


Gestione dell'Input / Output
Valutazione delle prestazioni

Prof. Alberto Borghese
Dipartimento di Informatica
alberto.borghese@unimi.it

Università degli Studi di Milano

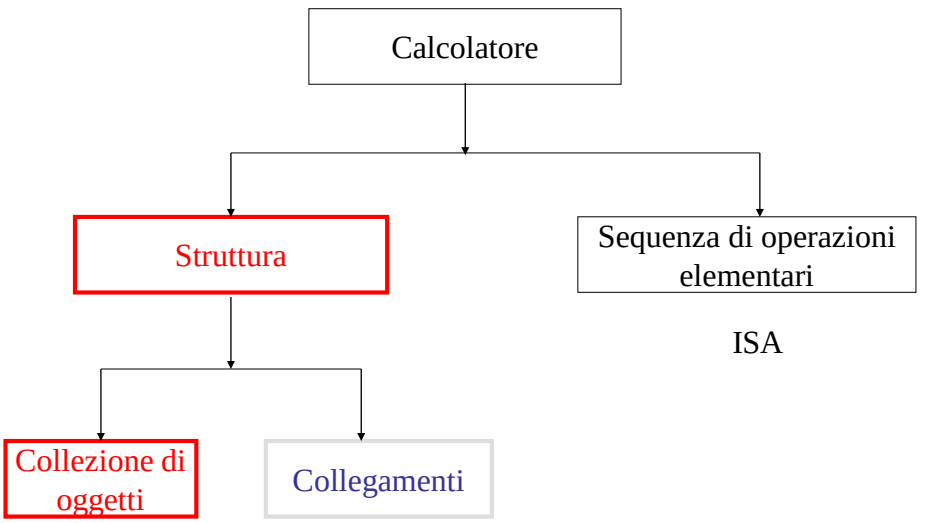
Riferimento Patterson: 6.3, 6.4 e 6.6


Sommario

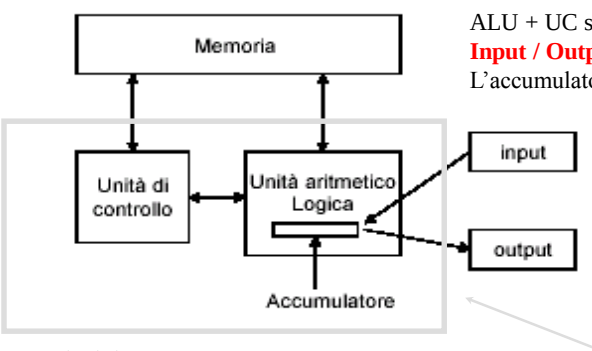
I collegamenti tra CPU e gli altri componenti

Trasferimento tra I/O e CPU

Descrizione di un elaboratore



Architettura di Von Neumann



ALU + UC sono incorporate nella CPU.
Input / Output vengono in realtà trasferiti via bus.
L'accumulatore è un possibile tipo di architettura.

I principi:
• I dati e le istruzioni sono memorizzate in una memoria read/write.
• Il contenuto della memoria può essere recuperato in base alla sua posizione, e non è funzione del tipo di dato.
• L'esecuzione procede sequenzialmente da un'istruzione alla seguente.

•Gia' vista e modificata (evoluzione nel tempo).

I/O

E' la parte più complessa di un'architettura per la sua variabilità

Dispositivi eterogenei per:

velocità di trasferimento.

latenze.

sincronismi.

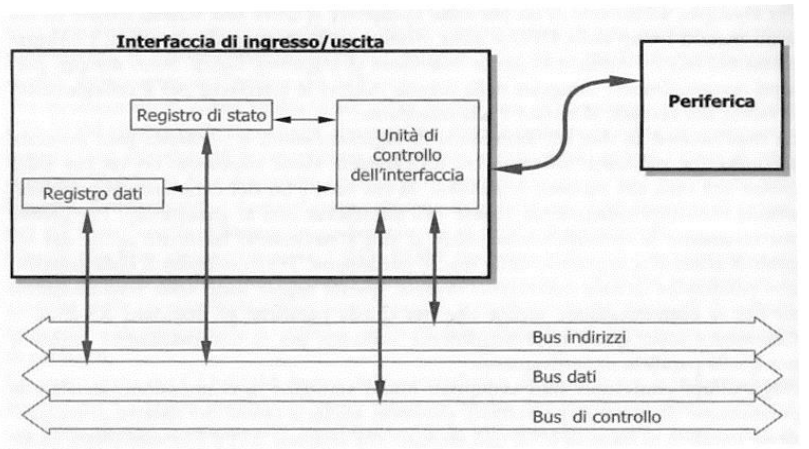
modalità di interazione (con l'uomo o con una macchina)

Dispositivi di I/O - classificazione

Device	Behavior	Partner	Data rate (Mbit/sec)
Keyboard	Input	Human	0.0001
Mouse	Input	Human	0.0038
Voice input	Input	Human	0.2640
Sound input	Input	Machine	3.0000
Scanner	Input	Human	3.2000
Voice output	Output	Human	0.2640
Sound output	Output	Human	8.0000
Laser printer	Output	Human	3.2000
Graphics display	Output	Human	800.0000–8000.0000
Cable modem	Input or output	Machine	0.1280–6.0000
Network/LAN	Input or output	Machine	100.0000–10000.0000
Network/wireless LAN	Input or output	Machine	11.0000–54.0000
Optical disk	Storage	Machine	80.0000–220.0000
Magnetic tape	Storage	Machine	5.0000–120.0000
Flash memory	Storage	Machine	32.0000–200.0000
Magnetic disk	Storage	Machine	800.0000–3000.0000

