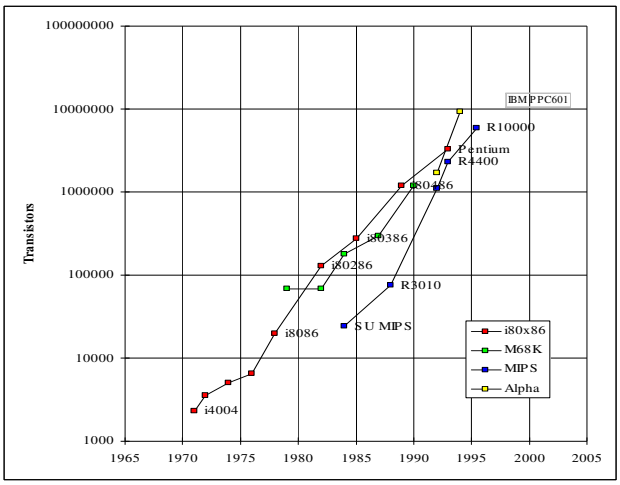
La Valutazione delle Prestazioni

Maurizio Palesi

Trend Tecnologico: Microprocessori



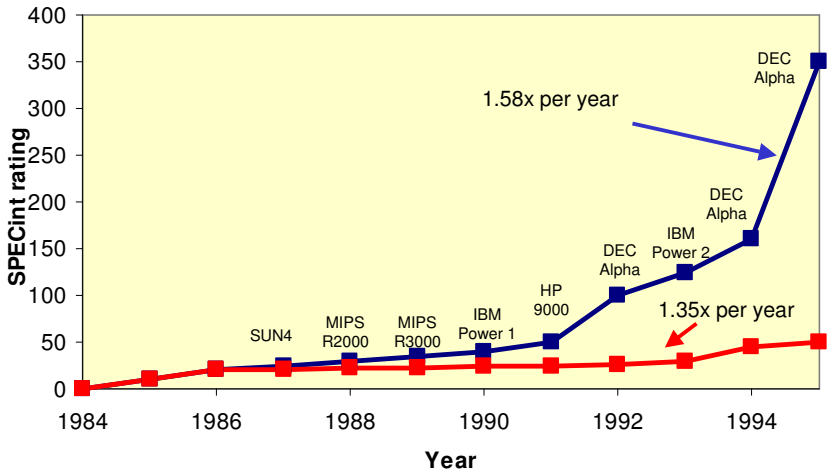
Alpha 21264: 15 milion
Pentium Pro: 5.5 million
PowerPC 620: 6.9 million
Alpha 21164: 9.3 million
Sparc Ultra: 5.2 million

2X transistor/Chip ogni 1,5 anni

Trend Tecnologici

- Processore
  - ➔Densità Logica: +30% per anno
  - ➔Frequenza Clock: +20% per anno
- Memoria
  - ➔Capacità DRAM: +60% per anno
  - ➔Velocità Memoria: +10% per anno
  - ➔Costo per bit: -25% per anno
- Dischi
  - ➔Capacità: +60% per anno
- Larghezza di banda della rete
  - ➔+100% per anno!

Trend Prestazioni



L'Approccio Quantitativo alla Progettazione

- L'incremento delle prestazioni è superiore a quello tecnologico
- Ciò è stato possibile per l'affermazione di un nuovo approccio nella progettazione
  - ➔ Approccio quantitativo ovvero basato su misure
- Principio da seguire nella progettazione
  - ➔ Rendere veloce il caso più frequente

Conseguenze

- Una diretta conseguenza di questo nuovo approccio è il passaggio da calcolatori con un numero elevato di istruzioni anche molto complesse (approccio CISC), a calcolatori con un ridotto insieme di istruzioni (approccio RISC)
- Da misure sul comportamento dei programmi si è visto che
  - ➔ **L'80%** delle istruzioni eseguite corrispondeva **al solo 20%** del repertorio
  - ➔ Conviene investire nella riduzione dei tempi di esecuzione di quel 20%, anziché aggiungere raffinate istruzioni, quasi mai usate, ma responsabili dell'allungamento del tempo di ciclo di macchina
  - ➔ Conviene costruire processori molto veloci, necessariamente con repertori semplici, e contare sull'ottimizzazione del compilatore

