

INFO SUL CORSO:

PROF: RICCARDO TOLONE

CFU: 6

MAIL: riccardo.tolone@uminoma3.it / tolone@dia.uminoma3.it

RICEVIMENTO: VENERDÌ MATTINA

OBIETTIVI: CONOSCERE LA STRUTTURA DELL'HARDWARE NEI COMPUTER E NON + ASSEMBLER

LIBRO: "ARCHITETTURA DEI CALCOLATORI: UN APPROCCIO STRUTTURALE" 6^a edizione**STRUTTURA ESAME:**

• MODALITÀ CLASSICA:

- Esercizi al computer (2 h)
- domande sul programma (1 h)

• MODALITÀ CON "ESONERI":

- 3/6 HOMEWORK CON AUTOVALUTAZIONE
- 26 APRILE SCRITTO CON VALUTAZIONE
- 3/6 HOMEWORK CON AUTOVALUTAZIONE
- LUGLIO SCRITTO CON VALUTAZIONE

GLI HOMEWORK
SONO OBBLIGATORI

STORIA DELLE ARCHITETTURE

L'evoluzione dei calcolatori parte dal 1836 ai primi anni 2000.

GENERAZIONE 0 (1600-1965)

Nascono le prime calcolatrici ad opera di Pascal e Leibniz anche se il padre dei calcolatori viene considerato Babbage che realizza il progetto di una calcolatrice programmabile.

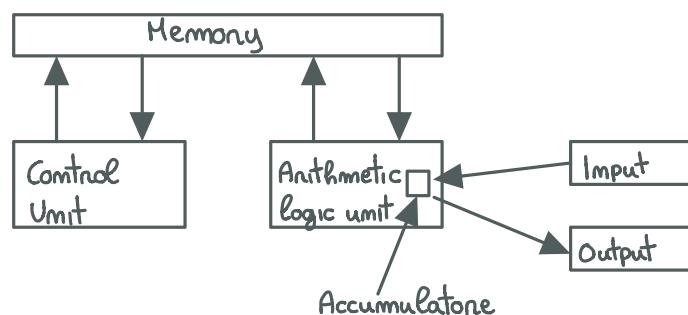
Dal 1930 si sono investiti molti fondi per la ricerca di calcolatori a scopo bellico, tra cui il primo calcolatore che operava coi numeri binari.

GENERAZIONE 1 (1965-1955)

Dal 1950 al 1955 si sono sviluppati i calcolatori colossus realizzato da Turing per decodificare Enigma ed il progetto Eimac (a scopo più generale) realizzati con il sistema delle valvole.

Negli anni '50 creiamo la società UNIVAC per commercializzare soluzioni come ENIAC.

Sempre negli anni '50 Neumann teorizza il modello architettonale di un calcolatore, così strutturato:



Nel 1953 nasce l'IBM che diventa un colosso fino agli anni '80

GENERAZIONE 2 (1955-1965)

Successivamente nascono i primi MINICOMPUTER, con architettura basata su Von Neumann, con architettura pesantemente incentrata su I/O.

Nascono anche i sistemi commerciali come i MAINFRAME, basati sui TRANSISTOR, e versioni più piccole per aziende minori.

GENERAZIONE 3 (1965-1980)

Qui si evolvono sia dal punto di vista HARDWARE (micropogrammazione, unità veloci floating point, processori ausiliari per gestire I/O), che dal punto di vista SOFTWARE (virtualizzazione delle risorse, programmazione concurrentiale, memoria virtuale).

Tra questi spiccano la serie IBM System/360, la serie DEC PDP-11 e UNIX

GENERAZIONE 4 (PC) 1986 - 2000

È un discendente del minicomputer, prodotti dall'IBM, si passa dai grandi centri di elaborazione a un contesto di Informatica Distribuita.

GENERAZIONE 5 (COMPUTER INVISIBILI)

La Apple introduce il primo computer palmare.

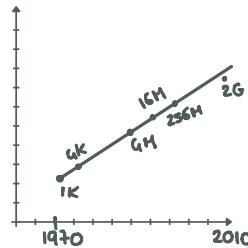
Successivamente si sono diffusi i computer EMBEDDED (integrati in altri dispositivi)

Il modello che si segue è quello "OBVIOUS" COMPUTING o dell' INTERNET OF THINGS

LEZIONE 2

05/03/21

LA LEGGE DI MOORE (1965)



"Il numero di transistors su di un chip raddoppia ogni 18 mesi"

Più transistor in una CPU:

- Eseguire direttamente istruzioni più complesse
- Maggiore memoria sul chip (cache)

Im fruturo l'andamento della legge di Moore, seguito da una nuova filosofia nei calcolatori (es. Quantum Computing)

LEGGE DI NATHAN

"Il software è un gas: riempie sempre completamente qualsiasi contenitore in cui lo si metta."

TIPI DI COMPUTER

- DISPOSABLE COMPUTER (Biglietti d'auguri)
- MICROCONTROLLORI (Ondoli, macchine)
- Mobile and game computer (Console e smartphone)
- Personal computer (Desktop o notebook)
- Server (Network server)
- Mainframe (Batch Data processing, in banca)

RFID (Radio Frequency Identification)

Appartengono alla categoria usa-e-getta

- Su chip
- Senza batteria
- 0,5 mm di diametro
- Dotati di traspondere radio
- Memorizzano un numero di 128 bit

MICROCONTROLLORI

Piccoli computer inclusi in vari dispositivi, tipicamente in rete

- CPU
- Una piccola memoria
- Qualche dispositivo I/O

GAME COMPUTERS (CONSOLE)

- Effetti grafici speciali
 - Software di base limitato
 - Non estendibili
- } Sono sistemi chiusi e specializzati

TABLET

Quasi dei computer normali con consumi ridotti

SERVER

- Su rete Locale o Web Server
- Memorie di diversi GB
- Diversi TB di disco
- Gestione di rete efficiente

COW (CLUSTER OF WORKSTATION)

- Sistema multiprocessore ad accoppiamento falso
- Hardware di tipo standard: costi contenuti
- Strutture di comunicazione veloci
- Elevata affidabilità
- Anche detto SERVER FARM

MAINFRAME:

- Dinetti discendenti della serie 360
- Gestione efficiente di I/O
- Periferie a dischi di molti Tbyte

UNITÀ DI MISURA

10^{-3} MILLI	10^3 KILO	Imuove nelle memorie si usano le potenze di 2: $1\text{ KB} \rightarrow 2^{10} = 1.024 \approx 10^3$
10^{-6} MICRO	10^6 MEGA	
10^{-9} NANO	10^9 GIGA	
10^{-12} PICO	10^{12} TERA	

NUMERO: Entità astratta

NUMERALE: Stringa di caratteri che rappresenta un numero in un dato sistema di numerazione

ESEMPIO: 11 vale undici in decimale, tre in binario

NUMERI A PRECISIONE FINITA:

Com questi numeri si prendono alcune proprietà:

- Alcune operazioni diventano chiuse (+, -, ×)
- Proprietà associativa e distributiva
- Esononi di arrotondamento
- Buchi nella rappresentazione dei reali

ESEMPI:

- 2 cifre decimali con segno [-99; +99]
- $78+36=114$ (overflow)
- $60+(50-40) \neq (60+50)-40$ (associatività)

IL SISTEMA POSITIONALE

- Ciascuna cifra rappresenta il coefficiente di una potenza della base
- d'espONENTE è dato dalla posizione della cifra

$$N = \sum_{i=-k}^m a_i b^i \quad \boxed{a_m \dots a_1 a_0 a_{-1} a_{-2} \dots a_{-k}} \quad \left. \begin{array}{l} b = \text{base} \\ 0 \leq a_i \leq b-1 \end{array} \right\}$$

ESEMPIO:

$$\begin{array}{r} 125.62 \\ 10^2 \downarrow 10^1 \downarrow 10^0 \\ 125.62 \end{array} \quad \begin{array}{r} 10^{-1} 10^{-2} \\ \nearrow \quad \nearrow \end{array}$$

Se la base è b occorrono b simboli:

- $b = 10 \{0, 1, \dots, 9\}$
- $b = 8 \{0, 1, \dots, 7\}$
- $b = 2 \{0, 1\}$
- $b = 16 \{0, 1, \dots, 9, A, B, C, D, E, F\}$

BASE BINARIA (VIRGOLA FISSA):

$$\begin{array}{r} 1010.01 \\ 2^3 \downarrow 2^2 \downarrow 2^1 \downarrow 2^0 \downarrow 2^{-1} \downarrow 2^{-2} \end{array} \quad \left. \begin{array}{l} 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 10,25 \end{array} \right\}$$

BASE OTTALE (VIRGOLA FISSA):

$$\begin{array}{r} 2107.65 \\ 8^3 \downarrow 8^2 \downarrow 8^1 \downarrow 8^0 \downarrow 8^{-1} \downarrow 8^{-2} \end{array} \quad \left. \begin{array}{l} 2 \cdot 8^3 + 1 \cdot 8^2 + 0 \cdot 8^1 + 7 \cdot 8^0 + 6 \cdot 8^{-1} + 5 \cdot 8^{-2} = 1095,578125 \end{array} \right\}$$

I NUMERI NATURALI IN BINARIO

$$m=3 \quad [0,7]$$

0	000	Com m bit coprono da 0 a $2^m - 1$
1	001	
2	010	
3	011	
4	100	
5	101	
6	110	
7	111	

ADDITIONI TRA NUMERI NATURALI IN BINARIO

Si sommano cifra a cifra come in decimale portando il riporto alla cifra successiva

$$0+0=0$$

$$0+1=1$$

$$1+0=1$$

$$1+1=0 \text{ con riporto di } 1$$

ESEMPIO:

$$\begin{array}{r} 0011 \\ + 0010 \\ \hline 0101 \end{array}$$

N.B.: Se il numero di cifre non permette di rappresentare il numero si ha un overflow

MOLTIPLICAZIONI FRA NUMERI INTERI:

x	0	1
0	0	0
1	0	1

Si segue questa semplice tabella e si procede come per i decimali

$$\begin{array}{r} 11_{10} \\ \times 5_{10} \\ \hline 0101 = \\ 1011 \\ 0000 - \\ 1011 - \\ \hline 11011 \end{array}$$

NUMERI IN VIRGOLA FISSA SENZA SEGNO:

- Si stabilisce il numero di bit
- Viene fissata la posizione della virgola
- Si usa il meccanismo posizionale

SOMMA CON VIRGOLA FISSA

$$3,5 + 2,75 = 6,25$$

$$\begin{array}{r} 0011.10 + \\ 0010.11 = \\ \hline 0110.01 \end{array} \quad \begin{array}{l} \text{la virgola rimane} \\ \text{allo stesso posto} \end{array}$$

MOLTIPLICAZIONE VIRGOLA FISSA

$$\begin{array}{r} 2,75_{10} \quad 10.11 \times \\ 1,25_{10} \quad \underline{-01.01} = \\ \hline 10.11 \\ 0000 - \\ 1011 - \\ \hline 11.0111 \end{array}$$

Si sposta la virgola a sinistra di un numero di posti pari alla somma del numero di cifre dopo la virgola di entrambi i numeri

NUMERI CON SEGNO:

A parità di cifre, per rappresentarli, si dimentica l'intervallo dei valori assoluti.

