamente all'interno della stessa CPU tramite tecniche di pipelining e processori superscalari

Nel parallelismo a livello di core o di processore più core/CPU cooperano per la soluzione dello stesso problema.

Aumento frequenza di clock

Nel ventennio 1980-2000 le frequenze di clock sono aumentate di **oltre 1000 volte**: da circa 1-10 MHz (IBM XT) a oltre 3 GHz (Pentium 4). La potenza di calcolo è aumentata ben più di 1000 volte, e ciò dimostra che aumentare la frequenza di clock è uno solo dei modi per rendere un calcolatore più potente.

Nel 2004 con il Pentium 4 sono state raggiunte frequenze di circa 3.8 GHz, che negli anni successivi sono state superate solo di poco.

Record di frequenze (con Overclocking): https://valid.x86.fr/records.html

Nel futuro la frequenza di clock non potrà aumentare significativamente (a meno di scoperte scientifiche straordinarie); infatti siamo oramai giunti nei pressi dei **limiti fisici**:

- le alte frequenze creano disturbi e aumentano il calore da dissipare
- ci sono ritardi nella propagazione del segnale
- bus skew (i segnali su linee diverse viaggiano a velocità diverse): problemi di sincronizzazioni.
- nelle architetture superscalari, ci sono anche limitazioni dovute alla suddivisione delle operazioni (meglio chiarito nel seguito)

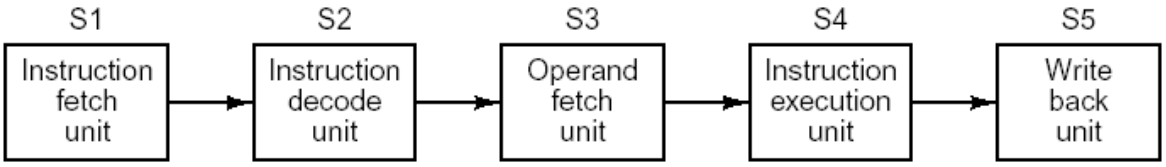
Esempio:

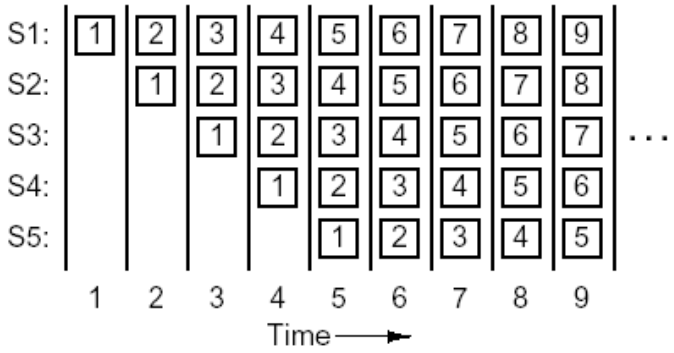
in un nanosecondo (frequenza 1 GHz) che distanza può percorrere un impulso digitale ?

Anche se questo viaggiasse alla velocità della luce (300.000 Km/sec; in realtà la propagazione di elettroni nei conduttori non è elevata come quella nel vuoto) il segnale potrebbe propagarsi in un ns di soli $3 \cdot 10^8 \cdot 10^{-9}$ metri ovvero di circa 33 centimetri (ovvero circa della lunghezza di una scheda madre !)

Pipelining (1)

Con la tecnica del pipelining l'esecuzione di ogni istruzione viene suddivisa in più fasi (chiamate stadi) ognuna delle quali viene gestita da un hardware dedicato.





- Se il ciclo di clock della macchina è di 2 nsec sono necessari 10 nsec per completare l'esecuzione della prima istruzione (nessun risparmio rispetto all'assenza di pipelining), **ma una volta riempita la pipeline, si completerà una istruzione ogni 2 nsec**.

- È necessario garantire che le istruzioni non siano in conflitto ossia che non dipendano l'una dall'altra. In caso contrario sarà necessario attendere il completamento dell'istruzione precedente prima di avviare la successiva.

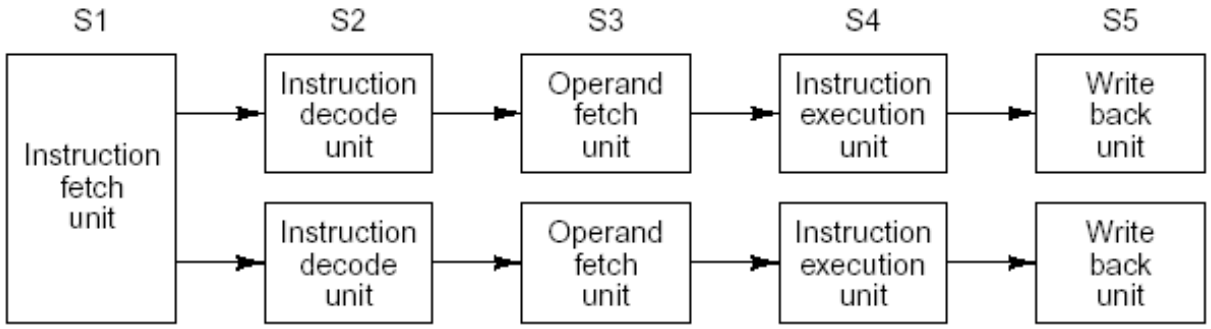
```
1)  x=a+b;
2)  y=x+c;
```

L'istruzione 2 ha come operando il risultato dell'istruzione 1. Finché la pipeline per 1 non termina 2 non può essere terminata.

- Le pipeline vengono normalmente utilizzate su macchine RISC tuttavia anche Intel adotta questa tecnica a partire dal processore 486 che è dotato di una pipeline a 5 stadi.

