ANALISI DELLE PRESTAZIONI DEI CALCOLATORI

1. Evoluzioni dei calcolatori e prestazioni

Una ***prima evoluzione sta nella nascita dell'ISA***. Negli anni '60 l'IBM decise di integrare 4 serie di calcolatori incompatibili in una singola ISA → incompatibile con l'unità di controllo che era realizzata specificatamente per ogni processore → *Unità di controllo doveva essere rifatta e riprogettata anche ad ogni minimo cambio dell'ISA.* Per questo si introdusse l'idea di creare l'unità di controllo come una memoria ROM contenente il Microcodice

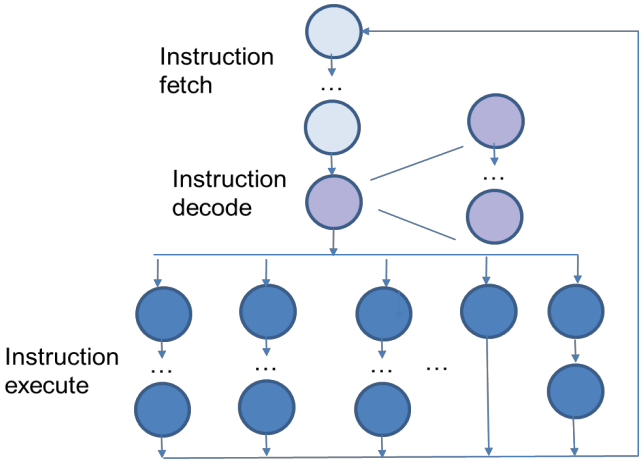


Fig.1 ciclo di istruzione

L'evoluzione della CPU ha portato a studiate come:

- **Ridurre numero di micro-operazioni per eseguire ogni istruzione** → influenza n° clock per ogni istruzione
- **Ridurre/semplificare operazioni per ogni micro-operazione**
- **Sovrapporre nel tempo e nello spazio l'esecuzione di più microoperazioni**/istruzioni per miglioramento del throughtput

Dal punto di vista della rete logica l'unità di controllo è stati inizialmente progettata come una rete sequenziale sincrona specifica cablata per la specifica ISA (**UNITÀ di CONTROLLO HARDWIRED**).

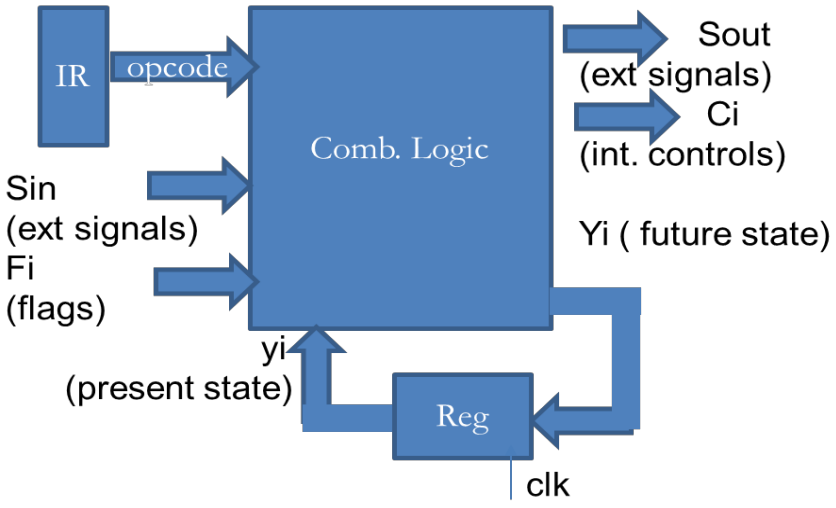


Fig 2 Unità di controllo hardwired

Per questo motivo, nel momento in cui le CPU divennero più complesse, si decise di realizzare l'unità di controllo in modo più modulare, senza doverle riprogettare ad hoc per ogni ISA differente. Fu impiegato cos' il modello di **UNITÀ DI CONTROLLO MICROPROGRAMMATA:**

