**Linguaggi, livelli e macchine virtuali**

- Traduzione: consiste nel sostituire, in una fase iniziale, ogni sua istruzione con un equivalente sequenza di istruzioni in LO. Il programma che ne risulta è costituito interamente da istruzioni di LO e può essere eseguito dal computer al posto del programma L1 originale.

- Interpretazione: consiste nello scrivere un programma in LO che accetta come dati d’ingresso programmi in L1; tale programma li esegue esaminando un’istruzione alla volta e sostituendola direttamente con l’equivalente sequenza di istruzioni LO. Il programma che la esegue è detto interprete.

La differenza è che, nel caso della traduzione, il programma L1 viene, all’inizio, convertito interamente in un programma LO. Nell’interpretazione ciascuna istruzione L1 viene esaminata e decodificata, e quindi eseguita direttamente senza generare alcun programma tradotto.

Immagine che contiene testo, ricevuta, documento, Carattere

Descrizione generata automaticamente

Il livello 0, che si trova alla base, rappresenta il vero e proprio hardware della macchina. Gli oggetti di interesse sono le porte logiche e i registri (combinazioni di porte logiche).

Subito dopo troviamo il livello di microarchitettura. Qui vi è una memoria locale, formata da un gruppo di registri (in genere da 8 a 32), e un circuito chiamato ALU (Arithmetic Logic Unit), capace di effettuare semplici operazioni aritmetiche. I registri sono connessi alla ALU per formare un percorso dati (data path) lungo il quale questi ultimi si spostano.

Il livello 2 consiste nelle istruzioni eseguite dall’architettura della macchina interpretate dal microprogramma.

Il livello 3 è un ibrido in quanto la maggior parte delle istruzioni del suo linguaggio fa parte anche del livello ISA, però vengono introdotte ulteriori istruzioni, una diversa organizzazione della memoria e la capacità di eseguire programmi in modo concorrente.

Il livello 4 consente di scrivere programmi per i livelli 1, 2 e 3 in maniera meno difficoltosa. I programmi sono tradotti nei linguaggi del livello sottostanti. Il programma che esegue la traduzione è chiamato assemblatore.

Il livello 5 definisce linguaggi di programmazione ad alto livello che sono tradotti da un compilatore al livello 3 o al livello 4.

**Evoluzione delle architetture multilivello**

- Invenzione della microprogrammazione

- Invenzione del sistema operativo

- Migrazione delle funzionalità verso il microcodice

- Eliminazione della microprogrammazione

- Gli attuali processori si affidano di nuovo alla microprogrammazione

**La macchina di Von Neumann**

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Si compone delle seguenti parti:

- *memoria*: conserva sia il programma che i dati su cui deve lavorare il programma;

- *CPU*: è l’unità di elaborazione composta da tre elementi principali:

* *ALU (Arithmetic Logic Unit)*: esegue le istruzioni elementari come quelle aritmetiche e logiche;
* *CU (Control Unit)*: recupera le istruzioni in memoria secondo l’ordine stabilito dall’algoritmo e permette la loro esecuzione;
* *Accumulatore*: è una memoria interna della CPU che viene utilizzata per contenere gli operandi delle istruzioni eseguite dalla ALU.

- *Input/Output (I/O)*: costituisce l’interfacciamento del calcolatore verso l’esterno;

- *bus di comunicazione*: è il canale che permette la comunicazione tra le unità appena descritte.

**Processori**

La CPU è il cervello del computer, ed è composta dall’unità di controllo (CU), l’unità aritmetica e logica (ALU) e alcuni registri (piccole memorie ad altissima velocità).

Tra i registri sono molto importanti:  
- Program Counter (PC): punta alla prossima istruzione da prelevare per l’esecuzione;

- Instruction Register (IR): mantiene l’istruzione corrente in fase di esecuzione.

Le componenti di un computer sono collegate attraverso un bus: una collezione di cavi paralleli utilizzati per trasferire indirizzi, dati e segnali di controllo.