Pipelining (2)

È anche possibile avere più di una pipeline:



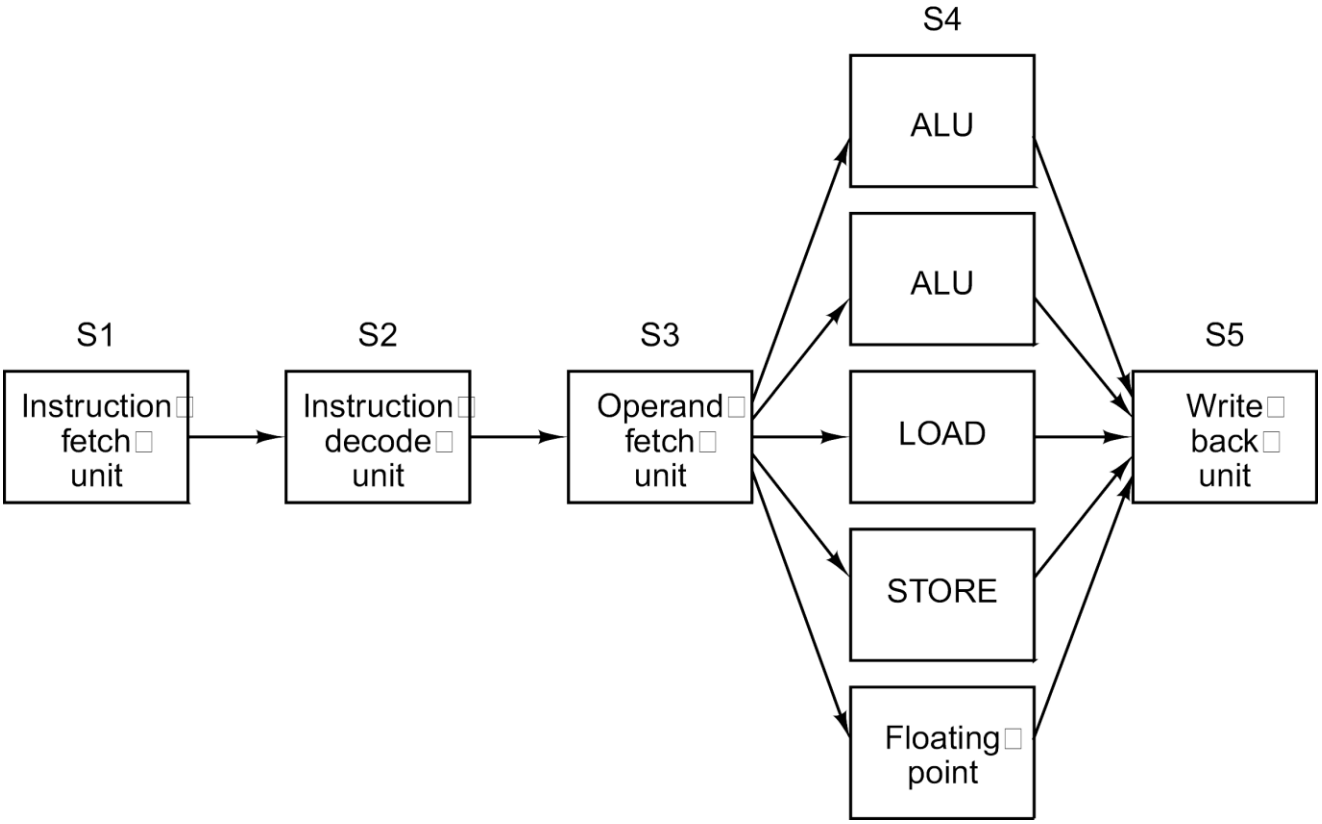
- Vengono lette due istruzioni alla volta che vengono eseguite su pipeline diverse.
- Anche in questo caso è necessario gestire eventuali conflitti tra le istruzioni.
- Il processore Intel Pentium utilizza due pipeline: la prima (pipeline u) esegue qualsiasi istruzione, la seconda (pipeline v) esegue solo istruzioni semplici su interi.

Pipelining e Frequenza della CPU

La frequenza di lavoro massima del sistema dipende dal tempo impiegato dalla fase più onerosa.

Architetture superscalari

Una soluzione alternativa è quella di avere una sola pipeline dotata di unità funzionali multiple (**architettura superscalare**)



Questa soluzione porta a un aumento delle prestazioni se il tempo di esecuzione delle unità funzionali dello stadio S4 è superiore al tempo di esecuzione degli altri stadi. In questo caso la maggior velocità dei rimanenti stadi sarebbe compensata dal parallelismo.

Una soluzione simile a quella rappresentata in figura è implementata nel processore Intel Pentium II.

Predizione di salto

Le architetture con pipeline (praticamente tutte le architetture moderne) funzionano molto bene se il codice viene eseguito **sequenzialmente senza salti**. Infatti, a seguito di un salto che determina un cambiamento nella sequenza di istruzioni da eseguire, una parte del lavoro già eseguito (stadi S1, S2, ed S3 nell'esempio precedente) viene buttato !

La figura mostra un esempio di codice (C) e il corrispondente assembler: sapere se *i* sarà uguale a 0 sarebbe di grande aiuto per eseguire il pre-fetching corretto !

```
if (i == 0)          CMP i,0; compare i to 0
    k = 1;           BNE Else; branch to Else if not equal
else                 Then:   MOV k,1; move 1 to k
    k = 2;           BR Next; unconditional branch to Next
                     Else: MOV k,2; move 2 to k
                     Next:
```

Per **ottimizzare** le operazioni di pre-fetching, le CPU moderne possono utilizzare tecniche di **predizione di salto**, che a fronte di istruzioni di salto condizionale cercano di prevedere (*indovinare?*) se il programma salterà oppure no:

- **predizione statica** vengono utilizzati criteri di buon senso derivati dallo studio delle abitudini dei programmatori e del comportamento dei compilatori; ad esempio: assumiamo che tutti i salti condizionali all'indietro vengano eseguiti.

- **predizione dinamica** vengono mantenute statistiche (tabelle interne) circa la frequenza con cui i recenti salti condizionali sono stati eseguiti. Sulla base di queste statistiche la CPU esegue la predizione. *Esempio*: se nelle ultime *n* volte il salto corrispondente a una data istruzione è stato eseguito almeno *n*/2 volte, allora probabilmente sarà ancora eseguito.

Esecuzione fuori ordine ed esecuzione speculativa

Gran parte delle CPU moderne sono sia pipelined sia superscalari. La progettazione di una CPU è più semplice se tutte le istruzioni vengono eseguite nell'ordine in cui vengono lette; ciò però non sempre porta a prestazioni ottimali per via delle **dipendenze** esistenti tra le varie operazioni:

Infatti, se un'istruzione A richiede un valore calcolato da un'istruzione precedente B, A non può essere eseguita fino a che l'esecuzione di B non è terminata.

Nel tentativo di ovviare a questi problemi e di massimizzare le prestazioni, alcune CPU consentono di saltare temporaneamente istruzioni che hanno dipendenze (lasciandole in attesa) e di passare ad eseguire istruzioni successive non dipendenti. Questo tipo di tecnica prende il nome di **esecuzione fuori ordine** e deve comunque garantire di ottenere esattamente gli stessi risultati dell'esecuzione ordinata.