ogni istruzione *altro non è che interpretata come la chiamata a un microprogramma che viene eseguito dall'unità di controllo*. Il ***codice operativo*** *è un puntatore ad un indirizzo di partenza* che legge su una memoria interna (micro ROM o RAM) la sequenza di microoperazioni da eseguire. Il ***dato memorizzato*** nella microROM è *una sequenza di segnali di controllo* che attraverso un micro-registro vanno sia ai segnali di controllo interni che esterni. ***Esiste poi un sequenziatore***, un contatore *che per ogni istruzione incrementa il micro-PC* affinchè venga eseguita tutta la sequenza di microcodice. Quindi ogni istruzione è a sua volta tradotta ed interpretata come una sequenza di microoperazioni. Una architettura di questo tipo è più lenta della precedente ma i vantaggi dell'interpretazione sono stati tanti, come la possibilità di emulare su macchine semplici ISA complesse, di correggere l'ISA nel tempo fino a realizzare l'interprete software come la Java Virtual Machine.

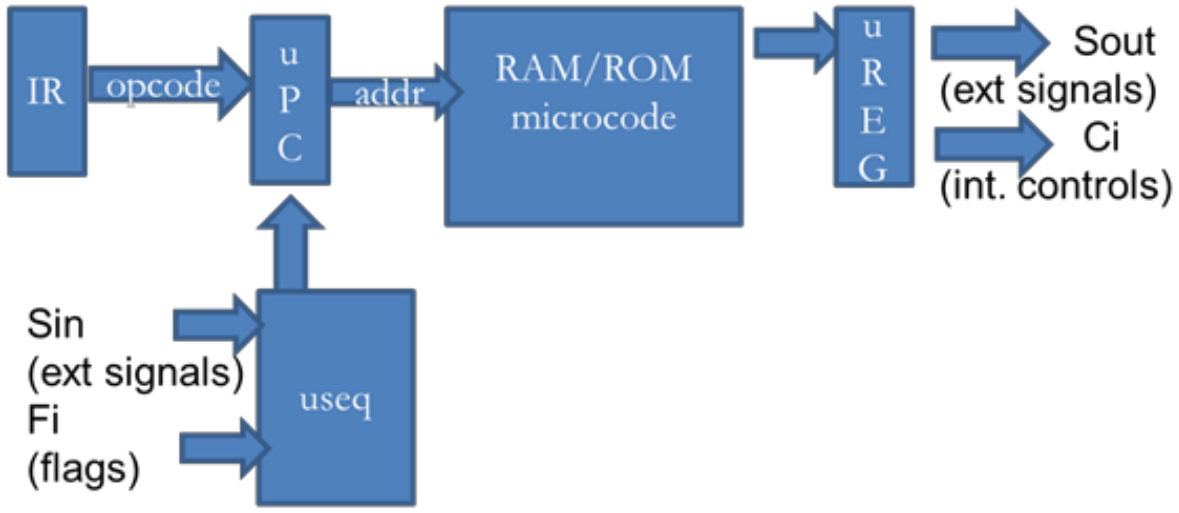


Fig.3 unità microprogrammata

se un progettista vuole cambiare l'ISA o a parità di ISA vuole con la stessa istruzione eseguire microoperazioni diverse, deve solo riscrivere la sequenza di microistruzioni da memorizzare nella microROM. → concetto in inizialmente chiamato **FIRMWARE →macchina più complicata: VAX-11/780 della DEC**

Si iniziò a progettare una architettura RISC (reduced instruction set computer) con alcune assunzioni:

- **32 istruzioni → Unità di controllo cablata**
- **Togliendo memoria x microcodice → spazio x piccole memorie veloci per istruzioni e dati**
- **Registri ortogonali da usare**
- **'80 legge di Moore riuscì a mettere nello stesso microprocessore memoria/hardware in modo regolare**

