Appendice A: Caratteristiche di processori esistenti

Queste note riportano le caratteristiche di alcuni processori esistenti, selezionati in modo da rappresentare le famiglie più diffuse ed, al tempo stesso, alcuni degli esemplari più recenti e tecnologicamente più avanzati.

Le caratteristiche riguardano aspetti architetturali (macchina assembler Risc *vs* Cisc, ciclo di clock, memorie, cache su più livelli, tecnologia dei chip), prestazioni sulla base di benchmark standard (SPEC) e sulla base di applicazioni complete.

Per quanto tutti i processori abbiano architettura interna parallela (studiata nel corso di "Architetture Parallele e Distribuite" della Laurea Specialistica) pipeline, superscalare, multi-threading, hyper-threading, SMP), le caratteristiche analizzate sono largamente indipendenti da questo aspetto.

Il presente materiale non fa parte del programma di esame, ma vuole servire da informazione ed esemplificazione per gli studenti che siano interessati all'evoluzione tecnologica nel settore, permettendo di ricondurre le caratteristiche tecnologiche di prodotti di mercato a quanto studiato nel corso di Architettura degli Elaboratori.

Una analisi più avanzata (processori ad alte prestazioni, multiprocessor, cluster, reti ad alta velocità), sempre riferita a prodotti innovativi esistenti, sarà effettuata nel suddetto corso di Architetture Parallele e Distribuite.

Gli interessati possono consultare il testo del seminario "Introduzione ai Sistemi ad Alte Prestazioni", allegato alle presenti note (Appendice B).

Caratteristiche architetturali di processori attualmente sul mercato

Model	G4 - PPC745x	G5 (PPC 970FX)	Pentium4 HT (Prestonia)	Power5	Itanium2 – MP (Madison)
Manufacturer	IBM-Motorola	IBM	Intel	IBM	Intel
Classification	RISC	RISC	CISC Hyperthreading	Symmetric Multithreading doublecore on a single chip	EPIC (Explicitly Parallel Instruction Computing)
Shipping Date	2001	Oct 2002	Feb 2002	Nov 2004	Dec 2004
Core					
Word (Bits)	32	64	32	64	64
Clock Speed (GHz)	0.55 – 1.42	1.6 – 2.0	2.0 – 3.20	1.5 – 1.9	1.0 – 1.6
Cache					
cache block width (bytes)	32 ($\sigma = 8$)	128 ($\sigma = 16$)	32 ($\sigma = 8$)	L1, L2 = 128 ($\sigma = 16$) L3 = 256 ($\sigma = 32$)	128 ($\sigma = 16$)
L1 Instr. size	32K, 8-set.	64K, direct-map	8K + 12K trace cache	64K, 2-set	16K, 4-set (min lat. 1 τ)
L1 Data size	32K, write-back, 8-set.	32K, write-thru, 2-set.	8K, 4-set	32K, 4-set	16K, (min lat. 1 τ)
L2 (KB)	256	512	512 (8-set)	1900, 10-set, shared (3 independent banks), 2	256, 8-set (min lat. 5 τ)
L3 (MB)	2	0	2 on die	36, 12-set (32 GB/s)	9 (min lat. 12 τ, 32 GB/s)
Memory					
P-M data width (bits)	64	128	64	2 x 128	128
P-M address width (bits)	36	42	32	42	50
P-M bandwidth (GB/sec)	1.3	8.0	6.4	16 per chip	6.4
Mem bandwidth (GB/sec)	2.7	6.4	System dep	> 20 GB/s per chip	System dep.
Mem latency (nsec)	93	135	System dep	System dep.	System dep.
Technology					
Process	0.18	0.13	0.13	0.13	0.13
Die size	106 mm2	131 mm2	146 mm2	389mm2	432mm2
Transistor count	33 M	55 M	55 M	276M	592M
Voltage/Power	1.6V/30W	1.3V/42W	1.525V/68.4W	1.3V/ ? W	1.35V/130W