Un'altra tecnica correlata all'esecuzione fuori ordine prende il nome di **esecuzione speculativa**: essa consiste nell'anticipare il più possibile l'esecuzione di alcune parti del codice (gravose) prima ancora di sapere se queste serviranno.

L'anticipazione (hoisting = atto di alzare / portare in alto il codice) può essere ad esempio relativa a un'operazione floating-point o a una lettura da memoria, ... Ovviamente nessuna delle istruzioni anticipate deve produrre effetti irrevocabili.

In realtà, molto spesso l'hardware della CPU non è in grado da solo di anticipare istruzioni, se non nei casi banali o se non con il supporto dei compilatori. In alcuni processori di recente progettazione, vengono previste particolari istruzioni o direttive che il compilatore può utilizzare per ottimizzare il comportamento della CPU.

Bug "architetturali"

A inizio 2018 sono stati resi pubblici da ricercatori dell'Università di Graz e da Google importanti bug di sicurezza (noti come **Meltdown** e **Spectre**) che coinvolgono diverse CPU moderne (Intel, AMD, ARM) che fanno uso di Cache ed Esecuzione Fuori Ordine.

- https://it.wikipedia.org/wiki/Meltdown_(vulnerabilit%C3%A0_di_sicurezza)
- https://it.wikipedia.org/wiki/Spectre_(vulnerabilit%C3%A0_di_sicurezza)

L'attacco **Meltdown** permette di leggere il contenuto di tutta la memoria fisica eludendo i vincoli che non consentono a un processo di vedere la memoria al di fuori della porzione a lui dedicata. La cosa è particolarmente grave nel caso di macchine virtuali e sistemi in cloud dove i dati di un'azienda potrebbero essere spiati da altri processi in esecuzione sulla stessa macchina fisica.

In sostanza:

- si generano accessi a locazioni proibite (che si vogliono spiare).
- l'esecuzione fuori ordine permette la lettura di queste locazioni prima ancora della verifica dei permessi (che avviene concorrentemente).
- una volta verificato che i permessi non consentono l'accesso si genera un'eccezione e non si rende disponibile il contenuto della memoria.
- nel frattempo però la lettura ha provocato effetti collaterali nella cache che non sono annullati.
- interrogando la cache in modo opportuno (se una riga è presente la risposta è molto più veloce) si può risalire al contenuto della cella di RAM.

Dettagli nel paper: https://meltdownattack.com/meltdown.pdf

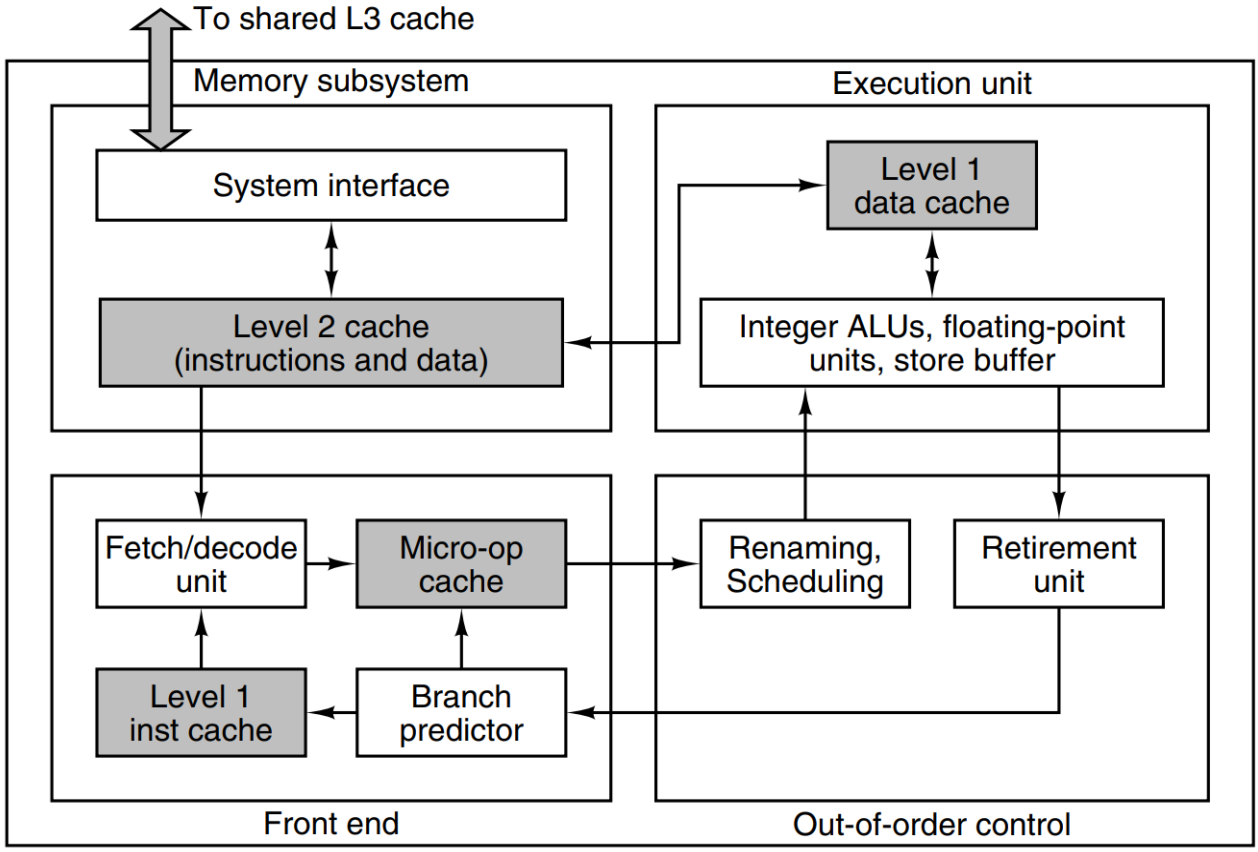
Architettura Core i7

Esternamente Core i7 si presenta come una tradizionale CPU **CISC** a **32/64 bit**, basata sullo stesso ISA (**IA-32** e sua estensione **x86-64**) dei modelli precedenti. Infatti, in disaccordo con i principi di progettazione RISC:

- le istruzioni sono molte e complesse
- i registri generali visibili sono solo 8
- la lunghezza delle istruzioni (codifica) è variabile da 1 a 17 byte.

Al suo interno però contiene un nucleo RISC moderno che fa largo uso di **pipelining** e **architettura superscalare**.