LE PRESTAZIONI → SI VALUTANO SOTTO DUE PUNTI DI VISTA:

- **Efficacia:** *calcolatore deve essere in grado di compiere il lavoro previsto* **→ si ottiene progettando una ISA generale capace di eseguire tutte le istruzioni tipiche di un linguaggio di alto livello e scrivere un codice corretto.**
- **Efficienza:** *la possibilità di svolgere il compito con minori risorse possibili*

2. Parametri di prestazioni

Si caratterizzano per la quantità di lavoro utile compiuto da un sistema informatico rispetto al tempo e le risorse utilizzate → si misurano soprattutto in termini di **VELOCITÀ** di risposta, di esecuzione, di trasferimento. *Nei calcolatori general purpose a parità del resto le misure di prestazioni sono misure di TEMPO.*

Nei sistemi di piccole dimensioni i fattori che maggiormente intervengono sono la **velocità, e il costo ed i consumi,** a cui si sommano altri fattori commerciali (peso, il design …) soprattutto per il mercato *consumer*.

Nei sistemi di grandi dimensioni sono altri fattori che incidono quali, oltre alla **velocità**, la disponibilità e l'efficienza, l'**affidabilità**, la **scalabilità, la sicurezza, la sostenibilità, la resilienza, la sicurezza**

a) **Scalabilità:** *è la capacità di un sistema di allargarsi al crescere delle esigenze di lavoro per mantenere le proprie prestazioni*
b) **Modularità:** *è la capacità di un sistema di essere progettato per parti* (moduli) *creati indipendentemente e usati per ottenere diverse funzionalità*
c) **Affidabilità (Reliability):** *indica il mantenimento delle prestazioni al cambiare del tempo, della domanda e delle richieste.* → **esiste una metrica molto importante:**

- **La Service/System Availability**: che *indica la percentuale di tempo in cui il sistema è disponibile* che si misura di solito come un rapporto medio tra il tempo in cui il sistema e servizio è disponibile (uptime) rispetto al tempo totale.

Inoltre una metrica molto importante di affidabilità è:

- ***MTTF (mean time to failure):*** *tempo medio fra due guasti*
- ***POFOD (probability of failure on demand):*** *probabilità di avere guasti quando ho richiesta*
- ***ROCOF (rate of failure occurrence):*** *quantità di eventi inaspettati nel tempo*

d) **Disponibilità (Aviability):** *è la capacità di reperimento nel mercato del calcolatore, dei suoi componenti modulari e dei pezzi di ricambio.*
e) **Resilienza (Resilience):** *è la capacità di mantenere un adeguato livello di servizio a fronte di guasti o di cambiamenti delle normali operazioni di lavoro*

3. Prestazioni in termini di tempo

TEMPO DI ESECUZIONE/RISPOSTA (chiamato in modo diverso Texec- *tempo di esecuzione, response time, latenza*): è il ritardo calcolato dall'inizio alla fine di una esecuzione (spesso misurato in cicli di clock). → si usano i termini:

- **tempo di risposta (Latency)** *usato nel contest di sistemi operativi*
- **tempo di accesso (Access Time)** si riferisce a memorie e I/O
- **ritardo (modal delay)** *per la rete*
- **tempo di esecuzione (Execution time)** *per valutare le CPU*

THROUGHPUT (*Th anche larghezza di banda*): è la quantità di lavoro (di operazioni o processi) ottenuta in un fissato periodo di tempo

Come calcolare le prestazioni? Esistono molte misure commerciali nelle CPU come il ***"numero di istruzioni per secondo"*** (misurate in milioni **MIPS Mega Instruction per Second** o miliardi **GIPS Giga instruction per second**) o la ***frequenza di clock*** (MHz, GHz) per eseguire un'operazione elementare.