+ latenza

Modalità trasferimento dati

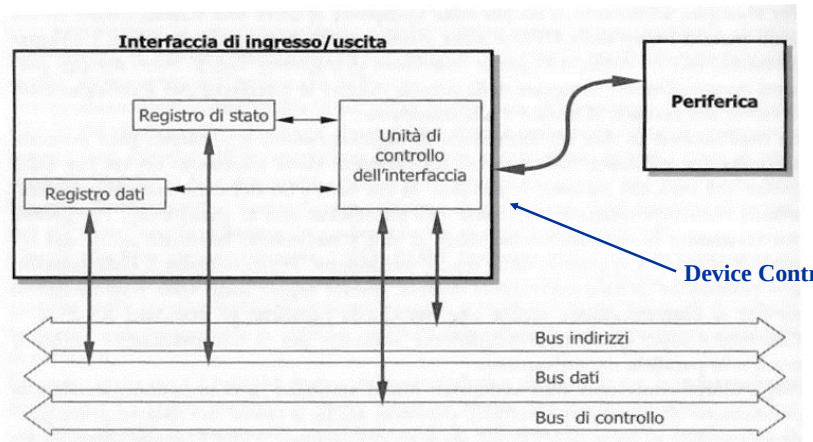


Parallela (bus PCI, bus processore-memoria).
Seriale (RS232, RS432). 1 bit alla volta.
Seriale ad alta velocità: USB, Firewire (IEEE 1394), PCIe.

Bus & buffer

- I dispositivi sono collegati al bus tramite porte.
- I dispositivi collegati al bus variano in termini di velocità dell'esecuzione delle operazioni ⇒ necessario un meccanismo di sincronizzazione per garantire il trasferimento efficiente delle informazioni sul bus.
- Tipicamente all'interno delle unità che utilizzano il bus sono presenti dei **registri di buffer** per mantenere l'informazione durante i trasferimenti e non vincolarsi alla velocità del dispositivo più lento connesso al bus.
- All'interno dell'ampiezza di banda massima si può:
  - **Aumentare la velocità di trasferimento.** Buffer grossi.
  - **Ridurre i tempi di risposta (latenza).** Buffer piccoli.

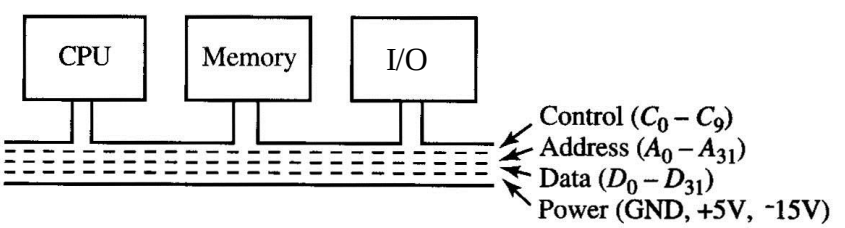
Porta I/O



Segnali di controllo: Busy, Ack, Interrupt...
Registri:
•Dati (registro o buffer di memoria)
•Stato: situazione della periferica (idle, busy, down….) e comando in esecuzione.

Il driver agisce inviando alla periferica comandi opportuni e dati.

Il bus (connessione a cammino comune)

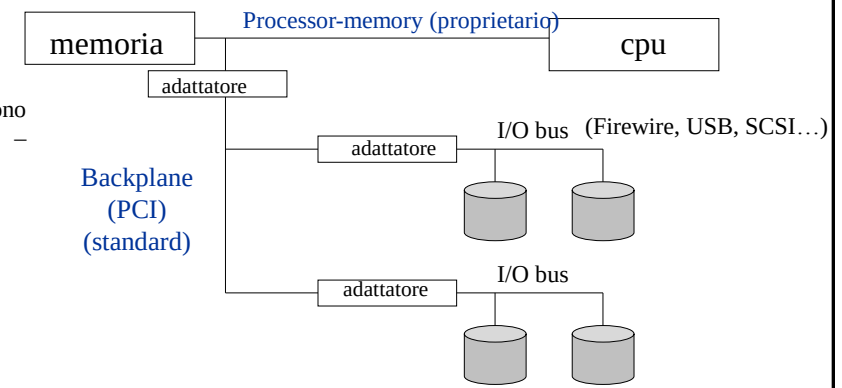


Pathway che connette tutti i dispositivi in modo bidirezionale (a partire dal PDP-8, omnibus, 1965). Trasferimento parallelo dei dati. Connessioni bidirezionali.

Principali vantaggi della struttura a bus singolo:
• elevata flessibilità
• bassi costi.
• Problema: i dispositivi non possono trasmettere contemporaneamente.

Le architetture contengono uno o più bus che collegano questi tre componenti.

La gerarchia dei bus



Gli adattatori sono dispositivi attivi – bridges.

Processore-Memoria (cache): lunghezza ridotta, molto veloci, tipicamente sincroni.
Back-plane: notevole lunghezza, molti device connessi.
I/O: servono per far coesistere la memoria, il processore e i dispositivi di I/O su di un unico bus. Tipicamente asincroni.

Struttura gerarchica – gerarchia + stella

Il PCI Express

Caratteristica	PCI	PCI Express
Tipo di bus	Back-plane	Versatile (back-plane, I/O)
Ampiezza di base del bus (numero di segnali per i dati)	32-64 (collegamento bidirezionale)	4 + 4 (In/Out) (collegamento monodirezionale)
Numero di dispositivi master	molti	1
Temporizzazione	Sincrono 33-66Mhz	Sincronizato: 2.5GHz
Modalità di funzionamento	Sincrona parallela (1,024Mbit/s – 4,096Mbit/s)	Sincrona seriale (2,5Gbit/s)
Ampiezza di banda di picco teorica	133-512MB/s (PCI64)	300MB/s per direzione
Ampiezza di banda stimata raggiungibile per bus di base	80MB/s	1,200MB/s (x 4 linee)
Massimo numero di dispositivi	1024 (32 dispositivi per segmento)	1
Massima lunghezza del bus	0,5 metri	0,5 metri
#Mbyte / s / linea	4-16	75 + 75
Nome dello standard	PCI	PCI - Express

bottleneck

La comunicazione tra i componenti

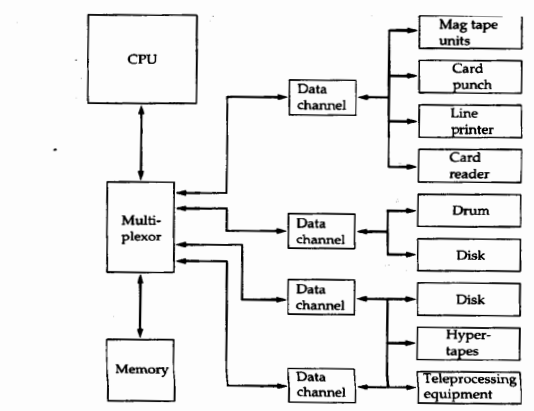


Figure 2.5 An IBM 7094 Configuration

Switch centralizzato (multiplexor)