Note:

- ***L1, L2, L3****: livelli di cache. L1 suddivisa in due unità distinte, una per solo istruzioni e una per solo dati.*
- ***P-M xxx width****: ampiezza del collegamento (in bit) per informazione di tipo xxx tra CPU e memoria esterna al chip.*

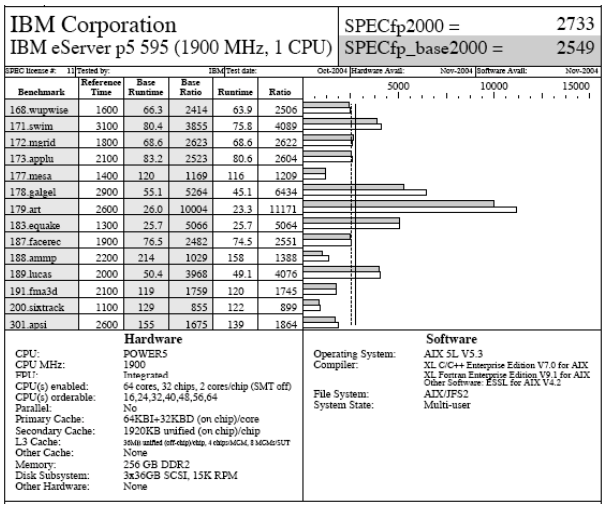
Risultati di benchmark

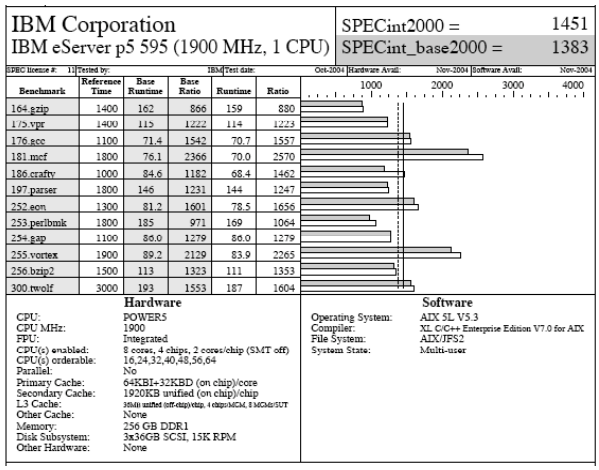
Standard Performance Evaluation Corporation (SPEC, www.spec.org)

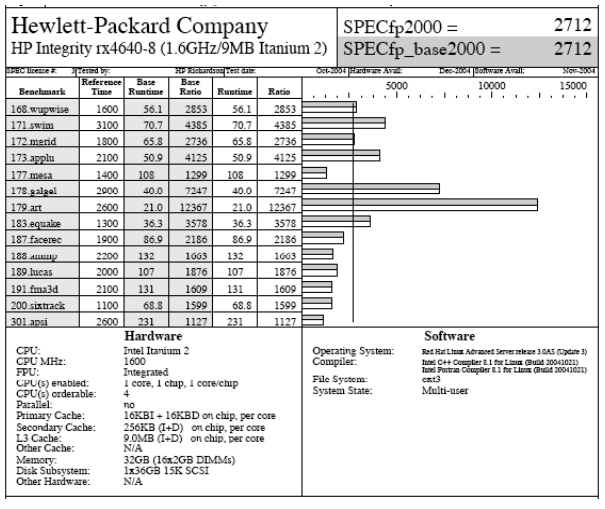
Computer type		CPU					Memory	SPEC	
Company	Model	Model	Freq	L1	L2	L3		int	fp
Intel	D815EEA2	Pentium III	1.0 GHz	16K+16 K	256K	NA	PC333	457	264
Intel	D850EMV2	Pentium 4	2.0 GHz	12Kops+ 8K	256K	NA	PC800-40 RDRAM	684	745
Intel	D850EMVR	Pentium 4 HT	3.06 GHz	12Kops+ 8K	512K	NA	PC1066-32 RDRAM	1107	1091
Intel	D925XECV2	Pentium 4 HT	3.80 GHz	12Kops+ 16K	1M	NA	PC2-4300 DDRII	1671	1842
AMD	MSI K8N Neo2	Athlon 64 FX-55	2.6 GHz	64K+64 K	1M	NA	PC3200 DDR	1735	1878
IBM	eServer p5 595	Power5	1.9 GHz	64K+32 K	1.9M	36 M	DDR2	1451	2733
HP	Integrity 4640-8	HP-Intel Itanium2	1.6 GHz	16K+16 K	256K	9M	DIMM	1590	2712