Motivazioni

- Misura/valutazione di un insieme di parametri quantitativi per
  - ➔Quantificare le caratteristiche di una macchina (velocità, ecc.)
  - ➔Fare scelte intelligenti (es. miglioramento dell'hardware vs. installazione nuovo software)
  - ➔Orientarsi nell'acquisto del calcolatore più adatto per l'applicazione data

Indici Prestazionali

- Tempo di risposta (o tempo di esecuzione o latenza)
  - ➔Tempo tra l'inizio e il completamento di un lavoro o compito elaborativo
    - ✓Durata dell'esecuzione del mio programma
    - ✓Attesa per l'accesso ad un sito web
- Throughput
  - ➔Ammontare complessivo di lavoro svolto in un dato tempo
    - ✓Numero di programmi eseguiti nell'unità di tempo
    - ✓Numero di lavori (job, transazioni, interrogazioni a basi di dati) svolti nell'unità di tempo
    - ✓Numero di programmi eseguibili da una macchina contemporaneamente

Relazione tra le prestazioni di due macchine

- La frase "X è più veloce di Y" è usata per indicare che il tempo di risposta o di esecuzione, per un dato lavoro, è più basso in X che in Y

$$\text{"X è n\% più veloce di Y"} \rightarrow \frac{\text{Tempo di esecuzione di Y}}{\text{Tempo di esecuzione di X}} = 1 + n/100$$

- Il tempo di esecuzione è il reciproco della prestazione

$$1 + n/100 = \frac{1/\text{Prestazioni di Y}}{1/\text{Prestazioni di X}} = \frac{\text{Prestazioni di X}}{\text{Prestazioni di Y}}$$

- Si osservi che

$$n = \frac{(\text{Prestazioni di X} - \text{Prestazioni di Y})}{\text{Prestazioni di Y}}$$

  Cioè n% rappresenta l'incremento percentuale delle prestazioni di X rispetto a Y

Clock

- Ciclo di Clock
  - ➔ Durata del periodo di un'oscillazione completa del segnale di sincronizzazione
    - ✓ Si misura in secondi
- Frequenza di Clock
  - ➔ Numero di cicli di clock per secondo
    - ✓ Si misura in Hertz
    - ✓ 1 Hz = 1/sec
- Es. Un calcolatore con frequenza di clock di 1GHz possiede un ciclo clock di durata 1/1GHz = 10^{-9} sec = 1 nsec

Tempo di Esecuzione

$$\frac{\text{Numero di cicli di clock necessari per completare il programma} \times \text{Durata del periodo di clock}}{\text{Tempo di esecuzione del programma}} =$$

- Per migliorare (diminuire) il tempo di esecuzione di un programma
  - ➔ Diminuire il numero di cicli di clock
  - ➔ Diminuire la durata del clock (o aumentarne la frequenza)

Principi Quntitativi per la Progettazione

- Rendere veloce il caso più comune
  - ➔ Si deve favorire il caso più frequente a discapito del più raro
  - ➔ Il caso più frequente è spesso il più semplice e può essere reso più veloce del caso infrequente
- **Legge di Amdahl**
  - ➔ *Il miglioramento di prestazione che può essere ottenuto usando alcune modalità di esecuzione più veloci è limitato dalla frazione di tempo nella quale tali modalità possono venire impiegate*

Speedup o Accelerazione

$$\text{Speedup} = \frac{\text{Prestazione per l'intero lavoro usando quando possibile il miglioramento}}{\text{Prestazione per l'intero lavoro non usando il miglioramento}}$$

Oppure

$$\text{Speedup} = \frac{\text{Tempo di esecuzione per l'intero lavoro non usando il miglioramento}}{\text{Tempo di esecuzione per l'intero lavoro usando quando possibile il miglioramento}}$$