**Organizzazione della CPU**

Il percorso dati (data path) di una tipica CPU di von Neumann è composta dai registri (da 1 a 32), dalla ALU e da alcuni bus che connettono fra loro le diverse parti.

La maggior parte delle istruzioni può essere divisa in due categorie principali:

- istruzioni registro-memoria (necessita di una fase di caricamento delle parole[[1]](#footnote-1) della memoria nei registri): permettono di prelevare parole di memoria per portarle all’interno dei registri (o viceversa), dove sono utilizzabili, per esempio, come input della ALU per effettuare istruzioni successive.

- istruzioni registro-registro (gli operandi sono già pronti nei registri): preleva due operandi dai registri, li porta all’interno dei registri di input della ALU, esegue su di loro una qualche operazione e ne memorizza il risultato in uno dei registri.

**Esecuzione dell’istruzione**

La CPU esegue ogni istruzione compiendo una serie di passi:

1. prelevare la successiva istruzione dalla memoria per portarla nell’IR;

2. aggiornare il PC per farlo puntare all’istruzione seguente;

3. determinare il tipo dell’istruzione appena prelevata (decodifica dell’istruzione);

4. se l’istruzione usa una parola in memoria, determinare dove si trova;

5. se necessario, prelevare la parola per portarla in un registro della CPU;

6. eseguire l’istruzione;

7. tornare al punto 1 per iniziare l’esecuzione dell’istruzione successiva.

Spesso ci si riferisce a questa sequenza di passi con il termine di ciclo esecutivo delle istruzioni, o ciclo di prelievo-decodifica-esecuzione (fetch-decode-execute).

**Strategie di progettazione delle CPU**

- CISC (Complex Instruction Set Computer): la CPU è in grado di comprendere molte istruzioni complesse nativamente (è il più alto livello di astrazione riconosciuto dalla macchina);

- RISC (Reduced Instruction Set Computer): si basa sull’idea che se le istruzioni sono semplici e poche, esse possono essere eseguite rapidamente (è necessario un solo ciclo nel datapath);

- Ibrido: a partire dal x486, le CPU intel contengono un sottoinsieme di istruzioni RISC (quelle più comuni) che possono essere eseguite in un singolo ciclo nel datapath, mentre le altre complesse sono interpretate secondo la classica modalità CISC.

**Principi di progettazione dei calcolatori moderni**

**- Tutte le istruzioni devono essere eseguite direttamente dall’hardware**

**- Massimizzare la frequenza di emissione delle istruzioni**

**- Le istruzioni devono essere facili da decodificare**

**- Solo le istruzioni Load e Store fanno riferimento alla memoria**

**- Molti registri disponibili**

**Parallelismo a livello di istruzione**

Poiché l’incremento del clock del processore ha raggiunto un limite fisico, i progettisti di CPU guardano al parallelismo (più istruzioni nello stesso tempo) per incrementare le prestazioni.

Il parallelismo si può ottenere in due diversi modi:  
- Parallelismo a livello di istruzione: il parallelismo è sfruttato all’interno delle istruzioni per ottenere un maggior numero di istruzioni al secondo;

- Parallelismo a livello di processore: più CPU collaborano per risolvere lo stesso problema.

**Pipelining**

Una limitazione nella velocità di esecuzione delle istruzioni è rappresentato dal prelievo delle istruzioni dalla memoria.

Per alleviare questo problema, i calcolatori sono stati dotati della capacità di poter prelevare in anticipo le istruzioni dalla memoria, in modo da averle già a disposizione nel momento in cui dovessero rendersi necessarie.

Il pipeline divide l’esecuzione di un’istruzione in un numero maggiore di parti che possono essere eseguite in parallelo

Il pipelining permette di bilanciare la latenza (quanto dura l’esecuzione di una istruzione) con la banda del processore (quanti MIPS la CPU è in grado di emettere).

