Appunti di Architettura degli Elaboratori

Andreas Araya Osorio

November 7, 2021

1 Rappresentazione dei numeri

I numeri possono essere rappresentati in qualsiasi base, noi utilizziamo la base 10 come conseguenza del numero delle nostre dita.
I **calcolatori** utilizzano la base 2, ovvero il binario, che può facilmente essere ricondotta alla condizione di uno stato elettrico o **positivo (1)** o **negativo (0)**
Per la rappresentazione di un numero in qualsiasi base:

Definizione 1:
La sequenza di k cifre:

$$d_k \cdot d_{k-1} \cdot \dots \cdot d_1 \cdot d_0 \tag{1}$$

E questa sequenza la moltiplichiamo per la base scelta:

$$d_k \times b^k \cdot d_{k-1} \times b^{k-1} \cdot \dots \cdot d_1 \times b^1 \cdot d_0 \times b^0 \tag{2}$$

*dove **b** è la base da noi scelta.*

Definizione 2 (Numero di cifre in una base N)**:**
In una qualsiasi base N il numero di cifre equivale a:

$$cifre = N-1, N-2, \dots, 1, 0. \tag{3}$$

1.1 Binario

Con la base 2 abbiamo le cifre dallo 0 allo 1.
Per la definizione precedente la rappresentazione sarà:

$$d_k \times 2^k \cdot d_{k-1} \times 2^{k-1} \cdot \dots \cdot d_1 \times 2^1 \cdot d_0 \times 2^0 \tag{4}$$

Definizione 3 (Valore minimo e massimo)**:**
Il valore minmo e massimo di un numero di n cifre è:

- ***Valore minimo*** $= 000000\ldots00(n\ times) = 0$
- ***Valore massimo*** $=111111\ldots11(n\ times) = 2^n - 1$

$$2^{n-1} + 2^{n-2} + \cdots + 2^2 + 2^1 + 2^0 = 2^n - 1 \tag{5}$$

Esempio 1.

$$n = 3 \implies 111 = 2^2 + 2^1 + 2^0 = 7 = 2^3 - 1 = 8 - 1 \tag{6}$$

1.2 Notazione

Definizione 4 (BIT)**:**
Bit *= binary digit, uno dei due simboli (0, 1) del sistema numerico binario. Esso è l'unità elementare dell'informazione trattata da un elaboratore. Numeri di 8, 16, 32 bit equivale in base 10 a parlare di numeri a 3, 4, 5, ... cifre*

Definizione 5 (BYTE)**:**
Byte *= 8 bit, è una sequenza di bit, convenzionalmente l'unità di misura delle capacità di una memoria.*
Può assumere $2^8 = 256(0\ -\ 255)$ *possibili valori*

Definizione 6 (Parola)**:**
Word/Parola *= corrisponde a 16, 32 o 64 bit in base al tipo di IS (Instruction Set), essa è l'unità più piccola di informazione su cui un elaboratore può intervenire.*

Nome	Simbolo	Multiplo in base 10	Multiplo in base 2
chilobyte	kB	10^3	2^{10}
megabyte	MB	10^6	2^{20}
gigabyte	GB	10^9	2^{30}
terabyte	TB	10^{12}	2^{40}
petabyte	PB	10^{15}	2^{50}

Convenzionalmente si utlizzano come unità di misura:

- il B(yte) per la capacità di una memoria
- il b(it)/s per la velocità di trasmissione di dati.

1.3 Decimale

Nella rappresentazione decimale abbiamo le cifre dallo 0 al 9.
Un qualsiasi numero in base decimale si può rappresentare come:

$$d_k \times 10^k \cdot d_{k-1} \times 10^{k-1} \cdot \dots \cdot d_1 \times 10^1 \cdot d_0 \times 10^0 \tag{7}$$

Binario	Esadecimale
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

1.4 Ottale

Nella rappresentazione ottale si hanno le cifre dallo 0 al 7.
Questa base viene utilizzata perchè essendo un multiplo di 2 si possono facilmente convertire numeri ottali in binario:
Con 3 cifre si possono rappresentare 8 bit. In questo modo si possono avere numeri più facilmente maneggiabili da umani, rispetto a lunge stringhe di 0 o 1.

Binario	Ottale
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

1.5 Esadecimale

Nella rappresentazione esadecimale si hanno 15 cifre: dallo 0, ..., 9, A, ..., F.
Un simbolo (cifra) in questa base rappresenta 4 cifre binarie (4 bit).