- Lo *speedup* fornisce informazioni su quanto più velocemente un lavoro verrà eseguito usando la macchina con la miglioria rispetto alla macchina originale

Legge di Amdahl

- $\text{Frazione}_{\text{migliorato}}$ (≤ 1)
  - ➔ Frazione del tempo di calcolo della macchina in cui è utilizzata la miglioria
- $\text{Speedup}_{\text{migliorato}}$ (≥ 1)
  - ➔ Miglioramento ottenuto dal modo di esecuzione più veloce

$$\textbf{Tempo di esecuzione}_{\text{nuovo}} = (1 - \textbf{Frazione}_{\text{migliorato}}) \times \textbf{Tempo di esecuzione}_{\text{vecchio}} + \textbf{Frazione}_{\text{migliorato}} \times \textbf{Tempo di esecuzione}_{\text{vecchio}} \times 1/\textbf{Speedup}_{\text{migliorato}}$$

$$\textbf{Speedup}_{\text{globale}} = \frac{\textbf{Tempo di esecuzione}_{\text{vecchio}}}{\textbf{Tempo di esecuzione}_{\text{nuovo}}} = \frac{1}{(1 - \textbf{Frazione}_{\text{migliorato}}) + \frac{\textbf{Frazione}_{\text{migliorato}}}{\textbf{Speedup}_{\text{migliorato}}}}$$

Esempio 1

Si consideri un calcolatore (CALC1) che possiede un'unità aritmetico–logica intera (INT_ALU) e un'unità aritmetico-logica in virgola mobile (FP_ALU).

CALC1 esegue un'applicazioni software che tipicamente prevede il 90% di istruzioni in aritmetica intera e il 10% in virgola mobile.

Si consideri un nuovo calcolatore (CALC2) che possiede una INT_ALU ottimizzata che gli permette di raggiungere un miglioramento pari a **2x** nell'esecuzione delle istruzioni intere.

Si valuti il miglioramento globale ottenuto utilizzando CALC2 per l'applicazione software data.

Soluzione

$$\text{Speedup} = \frac{1}{(1 - 0.9) + \frac{0.9}{2}} = 1.81$$

Corollario

- Se un miglioramento è utilizzabile solo per una frazione del lavoro complessivo, allora non è possibile accelerare il lavoro più del reciproco di uno meno tale frazione

$$\text{Speedup}_{\text{globale}} = \frac{1}{(1 - \text{Frazione}_{\text{migliorato}}) + \frac{\text{Frazione}_{\text{migliorato}}}{\text{Speedup}_{\text{migliorato}}}} \leq \frac{1}{1 - \text{Frazione}_{\text{migliorato}}}$$

Esempio 2

Si consideri un miglioramento che consente un funzionamento **10** volte più veloce rispetto alla macchina originaria, ma che sia utilizzabile solo per il **40%** del tempo. Qual è il guadagno complessivo che si ottiene incorporando detto miglioramento?

Soluzione

$$\text{Frazione}_{\text{migliorato}} = 0.4, \quad \text{Speedup}_{\text{migliorato}} = 10$$

$$\text{Speedup}_{\text{globale}} = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} \cong 1.56$$

Esempio 3

Supponiamo di potere aumentare la velocità della CPU della nostra macchina di un fattore 5 (senza influenzare le prestazioni di I/O) con un costo 5 volte superiore.

Assumiamo inoltre che la CPU sia utilizzata per il 50% del tempo ed il rimanente sia destinato ad attesa per operazioni di I/O.

Se la CPU è un terzo del costo totale del computer è un buon investimento da un punto di vista costo/prestazioni, aumentare di un fattore cinque la velocità della CPU?

Soluzione

$$\text{Speedup}_{\text{globale}} = \frac{1}{0.5 + \frac{0.5}{5}} = 1.67 \qquad \text{Incremento di costo} = \frac{2}{3} \times 1 + \frac{1}{3} \times 5 = 2.33$$

L'incremento di costo è quindi più grande del miglioramento di prestazioni: la modifica *non* migliora il rapporto costo/prestazioni.