- Si possono usare diversi metodi:

MODULO E SEGNO:

- Si usa un bit per il segno
- 0 per + e 1 per -
- $m-1$ bit per il modulo
- intervallo $[-2^{m-1}, +2^{m-1}]$

$$\left. \begin{array}{l} \text{ES. } m=4 \text{ bit} \\ S = 0101 \\ -S = 1101 \end{array} \right\}$$

COMPLEMENTO A 1

- Si aggiunge 1 a sinistra
- Per cambiare segno si invertono 0 e 1
- I numeri positivi iniziano con 0, i negativi con 1

$$\left. \begin{array}{l} m=4 \text{ bit} \\ S = 0101 \\ -S = 1010 \end{array} \right\}$$

COMPLEMENTO A 2:

- Si calcola il complemento a 1 del numero
- se positivo si rappresenta come il complemento a 1
- se negativo si somma 1 al numero positivo complementato

VANTAGGI: • Intervallo più esteso

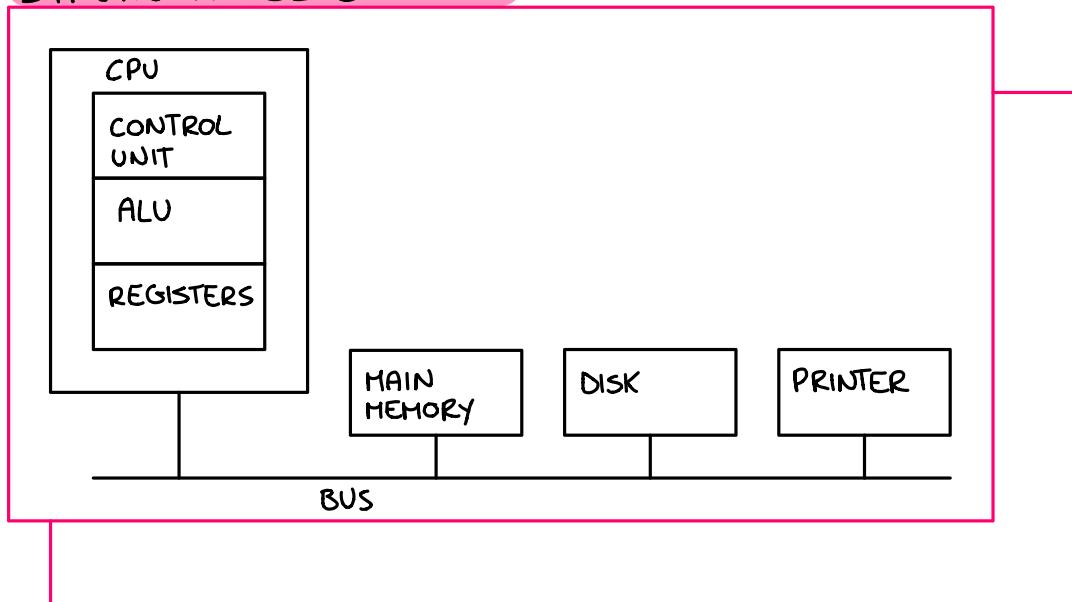
• Una sola rapp. dello 0

Struttura di un calcolatore

ALCUNE TERMINOLOGIE

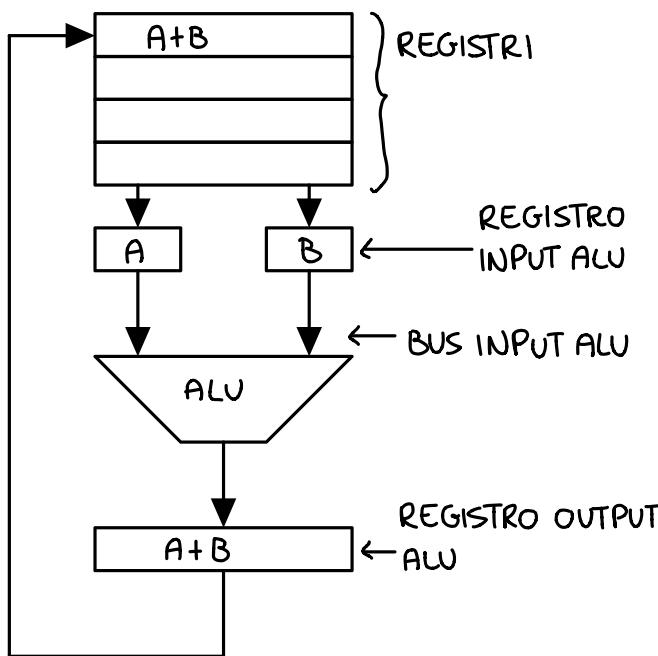
- CALCOLATORE ELETTRONICO: Macchina fatta di dispositivi elettronici che può risolvere problemi eseguendo istruzioni fornitegli
- PROGRAMMA: Sequenza di istruzioni in un linguaggio
- LINGUAGGIO MACCHINA: Eseguibile direttamente da un calcolatore

STRUTTURA DEL COMPUTER



- La memoria contiene sia i dati che le istruzioni.
- Il contenuto dei registri può essere scambiato con la memoria e l'I/O.
- Le istruzioni trasferiscono i dati e modificano il contenuto dei registri.
- Registri particolari:
 - PC: Indirizza la prossima istruzione.
 - IR: Contiene l'istruzione corrente.

STRUTTURA DELLA CPU



- Esecuzione di operazioni aritmetiche e logiche sui dati contenuti nei registri
- Spostamento di dati fra registri e fra registri e memoria
- Ciclo elementare: due operandi sono inviati all'ALU e il risultato messo in un registro

CICLO FETCH-DECODE EXECUTE

- 1- Carica l'istruzione dalla memoria in IR (Instruction Register) **FETCH**
- 2- Incremento PC (Program Counter)
- 3- Decodifica l'istruzione **DECODE**
- 4- Se l'istruzione usa un dato in memoria calcola l'indirizzo
- 5- Carica l'operando in un registro
- 6- Esegui l'istruzione **EXECUTE**
- 7- Torna al passo 1 per l'istruzione successiva

ESECUZIONE VS INTERPRETAZIONE

ESECUZIONE DIRETTA:

- Le istruzioni possono essere eseguite direttamente dai circuiti hardware
- Appoggio molto complesso:
 - Repertorio istruzioni limitato
 - Progettazione dell'HW complessa
 - Esecuzione molto efficiente

INTERPRETAZIONE:

- d' HW puo' eseguire solo alcune operazioni elementari dette microistruzioni
- Ciascuna microistruzione è scomposta in una successione di microistruzioni poi eseguite dall' HW
- Vantaggi:
 - Repertorio istruzioni esteso
 - HW più compatto
 - Flessibilità di progetto

ARCHITETTURA CISC E RISC

RISC: Reduced Instruction Set Computer

- Esecuzione diretta
- Repertorio ristretto
- Istruzioni principalmente sui registri
- Una istruzione per ciclo di macchina

CISC: Complex Instruction Set Computer

- Interpretazione tramite microprogramma
- Repertorio esteso
- Istruzioni anche su memoria
- Molti cicli di macchina per istruzione

PRINCIPI DEI COMPUTER MODERNI

- Eseguire tutte le istruzioni dall'hardware
- Massimizzare la velocità d'esecuzione delle istruzioni
- Semplificare la decodifica delle istruzioni
- Limitare i riferimenti alla memoria (ACCESSI)
- Ampliare il numero di registri

TIPI DI PARALLELISMO:

1) A LIVELLO DI ISTRUZIONI:

- Diverse istruzioni eseguite insieme
- Diverse fasi della stessa istruzione eseguite insieme

2) A LIVELLO DI PROCESSORI:

- Molti processori lavorano insieme allo stesso problema
- Fattori di parallelismo molto elevati
- Diversi tipi di interconnessione e di cooperazione

PARALLELISMO A LIVELLO DI ISTRUZIONI: PIPELINING



S1	1	2	3	4	5	6	7
S2	-	1	2	3	4	5	6
S3	-	-	1	2	3	4	5
S4	-	-	-	1	2	3	4
S5	-	-	-		1	2	3

Time
→

- Ciascuna istruzione è divisa in fasi
- d'esecuzione avviene in una pipeline a più stadi
- Più istruzioni in esecuzione contemporanea
- Una istruzione completata per ogni ciclo

CARATTERISTICHE DI UNA PIPELINE:

Consente un compromesso tra:

- LATENZA: Tempo per eseguire un'istruzione
- AMPIEZZA DI BANDA: Numero di istruzioni completate per ciclo di tempo (TFLOPS)