Con 4 cifre esadecimali si possono rappresentare 16 bit.

Binario	Esadecimale
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

1.6 Algebra di Boole

Essa viene utillizzata per la specifica di funzioni logiche.
Una qualsiasi variabile può assumere 2 valori: **vero** o **falso**

Definizione 7 (Operazoini logiche di base)**:**

$$\begin{aligned} A\ AND\ B &= A \cdot B \\ A\ OR\ B &= A + B \\ NOT\ A &= \overline{A} \end{aligned} \tag{8}$$

A	B	$A\ AND\ B$	$A\ OR\ B$	$NOT\ A$
0	*0*	*0*	*0*	*1*
0	*1*	*0*	*1*	*1*
1	*0*	*0*	*1*	*0*
1	*1*	*1*	*1*	*0*

Si indica con: $\oplus = xor$, è l'or esclusivo, esso è vero solo quando una delle due variabili è vera, non quando lo sono entrambe.
$\overline{A \cdot B} = nand$, l'opposto dell'AND classico

Grazie a XOR e NAND si possono rappresentare tutte le altre funzioni logiche attraverso delle combinazioni di questi due.

A	B	$(\overline{A})$	$A \cdot B$	$A + B$	$\overline{A \cdot B}$	$\overline{A + B}$	$A \oplus B$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	1	0	1
1	1	0	1	1	0	0	0

2 APPROACH

There's a difference between
Computer architecture and **Computer organization**.

- C.Architecture = attributes of a system visible to a programmer.
  ISA = Instruction set architecture, is a synonym of C.A.
- C.Organization = operational units and their interconnections that realize the architectural specifications. Examples:
  - instruction set
  - number of bits used to represent various data types
  - I/O mechanisms and techniques for adressing memory

2.1 Structure and function

A modern computer is a hierarchical system, and is a set of interrelated subssytems. These have, in turn, have subsystems of their own until we reach the lowest level of subsystems. There's a huge difference between:

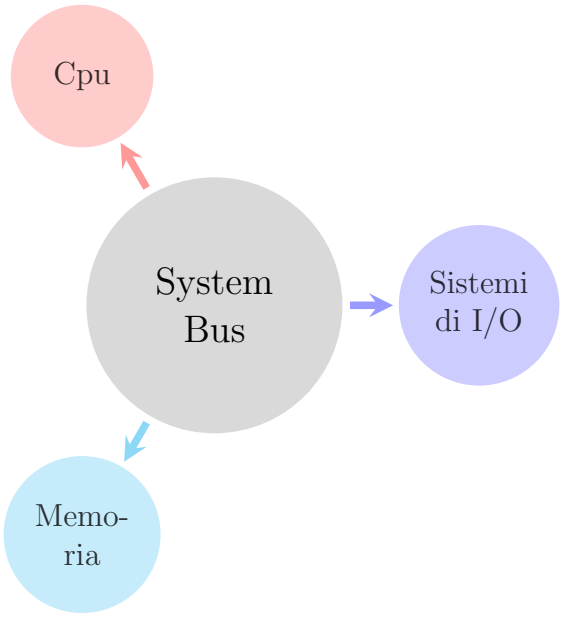
- **structure** the way components are interrelated
- **function** the operation of each individual

The way that is considered most efficient and the clearest approach for describing pc components is:**TOP-DOWN**

3 Componenti principali

Le componenti principali di un elaboratore sono:

- CPU
- Memoria
- Sistemi di I/O
- Interconnessioni



Definizione 8 (Architettura di Von Neumann)**:**
Secondo questo tipo di architettura, un elaboratore è composto di questi principali componenti:

- *dati e istruzioni in memoria*
- *memoria accessibile per indirizzo*
- *esecuzione sequenziale delle istruzioni*

Definizione 9 (Programma cablato)**:**
Conaiste nel costruire i componenti logici in modo tale che il risultato sia quello voluto e non può essere modificato in seguito.
Vuol dire "programmare" a livello hardware, ovvero con le componenti fisiche.
Non è un sistema flessibile, esegue solo operazioni predeterminate.

Definizione 10 (Programma)**:**
Un programma è una sequenza di passi
Ogni passo corrsiponde ad un'operazionie logica.
Ogni operazione determina un diverso insieme di segnali di controllo.