Architettura a nodo comune (a bus) (cf. bus PCI)

Programma di "canale"

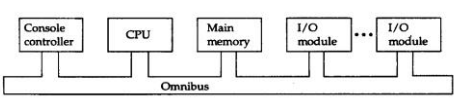


Figure 2.9 PDP-8 Bus Structure

	Intel 5000P chip set	Intel 975X chip set	AMD 580X CrossFiret
Target segment	Server	Performance PC	Server/Performance PC
Front Side Bus (64 bit)	1066/1333 MHz	800/1066 MHz	—
Memory controller hub ("north bridge")			
Product name	Blackbird 5000P MCH	975X MCH	
Pins	1432	1202	
Memory type, speed	DDR2 FBDIMM 667/533	DDR2 800/667/533	
Memory buses, widths	4 × 72	1 × 72	
Number of DIMMs, DRAM/DIMM	16, 1 GB/2 GB/4 GB	4, 1 GB/2 GB	
Maximum memory capacity	64 GB	8 GB	
Memory error correction available?	Yes	No	
PCIe/External Graphics Interface	1 PCIe x16 or 2 PCIe x	1 PCIe x16 or 2 PCIe x8	
South bridge interface	PCIe x8, ESI	PCIe x8	
I/O controller hub ("south bridge")			
Product name	6321 ESB	ICH7	580X CrossFire
Package size, pins	1284	652	549
PCI-bus: width, speed	Two 64-bit, 133 MHz	32-bit, 33 MHz, 6 masters	—
PCI Express ports	Three PCIe x4		Two PCIe x16, Four PCI x1
Ethernet MAC controller, interface	—	1000/100/10 Mbit	—
USB 2.0 ports, controllers	6	8	10
ATA ports, speed	One 100	Two 100	One 133
Serial ATA ports	6	2	4
AC-97 audio controller, interface	—	Yes	Yes
I/O management	SMbus 2.0, GPIO	SMbus 2.0, GPIO	ASF 2.0, GPIO

I bridge

Arbitraggio del bus

Protocollo La comunicazione su bus deve essere regolata attraverso un **protocollo di comunicazione**.

Viene introdotto il concetto di **bus master (padrone del bus)**, il cui scopo è quello di controllare l'accesso al bus.

L'architettura più semplice è quella che prevede un unico bus master (il processore) in cui tutte le comunicazioni vengono mediate dal processore stesso.

Questo può creare un collo di bottiglia. Ad esempio nel caso di trasferimento di dati da I/O a memoria.

Si utilizza allora un'architettura con più dispositivi master.

In questo caso occorre definire e rispettare una policy che coordini i vari dispositivi bus master. Questa policy si chiama di **arbitraggio** del bus. Un solo dispositivo alla volta può essere master, tutti gli altri ascoltano.

Questo è il principale inconveniente dei bus a nodo comune.

Protocollo di arbitraggio

Occorre stabilire quale master autorizzare all'utilizzo del bus.

Ad ogni dispositivo viene assegnata una *priorità*.

Il dispositivo a priorità maggiore può accedere prima al bus.

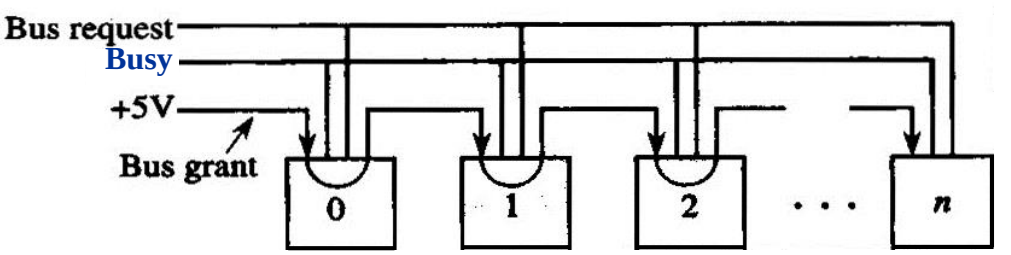
Meccanismo di accesso al bus diventa:
1. Richiesta del bus (*bus request*)
2. Assegnamento del bus (*bus grant*)

Problema è assicurare una *fairness*.

Compromesso tra fairness e priorità.

Un arbitro si preoccupa quindi di gestire *bus request* e *bus grant.*

Arbitraggio distrubuito in Daisy Chain



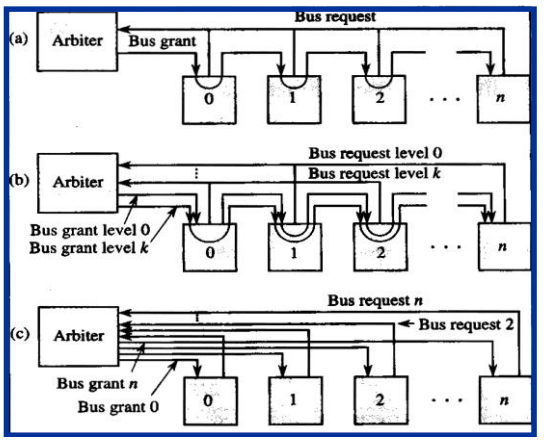
Viene introdotto il segnale di Busy.

Schemi di arbitraggio centralizzati

Arbitraggio
In Daisy Chain

Arbitraggio
Centralizzato
Con priorità

Arbitraggio
Centralizzato
Parallelo



- Arbitraggio centralizzato parallelo non scala con il numero di dispositivi.
- Nell'arbitraggio centralizzato con priorità, un dispositivo elabora un solo segnale di grant, gli altri segnali di grant vengono fatti passare inalterati.