Tra le diverse revisioni architetturali quella qui considerata è denominata **Sandy Bridge**. Core i7 Sandy Bridge (realizzato a partire dal 2011 con tecnologia a 32 nm) contiene al suo un numero di core variabile tra 2 e 6. Nella figura sotto lo schema architettura di un singolo core (gli altri realizzabili con copy&paste).



Architettura Core i7 (2)

Ciascun core è costituito da quattro parti principali: il sottosistema di memoria, il front end, il controllo dell'esecuzione fuori sequenza e le unità esecutive.

Sottosistema di memoria

include cache privata L2 (unificata dati e istruzioni insieme) di dimensione 256KB (associativa a 8-vie, linee di 64 byte). La cache L2 in caso di miss si interfaccia a una cache L3 condivisa dai diversi core ma sempre realizzata dentro al chip. L3 ha dimensione variabile tra 1 e 20 MB, è associativa a 12-vie con linee a 64 byte. In caso di miss anche in L3, il chip reperisce informazioni sulla RAM DDR3 (in modalità burst).

Front end

ha il compito di prelevare le istruzioni dalla cache L1 istruzioni (32KB) e di decodificarle nell'ordine del programma. La decodifica scompone ciascuna istruzione (CISC) in una sequenza di micro-operazioni (tipo RISC) successivamente eseguite dal cuore RISC del processore. Le micro-operazioni vengono salvate nella Micro-op cache (una sorta di cache L0): questo consente di evitare la decodifica per istruzioni successive identiche se queste si trovano già in cache. Fa parte di questo livello anche la predizione di salto (dinamica+statica) i cui dettagli non sono noti.

Controllo dell'esecuzione fuori sequenza

Le micro-operazioni sono trasferite dalla Micro-op cache allo schedulatore che può mandarle in esecuzione fuori ordine (per ottimizzare le prestazioni) allocandole su registri interni. L'unità di ritiro garantirà poi che i risultati prodotti all'esterno siano equivalenti rispetto a un'esecuzione in ordine.

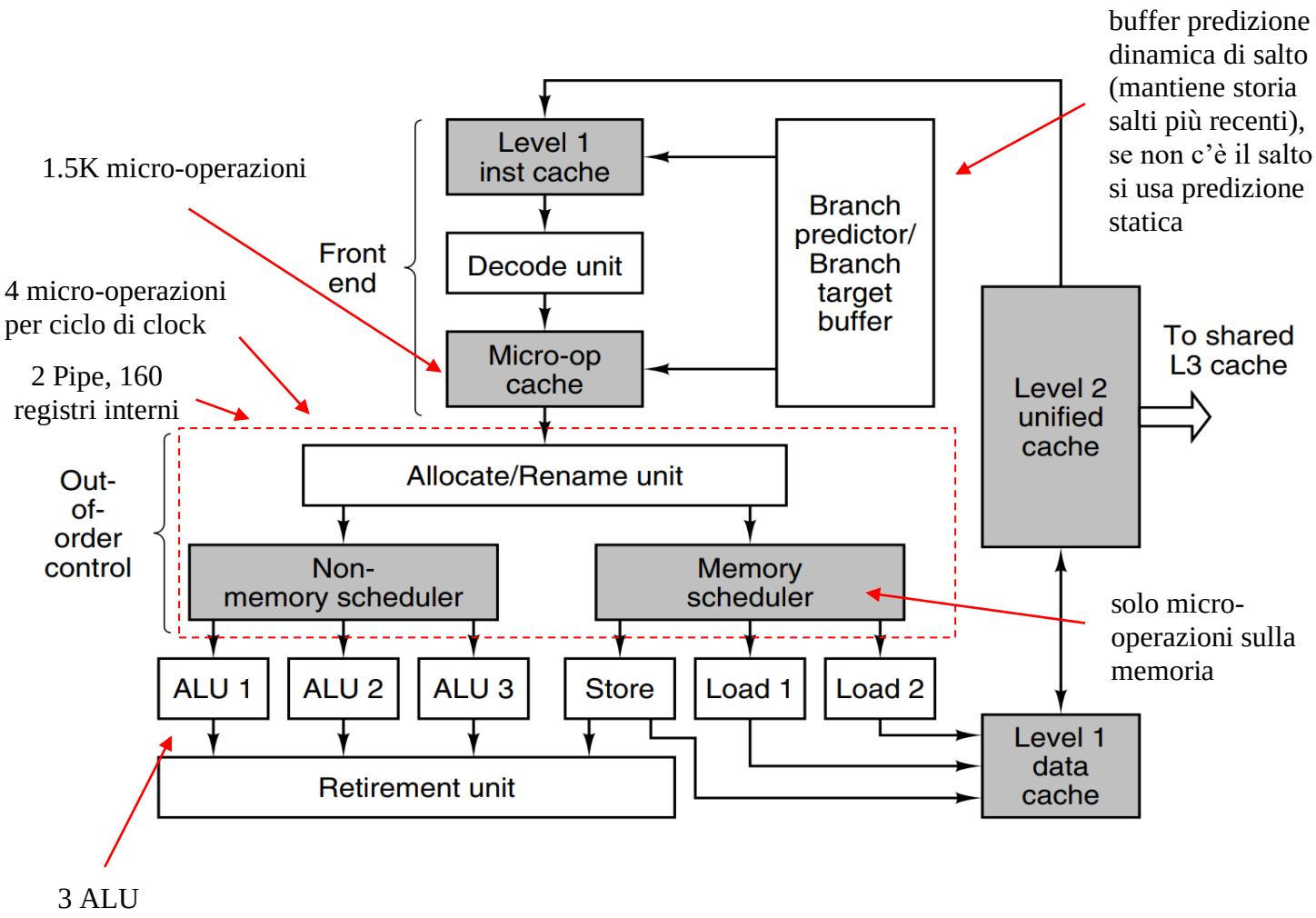
Unità esecutive

Le unità di esecuzione eseguono le operazioni tra interi, in virgola mobile o operazione speciali SIMD. Gli operandi vengono reperiti tramite la cache dati L1.

Architettura Core i7 (3)

La figura illustra l'architettura mettendo in evidenza il pipelining e le componenti superscalari (6 unità funzionali di cui 3 ALU).

La scomposizione in micro-op RISC e la loro esecuzione fuori ordine su architettura pipelining con componenti superscalari consente alta efficienza anche in presenza di dipendenze funzionali. Infatti, i 160 registri interni (nascosti al programmatore) possono contenere valori temporanei di istruzioni in attesa.