Definizione 11 (Programmazione software)**:**
Nasce con Von Neumann, si parte da un hardware generico, si ha una parte che preleva il codice di una istruzione, è generale: L'hardware di cui parliamo si dice ***general purpose****, utile a vari scopi. Si hanno poi dei segnali di controllo corrispondenti.*
Questo sistema è molto più flessibile di quello "cablato".
La CPU assume delle funzioni diverse ovvero:

- *interprete delle istruzioni*
- *generico modulo per operazioni aritmetico logiche = ALU*

I segnali di controllo sono necessari per far eseguire al giusto modulo la giusta operazione: ALU
ALU prende segnali di controllo ed esegue le istruzioni codificate
In questo sistema si ha la codifica delle istruzioni e la decodifica delle istruzioni.

Definizione 12 (Memoria principale nell'architettura di Von Neumann)**:**
Si ha la possibilità di salti oltre che all'esecuzione sequenziale(in serie)
Per esempio con le operazioni che richiedono accesso a più dati in memoria nello stesso momento.
Inoltre essa ha il compito di immaggazzinare dati e istruzioni

Esempio 2.
Somma con 2 numeri in locazione di memoria diverse

3.1 CPU

Essa non deve solo eseguire istruzioni ma anche gestire dei segnali di controllo e getstire delle risorse.
Composta da Vari componenti principali:

- btEU = execution unit = alu
- btIR = instruction register, registro che contiene l'istruziuone da eseguire successivamente a quella nel PC.

- **PC** = program counter, puntatore all'istruzione indirizzo dell'istruzione da eseguire presente nella memoria.
- **MAR** = memory address register, registro di interfaccia con il bus di sistema, contiene solo registri
- **MBR** = memory buffer register, contiene solo dati.
  Il MAR e il MBR mantengono le infromazioni fino a che non è disponibile il bus di sistema per essere impiegato.
  - in caso di lettura raccolgono il dato dal bus.
  - in caso di scrittura contengono il dato.
- **I/O AR** indirizzo periferica con cui scambiare dati, specificare periferica
- **I/O BR** raccolta dati

Quando si ha un salto nell'esecuzione delle istruzioni incrementa l'indirizzo del PC
Inoltre è presente un buffer nel modulo I/O: è una memoria interna al sistema di input output, esso serve perchè la CPU invia dati troppo velocemente rispetto ed esso non può riceverli alla stessa velocità, per questo il buffer dell'I/O, mantiene in memoria i dati inviati dalla più veloce CPU

3.2 Ciclo CPU esecuzione

- **Fetch**: reperimento, prelievo dell'istruzione dalla memoria
- **Execute**: esecuzione dell'istruzione prelevata dalla memoria

Il registro **PC** contiene l'indirizzo di memoria della cella di Memoria contenente l'istruzione da eseguire. Quando si ha un prelievo di istruzioni dalla memoria, si ha un incremento del PC.
L'istruzione prelevata viene messa in IR poi viene eseguita.

3.3 Tipi di Operazioni

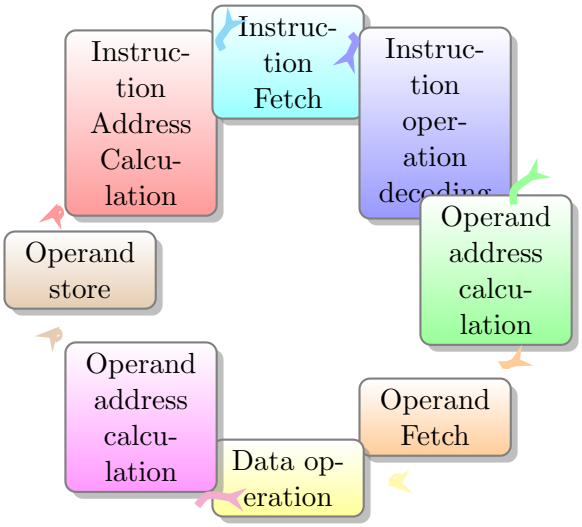
1. Processore-memoria: trasferimento dati dalla CPU alla Memoria R/W
2. Processore-I/O: trasferimento dati da CPU a I/O R/W
3. Elaborazione dati: operazioni logiche e aritmetiche sui dati operazioni della ALU
4. Controllo: può alterare la sequenza delle istruzioni, per esempio il salto

Esempio 3.
Parola = 16bit
Istruzione = 16bit
Codici operativi = 4 bit a sinistra = 2^4 combinazioni = 16

- 0001 carica in AC (accumulatore) una cella di M
- 0010 scrive in M il contenuto di AC
- 0101 somma una cella di M ad AC