4. Misure di prestazioni

Nei calcolatori le misure delle prestazioni si ottengono secondo tre diverse modalità

- **Simulazione e Profilazione → simulazione** *è il processo di valutazione impiegato normalmente in fase progettuale.*

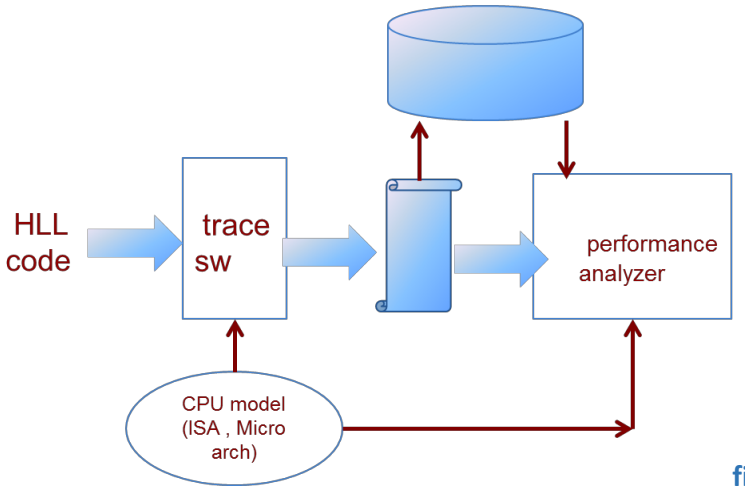


fig.5 Simulatori

I simulatori si usano anche in fase di analisi per verificare se una particolare architettura o un software su una particolare architettura risponde alle richieste di prestazioni in termini di velocità e di throughput.

I Profilatori (**program profiling**), *sono programmi che si impiegano per analizzare il comportamento dinamico dei programmi* → **I più semplici sono i debuggers ed il più noto in commercio nel mondo Intel è VTUNE**

- **Modelli analitici** → *sono modelli matematici che cercano di calcolare a priori le prestazioni ottenibili.*
- **Benchmark** → *sono programmi campione o insieme di programmi campione, di cui si conosce il tipo di istruzioni necessarie usati per misurare le prestazioni in termini di tempo o di risorse impiegate.*

5. Modelli analitici di prestazioni

- CPU-Bound: applicazioni o programmi che sono limitate dalle prestazioni della CPU, tendenzialmente programmi di calcolo, di grafica, dove devono essere fatte molte operazioni di ALU e di memoria

- I/O Bound: applicazioni o programmi che sono limitate dalle prestazioni delle periferiche di I/O, dai dischi anche distribuiti, dalla rete e dall'uomo; come ad esempio l'accesso ai Database.

TCPU (relativo a un programma): Ncc numero di cicli di clock della CPU relativi ad un programma, **NI numero di istruzioni in un programma** (in linguaggio macchina) e **sia Tck il tempo di clock** reciproco della frequenza f di lavoro

Tcpu = Ncc Tck

Tcpu = Ncc / f

CPI=Ncc/NI ***Clock per instruction (Medio)*** → *Per calcolare il CPI medio bisogna conoscere il CPIi dell'istruzione iesima e l'occorrenza media Fi (in percentuale) dell'istruzione iesima in un programma xi:*

$$CPI = \sum_{i=1}^{n} (CPI_i \times F_i)$$

Il tempo di CPU dipende dalla quantità e dal tipo di istruzioni come sono definiti dall'ISA che indica le istruzioni eseguibili, mentre il CPI per ogni istruzione dipende dalla micro-architettura e dalla architettura di tutto il calcolatore

Tcpu = Ni CPI Tck = Sec/Prog

Tcpu = Ni CPI /f ***" the Classic CPU Performance Equation"***

Tcpu = (Ni/Prog) (Ncc/Ni) (Sec/Ciclo)

Tcpu= Istruzioni x CPI x Clock

- **Ni:** *dipende dal repertorio di istruzioni (ISA) e dal grado di ottimizzazione del compilatore.*
- **CPI:** *dipende dall'architettura e dal repertorio delle istruzioni (e dal calcolatore)*
- **Tck (o la freq):** *è legato alla tecnologia e all'organizzazione architetturale della CPU.*