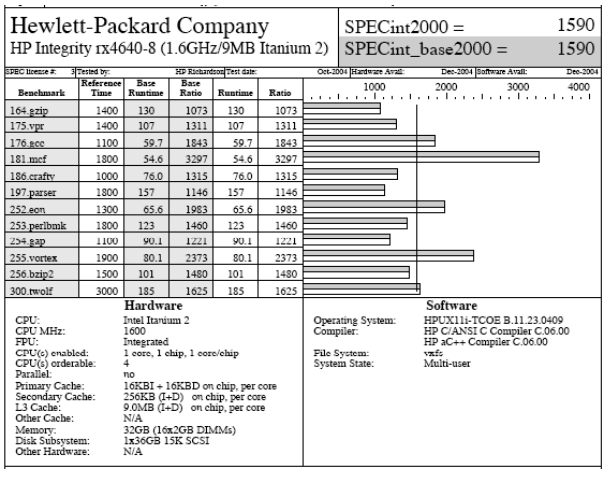
Nota: *SPEC int e SPEC fp si riferiscono a benchmark standard su tipi di dato intero e reale rispettivamente.*

Si noti che i prodotti Intel sono noti per totalizzare altissimi risultati sui benchmark SPEC anche grazie al compilatore (realizzato dalla Intel stessa) che è tarato proprio per generare codici altamente ottimizzati in relazione agli algoritmi prescritti dai benchmark.