3.4 Ciclo di esecuzione

1. Instruction Address Calculation
2. Instruction Fetch
3. Instruction operation decoding
4. Operand address calculation
5. Operand Fetch
6. Data operation
7. Operand address calculation
8. Operand Store



Per l'esecuzione di una singola operazione, bisogna prima eseguire una serie di altri sottocompiti, che devono essere ben eseguiti.
Per questo è essenziale **l'unità di controllo**, per accertarsi che ogni operazione venga eseguita correttamente e nel giusto ordine.
La CPU può eseguire **più operazioni** momentaneamente, mantenendo ogni sua parte attiva , breve introduzione al concetto di pipeline

3.5 Interruzioni

Il meccansimo tramite il quale dei moduli possono interrompere la normale di sequenza di esecuzione.
Il ciclo di esecuzione con interruzioni è differente: al **termine** di un ciclo si ha il controllo delle interruzioni,

- se ce ne sono, esse vengono risolte
- se non ce ne sono, il ciclo ricomincia

Tipi di interruzioni:

- Program, overflow, divisione per zero
- Timer, da un timer inerno alla CPU
- I/O, termine di un'operazione di I/O
- Guasto Hardware

Si interrompe per

- efficienza elaborazione

Ciclo interruzione:

- viene aggiunto al ciclo di esecuzione
- la cpu controlla (fetch) le interruzioni pendenti
- se non ce ne sono, prende la prossima istruzione
- se ce ne sono:
  - sospende esecuzione
  - salva contesto
  - imposta il pc all'indirizzo di inizio del programma di gestione
  - esegue il programma di gestione dell'hardware
  - rimette il contesto al suo posto e continua il programma interrotto

Lunghezza di attesa:

- **breve attesa**, tempo di operazione di I/O minore del tempo tra due istruzione WRITE
- **lunga attesa**,

3.5.1 Interruzioni multiple

In caso di **interruzioni multiple**: esistono vari livelli di interruzione e differenti tipi di politica di gestione.
Esistono interruzioni di alto livello e basso livello, a seconda dell'importanza che hanno per il funzionamento del sistema.
Politiche di interruzione:

- Disabilitare le interruzioni:
  - La CPU **ignora** le altre interruzioni e gestisce la prima e "maschera" le altre
  - Le interruzioni rimangono **pendenti**
  - vengono gestite nell'**ordine** in cui arrivano
- Definire le priorità
  - Interruzioni di bassa priorità vengono interrotte quando si presentano quelle di alta priorità

- Quando è stata gestita la priorità di alto livello viene maneggiata quella di basso livello
- Si hanno delle interruzioni **annidate**

3.6 Connessioni

Tutti i componenti **devono** essere connessi
Esistono vari tipi di connessioni per vari tipi di componenti, il bus collega:

- CPU
- Memoria
- I/O

Composto da 50 a qualche centinaio di linee

3.7 Bus

Tutti i dispostivi sono collegati dal bus di sistema
Il bus:

1. collega **2 o più** dispositivi
2. mezzo trasmissione condiviso
3. un segnale trasmesso ad un bus è disponibile a tutti i dispositivi
4. arbitro bus: solo un dispostivo alla volta può trasmettere
5. varie linee di comunicazione (trasmettono uno 0 o un 1)
6. varie linee trasmettono in parallelo numeri binari. Un bus da 8 bit trasmette un dato di 8 bit

Solitamente l'ampiezza del bus corrisponde ad un multiplo della parola.

3.7.1 Bus di sistema:

- connette cpu, i/o, M
- da 50 a qualche centinaio di linee
- 3 gruppi di linee
    1. bus dati
    2. indirizzi
    3. controllo

3.7.2 Bus dati:

- trasporta dati o istruzioni
- ampiezza –> efficienza del sistema
  - con poche linee –> maggiori accessi in memoria

3.7.3 Bus indirizzi

- indica sorgente o destinazione dati
- l'ampiezza determina la massima quantità di M indirizzabile

Deve essere di almeno 1 parola.

Esempio 4.
Architettura a 64 bit = $2^{64} - 1$ indirizzi
Il bus deve essere di almeno 64 bit.

3.7.4 Bus controllo

Per controllare accesso e uso di:

- **linee dati**
- **indirizzi**
  1. M write
  2. M read
  3. richiesta bus
  4. bus grant
  5. interrupt request
  6. clock

3.7.5 Uso del bus

Con modulo intendiamo una generica componente del computer.
se un modulo vuole inviare dati ad un altro:

- bus grant
- data transfer

se un module vuole ricevere dati da un altro:

- bus grant
- trasferire una richiesta all'altro modulo sulle linee di controllo
- attendere invio dati

3.7.6 Bus singoli e multipli

- singolo bus = ritardo e congestione
- vari bus = risoluzione problema

Esempio di Bus multiplo:

- **connessione punto a punto** fra CPU e cache, il che rende il trasferimento dei dati molto più veloce ed efficiente.