Com:

$$\cdot \text{VELOCITÀ DI CLOCK} = T \text{ msec}$$

$$\cdot \text{NUM. STADI} = m$$

Abbiamo:

$$\cdot \text{LATENZA} = mT$$

• AMPIEZZA DI BANDA: 1 istruzione ogni $T \text{ msec}$, esempio $1000/T \text{ MIPS}$

ARCHITETTURE SUPERSCALARI:

Sono architetture nelle quali si avranno più istruzioni (4-6) insieme

Si aumenta ulteriormente il parallelismo avendo più di una pipeline nel microprocessore

PROBLEMA! Compatibilità dell'esecuzione parallela

- Indipendenza tra le istruzioni
- Ciascuna istruzione non deve utilizzare i risultati dell'altra

UNITÀ FUNZIONALI MULTIPLE

- Solo lo stadio più lento della pipeline viene parallelizzato
- da CPU contiene al suo interno diverse unità funzionali indipendenti
- Architettura usata nelle CPU Intel Core

PARALLELISMO A LIVELLO DI PROCESSORI

- Miglioramento prestazioni con parallelismo a livello di istruzioni di 5-10 volte
- Per migliorare ancora si usano CPU multiple

Possibili approcci:

DATA PARALLELISM (SIMD)

- Processori matriciali
- Processori vettoriali
- GPU

TASK PARALLELISM (MIMD)

- Multiprocessori
- Multicore
- Multicomputer

GPU

- Operazioni comuni sui pixel, vertici, archi, figure

ESEMPIO NVIDIA FERMI (2009)

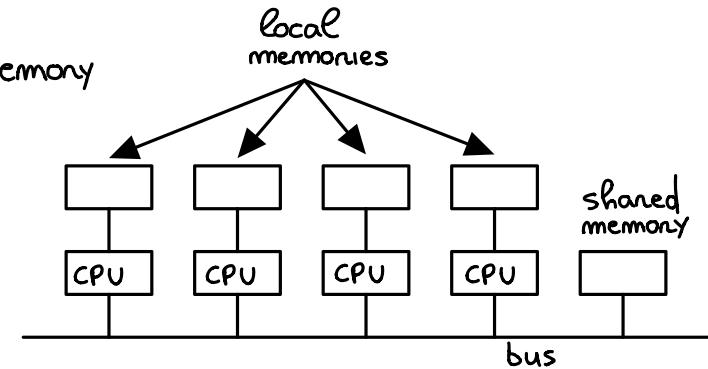
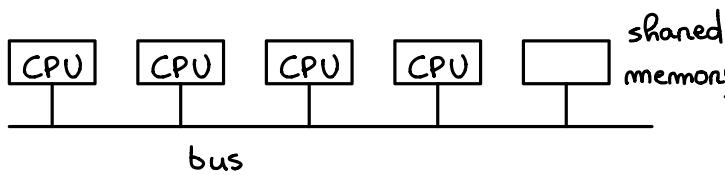
16 processori stream SIMD

Ogni processore ha 32 core

Fino a 512 operazioni per ciclo di clock

MULTIPROCESSORI

- le CPU lavorano indipendentemente
- SHARED MEMORY: il bus può diventare collo di bottiglia
- PRIVATE MEMORY: Contiene il codice e parte dei dati
- lo scambio dei dati avviene tramite la shared memory



ARCHITETTURE MULTICORE

- da CPU è composta da più core, ovvero da più nuclei di processori fisici sullo stesso package
- Ogni core:
 - È un processo indipendente.
 - Può essere dotato di cache autonoma
- Architetture omogenee o eterogenee
- Ogni core può essere multiscalare
- Accoppiamento dei core:
 - Stretto: Shared cache
 - Largo: Private cache
- Nascono a partire dal 2003

LA MEMORIA CENTRALE

- Contiene sia i programmi che i dati
- Memorizzazione binaria (bit)
- Cella: Unità indirizzabile
 - word: Insieme di K byte
- Indirizzo: Tramite il quale la CPU accede al dato nella cella
- Indirizzi binari a m bit: spazio di indirizzamento 2^m celle

CODICI A CORREZIONE DI ERRORE:

Tecniche per garantire maggiore affidabilità nella registrazione/trasmissione di informazioni binarie

Recupero degli errori hardware tramite codifiche ridondanti

Codifiche con $m = m+n$ bit

$$\left\{ \begin{array}{l} m \rightarrow \text{bit complessivi della codifica} \\ m \rightarrow \text{bit dati} \\ n \rightarrow \text{check bit} \end{array} \right.$$

Si usa solo un sottoinsieme di codifiche valide

ESEMPIO: Codice con $m=10, m=2, n=8$

0000000000
 0000011111
 1111100000
 1111111111

$\left. \begin{array}{c} 0000000000 \\ 0000011111 \\ 1111100000 \\ 1111111111 \end{array} \right\} 2^m = 4 \text{ codifiche valide (su } 2^{10})$

DISTANZA DI HAMMING :

Distanza di Hamming tra due codifiche: Indica il numero di bit diversi

ESEMPIO: 0101 e 1001 sono a distanza 2

Distanza di Hamming di un codice: $h \rightarrow$ dist. di Hamming minima tra due codifiche valide del codice

ESEMPIO:

0000000000
0000011111
1111100000
1111111111

} Distanza di Hamming del codice $h=5$

- Per rilevare errori su K bit occorre che sia:

- Almeno $h = K+1$ ovvero $K \leq h-1$

- Per correggere errori su K bit occorre che sia:

- Almeno $h = 2K+1$ ovvero $K \leq (h-1)/2$

ESEMPIO:

0000000000
0000011111
1111100000
1111111111

} Distanza di Hamming del codice $h=5$

$h=5 = K+1 \rightarrow$ E' possibile rilevare errori quadrupli

$0000011111 \rightarrow$ 1111011111 viene riconosciuto come errato

$h=5 = 2K+1 \rightarrow$ E' possibile correggere errori doppi

$0000011111 \rightarrow$ 1100011111 viene corretto in 0000011111

CONTROLLO DI PARITA':

Si usa per rilevare errori singoli

Basta aggiungere un solo check bit $n=1$, $m=m+1$

- **BIT DI PARITA'**: Scelto un modo che il numero complessivo di 1 nella codifica sia sempre pari (o dispari)

Questo codice ha distanza 2, viene rilevato da circuiti molto semplici.
Le memorie segnalano Parity Error quando un errore si manifesta

ESEMPIO: 11011010 bit di parita': 1 \rightarrow 110110101 OK
01100101 bit di parita': 0 \rightarrow 011011010 ERROR

CORREZIONE DI ERRORE SINGOLO:

- m data bit, n check bit, $m+n$ bit totali
- 2^m codifiche valide
- m codifiche errate a distanza 1 da ciascuna delle valide
- Ogni codifica valida me richiede in tutto $m+1$

ESEMPIO: da codifica: 0000

Richiede le codifiche errate:

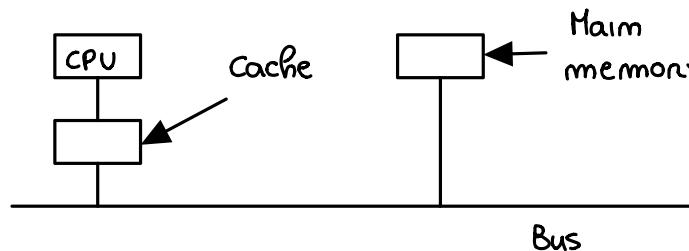
1000	{
0100	
0010	
0001	

Se ogni codifica valida me richiede $m+1$ deve essere:

$$(m+1)2^m \leq 2^n \text{ cioè } (m+n+1) \leq 2^n$$

MEMORIA CACHE

- Viene creata per tamponare il bottleneck tra la velocità della CPU e quella della memoria
- Memorie veloci esistono ma solo di piccole dimensioni
- La cache funziona alla stessa velocità del processore
- Contiene le ultime porzioni di memoria acceduta
- Funziona bene a causa della località degli accessi



CACHE HIT RATIO

Se una parola viene letta K volte di seguito, $K-1$ volte sarà trovata in cache

$$H = (K-1)/K$$

TEMPO MEDIO DI ACCESSO A MEMORIA:

$$A = c + (1 - H)m$$

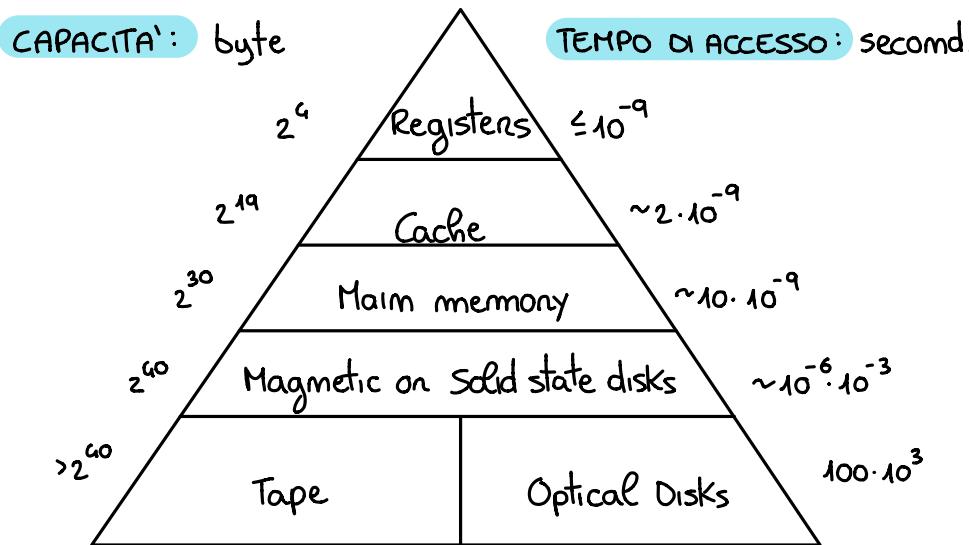
} m: tempo di accesso della memoria
c: tempo di accesso della cache