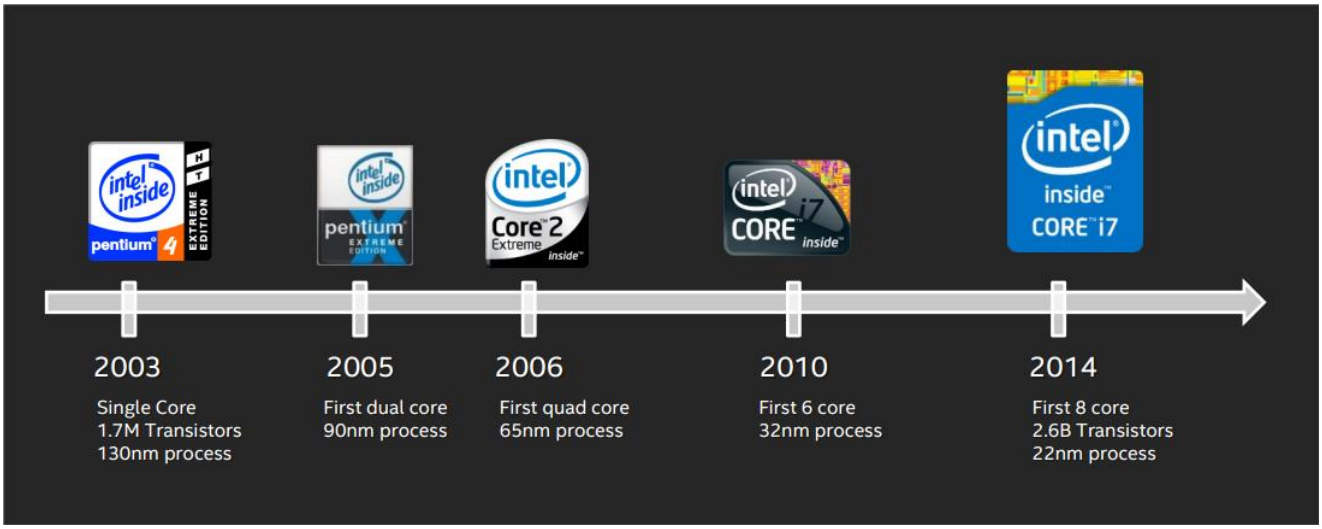


Intel multi-core (1)

Non riuscendo più ad aumentare significativamente le prestazioni con l'aumento di frequenza, Intel ha deciso di puntare al parallelismo proponendo come successori del Pentium IV dapprima **Pentium D** (2005), poi a partire dal 2006: **Core 2**, **Core i3, Core i5, Core i7** e, più recentemente, **Core i9**.

Una **CPU multi core** (fino a 24 nei modelli sopracitati) unisce più processori indipendenti e le rispettive cache in un singolo package (rendendo possibile tra un'implementazione "fisica" dell'Hyper Threading). Inoltre, consente di aumentare la potenza di calcolo senza aumentare la frequenza di lavoro, a tutto vantaggio del calore dissipato (che aveva oramai superato i 100W per CPU single core).

Con l'avvento dei processori della serie **Core 2** per la prima volta sia il mercato dei sistemi portatili che dei sistemi desktop, si basano su un unico processore.



Per trarre pieno beneficio dalla tecnologia multi core i programmi devono essere pensati per un uso ottimale di multi-thread, in caso contrario essi impegneranno solo uno dei due core, lasciando l'altro pressoché inutilizzato.

Intel multi-core (2)

Xeon Skylake-X (2017):

- fino a 18 core;
- supporto integrato DDR4-2666;
- 40 canali PCIe 3.0 integrati;
- Frequenza 2.6-4.3 GHz;
- Cache:
  - L1 (64 KB) per ogni core;
  - L2 (1MB) per ogni core;
  - L3 (1.375MB) per core;
- Istruzioni AVX512, SSE4.2, FMA3;
- 165 Watt;

In un'architettura multi-core moderna con istruzioni SIMD, il calcolo delle prestazioni teoriche è complesso:

```
CPU GHz * number of cores * vector ops (AVX) * special instructions (FMA3)
```

che nel caso di questa CPU si istanzia con:

```
3.3 (Frequenza effettiva con tutti i core attivi) * 18 * 8 (AVX512) * 2 (FMA3) = 950 GFLOPS
```

molto vicino alla prestazione di 977 GigaFlops misurata su Linpack nei primi test apparsi su Internet nel 2017.

Sistemi multi-processore

Attenzione a non confondere sistemi multi-core (*più core nello stesso chip, che condividono il BUS*) e sistemi multi-processore (*chip separati*). La soluzione multi-processore è più costosa ma anche più potente.

Nella famiglia **Xeon**, erano da tempo previste soluzioni multi-processore:

- **Xeon DP** in ambito dual processor
- **Xeon MP** per sistemi a 4 o più vie

Famiglie Intel x86

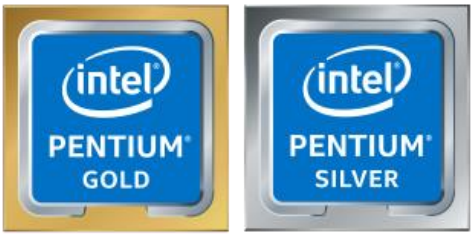
Famiglie di processori Intel con architettura x86 (aggiornato al 2022):

- Linea Core: CPU destinate a computer di fascia medio-alta e workstation. Le principali famiglie sono (Core i3, Core i5, Core i7, Core i9). Disponibili versioni fino a 24 core e 36MB di cache L3.

- Linea Xeon: CPU destinate a workstation e server di fascia alta, derivate dalla linea Core, ma con caratteristiche più avanzate, come il supporto per memoria ECC, numero di core più alto, memoria cache più ampia, supporto per sistemi multiprocessore. Disponibili versioni fino a 60 core e 112,5MB di cache L3.

- Linea Atom: CPU per dispositivi ultra-portatili (es. Netbook, schede per sistemi embedded, IoT), caratterizzate da piccole dimensioni e consumi molto contenuti. Ultime versioni del 2013, in declino.

- Linea Pentium: CPU destinate a computer di fascia medio-bassa, disponibili versioni fino a 5 core e 8MB di cache L2.



- Linea Celeron: CPU più economiche destinate a computer di fascia bassa, disponibili versioni fino a 5 core e 8MB di cache L2.

Non solo Intel: le CPU AMD

AMD (**A**dvanced **M**icro **D**evices) è il secondo produttore mondiale di microprocessori (dopo Intel).