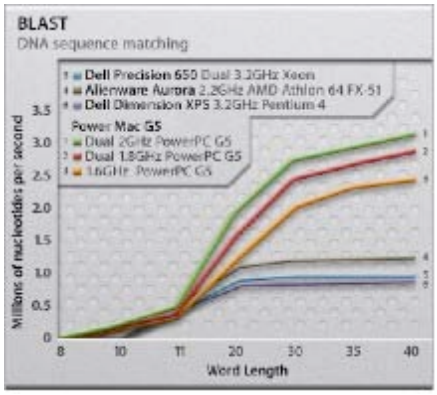


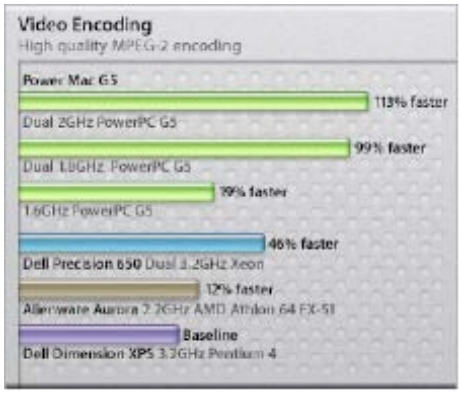


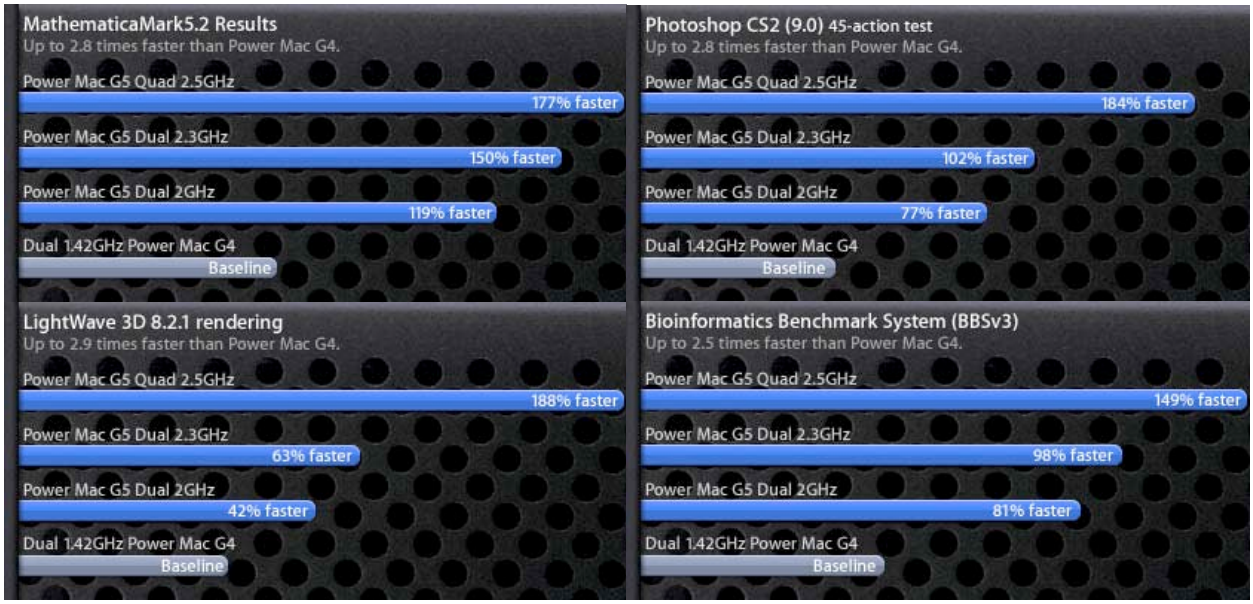


Esempi di benchmark su applicazioni complete

Mentre i benchmark SPEC si riferiscono ad una collezione di algoritmi, o “kernel”, spesso ha interesse avere dei benchmark riferiti ad applicazioni complete, come nei casi seguenti:







Alcuni schemi di CPU commerciali

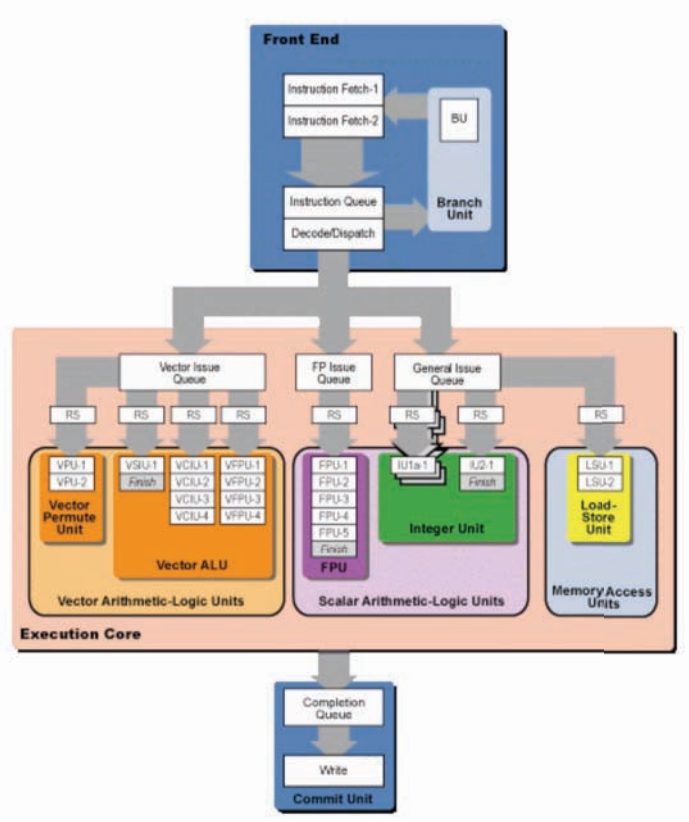


Figure 1: IBM-Motorola G4 (PPC 745x)