- La memoria è organizzata in blocchi
- Per ogni cache miss un intero blocco è spostato in cache

TIPI DI RAM:

- SIMM (single inline memory module)
- DIMM (double inline memory module)
- SO-DIMM (small outline DIMM)
- DDR, DDR2, DDR3, DDR4 (double data rate)

GERARCHIE DI MEMORIA



Scendendo nella piramide:

- Cresce il tempo d'accesso
- Aumenta la capacità
- Diminuisce il costo per bit

DISCHI MAGNETICI

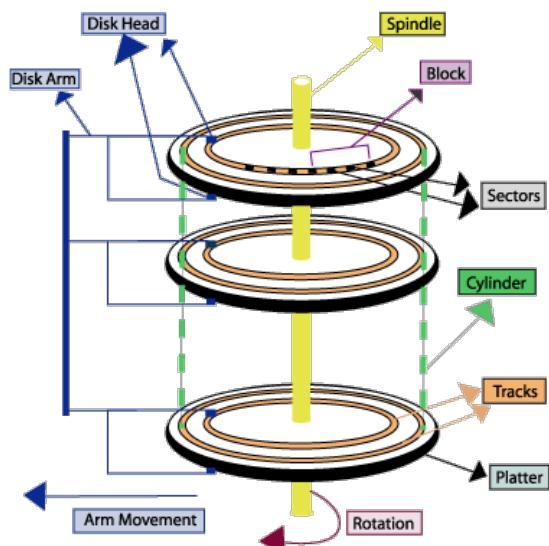
- **DIMENTIONI:** < 10 cm, densità: 25 Gb/cm²
- **REGISTRAZIONE:** Segnale su tracce concentriche
- 50'000 tracce/cm (larghe ~ 200 mm)
- Dischi ad alta densità con bit registrati perpendicolarmente
- Tracce divise in settori con dati, un preamble e un ECC (Error-Correcting-Code)
- Velocità di rotazione costante
- Velocità di trasferimento di 150 MB/sec
- **BURST RATE:** Velocità da quando la testina è sul primo bit
- **SUSTAINED RATE:** Velocità di trasferimento in un certo intervallo

- **CILINDRO:** Insieme di tracce sulla stessa verticale
- **TEMPO DI SEEK:** t_{seek} spostamento delle testine sul cilindro desiderato, dipende in parte dalla distanza
- **TEMPO DI LATENCY:** t_{lat} spostamento sul settore desiderato

TEMPO DI ACCESSO:

$$t_{\text{acc}} = t_{\text{seek}} + t_{\text{lat}}$$

L'organizzazione dei dati è gestita da **CONTROLLORI DI DISCO** (CPU specializzate)



DISCHI RAID

PROBLEMA: Miglioramento lento delle prestazioni dei dischi

SOLUZIONE: RAID (Redundant Array of Inexpensive Disks)

- Dividere i dati su più dischi
- Parallelizzare l'accesso
- Aumentare le data rate
- Introdurre una resistenza ai guasti

Vanno opposti agli SLED (Single Large Expensive Disk)

DATA STRIPPING: Dati consecutivi nello stesso file vengono "affettati" e disposti su dischi diversi, da quali possono essere letti (e scritti) in parallelo

RAID LEVEL 0

- Su m dischi si può ottenere un fattore m sia in lettura che in scrittura
- Il MTBF (Mean Time Between Failures) peggiora
- Non c'è ridondanza

RAID LEVEL 1

- Ciascun disco è duplicato: shadowing
- Ottime prestazioni soprattutto in lettura: molte possibilità di bilanciare il carico
- Eccellente resistenza ai guasti
- Supportato da vari s.o.

RAID LEVEL 2

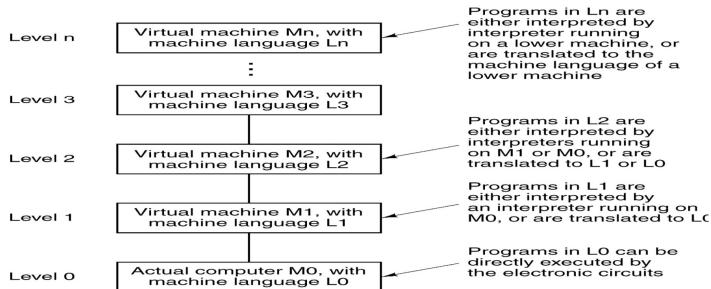
- Striping a livello di WORD o BYTE
- Esempio: un nibble (metà byte) più 3 bit: codice di Hamming a 7 bit
- Registrazione ad 1 bit per ogni disco
- Rotazione dei dischi sincronizzata
- Resiste a guasti semplici
- Guadagna un fattore di q in read e write
- Fante overhead
 - 32 bit + $(6+1)$ parità $\rightarrow 39$ dischi
 - Overhead del 19%
 - Guadagna un fattore 32 in read e write
- Ha senso con molti dischi:

RAID LEVEL 3

- Versione semplificata del 2
- Resiste a guasti semplici! 16 bit di parità, sapendo quale drive è rotto, consente la connessione
- Overhead abbastanza contenuto

Logica digitale e memorie

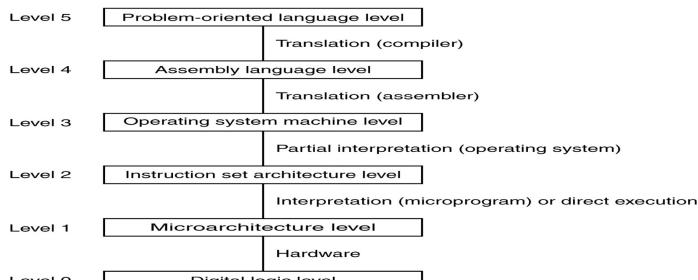
d'architettura di un computer viene detta "a livelli" in cui ogni livello, più si sale, aggiunge una macchina virtuale che utilizza un suo linguaggio di programmazione proprio del livello di astrazione



- Per esempio il linguaggio del liv. 3 sarà tradotto nel linguaggio del liv. 2 o interpretato dalla macchina virtuale del livello 2

l'obiettivo della stratificazione è quello di rendere la macchina più facilmente programmabile (usando i livelli più alti) mantenendo allo stesso tempo una più facile realizzazione hardware tramite i livelli più bassi

TIPICA STRUTTURA A LIVELLI:



- Il livello più basso al quale si può programmare è al livello 2
- Normalmente si programma al livello 5

CIRCUITO DIGITALE:



- Sono circuiti che possono assumere solo 2 livelli (1,0) sia in ingresso che in uscita

- Al circuito sono associate funzioni che me calcolano l'uscita

$$\begin{cases} o_1 = f_1(i_1, \dots, i_m) \\ \vdots \\ o_m = f_m(i_1, \dots, i_m) \end{cases}$$

FUNZIONI BOOLEANE

Si definiscono tramite tabelle di verità, strutturate così:

$x_1 \ x_2 \ \dots \ x_{m-1} \ x_m$	f
0 0 ... 0 0	0
0 0 ... 0 1	1
...	...
1 1 ... 1 1	0

- Ci sono 2^m combinazioni di ingresso
- 2^{2^m} funzioni distinte di m variabili

ESEMPI

$m=1$ con 4 funzioni

x_1	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

f_1 è detta NOT

$m=2$ con 16 funzioni

$x_1 \ x_2$	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0 0	0	0	0	0	0	0	0	0
0 1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1

- f_1 è detta AND
- f_7 è detta OR

NOTAZIONE BOOLEANA:

Date x, y due variabili booleane

- AND si indica con $x \cdot y$ (o xy)
- OR si indica con $x+y$
- NOT si indica con \bar{x}

TEOREMA ESPRESSIONI ALGEBRICHE: "Ogni funzione booleana è algebrica, cioè è rappresentabile con un'espressione algebrica"

PRIMA FORMA CANONICA DI FUNZIONI A N VARIABILI:

$$f = \sum_{j=1, \dots, m} \prod_{i=1, \dots, n} x_{ij}^*$$

• x_{ij}^* vale x_i oppure \bar{x}_i

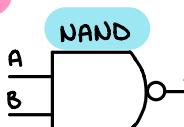
• f è espressa come OR delle combinazioni per cui la funzione è vera (somma di mintermimi)

• In base al teorema, qualsiasi funzione booleana può essere espressa in questa forma

PORTE LOGICHE



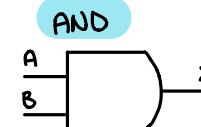
A	x
0	1
1	0



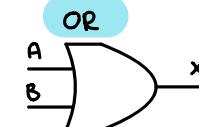
A	B	x
0	0	1
0	1	0
1	0	0
1	1	0



A	B	x
0	0	1
0	1	0
1	0	0
1	1	0



A	B	x
0	0	0
0	1	0
1	0	0
1	1	1



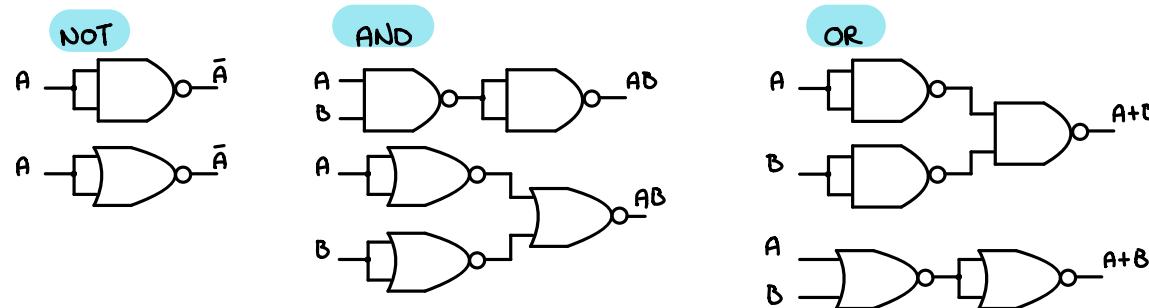
A	B	x
0	0	0
0	1	1
1	0	1
1	1	1

• Si può realizzare qualsiasi funzione booleana tramite combinazioni di AND, OR e NOT