Esempio 4

Si deve valutare un miglioramento di una macchina per l'aggiunta di una modalità vettoriale. La computazione vettoriale è 20 volte più veloce di quella normale. La percentuale di vettorizzazione è la porzione del tempo che può essere spesa usando la modalità vettoriale.

✓Disegnare un grafico che riporti lo speedup come percentuale della computazione effettuata in modo vettoriale.

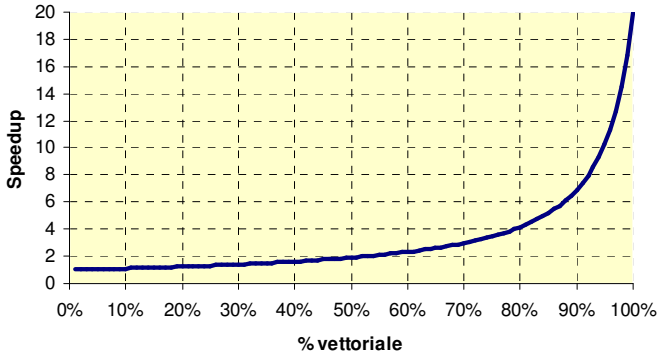
✓Quale percentuale di vettorizzazione è necessaria per uno speedup di 2? Quale per raggiungere la metà dello speedup massimo?

✓La percentuale di vettorizzazione misurata è del 70%. I progettisti hardware affermano di potere raddoppiare la velocità della parte vettoriale se vengono effettuati significativi investimenti. Il gruppo che si occupa dei compilatori può incrementare la percentuale d'uso della modalità vettoriale. Quale incremento della percentuale di vettorizzazione sarebbe necessario per ottenere lo stesso guadagno di prestazioni? Quale investimento raccomandereste?

Soluzione (Esercizio 4)

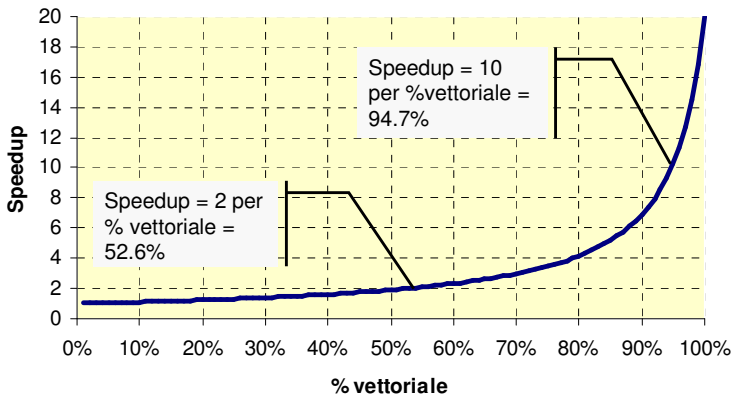
Disegnare un grafico che riporti lo speedup come percentuale della computazione effettuata in modo vettoriale

$$\text{Speedup} = \frac{1}{(1 - \%\text{vettoriale}) + \frac{\%\text{vettoriale}}{20}}$$



Soluzione (Esercizio 4)

Quale percentuale di vettorizzazione è necessaria per uno speedup di 2? Quale per raggiungere la metà dello speedup massimo?



Soluzione (Esercizio 4)

La percentuale di vettorizzazione misurata è del 70%. I progettisti hardware affermano di potere raddoppiare la velocità della parte vettoriale se vengono effettuati significativi investimenti. Il gruppo che si occupa dei compilatori può incrementare la percentuale d'uso della modalità vettoriale. Quale incremento della percentuale di vettorizzazione sarebbe necessario per ottenere lo stesso guadagno di prestazioni? Quale investimento raccomandereste?

$$Speedup_{orig} = \frac{1}{0.3 + \frac{0.7}{20}} = 2.99$$

$$Speedup_{HW} = \frac{1}{0.3 + \frac{0.7}{40}} = 3.15 \qquad Speedup_{comp} = \frac{1}{(1-x) + \frac{x}{20}}$$

$$Speedup_{comp} = Speedup_{HW} \rightarrow x = 0.72$$

La percentuale di vettorializzazione dovrebbe essere almeno del 72% per avere un miglioramento pari a quello del progettista hardware. Consiglierei di intervenire sul compilatore (mi basta miglioralo soltanto del 2%).