Arbitraggio distribuito con autoselezione

In questo schema un dispositivo che vuole prendere il controllo del bus, deve:

- 1) Inviare il segnale di richiesta del bus.
- 2) Scrivere sul bus il codice che lo identifica.
- 3) Controllare se il bus è libero.
- 4) Se il bus non è occupato ma ci sono richieste contemporanee di bus, controllare il codice dei dispositivi che hanno fatto richiesta.
- 5) Occupare il bus se è libero o i dispositivi hanno priorità minore: inviare 0 sulla linea di bus grant ai dispositivi successivi, asserisce la linea di busy (il bus è occupato), altrimenti non fare nulla. *Ciascun dispositivo è arbitro.*
- 6) Deasserire la richiesta del bus.

Problema: rilevamento delle collisioni. Occorre prevedere un segnale di ricevuto.

Sommario

I collegamenti tra CPU e gli altri componenti

Trasferimento tra I/O e CPU

Modalità di trasferimento dati

Controllo da programma:
con polling.

Mediante interrupt:
con DMA.

Funzionamento di un driver

Funzione dei driver:
- Controllano l'operato dei device controller.
- Gestiscono lo scambio dei dati dal controller (registro dati) e la memoria.

1. CPU richiede alla periferica (controller della periferica) l'esecuzione di un'operazione di read o di write.
2. I dati coinvolti nell'operazione devono essere trasferiti da e verso la memoria centrale.

Per potere eseguire un'operazione di read / write, occorre spesso una serie di operazioni sul dispositivo, che vengono eseguite attraverso il controller.

Esempio: una stampante ha 1 registro dati ed 1 registro di stato. Il registro di stato contiene il bit done, che viene impostato a 1 quando il carattere è stato stampato; ed il bit error che, indica se ci sono problemi. Il processore deve controllare che non ci siano errori e che il bit di done sia stato settato ad 1 prima di inviare un altro dato.

Per potere inviare i comandi al controller, occorre prima avere individuato il controller giusto!!
A ciascun dispositivo viene dato uno o più numeri -> indirizzo personalizzato.

2 modalità:
- Memory-mapped
- Istruzioni speciali di I/O

Istruzioni speciali di I/O

Istruzioni appartenente alla ISA che indirizzano direttamente il dispositivo (i registri del dispositivo):
- Numero del dispositivo
- Parola di comando (o indirizzo della parola che contiene il comando)

Sul bus è possibile inviare il numero del dispositivo su un insieme di linee dedicate.
Ci saranno linee dedicate anche ai segnali di controllo (read / write).
I dati viaggeranno sulle linee dedicate ai dati.

Rendendo le istruzioni illegali al di fuori del kernel mode del processore, i programmi utenti non accedono direttamente ai device controller.

Esempio di architetture di questo tipo: Intel IA-32, IBM370.

```
out 70h, ax    # Trasferisci alla porta di I/O 70, il contenuto di AX
```

Indirizzamento memory-mapped

•I registri del device controller sono considerati come celle di memoria RAM.
•I loro indirizzi saranno diversi da quelli delle celle di memoria RAM effettive.
•Il processore esegue operazioni di I/O come se fossero operazioni di lettura/scrittura in memoria.

Esempio:

sw $s0, indirizzo
lw $s0, indirizzo

dove l'*indirizzo* è al di fuori dallo spazio fisico della memoria.

• I controller ascoltano tutti i segnali in transito sul bus (*bus snooping*) e si attivano solamente quando riconoscono sul bus indirizzi, l'indirizzo corrispondente alla propria locazione di memoria.

• Gli indirizzi riservati ai registri del controller fanno di solito riferimento alla porzione di memoria riservata al SO e non accessibile quindi al programma utente.

• I programmi utente devono quindi passare dal SO per accedere a questi indirizzi riservati (**modalità kernel**) e quindi effettuare operazioni di I/O. Questo è quanto viene fatto ricorrendo alle System Call.

I/O a controllo di programma

E.g. chiamata syscall per la stampa di una stringa.

La periferica ha un ruolo passivo. Il processore esegue tutto il lavoro.

Svantaggio: La CPU dopo avere predisposto il controller all'esecuzione dell'I/O si ferma e si mette ad interrogare il registro di stato della periferica in attesa che il ready bit assuma un determinato valore. Stato *busy waiting* o *spin lock*.

```
begin
        1. Predisponi i registri del controller ad effettuare una
        operazione di lettura.
        2. While (ready-bit = = 0) do;          // spin lock (o busy waiting)
        3. Carica il dato acquisito;
end;
```

Esempio: Receiver (tastiera)

```
# NB i dispositivi vengono indirizzati tramite gli indirizzi "alti".

.text
.globl main
main:
        li $t0, 0x8000 0000 # indirizzo del receiver control register (2Gbyte)
        li $t2, 0x8000 0004  # indirizzo del receiver data register

# Ciclo di lettura di un carattere
ciclo:  lw $t1, 0($t0)              # Contenuto del registro di controllo
        beq $t1, $zero, ciclo       # if ($t1 != 0) esci

        lw $a0, 0($t2)              # Caricamento del dato in a0

        li $v0, 10                  # exit
        syscall
```


Polling

Interrogazione del registro di stato della periferica.

Ciclo di polling: durante un ciclo di busy-waiting su un dispositivo si esegue il **polling** sugli altri dispositivi di I/O.

Quando una periferica necessita di un qualche intervento, si soddisfa la richiesta e si prosegue il ciclo di polling sugli altri I/O.

```
// Leggi dato da perif_x
begin
a.   Predisponi i registri dei controller ad eseguire una read;
b.   if(ready_bit (perif_1) = = 1) servi perif_1;    #Esempio: Mouse
     if (ready_bit (perif_2) = = 1) servi perif_2;   #Esempio: Hard disk1
     if (ready_bit (perif_3) = = 1) servi perif_3;   #Esempio: Hard disk2

          ….
     if (ready_bit (perif_n) = = 1) servi perif_n;
     UpdateFunctions;                                #Programma di gestione in funzione della
                                                     #situazione delle periferiche (sistemi di controllo)
     goto b;
end;
```

Interrupt

E' la periferica a segnalare al processore (su una linea del bus dedicata) di avere bisogno di attenzione.