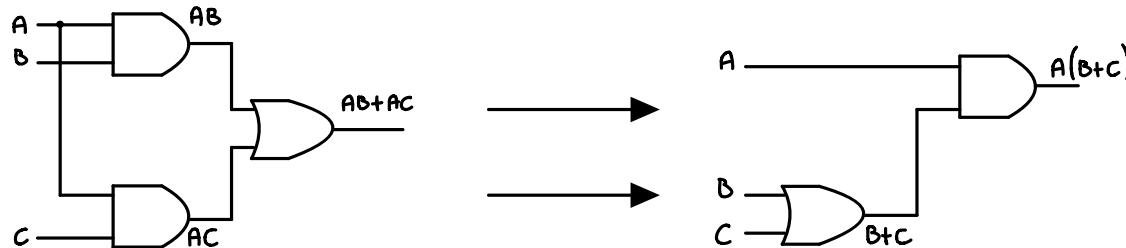
PROPRIETÀ DELL'ALGEBRA BOOLEANA

	AND form	OR form
• IDENTITY LAW	$1A = A$	$0+A = A$
• NULL LAW	$0A = 0$	$1+A = 1$
• IDEMPOTENT LAW	$AA = A$	$A+A = A$
• INVERSE LAW	$A\bar{A} = 0$	$A+\bar{A} = 1$
• COMMUTATIVE LAW	$AB = BA$	$A+B = B+A$
• ASSOCIATIVE LAW	$(AB)C = A(BC)$	$(A+B)+C = A+(B+C)$
• DISTRIBUTIVE LAW	$A+BC = (A+B)(A+C)$	$A(B+C) = AB+AC$
• ABSORPTION LAW	$A(A+B) = A$	$A+AB = A$
• DE MORGAN'S LAW	$\bar{AB} = \bar{A} + \bar{B}$	$\overline{A+B} = \bar{A}\bar{B}$

COMPLETEZZA DELLE PORTE NAND E NOR: È possibile simulare AND, OR e NOT usando solo NAND e NOR



OTTIMIZZAZIONE DEI CIRCUITI



A	B	C	AB	AC	AB+AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

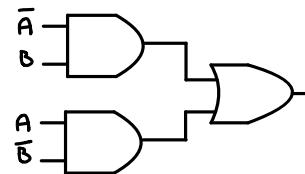
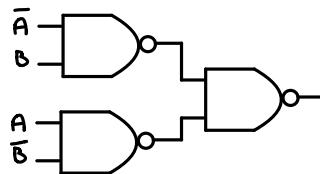
A	B	C	$B+C$	$A(B+C)$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

PORTA XOR

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0



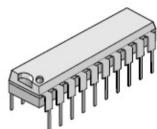
- Calcola la funzione XOR, da un'uscita 1 quando uno solo degli ingressi è 1
- Facilmente realizzabile con ponte AND, OR, NAND



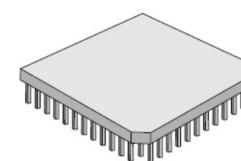
CIRCUITI INTEGRATI

- Molti ponte realizzate su una board di silicio (chip)
- Schede da 16 a 68 piedimi
- Vani livelli di integrazione:
 - SSI SMALL SCALE 1-10 PORTE
 - MSI MEDIUM SCALE 10-100 PORTE
 - LSI LARGE SCALE 10^2 - 10^5 PORTE
 - VLSI VERY LARGE SCALE $>10^5$ PORTE

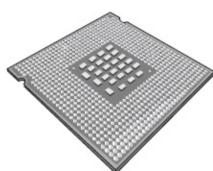
CIRCUITI INTEGRATI MODERNI



- Dual In-Line Packages (DIPs)

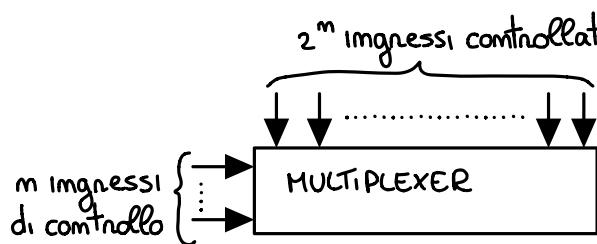


- Pin Grid Arrays (PGAs)



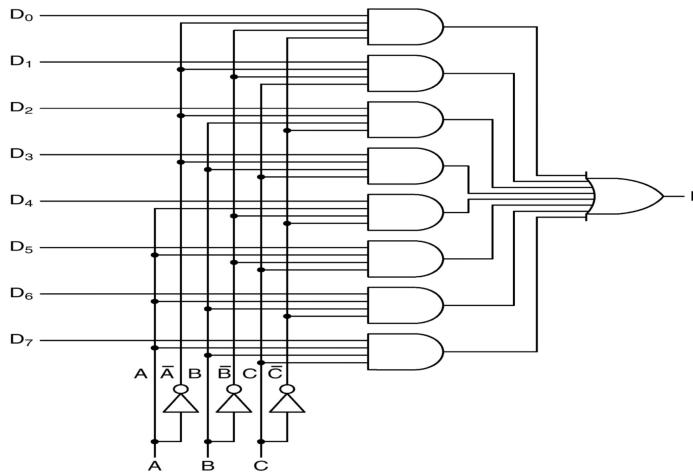
- Large Grid Arrays (LGAs)

CIRCUITI COMBINATORI: Sono circuiti in cui l'uscita dipende solo dagli ingressi, e non dallo stato (avvenuto dalla storia passata)



Gli ingressi di controllo selezionano quale degli ingressi controllati viene mandato in uscita

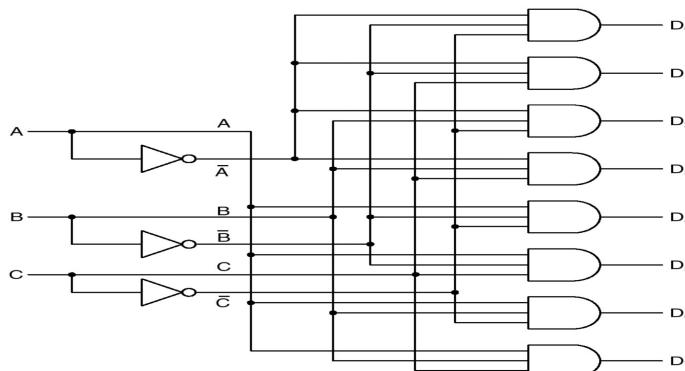
MULTIPLEXER



REALIZZAZIONE DI FUNZIONI BOOLEANE TRAMITE MULTIPLEXER

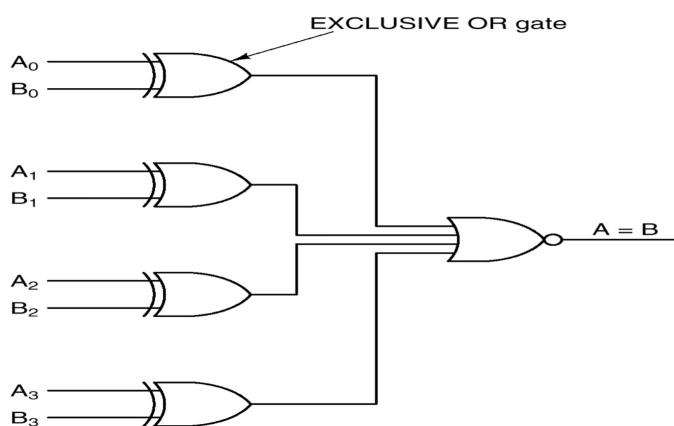
- Con un multiplexer ad m bit si puo' calcolare qualsiasi funzione di m variabili
- Gli ingressi controllati corrispondono ai minterminimi
- Si cablano a 0 o 1, a seconda che il minterm minime compaia o meno nella forma canonica

DECODIFICATORE



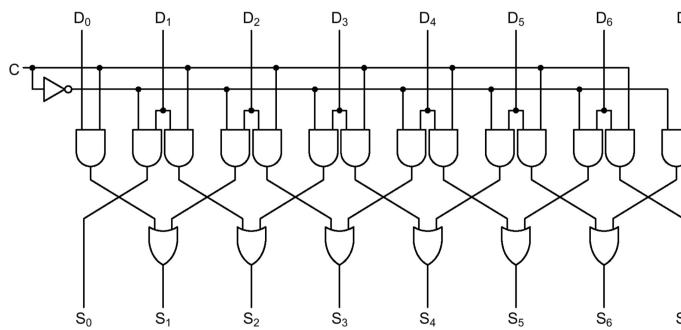
- Circuiti a m ingressi e 2^m uscite
- Una ed una sola delle 2^m uscite assume valore 1 in corrispondenza della configurazione di m bit in ingresso

COMPARATORE



- Compara i bit omologhi di due stringhe
- d'uscita vale 1 se e solo se $A_i = B_i \forall i$
- Se $A_i = B_i$ allora $A_i \text{ XOR } B_i = 0$
- Il NOR da uscita 1 solo quando tutti i suoi ingressi valgono 0

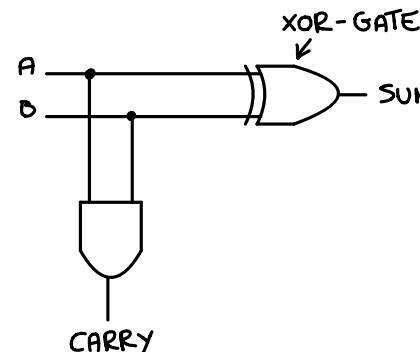
SHIFTER



• Il segnale C determina il verso dello shift (sinistra/destra) (si perde una cifra)