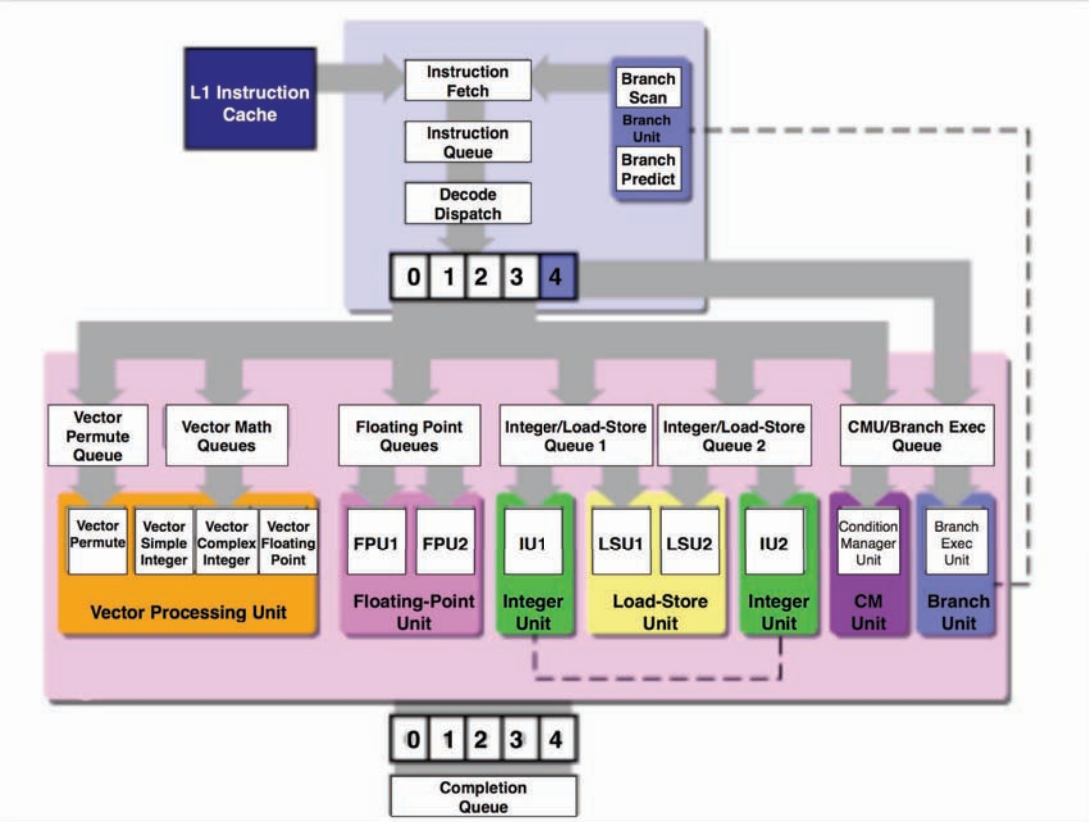


Figure 2: IBM G5 (PPC 970)