**Parallelismo a livello di processore**

Il parallelismo nel chip aiuta a migliorare le performance della CPU: con il pipelining e le architetture superscalari si può arrivare ad un fattore di miglioramento da 5 a 10.

Però per incrementare drasticamente le performance di un calcolatore occorre progettare sistemi con molte CPU, in questo caso si può arrivare ad ottenere un incremento di 50, 100, o anche più.

Esistono tre differenti approcci: computer con parallelismo sui dati, multiprocessori, multicomputer.

**Computer con parallelismo sui dati**

Ci sono due schemi differenti:

- Processori SIMD (Single Instruction-stream Multiple Data-stream): sono costituiti da un vasto numero di processori identici che eseguono la stessa sequenza di istruzioni su insieme differenti di dati;

- Processori vettoriali: un processore vettoriale esegue la stessa sequenza di operazioni su coppie di dati, ma tutte le addizioni sono svolte da un unico sommatore strutturato in pipeline.

**Multiprocessori**

È un’architettura costituita da più CPU che condividono una memoria comune.

Poiché ciascuna CPU può leggere o scrivere qualsiasi zona della memoria comune, le CPU devono sincronizzarsi via software.

In questo caso le CPU hanno la necessità di interagire in modo così profondo che il sistema è detto fortemente accoppiato (tightly coupled).

**Multicomputer**

Multiprocessori con molte CPU sono difficili da realizzare, per via del problema delle connessioni di ciascuna CPU verso la memoria comune.

I progettisti hanno superato il problema abbandonando il concetto di memoria comune e realizzando un elevato numero di CPU interconnesse, ciascuna con la propria memoria privata.

Le CPU in un multicomputer sono accoppiate in modo lasco (loosely coupled) e comunicano attraverso scambi di messaggi.

**Memoria cache**

Le CPU sono sempre state più veloci delle memorie. Questa differenza di prestazioni si nota quando la CPU lancia una richiesta alla memoria, ma non otterrà la parola desiderata se non dopo molti cicli di CPU. Più lenta è la memoria, più cicli dovrà attendere la CPU .

le parole di memoria usate più di frequente sono mantenute all’interno della cache. Quando la CPU necessita di una parola, la cerca nella cache e, solo nel caso in cui essa non sia presente, la richiede alla memoria centrale.

L’idea generale, quindi, prevede che quando una parola viene referenziata, la parola stessa e alcune parole vicine sono portate dalla grande e lenta memoria all’interno della cache, in modo che sia possibile accedervi velocemente in un secondo momento.

**Gerarchie di memoria**

La soluzione che viene tradizionalmente adottata per memorizzare una gran mole di dati consiste nell’organizzare gerarchicamente la memoria.

Immagine che contiene linea, diagramma, triangolo

Descrizione generata automaticamente

Nella parte alta della gerarchia si trovano i registri della CPU, ai quali si può accedere alla stessa velocità della CPU.

Successivamente vi è la memoria cache, la cui dimensione può variare da 32 KB fino ad alcuni megabyte.

La memoria centrale è il passo successivo e la sua dimensione è compresa tra 16 MB per i sistemi più economici fino a decine di gigabyte per quelli professionali.

Successivamente troviamo i dischi magnetici, la vera forza lavoro per quanto riguarda la memorizzazione permanente.

Infine ci sono i nastri magnetici e i dischi ottici utilizzati per l’archiviazione.

**RAID**

Il RAID livello 0: il disco virtuale simulato dal RAID è visto come se ognuno dei k settori fosse diviso in strip (“strisce”), con i settori da 0 a k - 1 che compongono la strip 0, i settori da k a 2k - 1 la strip 1 e così via; se k = 1 ogni strip è un settore, se k = 2 una strip è composta da due settori, ecc. L’organizzazione RAID livello 0 scrive sulle strip consecutive in modo ciclico (modalità round robin), come mostra la Figura 2.23(a) nel caso di un RAID con quattro unità. Il RAID livello 0 lavora meglio quando le richieste sono di grandi dimensioni.