- **Bus di espansione**, si interfaccia con i dispositivi I/O
- **Bus ad alta velocità**, si interfaccia con i dispositivi I/O e fornisce una trasmissione di dati più rapida
- **Bus di sistema**, fra cache e Memoria principale.

3.8 Temporizzazione

Coordinazione degli eventi su un bus

- Asincrona
  - più complessa da implementare
- Sincrona
  - clock determined events
  - single clock line, with an alternate sequence of 0 and 1, with equal length
  - single 1-0 sequence is a clock cicle
  - every device connected to the bus can read the clock line
  - every event starts at the beginning of a clock cycle

3.8.1 QPI

Interconnessione Punto a Punto, con l'aumentare della velocità dei processori è sempre più frequenta una distribuzione di dati densa e veloce.
Un bus condiviso però rallenta la comunicazione del processore (bottleneck) , per questo si adottano delle connessioni punto a punto.

- Connessioni dirette multiple:
  più componenti del sistema godono di connessioni dirette a coppie con altri componenti.
- Architettura di protocollo a strati
- Trassferimento Dati a pacchetto:
  i dati non sono inviati come flussi di dati non elaborato ma come una sequenza di pacchetti, essi includono intestazioni di controllo e codici di correzione dell'errore.

Livelli di QPI:

- **fisico**
  la parte hardware, l'unità di trasferimento è di 20 bit (Phit) , tutto è sincronizzato da un clock.
- **Link**
  trasmissione affidabile del controllo del flusso, l'unità di trasferimento è di 80 bit (Flit).
  Protocollo con pacchetti da 72 (dati) + 8 (codice di correzione errore) bit.
  - Controllo del flusso:
    il mittente non può inviare più dati di quelli che il destinatario può ricevere.
  - Controllo dell'erroreç 8 bit per rilevare errori di trasmissione sugli altri 72 bit.
    In caso di errore il mittente deve re-inviare il pacchetto errato.
- **Routing**
  Struttura per dirigere i pacchetti attraverso essa. Determina il percorso che un pacchetto deve seguire.
  Supportato da tabelle di instradamento:

  - definite da software di basso livello che contiene delle istruzioni
  - descrizione percorso che un pacchetto può seguire
  - utile in sistemi di maggiori dimensione

- **Protocollo**
  Insieme di regole ad alto livello per lo scambio di pacchetti. Un pacchetto è un numero intero di Flits. Il contenuto di un pacchetto è flessibile, per gestire esigenze diverse.
  Supporta il protocollo di coerenza della cache, per garantire coerenza fra i contenuti della cache dei core e la memoria principale

3.9 Memoria

Caratteristiche principali della memoria:

- Locazione: processore (cache), interna (principale RAM), esterna (secondaria)
- Capacità: dimensione parola (dipende dall 'architettura), numero di parole
- Unità di trasferimento: parola, fra cache e RAM in blocco (insieme contiguo di parole)
- Metodo di accesso:
  - sequenziale: accedere prima a tutte le informazioni che vengono prima: lento. Utilizzato nei nastri magnetici, backup sistemi.
  - diretto, organizzata in gruppi, accesso sequenziale ai gruppi. accesso diretto ad un insieme di informazioni. HDD
  - casuale, accesso diretto a quella locazione di memoria, non importa dove si trova la locazione di memoria dell'informazione. RAM
  - associativo, informazione non indviduata da indirizzo, ma da una parte dell'informazione. CACHE
- Prestazioni: tempo di accesso, ciclo e velocità di trasferimento
- Modello fisico:
  - semiconduttore (ROM)
  - magnetico, campi magnetici (HDD)

  - ottico, laser ottico (CD)
  - magnetico-ottico
- Caratteristiche fisiche:
  - volatile (cache, quando termina il flusso di corrente elettrica i dati vengono cancellati)
  - non volatile (hdd, i dati vegnono memorizzati permanentemente quando si spegne il computer)
  - riscrivibile (RAM, HDD, cache)
  - non riscrivibile (ROM, read only memory).
- Organizzazione, se suddivisa in un singolo chip oppure vari, più o meno moduli.