Fin dal 1975 AMD iniziò a produrre microprocessori **x86-compatibili** (partendo con una variante dell'Intel 8080 chiamata AMD8080) e instaurando una competizione perenne con Intel:

- un processore si dice **x86-compatibile** se in grado di supportare le istruzioni ISA dei processori x86 (i.e. IA-32). Tutti i programmi sviluppati per piattaforma IA-32 (i.e., Windows) "girano" su microprocessori x86-compatibili.
- due processori **x86-compatibili**, a seconda dell'architettura interna, possono però avere prestazioni molto differenti ed eseguire lo stesso programma in tempi diversi (infatti anche quando operano alla stessa frequenza, il numero di cicli di clock per eseguire la stessa istruzione ISA può essere diverso).

Nel corso degli anni AMD non si limita a produrre CPU gemelle (e generalmente più economiche) rispetto a Intel, ma spesso anticipa Intel dal punto di vista tecnologico, costringendo quest'ultima a correre ai ripari.

CPU AMD di successo sono:

- la serie **AMD K6** (metà anni '90), la serie **Athlon** (1999)
- **Athlon 64** (2003) è il primo processore desktop con supporto 64 bit della famiglia x86. Il modello AMD64 viene presto "imitato" da Intel, che inserisce EM64T nel Pentium 4, per "ricucire lo strappo".
- **Opteron** (2003, destinata alla fascia server, supporto 64 bit)
- **Phenom** (2007): quad core, **Phenom II** (2009): esa core
- **APU** (2011, …): **CPU + GPU** integrate → PS4, XBOX ONE.
- Le serie **Bulldozer, Jaguar, Puma** (2011-2015)
- Le serie **Zen** (2017) e **Zen 2** (2019): prime CPU a 7nm
- Le serie **Zen 3** (2020, …): ver. desktop **Ryzen 5**, **7**, e **9** e ver. server Ryzen Threadripper PRO (fino a 64 core e 256 MB cache).

CPU RISC non x86 per Server

(oramai abbandonate)

SPARC (Scalable Processor ARChitecture) è il nome di un'architettura "aperta" e non proprietaria per microprocessori RISC - big-endian (basata sul modello RISC originariamente sviluppato nell'università californiana di Berkley). L'architettura fu disegnata nel 1985 da Sun Microsystems; Workstation e server di Sun Microsystems e Fujitsu sono stati i maggiori fruitori di CPU Sparc. Nel corso degli anni l'architettura ha subito diverse revisioni, la modifica maggiore si è avuta con la versione 9 che ha introdotto la gestione dei dati a 64 bit. Poi è stato via via abbandonato. Nel 2018 anche Oracle (che ha acquisito Sun nel 2010) e Fujitsu hanno fermato lo sviluppo.

Power è un'architettura di microprocessori RISC (32 e 64 bit) creata da IBM. Nel 1991 l'alleanza Apple – IBM – Motorola (AIM) ne derivò un'implementazione di successo nota come PowerPC. Apple adottò PowerPC per il suo Machintosh; anche altri Sistemi Operativi (tra cui Windows NT) ben presto supportarono PowerPC, che a metà egli anni 90 era l'architettura più potente per personal computer. Sfortunatamente PowerPC non resse il confronto con la concorrenza (soprattutto per non compatibilità x86) e nel 2006 fu abbandonato anche da Apple. L'architettura Power è stata in seguito utilizzata come base per altri progetti non-x86 tra cui: Xenon (triple core dell'Xbox 360); Cell (cuore di Playstation 3). Al 2017, IBM propone sistemi con O.S. AIX basati su PowerPC.

Itanium è un'architettura RISC nativa a 64 bit (con ISA IA-64) progettata da Intel e HP e lanciata nel 2001. Il progetto eredita soluzioni innovative già implementate nell'architettura di successo Alpha (64 bit) sviluppata e prodotta dalla DEC (acquisita da Compaq, e poi successivamente da HP) a partire dal 1992. Nel 2017 Intel ha introdotto una nuova versione di processori Itanium, contenenti alcuni affinamenti architetturali, affermando però che sarebbe stata l'ultima. Nonostante le potenzialità, il mercato non ha mai premiato Itanium e IA-64 e nel 2019 l'architettura è stata abbandonata.

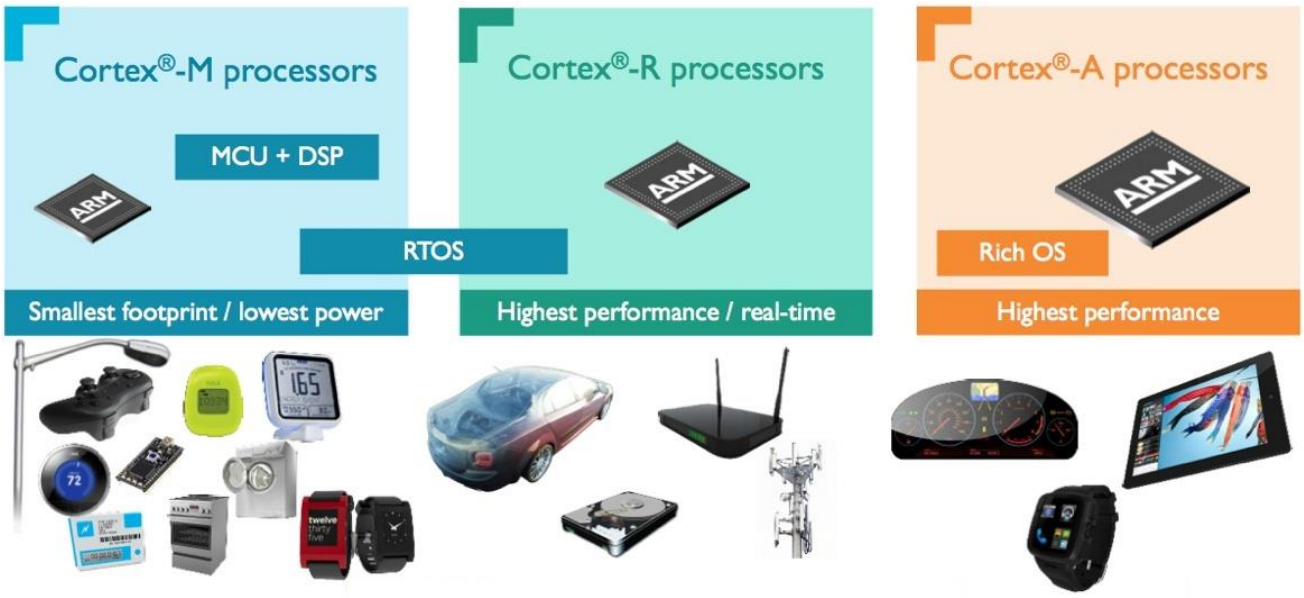
IA-64

Caratteristiche peculiari di questi ISA (alla base di CPU **Itanium**) sono:

- **parallelismo a livello di istruzione** che è esplicito nelle istruzioni macchina piuttosto che determinato dal processore durante l'esecuzione. Questo tipo di parallelismo è noto come EPIC (Explicit Parallel Instruction Computing). In questo caso è il compilatore (e non il processore) a dover capire quali istruzioni possono essere eseguite in parallelo e a generare il codice macchina relativo, unendole in un'unica istruzione più grande, da cui il nome di Very Long Instruction Word (VLIW).
- Pertanto, i processori EPIC non necessitano di circuiti complessi per l'esecuzione fuori sequenza.
- **sono previsti fino a 256 registri a 64 bit** (128 interi + 128 in virgola mobile nell'Itanium).
- **più pipeline parallele** (si prevedono implementazioni con 8 o più unità esecutive). Il pipelining può essere gestito a livello software.
- le istruzioni hanno un **formato a lunghezza fissa** di 41 bit. Un pacchetto (bundle) contiene tre istruzioni. Il processore può caricare uno o più pacchetti per volta.
- l'utilizzo dei **predicati di salto** con la maggior parte delle istruzioni (concetto diverso dalla predizione di salto). Questo si realizza aggiungendo un registro predicato davanti alle istruzioni: durante l'esecuzione se il valore del predicato è FALSE l'istruzione non viene eseguita ma non si procede a nessun salto (dannoso per il pipelining) e si continua semplicemente con la successiva.
- **il caricamento speculativo** e l'esecuzione speculativa (lungo entrambi i rami di un salto). Sono previsti appositi formati (modificatori delle istruzioni) per la gestione efficiente dell'esecuzione speculativa.

CPU per sistemi embedded

ARM®

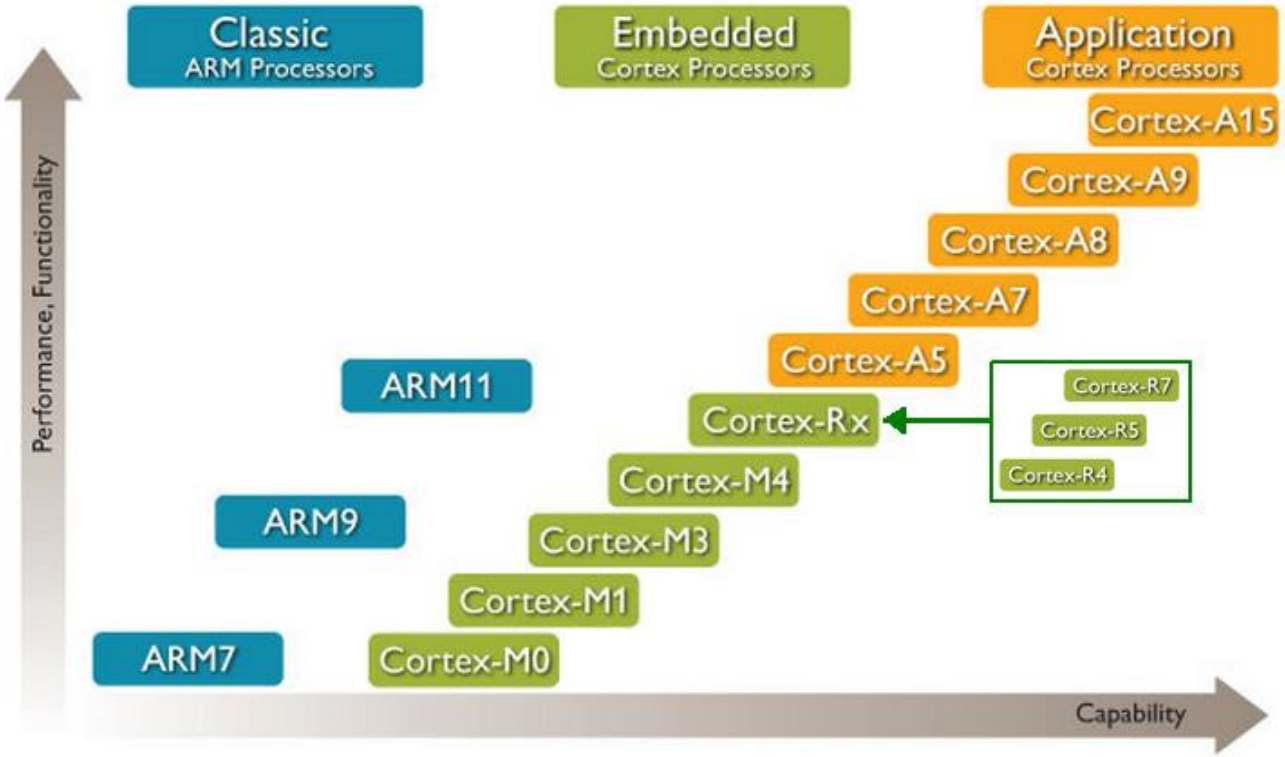


ARM (Advanced RISC Machine) indica una famiglia di processori RISC a 32 bit sviluppata dell'azienda inglese ARM Holdings che detiene la proprietà intellettuale di molti "core" di CPU embedded. ARM non produce direttamente hardware ma vende licenze di produzione dei suoi core a grandi produttori di CPU e System-On-Chip (SOC) tra cui: FreeScale, STMicroelectonics, NXP, Texas Instruments, Samsung, Broadcom, Qualcomm, Nvidia, Apple, ecc.
Si tratta di CPU RISC, di disegno architetturale semplice e pulito, caratterizzate da **basso consumo** (caratteristica fondamentale per dispositivi a batteria) e un buon compromesso: prestazioni ↔ costo

Cortex A9 alla frequenza di 1 GHz consuma meno di 250 mW per core
Un processore Intel Core-i7 multicore può consumare oltre 100 W

Nel 2013 chip basati su ARM erano presenti nel 60% dei dispositivi mobili esistenti.

CPU per sistemi embedded (2)



Core Classic più popolari:

Nati per sistemi embedded, ARM 9 e ARM 11 adottati da dispositivi mobili.

ARM 7: aritmetica intera, fino 20..60 MIPS

ARM 9: aritmetica intera, fino 100..200 MIPS

ARM 11: floating point, 300..1000 MIPS (2600 MIPS multicore). Modello a 700 MHz usato in Raspberry Pi (2012).