Il RAID livello 1, mostrato nella Figura 2.23(b), duplica tutti i dischi, risultando così composto da quattro dischi primari e da quattro di backup. Nel caso di una scrittura ogni strip viene scritta due volte, mentre nel caso di una lettura è possibile usare entrambe le copie, distribuendo il carico di lavoro su più unità.

Il RAID livello 2 funziona sulla base di una parola o, in alcuni casi, anche sulla base di un byte. Questa organizzazione presenta però alcuni difetti: la rotazione dei dischi deve essere sincronizzata e lo schema ha senso soltanto se si utilizza un numero significativo di unità (anche con 32 dischi di dati e 6 dischi di parità l’overhead è del 19%).

Il RAID livello 3, mostrato nella Figura 2.23(d), è una versione semplificata del RAID livello 2. Il bit di parità viene calcolato per ogni parola di dati e poi scritto su un apposito disco.

Il RAID livello 4, illustrato nella Figura 2.23(e), è come il RAID livello 0, con una parità strip-per-strip scritta su un disco aggiuntivo. Se un disco si guasta è possibile ricalcolare i byte persi grazie al disco di parità. Questo schema protegge dalla perdita di un disco, ma ha prestazioni scarse quando si aggiornano piccole quantità di dati. II disco di parità può inoltre diventare un collo di bottiglia, a causa del grande carico di lavoro che pesa su di esso.

Il RAID livello 5 elimina questo collo di bottiglia distribuendo uniformemente i bit di parità su tutti i dischi, in modalità round robin, com’è mostrato nella Figura 2.23(f). Tuttavia, quando si verifica un guasto a un disco, la ricostruzione del suo contenuto è un processo complesso.

Immagine che contiene testo, diagramma, carta, modello

Descrizione generata automaticamente

**Porte logiche**

Immagine che contiene calligrafia, diagramma, schizzo, testo

Descrizione generata automaticamente

**Come implementare un circuito che realizzi una qualsiasi funzione booleana**

1. Si scrive la tabella di verità della funzione.

2. Ci si munisce di invertitori per generare la negazione di ciascun input.

3. Si utilizza una porta AND per ciascun termine il cui valore nella colonna risultato è 1.

4. Si collegano le porte AND agli input appropriati.

5. Si connettono tutti gli output delle porte AND nella porta OR.

**Circuiti integrati**

Le porte logiche non sono prodotte individualmente, ma in unità chiamate circuiti integrati, alle quali spesso ci si riferisce con i termini IC o chip.

Un IC è un quadrato di silicio di circa 5x5 mm sul quale sono state posizionate alcune porte.

I chip possono essere approssimativamente classificati in base al numero di porte che contengono:

- circuiti SSI (Small Scale Integrated): da 1 a 10 porte;

- circuiti MSI (Medium Scale Integrated): da 10 a 100 porte;

- circuiti LSI (Large Scale Integrated): da 100 a 100.000 porte.

- circuiti VLSI (Very Large Scale Integrated): più di 100.000 porte.

**Reti combinatorie**

**Multiplexer**

In logica digitale un multiplexer è un circuito con 2*n* dati di input, un valore di output e *n* input di controllo; gli input di controllo permettono di selezionare uno dei dati di input, che viene “instradato” verso l’output.

Immagine che contiene diagramma, schizzo, Disegno tecnico, Piano

Descrizione generata automaticamente

**Decodificatori** **(Decoder)**

Accetta come input un numero a *n* bit e lo utilizza per impostare a 1 una sola delle 2*n* linee di output. La Figura 3.13 mostra questo circuito nel caso *n* = 3.

Per capire in quali situazioni può essere utile questo circuito immaginiamo una piccola memoria di otto chip, da 1 MB ciascuno. Quando si fornisce alla memoria un indirizzo, si utilizzano i suoi 3 bit più significativi per selezionare uno degli otto chip.