Il Tempo di CPU

- Il **tempo** è la misura delle prestazioni di un computer
  - ➔ Il computer che svolge la stessa quantità di lavoro nel minore tempo è il più veloce
- **Tempo di risposta** rappresenta la latenza per il completamento di un lavoro
  - ➔ Includendo accessi a disco, accessi a memoria, attività di I/O
- **Tempo di CPU** rappresenta il tempo speso dalla CPU per eseguire il programma dato
  - ➔ **Non** include il tempo di attesa per I/O o per l'esecuzione di altri programmi
  - ➔ Comprende il **tempo utente di CPU** (tempo speso dalla CPU per eseguire le linee di codice che stanno nel nostro programma) **+ tempo di CPU di sistema** (speso dal sistema operativo per eseguire i compiti richiesti dal programma)

```
% time find . -name "pippo"        % time ./stress
real 0m17.496s                     real 0m2.836s
user 0m0.050s                      user 0m2.830s
sys  0m0.170s                      sys  0m0.010s
```

$$\text{Tempo di CPU} = \text{cicli di clock della CPU} \times \text{durata periodo di clock}$$

$$\text{Tempo di CPU} = \frac{\text{cicli di clock della CPU}}{\text{frequenza di clock}}$$

Cicli di Clock per Istruzione (CPI)

- In genere, istruzioni di tipo diverso richiedono quantità diverse di tempo
  - ➔ La moltiplicazione richiede più tempo dell'addizione
  - ➔ L'accesso alla memoria richiede più tempo dell'accesso ai registri
- Fissata la durata del ciclo di clock, varia il numero di cicli di clock richiesti dalle diverse istruzioni
- Si può calcolare il numero medio di cicli di clock per istruzione di un dato programma

Cicli di Clock per Istruzione (CPI)

$$CPI = \frac{\text{cicli di clock della CPU per eseguire il programma}}{\text{numero di istruzioni eseguite}}$$

$$T_{CPU} = \text{numero di istruzioni eseguite} \times CPI \times \text{periodo di clock}$$

$$T_{CPU} = \frac{\text{numero di istruzioni eseguite} \times CPI}{\text{frequenza di clock}}$$

Utilizzeremo da ora in avanti i seguenti simboli

$$T_{CPU} = IC \times CPI \times T_{CK} = \frac{IC \times CPI}{f_{CK}}$$

CPI (IC)

$$T_{CPU} = IC \times CPI \times T_{CK} = \frac{IC \times CPI}{f_{CK}}$$

- IC dipende dal repertorio di istruzioni e dal grado di ottimizzazione del compilatore
  - ➔ Compilatori diversi possono dare luogo a IC diversi
  - ➔ Uno stesso compilatore che genera codice per due macchine diverse, darà IC diversi
  - ➔ Un repertorio CISC favorisce la riduzione del numero di istruzioni

CPI (T_{CK} o f_{CK})

$$T_{CPU} = IC \times CPI \times T_{CK} = \frac{IC \times CPI}{f_{CK}}$$

- f_{CK} (T_{CK}) è legata alla tecnologia e all'organizzazione architetturale della CPU
  - ➔ Oggi 1500÷3000 MHz sono la norma
  - ➔ Istruzioni complesse richiedono di norma frequenze di più basse
  - ➔ Istruzioni semplici (RISC) permettono di diminuire i ritardi di propagazione nella logica di controllo e, quindi, di diminuire l'ampiezza del periodo di clock