Generalmente all'aumentare del costo della memoria aumenta la sua velocità ma diminuisce la sua ampiezza.
In ordine decrescente per velocità, costo e crescente per ampiezza:

- Registro (1 parola) 16 - 64 bit
- Cache (1 indirizzo)
- SRAM
- DRAM (8gb - 64gb)
- SSD (1TB - 8TB)
- HDD (16TB-24TB)
- CD - DVD-ROM
- Nastro (vari PB)

L'aumento di prestazioni delle CPU si deve a migliorie tecnologiche e architetturali, mentre quello delle memorie **solo** ad avanzamenti tecnologici.
Proprieta' dei programmi:

- Statiche, dal file sorgente
- Dinamiche, dall'esecuzione
  - Linearita' dei riferimenti, spesso consecutivi

- Localita' dei riferimenti, indirizzi contigui sono piu' probabili

Definizione 13 (Congettura 90/10)**:**
Un programma impiega di solito il 90% del suo tempo di esecuzione alle prese con un numero di istruzioni pari a circa il 10% di tutte quelle che le compongono.

3.9.1 GERARCHIA DI MEMORIA

Tutte le locazioni di memoria sono suddivise in blocchi.

Conviene organizzare la memoria in vari livelli gerarchici:

- Cache la più veloce e suddivisa in diversi livelli
  - L1 cache:
    molto veloce e molto costosa, per i dati ad accesso probabile
    es. 10% di cui parlavamo prima
  - L2 cache, più capiente ma meno veloce della L1.
  - L3 cache, più capiente ma meno veloce della L2
- Ram, più lenta della cache ma più capiente ed economica

La memoria Ram è composta da:

1. indirizzo di memoria
2. blocco di memoria

Memoria a livelli:

1. Livello inferiore, supporti con capacità più alti, piu' lenti, meno costosi
2. A livelli piu' alti diventano progressivamente piu' veloci, piu' costosi e meno capienti.

La CPU usa il livello piu' alto. Ogni livello inferiore contiene tutti i dati presenti ai livelli superiori.

3.9.2 SCAMBIO DATI

I dati vengono scambiati sotto forma di WORD/PAROLE:

- CPU - Cache: scambio di PAROLE
- CACHE - MEMORIA P. scambio di BLOCCHI (multiplo di PAROLE)

La memoria lavora efficientemente se la CPU trova il dato cercato nella CACHE. Avendo piu' livelli di cache si ha una maggiore probabilità di trovare questo dato.

Il numero di parole in un blocco è una potenza di 2.
Una parola è composta da 4 byte, possiamo identificare i primi 14 bit come "indirizo" del bit, mentre i restanti 2 come identificativi del bit.

3.9.3 Gerarchia di memoria

Un indirizzo di linea di cache e' costituito generalmente da:

1. Etichetta (tag), indirizzo del blocco nella memoria principale.
2. Blocco di K parole

Un blocco di memoria richiesto dalla CPU può essere presente **hit** o non presente **miss** in memoria. (generalmente è presente).
Per guadagnare efficienza prestazionale un **hit** deve essere molto probabile ($> 90\%$), se fosse minore di questa percentuale non avrebbe senso utilizzare quella struttura di memoria.
Un **miss** avvia una procedura di scambio di dati con un livello inferiore.
Una linea di cache può memorizzare diversi blocchi diversi, si usano i bit piu' a destra per identificare la parola all'interno della linea e i bit piu' a sinistra qual'e' la linea di cache per identificare il blocco.

Definizione 14 (Organizzazione)**:**
La memoria principale e' suddivisa in blocchi logici
La cache e' suddivisa in un multiplo di blocchi.

Definizione 15 (Tempo medio di Accesso)**:**
T_a: Tempo medio di accesso ad un dato in memoria cache

$$T_a \ = \ T_h \times P_h + T_m(1 - P_h) \tag{9}$$

*T_h: tempo di accesso ad un dato presnte in cache T_m: tempo medio di accesso ad un dato **non** in cache (dimensione blocco) P_h: probabilità di hit*

Tecnica generale

1. Suddivisione della memoria centrale in blocchi logici
2. dimensionamento della cache in multiplo di blocchi
3. ogni indirizzo emesso dalla cpu
    - hit $\iff$ il dato viene fornito immediatamente alla cpu, (sotto forma di parola)
    - miss, il dato viene fornito sotto forma di blocco
        (a) la cache richiede il dato al livello inferiore (memoria principale RAM)
        (b) viene posto in cache
        (c) viene fornito alla cpu