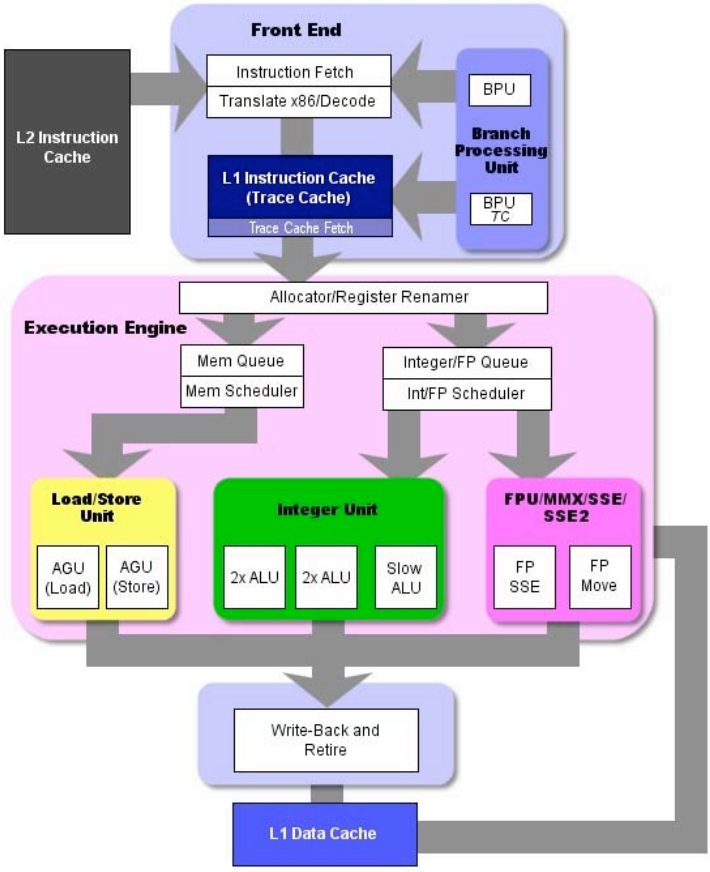


Figure 3: Intel Pentium 4

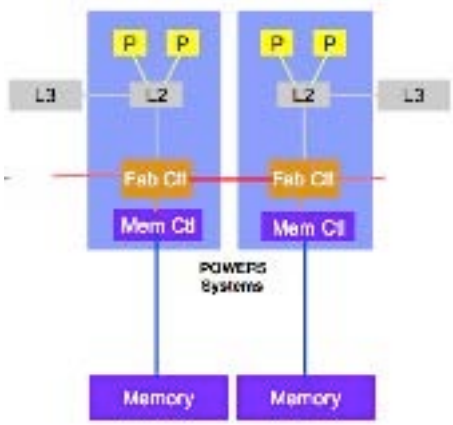


Figure 4: 4-way SMP (2 power5 chips)

Figure 5: 8-way SMP (4 power5 chips, in the middle) with 144MB L3 cache (4 chips external)