La segnalazione viene chiamata ***interrupt*** perché interrompe il normale funzionamento del processore (***interrupt request***).

Quando il processore "se ne accorge" (fase di fetch), riceve un segnale di ***interrupt acknowledge***.

Viene eseguita una procedura speciale, chiamata ***procedura di risposta all'interrupt***.

Problema: Il programma utente deve potere procedere dal punto in cui è stato interrotto ➔ *Salvataggio del contesto.*

DMA

Tra il momento in cui termina l'invio del comando al controller ed il momento in cui il dato è disponibile sul controller, la CPU può fare altro (tipicamente l'esecuzione di un altro programma).

Il meccanismo interrupt driven non svincola la CPU dal dovere eseguire le operazioni di trasferimento dati.

Per periferiche veloci, le operazioni di trasferimento dati occupano un tempo preponderante rispetto al tempo speso in spin lock.

Per evitare l'intervento della CPU nella fase di trasferimento dati, è stato introdotto il protocollo di trasferimento in Direct Memory Access (DMA).

Viene disaccoppiato il colloquio processore-Memoria dal colloquio IO-Memoria. Questo è reso più facile dalla struttura a bus gerarchici.

Il device controller che gestisce il trasferimento diventa **bus master**.

I passi della DMA

Il DMA controller è un processore specializzato nel trasferimento dati tra dispositivo di I/O e memoria centrale.

Per attivare il trasferimento viene richiesto alla CPU:
1. Spedire al DMA controller il tipo di operazione richiesta
2. Spedire al DMA controller l'indirizzo da cui iniziare a leggere/scrivere i dati.
3. Spedire al DMA controller il numero di byte riservati in memoria.

Per attivare il trasferimento al controller viene richiesta la corretta lettura dello stato della memoria e l'aggiornamento dell'indirizzo a cui trasferire il dato. *E' il controller che gestisce il trasferimento del singolo dato.*

Caratteristiche della DMA

La CPU si svincola completamente dall'esecuzione dell'operazione di I/O.

Il controller avvia l'operazione richiesta e trasferisce i dati da/verso memoria mentre la CPU sta facendo altro.

Dopo avere trasferito tutti i dati, il DMA invia un interrupt alla CPU per segnalare il completamento del trasferimento.

La CPU perciò controlla il (device) controller.

Sommario

I collegamenti tra CPU e gli altri componenti

Trasferimento tra I/O e CPU

Sommario

Valutazione delle prestazioni

Benchmark

Perché valutare le prestazioni?

- Misura/Valutazione quantitativa delle prestazioni (velocità….).
- Fare scelte intelligenti (e.g. installare nuovo hardware o nuovo sw).
- Orientarsi nell'acquisto di nuovo hw.
- Fatturazione delle prestazioni.

Le prestazioni migliorano perché:
- Incrementa la quantità di lavoro nell'unità di tempo (throughput)
- Diminuisce il tempo di esecuzione.

Domande:

Un processore più veloce cosa influenza?
Più processori dedicati, cosa modificano?

Misura del tempo di esecuzione

Tempo di esecuzione -> dipende dal mix di istruzioni

$$t_{medio} = \frac{\sum_{i=0}^{S} t_i l_i}{\sum_{i=0}^{S} l_i}$$

l_i numero di istruzioni di tipo i (=> frequenza)
t_i tempo di esecuzione dell'istruzione di tipo i

Numero di cicli di clock per istruzione (CPI) = $Cicli\_clock\_CPU_{programma}$ / Numero_istruzioni
➔ Tempo di esecuzione = CPI * T_{clock}.

Numero di istruzioni per secondo (MIPS) = (numero_istruzioni / 10^6) / tempo_esecuzione

Numero di operazioni in virgola mobile per secondo (MFLOPS) =
(numero_operazioniFP / 10^6) / tempo_esecuzione

MIPS = milioni di istruzioni per secondo

MIPS = (numero_istruzioni / 10^6) / tempo_esecuzione

Problemi:

· dipende dall'insieme di istruzioni, quindi è difficile confrontare computer con diversi insiemi di istruzioni;

• Il tempo totale di esecuzione dipende da diverse caratteristiche: dischi, sottosistema di I/O, sottosistema grafico …. Per questo motivo occorre menzionare la configurazione del sistema.

• varia a seconda del programma considerato;

• può variare in modo inversamente proporzionale alle prestazioni!

• valore di picco, scelgo il mix di istruzioni per massimizzare il MIPS misurato (fuorviante).

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!!

Problemi con MIPS di picco

Intel i860 (1989) dichiarava:
- 2 operazioni VM al secondo
- Clock di 50 Mhz

⇒ Prestazioni attese di 100 MFlops

MIPS R3000 (1989) dichiarava:
- 16 MFlops
- Clock a 33 Mhz

Su problemi reali l'i860 risultò 12% più lento del MIPS R3000

Intel i860 dichiarava i MFlops di picco, difficilmente raggiungibili e sostenibili.

I benchmark

MIPS / MFLOPS di picco poco significativi

Benchmarks = Programmi per valutare le prestazioni.

Benchmarks: Whetstone, 1976; Drystone, 1984.

Kernel benchmark. Loop Livermore, Linpack, 1980. Problema: polarizzazione del risultato.

Benchmark con programmi piccoli (10-100 linee, 1980). Problema: mal si adattano alle strutture gerarchiche di memoria.