Definizione 16 (**associazione diretta / direct mapping**)**:**
Ogni blocco del livello ineriore può essere allocato solo in una specifica posizione ***linea/slot*** *del livello superiore*

1. ***ILS*** *= indirizzo di livello superiore*
2. ***ILI*** *= indirizzo di livello inferiore*
3. $ILS = ILI \bmod N$*, divisione con resto intero*

1. *vantaggi*
    - *semplicità traduzione indirizzo ILI a ILS*
    - *determinazione velocità hit o miss*
2. *svantaggi*
    - *necessità di contraddistinguere blocco in ILS*
    - *swap frequenti per accesso a dati di blocchi adiacenti,*
      il primo blocco di ogni insieme avrà la stessa linea di cache, quindi solo uno di questi puo' essere in cache

Definizione 17 (**associazione completa / fully associative**)**:**
Ogni blocco del livello inferiore può essere posto in qualunque posizione del livello superiore.
Ad una cache di N blocchi viene associata una tabella di N posizioni contenenti il numero di blocco effettivo (tag)

- *vantaggi: massima efficienza di allocazione*
- *molto tempo per la corrispondenza ILS-ILI e della verifica hit/miss*
- *molto costoso dal punto di vista hardware e scarsa possibilità di hit*
- *difficile identificazione di hit o miss*

Definizione 18 (**associazinone a N-gruppi / N-way set associative**)**:** *Ogni blocco di un certo insieme di blocchi del livello inferiore può essere allocato liberamente in uno specifico gruppo di blocchi del livello superiore*

Esempio 5.
Per una cache di 32 linee con un N equivalente a 2, ogni gruppo avrà 16 linee.
Ci sono 16 gruppi da 2 linee e per ogni coppia avremo un blocco di una determinata posizione di qualunque insieme.

Questo tipo di associazione è una via di mezzo fra gli altri due tipi. La cache composta da R gruppi di N posizioni di blocco, si affiancano R tabelle di N elementi contenenti i tag.
Per ottenere facilmente il numero di linee basta semplicemente fare la moltiplicazione

$$Numero\ di\ linee\ della\ cache = N \times R \tag{10}$$

Ha una buona efficienza di allocazione, nonstante abbia una certa complessità, e prende i due punti di forza degli altri due tipi:

- *uso efficiente delle linee di cache*
- *facile determinazione di hit o miss*

Non e' necessario avere un grande numero di N per raggiungere il massimo dell'efficienza1

Definizione 19 (Politiche di rimpiazzo dei blocchi)**:**
Quando si ha un miss, come si decide quale blocco della cache dobbiamo rimpiazzare?
Nell'associazione diretta non ci si pone questo problema, perchè ogni linea della cache corrisponde un blocco della memoria centrale.

1. *casuale, viene occupato lo spazio omogeneamente, facile implementazione*

2. *First-In-First-Out(FIFO), il blocco rimasto più a lungo in cache, complicata implementazione*

3. *Least Frequently Used(LFU), il blocco con meno accessi, complicata implementazione hardware*

4. *Least Recently Used(LRU), il blocco con l'accesso più distante, per preservare quelli accessi più recentemente, implementazione difficile.*

A minor quantità di cache si hanno migliori prestazioni con il rimpiazzo LRU.
Questo e' il metodo piu' gettonato, e ad aumentare il livello di cache è sempre meno significativo il miglioramento offerto da queste tecnologie.

La scrittura dati determina incoerenza tra il blocco in cache e quello nei livelli inferiori

Definizione 20:
write through:

1. *scrittura contemporanea in cache e livello inferiore*

2. *aumento traffico per frequenti scritture nel medesimo blocco, dati coerenti fra blocchi*

3. *si ricorre a buffer asincroni verso la memoria, a causa di momenti di congestione del bus.*

La memoria contiene **istruzioni** e **dati** e solo il 50% delle operazioni sui dati sono scritture.

Definizione 21:
write back:

1. *scrittura in memoria inferiore differita al rimpiazzo del blocco di cache corrisp.*

2. *ridotto numero di modifiche nella memoria principale*

3. *occore ricordare operazioni di scrittura nel blocco*

4. *ottimizzazione del traffico tra livelli, riduzione congestione bus*

5. *periodi di incoerenza*

Occore ricordare che tra memoria centrale (RAM) e cache si passano **BLOCCHI** e non **PAROLE**.