CPI (CPI)

$$T_{CPU} = IC \times CPI \times T_{CK} = \frac{IC \times CPI}{f_{CK}}$$

- CPI dipende dall'architettura e dal repertorio delle istruzioni
  - ➔ Istruzioni semplici richiedono un minor numero di cicli
  - ➔ Attraverso tecniche come la pipeline è possibile portare CPI ad un valore molto vicino ad 1
  - ➔ L'aggiunta di più unità di esecuzione in parallello (macchine superscalari, VLIW) permette di rendere CPI minore di 1

Numero Totale di Cicli di Clock

$$\text{cicli di clock della CPU} = \sum_{i=1}^{n} (CPI_i \times I_i)$$

- Dove si è indicato
  - **n**: numero di istruzioni diverse eseguite
  - **CPI_i**: numero di cicli di clock necessari per eseguire l'istruzione di tipo *i*
  - **I_i**: numero di volte in cui l'istruzione di tipo *i* è stata eseguita
- Questa formula può essere usata per esprimere il tempo di CPU come

$$T_{CPU} = \sum_{i=1}^{n} (CPI_i \times I_i) \times T_{CK}$$

Nuova Espressione per il CPI

$$CPI = \frac{\sum_{i=1}^{n} (CPI_i \times I_i)}{IC} = \sum_{i=1}^{n} \left(CPI_i \times \frac{I_i}{IC} \right)$$

- Cioè ogni CPI_i viene moltiplicato (pesato) per la frazione delle occorrenze nel programma

Esempio 5

Si consideri un calcolatore in grado di eseguire le istruzioni riportate in tabella.

Calcolare CPI e il tempo di CPU per eseguire un programma composto da 100 istruzioni supponendo di usare una frequenza di clock pari a 500 MHz.

Tipo	Frequenza	Cicli di clock
ALU	43%	1
Load	21%	4
Store	12%	4
Branch	12%	2
Jump	12%	2

Soluzione

CPI = 1*0.43 + 4*0.21 + 4*0.12 + 2*0.12 + 2*0.12 = 2.23

T_{CPU} = IC * CPI / f_{CK} = 100 * 2.23 * 1/(500*10^6) = 446 ns

Esercizio

Si considerino le seguenti statistiche di esecuzione di un certo programma: 30% di istruzioni di load/store, 40% di istruzioni ALU, 20% di istruzioni di salto, 10% di istruzioni di calcolo in virgola mobile. Le istruzioni di load/store richiedono 32 cicli di clock, le istruzioni ALU 1 ciclo di clock, le istruzioni di salto 5 cicli di clock e le istruzioni di calcolo in virgola mobile richiedono 16 cicli di clock.
Se si migliorano le prestazioni dell'unità di calcolo in virgola mobile del 30%, qual è lo speed-up ottenuto?

Esercizio (Svolgimento 1)

Speedup = $TCPU_{vecchio}$ / $TCPU_{nuovo}$

$TCPU_{vecchio}$ = IC * $CPI_{vecchio}$ * T_{CK}
$TCPU_{nuovo}$ = IC * CPI_{nuovo} * T_{CK}

Quindi sostituendo, Speedup = $CPI_{vecchio}$ / CPI_{nuovo}

$CPI_{vecchio}$ = 30% * 32 + 40% * 1 + 20% * 5 + 10%* 16 = 12.6 [cicli/istruzione]
CPI_{nuovo} = 30% * 32 + 40% * 1 + 20% * 5 + 10%* (16/1.3) = 12.2 [cicli/istruzione]

L'unità in virgola mobile migliora del 30%

Per cui,

Speedup = 12.6/12.2 = 1.03 (le prestazioni migliorano del 3%)

Esercizio (Svolgimento 2)

Speedup = $1/[(1-F_{migliorata})+F_{migliorata}/Speedup_{migliorato}]$

$Speedup_{migliorato}$ = 1.3

$F_{migliorata}$ = Frazione di tempo in cui è utilizzata l'unità in virgola mobile =
= Frazione dei cicli in cui è usata l'unità in virgola mobile =
= (10%*16) / (30% * 32 + 40% * 1 + 20% * 5 + 10%* 16) = 13%