Evaluating Architecture performances

Throughput, Response time, Execution time

Small programs can be incredibly fast (kernel benchmarks)

Description	Name	Instruction Count x 10⁹	CPI	Clock cycle time (seconds x 10⁹)	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2,118	0.75	0.4	637	9,770	15.3
Block-sorting compression	bzip2	2,389	0.85	0.4	817	9,650	11.8
GNU C compiler	gcc	1,050	1.72	0.4	724	8,050	11.1
Combinatorial optimization	mcf	336	10.00	0.4	1,345	9,120	6.8
Go game (AI)	go	1,658	1.09	0.4	721	10,490	14.6
Search gene sequence	hmmer	2,783	0.80	0.4	890	9,330	10.5
Chess game (AI)	sjeng	2,176	0.96	0.4	837	12,100	14.5
Quantum computer simulation	libquantum	1,623	1.61	0.4	1,047	20,720	19.8
Video compression	h264avc	3,102	0.80	0.4	993	22,130	22.3
Discrete event simulation library	omnetpp	587	2.94	0.4	690	6,250	9.1
Games/path finding	astar	1,082	1.79	0.4	773	7,020	9.1
XML parsing	xalancbmk	1,058	2.70	0.4	1,143	6,900	6.0
Geometric Mean							11.7

SPEC (System Performance Evaluation Cooperative)

Indici SPEC (‘89, ‘92, ’95, ’00, ‘06, ‘16)

http://www.spec.org/. **The Standard Performance Evaluation Corporation (SPEC)** is a non-profit corporation formed to establish, maintain and endorse a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers. SPEC develops benchmark suites and also reviews and publishes submitted results from our member organizations and other benchmark licensees.

Insieme di programmi test.
Condizioni diverse: singolo / multiplo processore / time sharing.
Benchmark specifici per valutare S.O. e I/O.

SPEC’95 -> SPECint, SPECfp, base Sun SPARCstation 10/40.

Benchmark particolari:
SDM (Systems Development Multitasking).
SFS (System-level File Server).
SPEChpc96. Elaborazioni scientifiche ad alto livello.

Orientamento: Benchmark specifici.

SPEC CPU200 CINT2000

Benchmark	Language	Category	Full Descriptions
164.gzip	C	Compression	HTML Text
175.vpr	C	FPGA Circuit Placement and Routing	HTML Text
176.gcc	C	C Programming Language Compiler	HTML Text
181.mcf	C	Combinatorial Optimization	HTML Text
186.crafty	C	Game Playing: Chess	HTML Text
197.parser	C	Word Processing	HTML Text
252.eon	C++	Computer Visualization	HTML Text
253.perlbmk	C	PERL Programming Language	HTML Text
254.gap	C	Group Theory, Interpreter	HTML Text
255.vortex	C	Object-oriented Database	HTML Text
256.bzip2	C	Compression	HTML Text
300.twolf	C	Place and Route Simulator	HTML Text

Parallel SPEC

Weak scaling: La dimensione dei dati e programma (working set) cresce con il numero di nodi di elaborazione.

Strong scaling: la dimensione del programma e dei dati è fissa e aumentano le prestazioni con l'aumentare del numero dei nodi di elaborazione.

Anche valutazione dei cloud:
SPEC Cloud™ IaaS 2016

Anche Java server benchmark:
SPECjbb2015

Benchmark	Scaling?	Reprogram?	Description
Linpack	Weak	Yes	Dense matrix linear algebra [Dongarra, 1979]
SPECrate	Weak	No	Independent job parallelism [Henning, 2007]
Stanford Parallel Applications for Shared Memory SPLASH 2 [Woo et al., 1995]	Strong (although offers two problem sizes)	No	Complex 1D FFT Blocked LU Decomposition Blocked Sparse Cholesky Factorization Integer Radix Sort Barnes-Hut Adaptive Fast Multipole Ocean Simulation Hierarchical Radiosity Ray Tracer Volume Renderer Water Simulation with Spatial Data Structure Water Simulation without Spatial Data Structure
NAS Parallel Benchmarks [Bailey et al., 1991]	Weak	Yes (C or Fortran only)	EP: embarrassingly parallel MG: simplified multigrid CG: unstructured grid for a conjugate gradient method FT: 3-D partial differential equation solution using FFTs IS: large integer sort
PARSEC Benchmark Suite [Bienia et al., 2008]	Weak	No	Blackscholes—Option pricing with Black-Scholes PDE Bodytrack—Body tracking of a person Canneal—Simulated cache-aware annealing to optimize routing Dedup—Next-generation compression with data deduplication Facesim—Simulates the motions of a human face Ferret—Content similarity search server Fluidanimate—Fluid dynamics for animation with SPH method Freqmine—Frequent itemset mining Streamcluster—Online clustering of an input stream Swaptions—Pricing of a portfolio of swaptions Vips—Image processing x264—H.264 video encoding
Berkeley Design Patterns [Asanovic et al., 2006]	Strong or Weak	Yes	Finite-State Machine Combinational Logic Graph Traversal Structured Grid Dense Matrix Sparse Matrix Spectral Methods (FFT) Dynamic Programming N-Body MapReduce Backtrack/Branch and Bound Graphical Model Inference Unstructured Grid

Sommario

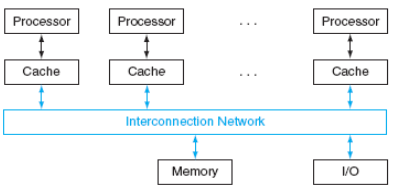
Valutazione delle prestazioni

Benchmark

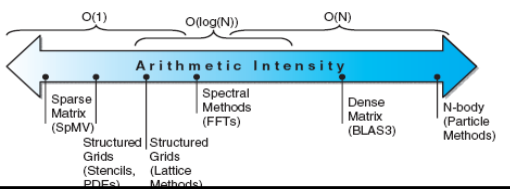
Arithmetic intensity

Velocità di calcolo FLOPS (floating point per second): velocità del singolo core: V_{core}.
In un'achitettura multi-core con P core la velocità di calcolo $V_{calc} = P * V_{core}$

Velocità di trasferimento dalla gerarchia di memoria. Per calcolarla dobbiamo capire quante operazioni devono essere fatte per ciascun byte caricato in cache, N_{op} / Byte (**Arithmetic Intensity**)



Se effettuiamo N_{op} >> 1 operazioni su ogni byte letto dalla memoria, avremo una velocità di calcolo massima pari a: $V_{max} = P * V_{core}$ = [Byte /s].