SEMIADDIZIONATORE

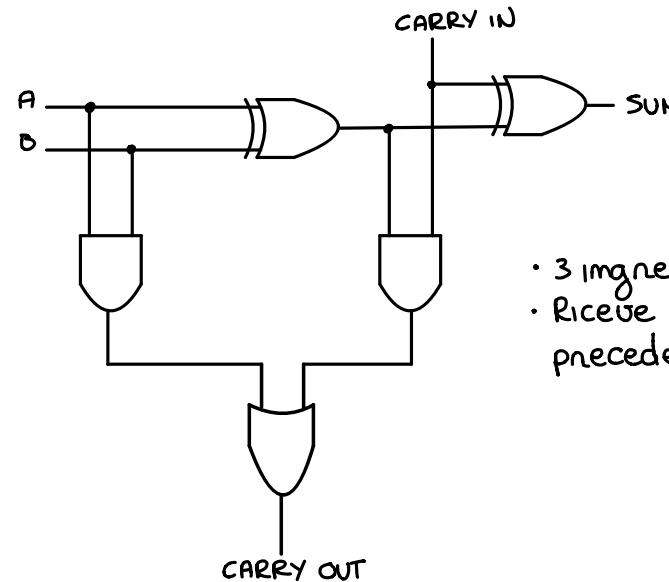
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



- 2 ingressi e 2 uscite: somma e riporto (carry)
- mom si puo' usare per la somma di numeri a piu' bit, dove serve sommare anche il carry della cifra precedente

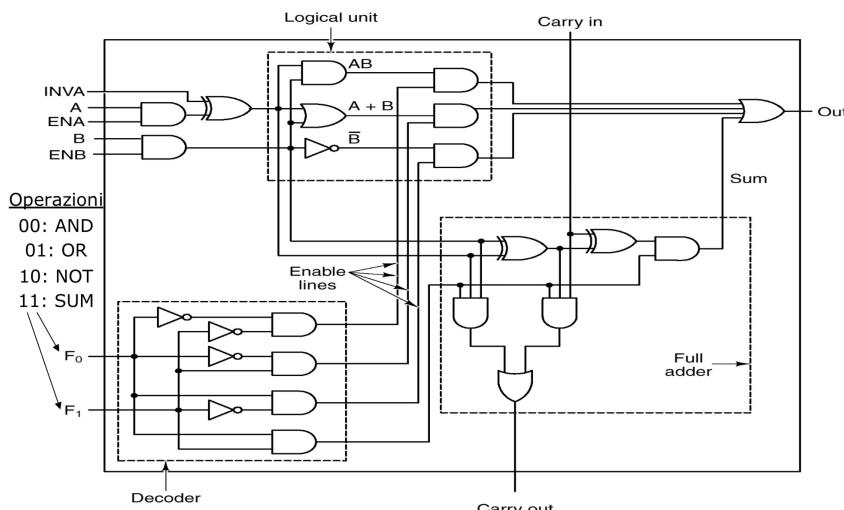
ADDIZIONATORE COMPLETO

A	B	CARRY IN	SUM	CARRY OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

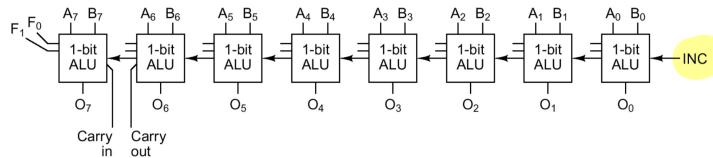


- 3 ingressi e 3 uscite
- Riceve il riporto della cifra precedente

ALU A 1 BIT

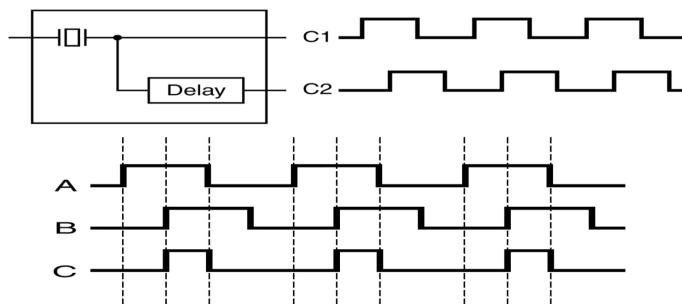


ALU AD N BIT



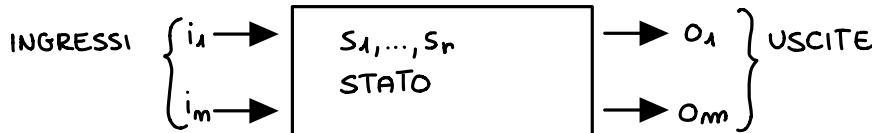
- Realizzata commettendo m ALU ad 1 bit
- INC incrementa la somma di 1 ($A+1, A+B+1$)
- PROBLEMA: Propagazione dei riporti
- Ciascuno stadio deve attendere il riporto dal precedente
- Tempo di addizione lineare con m

CLOCK



- Tutti i cambiamenti di stato vengono sincronizzati da un segnale di clock
- Da un clock primario ne vengono ricavati per sfasatura, sottrazione, etc...
- Le transizioni di stato del circuito possono avvenire:
 - In corrispondenza dei livelli
 - In corrispondenza dei fronti

CIRCUITI SEQUENZIALI

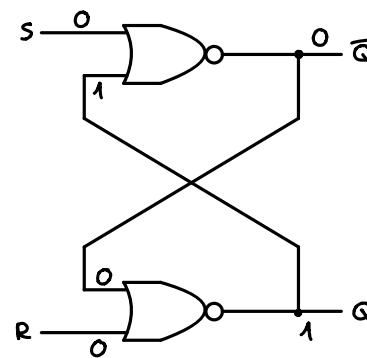
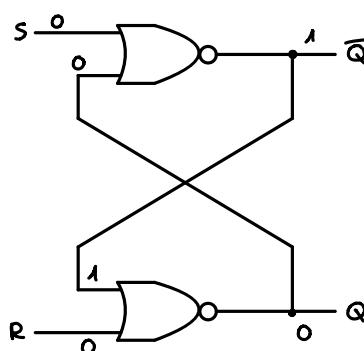


$$O_i = f_i(i_1, \dots, i_m, s_1, \dots, s_n) \quad i=1, \dots, m$$

$$s'_j = g_j(i_1, \dots, i_m, s_1, \dots, s_n) \quad j=1, \dots, n$$

- Le uscite del circuito dipendono sia dagli ingressi che dalla storia passata
- da storia passata è riassunta nello stato codificato nelle variabili di stato booleane s_1, \dots, s_n
- Le variabili di stato sono memorizzate in elementi di memoria binari
- Circuiti combinatori calcolano le uscite ed il nuovo valore dello stato

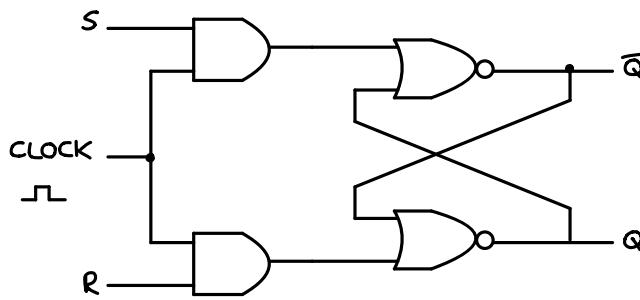
LATCH



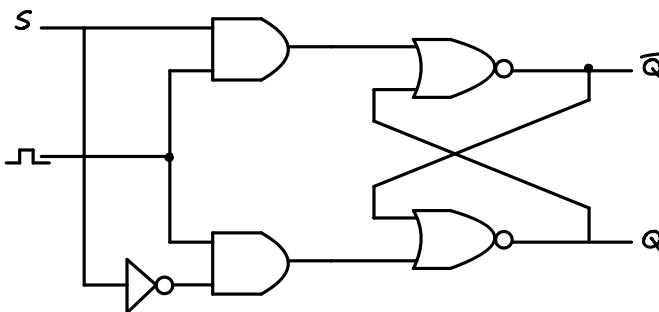
A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

- Dispositivo di memoria elementare
- Due stati stabili $Q=0$ e $Q=1$
 - S (SET): forza Q a 1
 - R (RESET): forza Q a 0
- Con $S=R=0$ il circuito mantiene lo stato
- Il circuito commuta sui livelli cioè quando S o R valgono 1
- S ed R non devono mai andare insieme ad 1

LATCH CON CLOCK, LATCH D



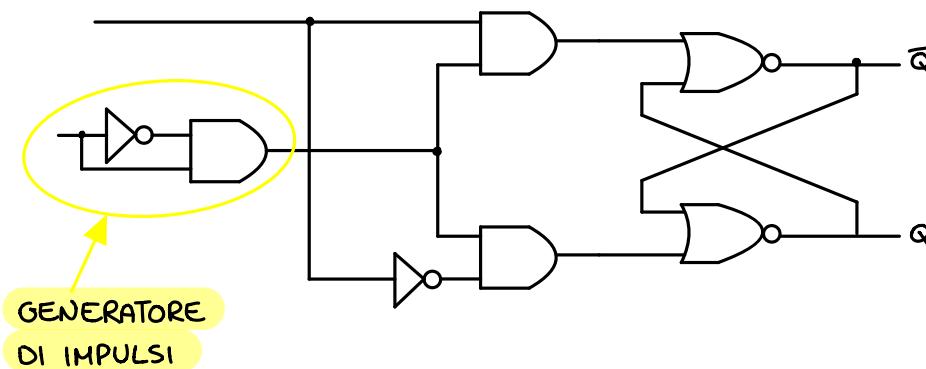
- R ed S vengono trasferiti sugli ingressi del latch solo quando il clock è ad 1
- Quando il clock è a 0 vengono ignorati