Quindi sostituendo,

Speedup = 1/(87% + 13%/1.3) = 1.03

MIPS

- **MIPS**: **M**ilioni di **I**struzioni **P**er **S**econdo

$$\text{MIPS} = \frac{\text{numero di istruzioni eseguite}}{\text{tempo di esecuzione} \times 10^6} = \frac{IC}{T_{CPU} \times 10^6}$$

$$\text{Ricordando che } T_{CPU} = \frac{IC \times CPI}{f_{CK}}$$

$$\text{MIPS} = \frac{f_{CK}}{CPI \times 10^6}$$

La caratteristica positiva del MIPS è che risulta semplice da comprendere infatti le macchine più veloci hanno valori di MIPS più elevati.

MIPS - Problemi

- Dipende dall'insieme di istruzioni, quindi è difficile confrontare computer con diversi insiemi di istruzioni
- Varia a seconda del programma considerato
- Può variare in modo inversamente proporzionale alle prestazioni!
- **Esempio**
  - ➔ Macchina con hardware opzionale per virgola mobile. Le istruzioni in virgola mobile richiedono più cicli di clock rispetto a quelle che lavorano con interi, quindi i programmi che usano l'hardware opzionale per la virgola mobile in luogo delle routine software per tali operazioni impiegano meno tempo ma hanno un MIPS più basso. L'implementazione software delle istruzioni in virgola mobile esegue semplici istruzioni, con il risultato di avere un elevato MIPS, ma ne esegue talmente tante da avere un più elevato tempo di esecuzione.

MIPS - Esempio

	IC	CPI	T_{CK}
Processore 1	2N	2	T
Processore 2	N	1,5	2T

- $T_{CPU1} = IC_1 * CPI_1 * T_{CK1} = 4*N*T$
- $T_{CPU2} = IC_2 * CPI_2 * T_{CK2} = 3*N*T$

$T_{CPU1} > T_{CPU2}$ → CPU_2 è migliore di CPU_1

- $MIPS_1 = 1/(CPI_1 * T_{CK1} * 10^6) = 1/(2T * 10^6)$
- $MIPS_2 = 1/(CPI_2 * T_{CK2} * 10^6) = 1/(3T * 10^6)$

$MIPS_1 > MIPS_2$ → CPU_1 è migliore di CPU_2

Benchmarks

- Benchmark = Programmi per valutare le prestazioni
  - → **Programmi reali**
    - ✓ Pur non sapendo il venditore quale frazione del tempo sarà dedicata a questi programmi, egli sa che alcuni utenti li useranno per risolvere problemi reali. Le prestazioni sono valutate meglio eseguendo un'applicazione reale che rappresenti un tipico carico di lavoro
  - → **Benchmark ridotti**
    - ✓ Tentativi di circoscrivere piccoli pezzi chiave dai programmi
  - → **Benchmark sintetici**
    - ✓ Cercano di simulare la frequenza media degli operandi e delle operazioni di un vasto insieme di programmi
- SPEC (System Performance Evaluation Cooperative)

Benchmark Suite

- Insieme di benchmarks
- Media aritmetica dei tempi di esecuzione
  - Una media dei tempi di esecuzione che rispetta il tempo di esecuzione totale è la *media aritmetica*
    $$\frac{1}{n}\sum_{i=1}^{n}\text{Tempo}_i$$
  - Tempo_i è il tempo per il programma i-esimo degli *n* programmi che costituiscono la suite
- Media aritmetica pesata dei tempi di esecuzione
  - Se il carico di lavoro è composto da un insieme di programmi non eseguiti tutto lo stesso numero di volte, si assegna un peso w_i ad ognuno di essi per indicare la frequenza relativa del programma entro il carico di lavoro. Si usa quindi una *media aritmetica pesata*
    $$\sum_{i=1}^{n}(\text{Peso}_i \times \text{Tempo}_i)$$
  - Peso_i rappresenta la frequenza del programma i-esimo entro il carico di lavoro e Tempo_i è il tempo di esecuzione di tale programma