Core Cortex:

Presentati nel 2005 come evoluzione futura ARM.

Cortex-M (M = microcontroller): naturale evoluzione di ARM 7 e ARM 9 per device embedded senza grandi necessità di performance (es. 100 MIPS).

Cortex-R (R = real-time): per sistemi embedded con requisiti prestazionali medio-elevati (es. 600 MIPS). Floating point opzionale.

CPU per sistemi embedded (3)

Cortex-A (A = applications): per sistemi con requisiti prestazionali elevati (es. smartphone, tablet). Include floating point, SIMD.

Disponibili versioni a 32 bit e 64 bit:

64-bit	
Year	Core
2016	Cortex-A34
2018	Cortex-A65
2021	Cortex-A510
2022	Cortex-A715

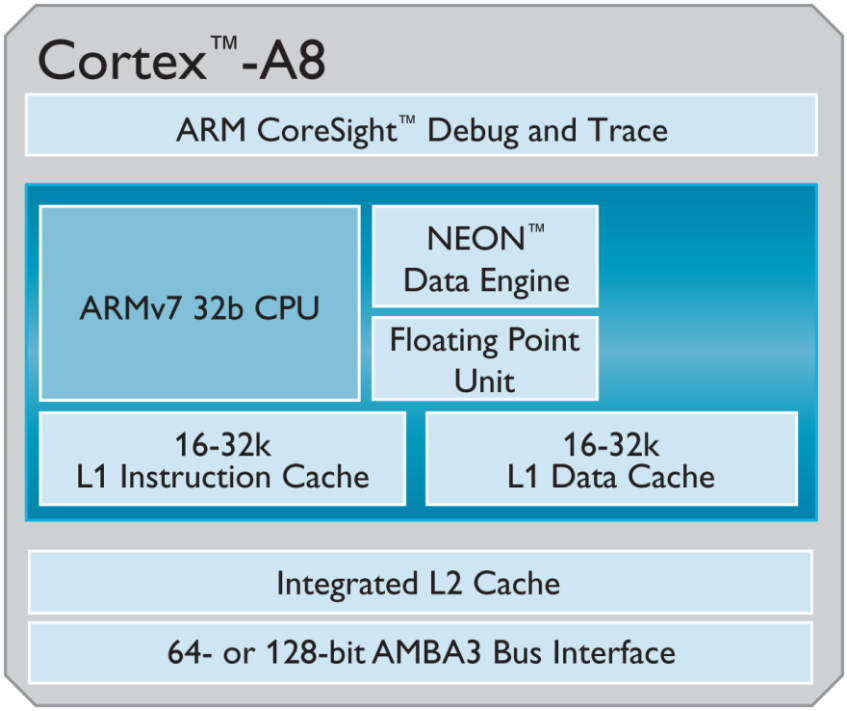
32/64-bit	
Year	Core
2012	Cortex-A53
2012	Cortex-A57
2015	Cortex-A35
2015	Cortex-A72
2016	Cortex-A73
2017	Cortex-A55
2017	Cortex-A75
2018	Cortex-A76
2019	Cortex-A77
2020	Cortex-A78
2021	Cortex-A710
2022	Cortex-A510 Refresh

32-bit	
Year	Core
2005	Cortex-A8
2007	Cortex-A9
2009	Cortex-A5
2010	Cortex-A15
2011	Cortex-A7
2013	Cortex-A12
2014	Cortex-A17
2016	Cortex-A32

- Il Cortex **A7** supporta fino a 4 core, ciascuno con una pipeline parzialmente superscalare a 8 stadi e unità predizione salto ma con esecuzione in ordine. 1.9 MIPS per MHz. Scelto per via del basso consumo, per il system-on-chip Broadcom per Raspberry Pi 2.
- Il Cortex **A72** supporta fino a 4 core per cluster (possibili più cluster). Ha una cache L2 fino a 4MB. Architettura superscalare a tre vie. Predizione di salto (sofisticata). Supporta esecuzione fuori ordine e speculativa. Rende disponibili 4.7 MIPS per MHz.
- Varianti dei recenti core "di punta" a 64 bit sono state usate nei SOC Exynos (di Samsung) e Snapdragon (di Qualcomm) usati in smartphone di Samsung e Apple.

CPU per sistemi embedded (4)

Cortex-A8



Frequenza: da 600 MHz a 1 GHz

Consumo: meno di 300 mW

Prestazioni: 2 MIPS per MHz (fino 2000 MIPS)

ISA ARM

Come da filosofia RISC, l'ISA ARM:

- ha un insieme ridotto di istruzioni (100 o meno).
- le istruzioni operano su registri (numerosi), mentre per accedere alla memoria si usano specifiche istruzioni di Load/Store (anche caricamenti multipli).
- la maggior parte delle istruzioni permette esecuzione condizionale.
- si può settare l'endianess (e operare in little o big endian)

Esistono varie generazioni di ISA per le diverse famiglio di CPU.

ARM family	ARM architecture
ARM7	ARM v4
ARM9	ARM v5
ARM11	ARM v6
Cortex-A	ARM v7-A
Cortex-R	ARM v7-R
Cortex-M	ARM v7-M

Tutorial e risorse:

https://azeria-labs.com/writing-arm-assembly-part-1/

Apple M1

In passato Apple basava la realizzazione dei propri Mac e dispositivi mobili su CPU prodotte da terzi: Motorola, PowerPC, Intel. Nel 2020 abbandona Intel e passa ad ARM. A differenza di altri però progetta e produce in casa i propri SoC (System on Chip), pagando ad ARM licenze per l'uso di proprietà intellettuale (es. Instruction Set a 64 bit).

M1 (uscito nel 2021) è il primo SoC progettato da Apple per Mac e per iPad Pro. Il Chip è prodotto con tecnologia 5nm e contiene 16 miliardi di transistor. È un progetto di grande successo capace di unire elevate prestazioni a consumi ridotti. Degli 8 core disponibili: 4 sono ad alte prestazioni (e maggior consumo), i rimanenti 4 a prestazioni e consumo ridotto. M1 integra inoltre:

- una GPU a 8 core
- un Neural Processing Unit a 16 core per accelerare reti neurali.
- UMA: una sorta di memoria cache L3 di grande capacità (in realtà tecnologia DDR4). Su questa memoria CPU, GPU e NPU possono condividere i dati senza copie/spostamenti fra memorie diverse (spesso un collo di bottiglia).