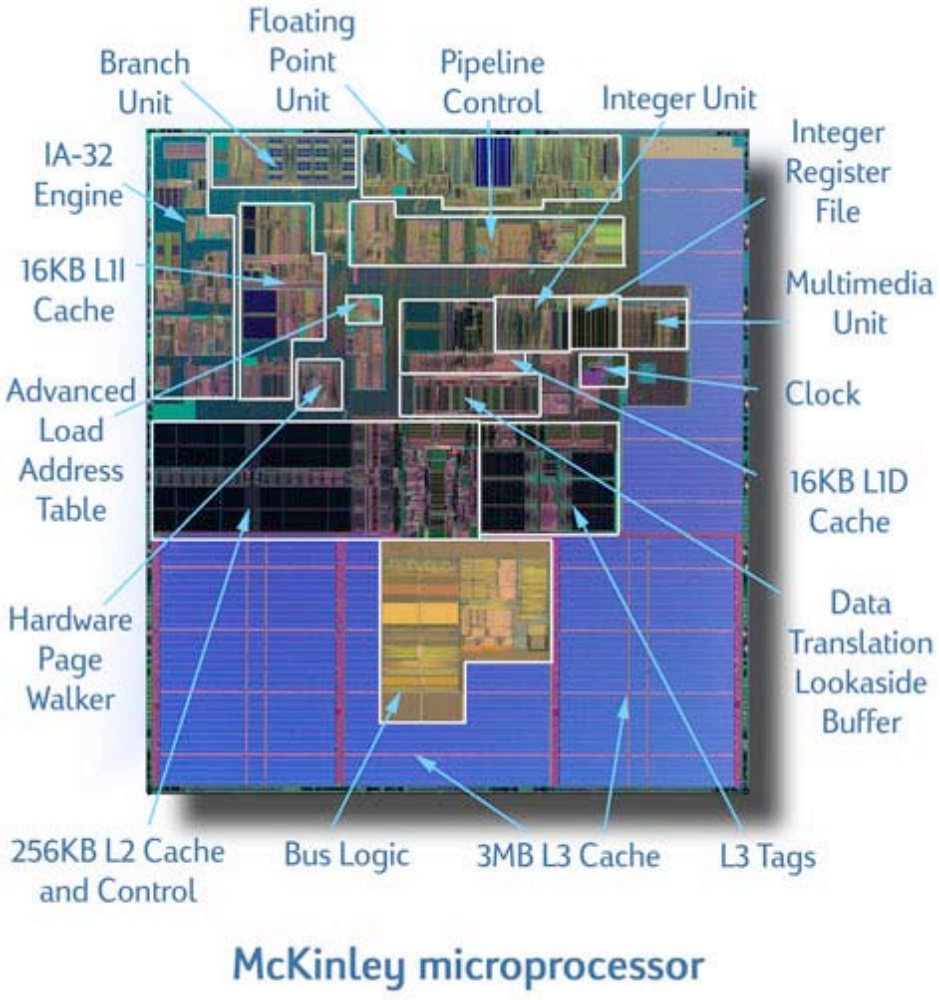


Figure 6: Intel Itanium 2 (MkKinley) chip

Esempi di ottimizzazioni di codice legate alla località

Originale | Ottimizzato

Prefetch software

```
int a[N], b[N];

for(i=0; i<N; i=i+1)
  a[i]=a[i]*b[i];
```

```
// Si assume N=SIGMA*K
int a[K][SIGMA], b[K][SIGMA]

for(i=0; i<K; i=i+1) {
  prefetch(a[i+1][0]);
  prefetch(b[i+1][0]);
  for (j=0; j< SIGMA; j=j+1)
    a[i][j]=a[i][j]*b[i][j];
}
```

Fusione di array

```
int a[N], b[N];

for(i=0; i<N; i=i+1)
    a[i]= a[i]+b[i];
```

```
struct merge{
  int a;
  int b;
};
struct merge ab[N];

for(i=0; i<N; i=i+1)
  ab.a[i]=ab.a[i]+ab.b[i];
```

Scambio di loop

```
int a[N][M];

for(j=0; j<M; j=j+1)
  for(i=0; i<N; i=i+1)
    a[i][j]= 5*a[i][j];
```

```
int a[N][M];

for(i=0; i<N; i=i+1)
  for(j=0; i<M; j=j+1)
    a[i][j]= 5*a[i][j];
```

Fusione di loop

```
int a[N][M], b[N][M], c[i][j], d[i][j];

for(i=0; i<N; i=i+1)
  for(j=0; i<M; j=j+1)
    a[i][j]= 5*b[i][j]*c[i][j];

for(i=0; i<N; i=i+1)
  for(j=0; i<M; j=j+1)
    d[i][j]= a[i][j]+c[i][j];
```

```
int a[N][M], b[N][M], c[i][j], d[i][j];

for(i=0; i<N; i=i+1)
  for(j=0; i<M; j=j+1) {
    a[i][j]= 5*b[i][j]*c[i][j];
    d[i][j]= a[i][j]+ c[i][j];
  }
```