Weak scaling, less memory request per byte

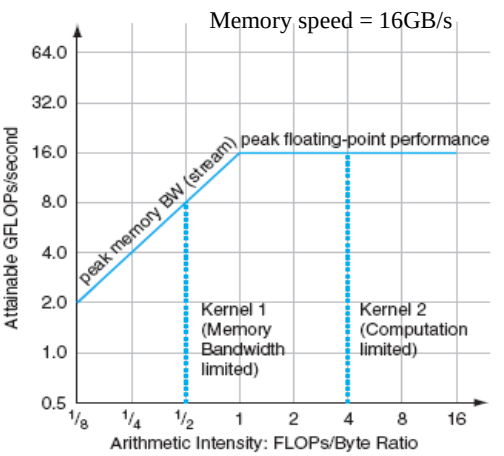
Il modello "roofline"

Nessun benchmark può essere contenuto interamente in cache.

2 elements:
- Computation
- Memory transfer

Per programmi con bassa intensità aritmetica (elevati accessi alla memoria per dato), il limite è offerto dal sistema di memoria.

Per programmi ad alta intensità, il limite è dato dalla capacità di elaborazione della CPU.



AMD Opteron X2

Il modello “roofline”: Computation

Se non ci fossero problemi con la memoria le prestazioni sarebbero una linea orizzontale pari alla massima capacità di calcolo:

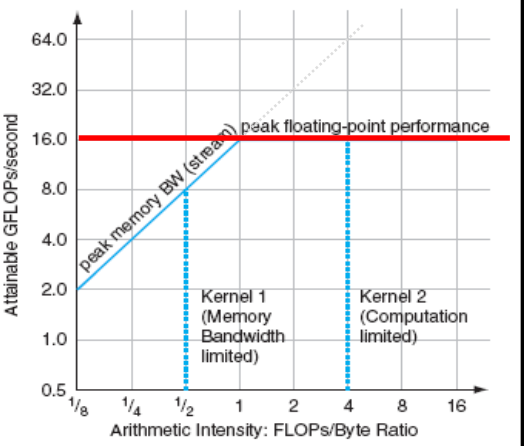
$V_{calc} = P * V_{core}$

V_{calc} = cost = 16 Gflops per Opteron X2

Tutto quello che viene letto dalla memoria può essere elaborato senza stalli.

Le prestazioni non dipendono dall’intensità aritmetica. Ma solo da P e V_{core}.

$V_{core} = N_{Cammini} * N_{Dati}$ / cammino * #issue / sec



AMD Opteron X2

Il modello “roofline”: Memory transfer

La memoria rifornisce la CPU.

I dati vengono letti dalla memoria con una certa velocità massima: V_{mem} = [Byte]/[s].

Tutti i dati letti dalla memoria vengono elaborati: per ogni byte effettuo un numero di operazioni FP FP definite dall’intensità aritmetica.

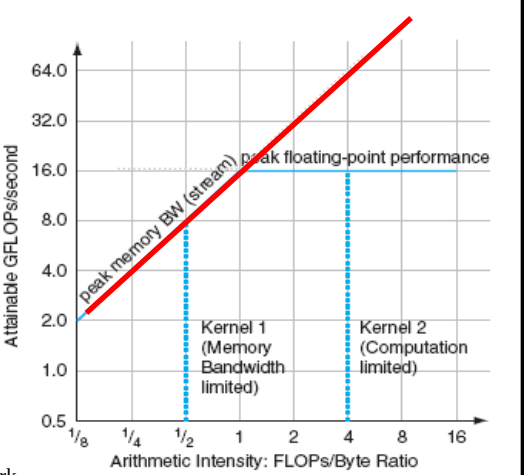
Maggiore è l’intensità, maggiore la velocità.

Come si inserisce nel grafico questo vincolo?

All’aumentare della velocità del sistema di memoria, la retta trasla verso sx.

Le prestazioni di memoria si valutano con un benchmark particolare: streaming benchmark.

Il sottosistema di memoria associato all’AMD Opteron X2 ha una velocità di trasferimento di picco di 16GFlop/s.



AMD Opteron X2

Il modello “roofline”: riassunto

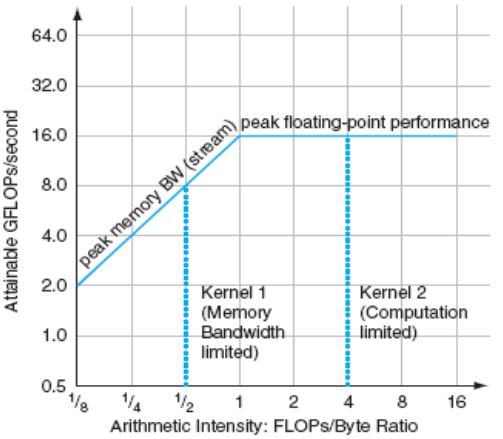
Nessun benchmark può essere contenuto interamente in cache.

AMD Opteron X2

2 elements:
- Computation
- Memory transfer

Per programmi con bassa intensità aritmetica (elevati accessi alla memoria per dato), il limite è offerto dal sistema di memoria.

Per programmi ad alta intensità, il limite è dato dalla capacità di elaborazione della CPU.

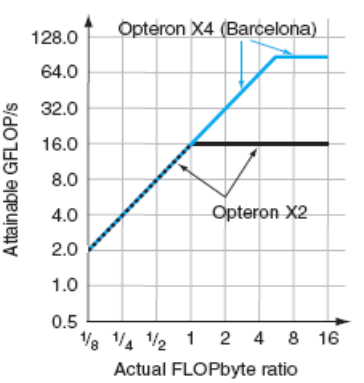


Attainable GFLOPs/sec (V_{calc})=
Min {Peak Memory BW × Arithmetic Intensity, Peak Floating-Point Performance}

Dall’Opteron X2 all’Opteron X4

Opteron X4 vs Opteron X2:

- **Stesso sistema di memoria**
- Numero doppio di processori (core)
- Numero quadruplo di operazionio in virgola mobile al secondo
  - Doppia capacità aritmetica della pipeline
  - Doppia capacità di fetch.



La velocità di elaborazione aumenta, ma solo per intensità aritmetiche superiori ad 1.

La velcocità di calcolo massima di 80 Gflops si raggiunge solo per un’intensità aritmetica pari a 5.