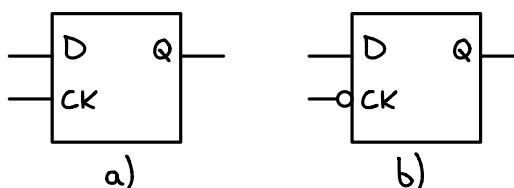
- Il latch D (Delay) quando il clock va ad 1 registra nello stato Q il valore dell'ingresso D

FLIP-FLOP

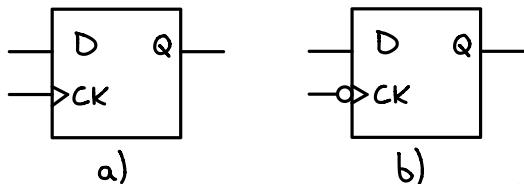
Il flip-flop è una varietà del latch che commuta sui fronti del clock



LATCH E FLIP-FLOP



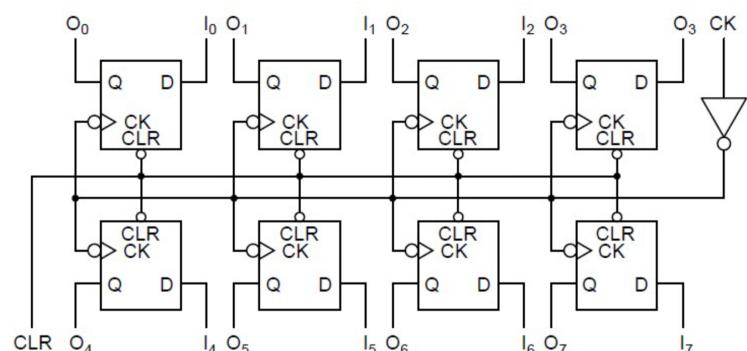
- I Latch commutano sui livelli del clock a) ALTO b) BASSO



- I Flip-Flop commutano sui fronti del clock:
- a) Commuta sul fronte di salita
- b) Commuta sul fronte di discesa

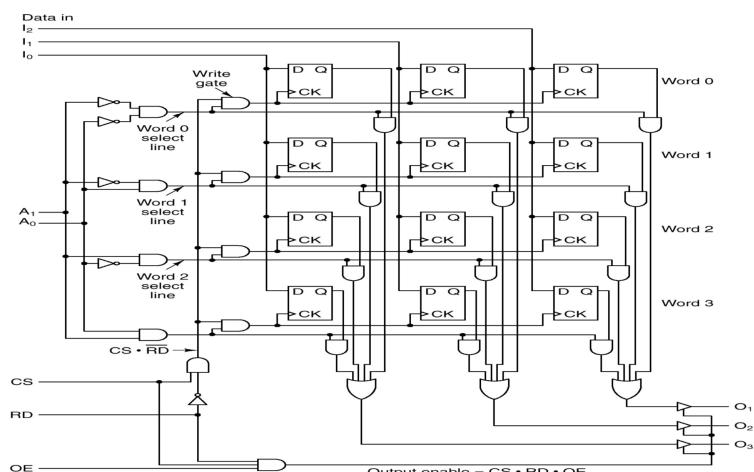


REGISTRI

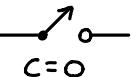
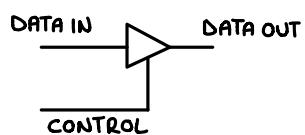


- I Flip-Flop sono gli elementi base di memorizzazione nel computer
- Molti Flip-Flop possono essere messi su un unico chip
- Occorrono in genere da 6 a 10 transistor per ogni Flip-Flop

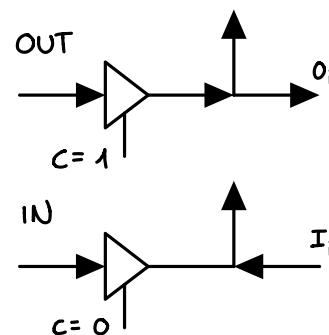
ORGANIZZAZIONE DELLA MEMORIA



DISPOSITIVI A 3 STATI

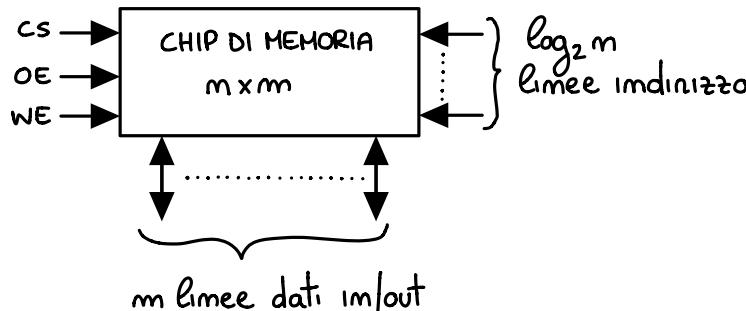


- In base ad un segnale di controllo C si comporta:
 - $C=1$ come circuito chiuso
 - $C=0$ come circuito aperto
- Tempo di commutazione: pochi msec



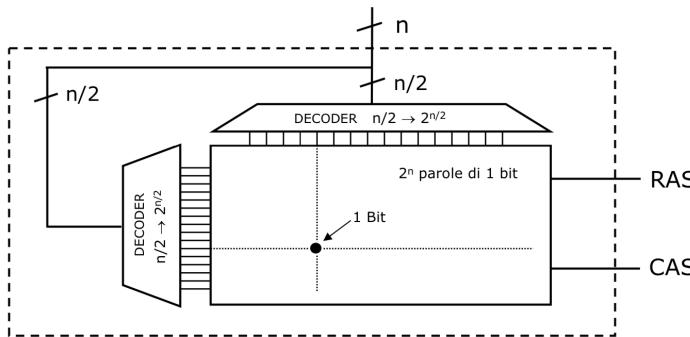
- Consente di usare gli stessi piedini sia per la lettura che per la scrittura
- Usato anche per la commissione ai bus e a qualsiasi linea bidirezionale

CHIP DI MEMORIA



- Chip da $m \times m$ bit complessivi (m parole da m bit)
- $m n$ linee dati bidirezionali
- $\log_2 m$ linee di indirizzo
- Segnali di controllo:
CS (Chip Select)
OE (Output Enable)
WE (Write Enable)
- Problema: numero limitato di piedini del contenitore

MATRICE DI SELEZIONE



• Si risparmia nella complessità della logica di decodifica

- Un decodice $m \rightarrow 2^m$ richiede 2^m ponte AND
- RAS (Row Address Strobe), CAS (Column Address Strobe)

ESEMPIO:

- $4 M$ parole da 1 bit $\rightarrow 22$ linee di indirizzo
- 1 decodice a $22 \rightarrow 4 M$ ponte AND
- 2 decodice a $11 \rightarrow 2 \cdot 2^{11} = 4 K$ ponte AND

SEGNALI ASERITI E NEGATI

In alcuni casi un segnale provoca l'azione corrispondente la sua tensione è alta, in altri quando è bassa

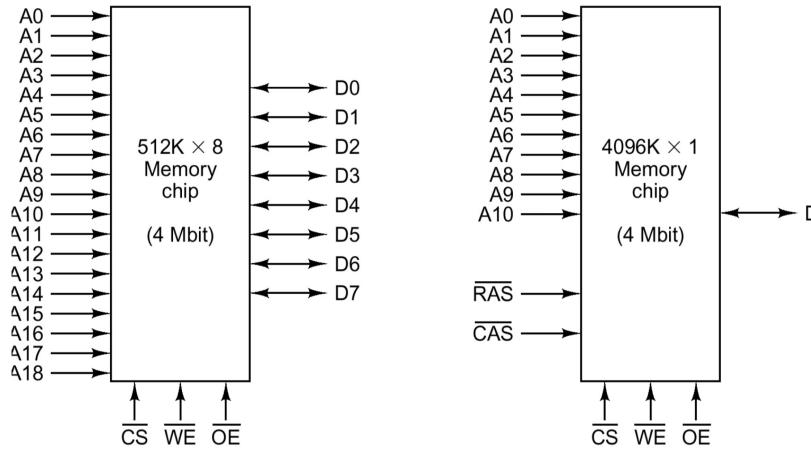
Si parla di:

- SEGNALE ASERITO quando assume il valore che provoca l'azione
- SEGNALE NEGATO altrimenti

con la seguente notazione:

- S: segnale che è assentito alto (s)
- \bar{S} : segnale che è assentito basso ($s\#$)

CHIP DI MEMORIA



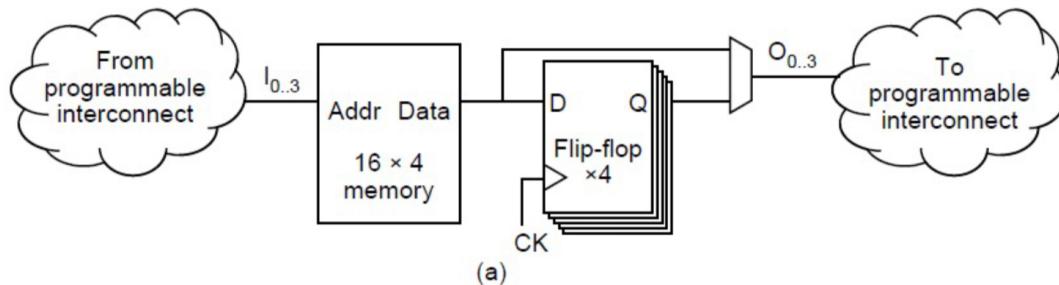
TIPI DI RAM E ROM:

- RAM (Random Access Memory)
- ROM (Read Only Memory)
- SDRAM (Static Ram) a FLIP-FLOP, molto veloce (~ 5 msec)
- DRAM (Dynamic RAM): basata su capacità parassite; richiede refresh, alta densità, basso costo (~ 70 msec)
 - FPM: Selezione a matrice
 - EDO: Extended Data Output, lettura im pipeline, più banda
- SDRAM (Synchronous DRAM)
 - Simultanea, prestazioni migliori
- DDR (Double Data Rate)
 - Lettura / scrittura im pipeline
 - DDR 2-3-4: 100 MHz / 1.6 GHz, fino a 25,6 GBs
- PROM (Programmable ROM)
- EPROM (Erasable PROM) raggi UV
- EEPROM: cancellabile elettricamente
- FLASH MEMORY: Tipo di EEPROM, ciclo 50 msec, max 100'000 riscritture

Type	Category	Erasure	Byte alterable	Volatile	Typical use
SRAM	Read/write	Electrical	Yes	Yes	Level 2 cache
DRAM	Read/write	Electrical	Yes	Yes	Main memory (old)
SDRAM	Read/write	Electrical	Yes	Yes	Main memory (new)
ROM	Read-only	Not possible	No	No	Large-volume appliances
PROM	Read-only	Not possible	No	No	Small-volume equipment
EPROM	Read-mostly	UV light	No	No	Device prototyping
EEPROM	Read-mostly	Electrical	Yes	No	Device prototyping
Flash	Read/write	Electrical	No	No	Film for digital camera

FIELD PROGRAMMABLE GATE ARRAY (FPGA)

- Consentono di realizzare circuiti logici arbitrari
- Due componenti replicati:
 - LOOKUP TABLES (LUT) Piccola memoria che si usa per implementare una qualsiasi funzione booleana
 - CONNESSIONI PROGRAMMABILI



I Bus

Architettura a più bus:

Costituita da bus interni ed esterni al chip, quelli interni collegano i registri all'ALU per esempio, quelli esterni invece collegano la CPU (come unità intera) e le varie memorie (RAM, dischi rigidi, dispositivi di input e di output).

L'importanza dei bus è alta, infatti devono soddisfare una serie di criteri come:

Alta velocità di trasferimento (Ampiezza di banda)

Numero di linee elevato (Parallelismo dei bus)

Retrocompatibilità (Per poter lavorare anche con dispositivi antecedenti ai bus che sto usando)

Nei pc moderni sono presenti almeno due bus esterni.

Comunicazione sul bus:

E' regolata da un protocollo di bus, si comunica sempre solo tra due dispositivi per volta (Master e Slave), che possono anche invertirsi in base alla situazione.

I dispositivi sono connessi al bus tramite il bust transceiver.

Per collegare un dispositivo ad un bus servono o dispositivi a tre stati o di tipo open collector.

Larghezza del Bus:

Corrisponde al numero di linee di un bus.

Le linee di indirizzo indicano lo spazio di memoria indirizzabile (2^n locazioni con n bit di indirizzo)

Le linee dati contengono i dati da trasmettere.

Per diminuire i costi si condividono più segnali sulla stessa linea, però così facendo al crescere

della velocità aumenta il bus skew (differenza nella velocità di propagazione dei segnali su linee diverse)

Segnali asseriti e negati:

Sono dei segnali che valgono 1 (quando la tensione è alta) e 0 (quando è bassa), ed in base al valore della tensione fanno qualcosa.

Si parla di segnale asserito alto quando è riportato il nome del segnale e basta, invece di segnale asserito basso quando è rappresentato con il nome con sopra un trattino o con S#.

Bus Sincroni:

Sono bus che vengono temporizzati in base ad un segnale di clock esterno (usando il fronte del segnale di clock), quindi qualsiasi azione del bus avviene durante il fronte del clock

Bus Sincrono: Temporizzazione

Ci sono due vincoli da dover rispettare:

Bus Asincrono: Ciclo di lettura

Si effettua un handshake tra dispositivi, consentendo anche di collegare dispositivi con velocità diverse.

Gli eventi avvengono in risposta ad altri eventi.

Si usa il FULL HANDSHAKE:

- MSYN# Asserito
- SSYN# asserito in risposta a MSYN#
- MSYN# negato in risposta a SSYN#
- SSYN# negato in risposta alla negazione di MSYN#

Arbitraggio del Bus:

- Permette di decidere quale dispositivo sarà il prossimo Bus Master risolvendo eventuali conflitti
- Spesso l'arbitro è nel chip del microprocessore
- Linea di richiesta condivisa
- Il Bus grant è propagato dall'arbitro prima dell'inizio del ciclo
- Viene intercettato dal futuro master
- Sono favoriti i dispositivi che si trovano vicino all'arbitro

Si può ovviare a ciò implementando LIVELLI MULTIPLO DI PRIORITA', in cui:

- Ci sono diverse linee di richiesta associate a diversi livelli di priorità
- In caso di conflitto sono favorite le catene a priorità più alta
- All'interno di ciascuna catena vale la posizione
- In genere se c'è un solo bus con anche la memoria, la CPU ha la priorità più bassa dei dispositivi di I/O

Arbitraggio Decentralizzato:

In sistemi semplici si può eseguire un arbitraggio anche in assenza di un arbitro, con le seguenti modalità.

- Quando nessun dispositivo vuole il bus la linea di arbitraggio è asserita con propagazione a tutti i dispositivi
- Quando un dispositivo vuole il bus:
 - Invia una richiesta di bus
 - Verifica se il bus è libero
 - Se In è asserito diventa master, nega Out e asserisce Busy
 - Se In non è asserito non diventa master e nega Out.

Non necessitando di arbitro risulta quindi più veloce e semplice.

CPU Struttura

Il livello di microarchitettura si trova ad un livello di astrazione maggiore rispetto a quello logico-digitale, e ci serve per studiare come vengono implementate le istruzioni macchina mediante i dispositivi digitali (hardware a sua disposizione).

Vediamo l'ALU come una scatola nera, come anche i registri, anch'essi visti come scatole nere, ai quali si aggiungono anche dei bus interni e delle linee di controllo.

POSSIBILI IMPLEMENTAZIONI:

- RISC: Le istruzioni possono venire eseguite direttamente dalla microarchitettura
- CISC: Le istruzioni sono divise in microistruzioni ed eventualmente interpretate

ESEMPIO DI MICROARCHITETTURA:

La JVM può essere intesa come microarchitettura, però in questo corso ci limiteremo a studiare le seguenti cose:

- La microarchitettura - data path
- La temporizzazione di esecuzione
- L'accesso alla memoria
- Il formato delle micro-istruzioni
- La sezione di controllo

IL CAMMINO DEI DATI NELLA JVM:

//Inserire img

- E' strutturata con più registri
- I registri PC e MBR consentono il caricamento delle istruzioni in memoria, il PC contiene l'indirizzo dell'istruzione e MBR contiene l'istruzione stessa
- I registri sono tutti da 32 bit, tranne il MBR che è da 8 bit, in quanto nella JVM le istruzioni sono solo da 8 bit
- MAR e MDR servono per gestire i dati utilizzati dalle istruzioni, MAR si occupa degli indirizzi dei dati, MDR si occupa dei dati veri e propri
- H (Holding) (Accumulatore) serve per trattenere i dati da mandare in ingresso all'ALU (Serve per ridurre il numero di bus necessari da 3 a 2, con l'inconveniente che per poter fare operazioni binarie tra due registri devo precaricare il contenuto di uno dei due nel registro H passando prima per l'ALU, impiegando così un ciclo di macchina in più).
- L'ALU in questa microarchitettura non fa operazioni di scalatura in quanto di questo se ne occupa uno shifter collegato subito di seguito, l'ALU si occupa solo delle operazioni aritmetiche
- Le frecce con il numero sopra corrispondono a delle linee di abilitazione.

TEMPORIZZAZIONE DEL CICLO DI BASE

E' importante avere un segnale di clock per poter temporizzare/scandire le attività da compiere, il modo per fare ciò è utilizzare un fronte del clock (o salita o discesa) per far partire tutto, per esempio nella seguente immagine si usa il fronte di discesa. Poi devo tenere conto di un tempo di assestamento (Delta W) in cui i segnali di clock si assestano, un periodo (Delta X) che è il percorso dal bus sorgente fino all'ALU (Bus b), un tempo (delta Y) per attraversare l'ALU e lo shifter e un ultimo tempo delta (Delta Z) che è il tempo per andare dall'uscita dello shifter fino al registro successivo. Successivamente utilizzo il fronte di salita per registrare istantaneamente il registro con i dati contenuti sul bus C

ACCESSO ALLA MEMORIA

Accesso parallelo a due memorie:

- Memoria Dati: 32 bit indirizzabili a word in lettura e scrittura
- Memoria Istruzioni: 8 bit indirizzabili a byte solo in lettura

Registri coinvolti:

- MAR (Memory Address Register) Contiene l'indirizzo della word dati
- MDR (Memory Data Register) Contiene la word dati
- PC (Program Counter) Contiene l'indirizzo del byte di codice
- MBR (Memory Buffer Register) Riceve il byte di codice (sola lettura)

Caricamento di B da parte di MBR:

- Estensione a 32 bit con tutti 0
- Estensione del bit più significativo (per mantenere il segno)

STRUTTURA DELLE MICRO-ISTRUZIONI

Una micro istruzione a 36 bit deve contenere:

- Tutti i segnali di controllo da inviare al data path durante il ciclo
- Le informazioni per la scelta della microistruzione successiva

Segnali di controllo:

- 9 Selezione registri sul bus C
- 9 Selezione registro sul bus B
- 8 Funzioni ALU e shifter
- 2 Lettura e scrittura dati (MAR/MDR)
- 1 Lettura istruzioni (PC/MBR)

Selezione micro-istruzione successiva:

- 9 Indirizzo micro-istruzione (su 512)
- 3 Modalità di scelta

Dato che si invia su B solo un registro per volta, si codificano 9 segnali con 4

CACHE A MAPPATURA DIRETTA

- Spazio di memoria di 2^n byte, diviso in blocchi da 2^r byte
- Gli $n-r$ bit più significativi dell'indirizzo specificano il blocco
-