Riduzione a blocchi

```
int a[N][N], b[N][N], ab[N][N] = 0;

for(i=0;i<N;i=i+1)
  for(j=0;j<N;j=j+1) {
    r=0;
    for(k=0;k<N;k=k+1) {
      r = r+a[i][k]*b[k][j];
    }
    ab[i][j] = r;
  }
```

```
int a[N][N], b[N][N], ab[N][N] = 0;

for(jj=0;jj<N;jj=jj+B)
  for(kk=0;kk<N;kk=kk+B)
    for(i=0;i<N;i=i+i) {
      for(j=jj; j<min(jj+B-1,N);j=j+1) {
        r=0;
        for(k=kk;k<min(kk+B-1,N);k=k+1) {
        r = r + a[i][k]*b[k][j];
      }
      ab[i][j] = ab[i][j]+r;
    }
```

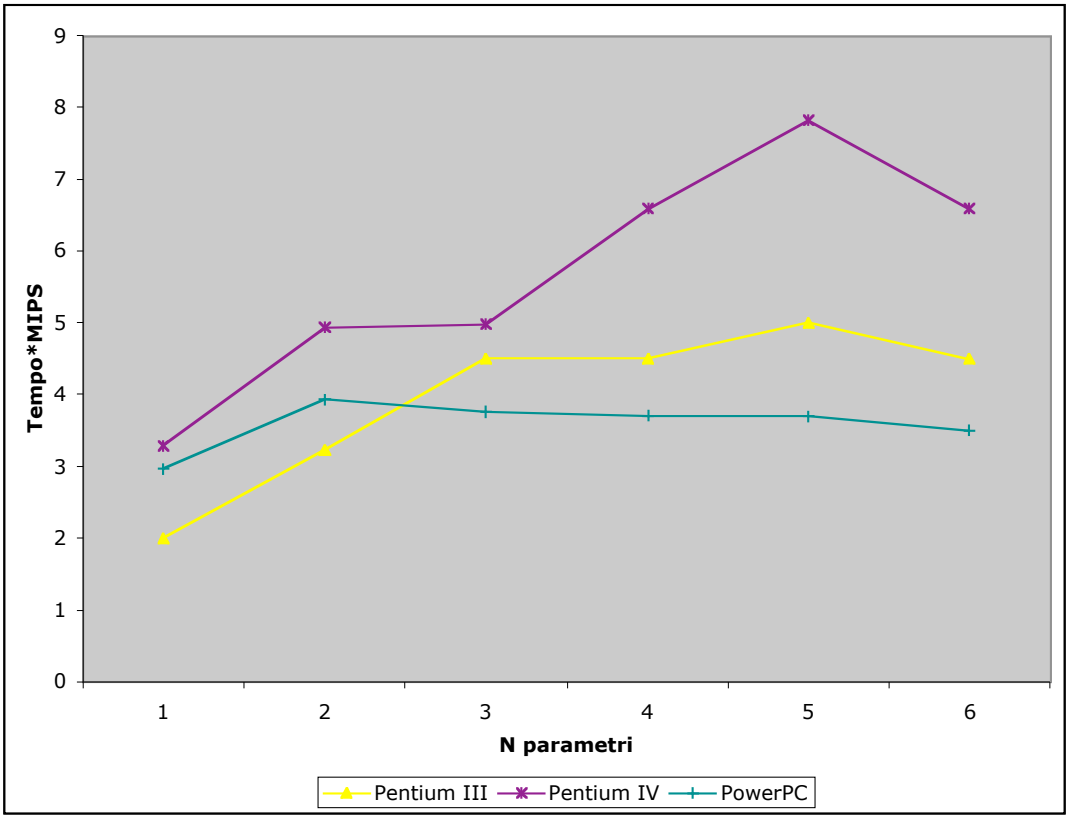


Figure 7: Performance della chiamata di funzione vs numero di parametri per processori con un numero diverso di registri (PowerPC = 32, Pentium X86 = 4+4)