Le prestazioni si misurano spesso soprattutto in campo commerciale con i **MIPS (Mega instruction per second):** è una misura utilizzabile solo per CPU con la stessa

ISA. Corrispondentemente si usano i **MFLOPS** Mega Floating point operation per second *o i Gflops.*

MIPS = N_I / (Tcpu * (10^6)) inoltre: MIPS = **f_{ck} / CPI_{medio}** (f_{ck} in MHz)

Quindi il numero di MIPS, da solo, non è un buon indicatore delle prestazioni di una CPU Si possono confrontare tra loro due CPU rispetto a un determinato programma solamente se le CPU hanno lo stesso set di istruzioni

6. Speedup

Definisce l'indice di miglioramento e definisce quanto è migliorata la prestazione globale del sistema in seguito ad un miglioramento anche solo di una sua parte.

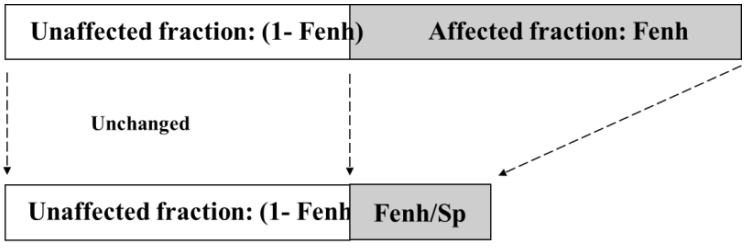
$$SP = \quad Pnew/\ Pold = Texec\ (old)/\ Texec\ (new)$$

In molti casi però bisogna valutare quello che si chiama **Speedup overall** ossia miglioramento complessivo che è diverso dal miglioramento di solo la parte modificata.

Da sempre si usa una legge nota come: **legge di Amdhal** che afferma: "***il miglioramento delle prestazioni dovute ad un miglioramento di esecuzione è limitato dalla frazione di tempo in cui tale miglioramento può essere applicato"***

$$Texec(new) = Texec(old)\ [\ (1- Fenh) + Fenh/Senh]$$

$$Speedup = \frac{Old\ time}{New\ time} = \frac{1}{1 - f_{enh} + f_{enh}/n}$$



Dove Fenh (fraction enhanced) è la percentuale di tempo in cui ha effetto il miglioramento e Senh (speedup enhanced) è il valore di tale miglioramento.

OSS: ***nella progettazione dei calcolatori è preferibile migliorare le prestazioni delle parti usate più frequentemente*** *(o per le parti usate da istruzioni più frequenti) rispetto ad apportare grossi miglioramenti in parti meno usate.*

Avendo definito la legge di Amdahl, ci domandiamo: vale ancora la legge di Moore?

Nel 2010 Intel ha deciso di smettere di migliorare CPU integrando CPU più semplici → distanza da legge di Moore e progettazione secondo tick-tock → ora la legge di Moore vale nell'ambito Multi-core.

7. Benchmark

I benchmark *sono specifici programmi campione o insiemi di programmi scelti per misurare le prestazioni.* Sono nati per fornire un workload significativo per stimare le prestazioni. **Ci sono 3 tipi di benchmark:**

a) **Benchmark reali:** veri e propri programmi o suite di programmi
b) **Benchmark ridotti:** sono insiemi di funzioni o librerie che stressano solo parti della CPU
c) **Benchmark sintetici:** simulano le operazioni più frequenti di una CPU (es wetstone, drystone) non sono programmi significativi ma sono molto usati in fase di progettazione o di simulazione delle singole parti di un calcolatore.

I benchmark più famosi sono le suite di kernel e programmi reali prodotti da **SPEC**, **Standard Performance Evaluation Corporation** → *associazione che definisce gli standard usati da tutte le casi costruttrici.*

EEMBC = *associazione nata x benchmark sistema embedded*