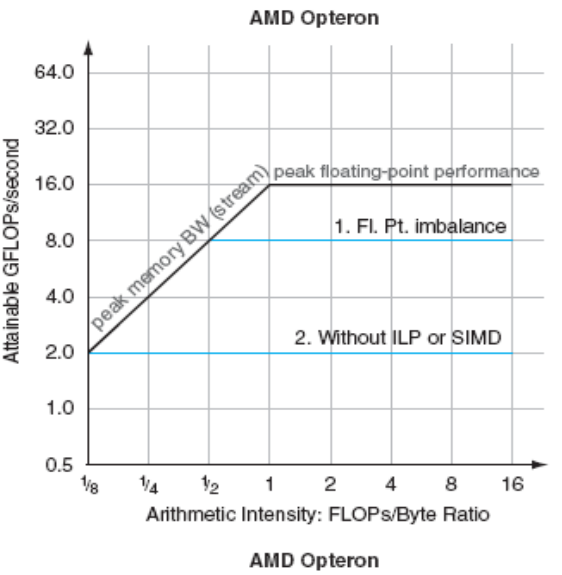
Ottimizzazioni sulla parte di calcolo - 1

Cosa succede se le prestazioni del vostro programma risultano scadenti rispetto alle prestazioni attese?

Riempire meglio le pipeline.

Migliorare il mix di operazioni:

- Significativa percentuale di operazioni floating point
- Bilanciamento tra Moltiplicazioni e addizioni



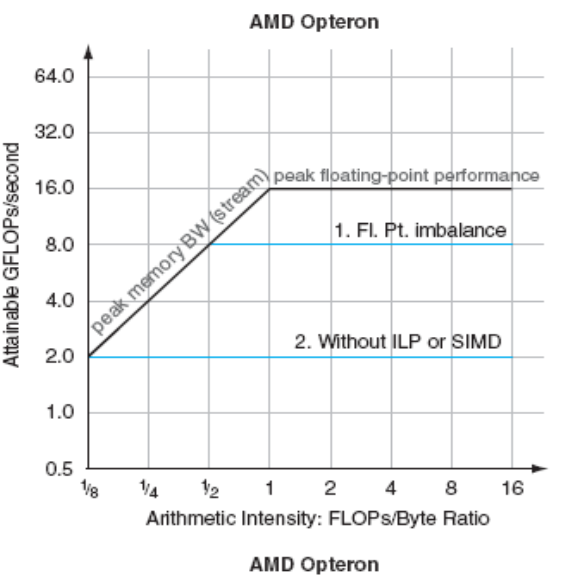
Ottimizzazioni sulla parte di calcolo - 2

Cosa succede se le prestazioni del vostro programma risultano scadenti rispetto alle prestazioni attese?

Riempire meglio le pipeline.

Aumentare la parallelizzazione dell'esecuzione:

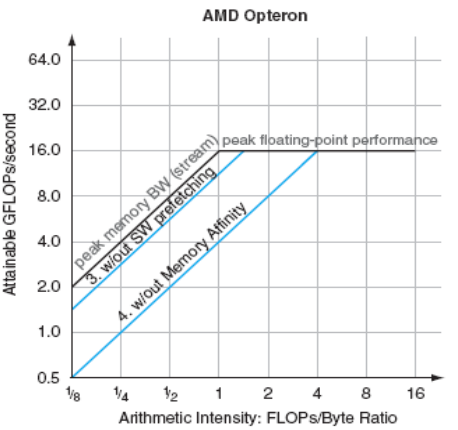
- Srotolamento dei cicli
- Ottimizzazione del mix delle istruzioni.



Ottimizzazioni sulla parte di memoria - 3

Cosa succede se le prestazioni del vostro programma risultano scadenti rispetto alle prestazioni attese?

Caricare meglio i dati in CPU



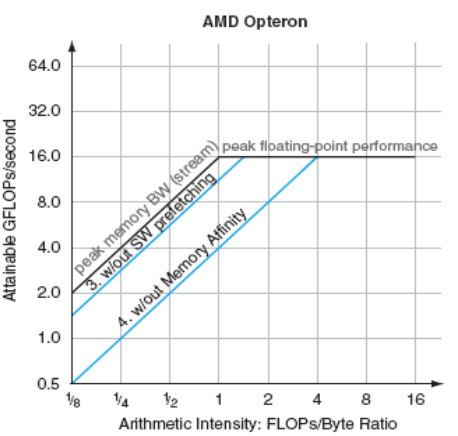
Software pre-fetching:

- Precaricamento dei dati in cache.
- Speculazione sui dati.

Ottimizzazioni sulla parte di memoria - 4

Cosa succede se le prestazioni del vostro programma risultano scadenti rispetto alle prestazioni attese?

Caricare meglio i dati in CPU



Affinità della memoria:

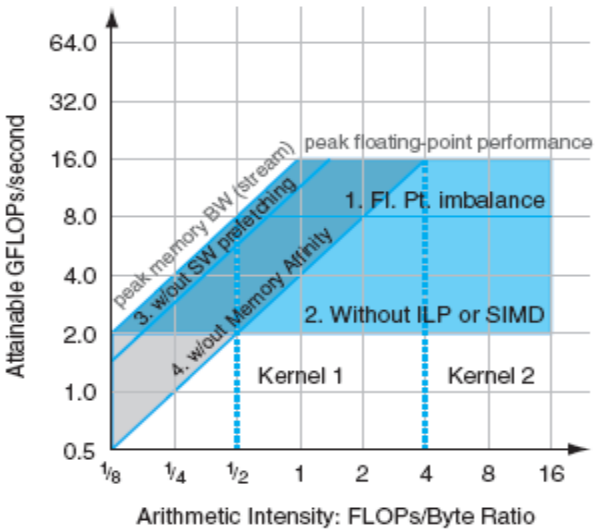
- Massimizzare gli hit.
- Separare il codice nei diversi core in modo che gli accessi in memoria siano all'interno della cache associato.
- Minimizzazione degli «invalidate».

Quali ottimizzazioni?

Occorre migliorare il codice perchè venga raggiunto il «tetto». Massima velocità di calcolo raggiungibile.

La distanza dal tetto indica quando si guadagna.

Ottimizzazioni da eseguire in sequenza.



Sommario

Valutazione delle prestazioni

Benchmark