Esempio 6 (scenario problematico).
- più dispositivi connessi allo stesso bus con cache locale
- memoria centrale condivisa

Nessun tipo di "write" (through, back) può assicurare coerenza.

Possibili soluzioni

- **monitoraggio del bus con write through**, controllori intercettano modifiche locazioni condivise
- **trasparenza hardware**, hardware aggiuntivo: modifica a RAM = modifica a cache
- **memoria non cacheable**, solo una porzione è condivisa e non cacheable

Cosa comporta la modifica di dati in una cache?

- invalida quella parola corrispondente nella memoria centrale
- invalida la parola nelle altre cache che la contengono
  per esempio in un sistema con CPU multi-core

Estendiamo il discorso alla memoria swap e RAM:
Quello che cambia e' che al livello cache-RAM e' tutto basato su un livello logico, mentre nell'altro caso si parla di un livello fisico fra RAM e memoria swap dei dispositivi di archiviazione eseterna.
Le problematiche che si incontreranno saranno le stesse: capienza RAM piena, politiche di rimpiazzo, modalita' di scrittura.
I blocchi diventano pagine di un programma.

3.9.4 Cache logica e fisica

Possiamo avere due tipi di Cache:

- Logica, riceve un indirizzo logico ma ad ogni cambiamento di contesto deve essere svuotata, conveniente se si prevede un utilizzo in un contesto con poche interruzioni

- Fisica, riceve un indirizzo fisico deve attendere la traduzione dell'indirizzo da logico a fisico, non deve essere svuotata ad ogni cambiamento di contesto, conveniente in un contesto con molte interruzioni

La cache deve essere svuotata ad ogni cambiamento di contesto, altrimenti non puo' funzionare bene.

3.9.5 Il problema dei miss

Esistono vari tipi di miss:

- di primo accesso, inevitabile non riducibile
- per capacità insufficiente, quando la cache non può contenere altri blocchi
- per conflitto, dipende dal tipo di associazione, quando vari blocchi possono corrispondere allo stesso gruppo

Soluzioni classiche:

- Maggior dimensione di blocco, aumento di miss, aumento linee utilizzo piu' efficiente dello spazio
- Maggiore associativita', incremento del tempo di localizzazione di un gruppo, soggetto alla regoal 2:1

Altre soluzioni:

- multilivello cache, fino a 3 livelli
- separazione cache dati / cache istruzioni
- ottimizzazione degli accessi mediante compilatori (C)

3.9.6 tipi di memorie a semiconduttore

Tipo	Categoria	Cancellamento	Mecc. scrittura	Volatile
RAM	read-write	elettric. byte-level	elettric.	si
ROM	read-only	non possibile	maschere	no
PROM	read-only	non possibile	elettric.	no
EPROM	read-mostly	luce UV, chip-level	elettric.	no
EEPROM	read-mostly	elettric. byte-level	elettric.	no
Memoria Flash	read-mostly	elettric. block-level	elettric.	no

Si dice di una memoria che e' volatile o no quando: se si toglie l'alimentazione si cancella l'intero contenuto della memoria.

Definizione 22 (DRAM)**:**
Dynamic RAM =

- *bit memorizzati in condensatori*
- *decadimento cariche con tempo*
- *refresh cariche, durante alimentazione*
- *semplice costruzione*
- *1 condensatore = 1 bit*
- *meno costose*
- *circuito per refresh*
- *meno veloci della SRAM*
- *usate nella RAM*
- *operazione analogica, la carica determina il valore (0 o 1)*

Definizione 23 (SRAM)**:**
Static RAM =

- *bit memorizzati tramite porte logiche*

- *non perde la carica*
- *no refresh*
- *piu' complesso, piu' elementi per bit (6 transistor)*
- *piu' costosa*
- *no refresh*
- *piu' veloce, utilizzata nella cache*
- *digitale*

3.9.7 ROM

Una ROM:

- memorizzazione permanente (non volatile)
- memorizzano:
  - microprogrammi
  - subroutine di libreria
  - programmi di sistema
  - funzioni tabulate (logaritmi, esponenziali, etc)

3.9.8 Codice correzione errore

Tipi di errori:

- Guasto hardware, non risolvibile
- errore software, possono accadere casualmente, a causa del decadimento della materia, i danni non sono permanenti

Quando un errore viene rilevato, puo' essere corretto attraverso i codici di Hamming (quelli che vedremo)
Quando un dato viene creato assieme ad esso si crea una sua 'impronta digitale' che verra' utilizzata per comparare la validita' del dato.