Immagine che contiene diagramma, schizzo, Disegno tecnico, Piano

Descrizione generata automaticamente

**Comparatori**

Un altro circuito particolarmente utile è il comparatore, che permette di confrontare due stringhe di bit. Il semplice comparatore mostrato nella Figura 3.14 accetta due input, A e B, ciascuno lungo 4 bit, e genera 1 se sono uguali, mentre 0 se sono diversi. Il circuito è basato sulla porta logica XOR (EXCLUSIVE OR), che produce in output un valore 0 se i suoi input sono uguali e un valore 1 se sono diversi. Se due stringhe in ingresso sono uguali, tutte e quattro le porte XOR devono generare come risultato 0. Questi quattro segnali possono poi essere connessi a una stessa porta logica OR in modo da produrre un valore 0 quando gli input sono uguali e un valore 1 nel caso contrario.

Immagine che contiene diagramma, schizzo, Piano, Disegno tecnico

Descrizione generata automaticamente

**Array logici programmabile**

Un chip molto generale che permette di calcolare somme di prodotti è l’array logico programmabile o PLA (Programmable Logic Array), di cui la Figura 3.15 mostra un semplice esempio. Questo chip ha 12 ingressi e al suo interno questi valori vengono invertiti; quindi il numero totale di segnali di input diventa 24. Il cuore del circuito è costituito da una schiera di 50 porte AND che possono avere come input un qualsiasi sottoinsieme dei 24 segnali di input. Una matrice di 24 x 50 bit fornita dall’utente determina le connessioni desiderate tra i segnali di input e le 50 porte AND.

Immagine che contiene diagramma, testo, Disegno tecnico, Piano

Descrizione generata automaticamente

**Registri a scorrimento**

Ha otto input e otto output (Figura 3.16). Gli input sono collegati alle linee D0,..., D7, mentre l’output, corrispondente all’input traslato di un bit, risulta disponibile sulle linee S0,…, S7. La linea di controllo, C, ha valore 0 se lo spostamento deve avvenire verso sinistra, e 1 in caso contrario. Nel caso di uno spostamento a sinistra si inserisce uno 0 nel bit 7, e nel caso di uno shift a destra si inserisce un 1 nel bit 0.

Immagine che contiene diagramma, disegno, schizzo

Descrizione generata automaticamente

**Sommatori**

La Figura 3.17(a) mostra la tabella di verità per la somma di interi a 1 bit, in cui sono presenti due output: la somma dei due input, A e B, e il riporto da sommare alla successiva posizione (a sinistra). La Figura 3.17(b) rappresenta un circuito, chiamato half adder (cioè semisommatore, non tenendo conto del riporto in ingresso), capace di calcolare il bit della somma e il bit del riporto.

Immagine che contiene diagramma, testo, Piano, Disegno tecnico

Descrizione generata automaticamente

Questo circuito è in grado di sommare correttamente i bit meno significativi di due stringhe binarie, ma non è in grado di eseguire correttamente la somma degli altri bit, dato che non riesce a gestire il riporto che arriva dalle posizioni precedenti.

Per far ciò è necessario un sommatore come quello mostrato nella Figura 3.18. Osservando attentamente il circuito dovrebbe apparire chiaro che un sommatore è costruito a partire da due semisommatori.

La linea di output *Sum* vale 1 se una o tre delle linee A, e *Curry in* valgono 1. *Carry out* vale 1 se sia A sia B valgono 1 (input di sinistra della porta OR) oppure se uno dei due vale 1 e anche *Carry in* vale 1. L’unione dei due semisommatori permette di generare in output sia il bit della somma sia il bit del riporto. Per esempio, per generare un sommatore di due parole a 16 bit occorre replicare il circuito della Figura 3.18(b) per 16 volte.

Immagine che contiene diagramma, Piano, Disegno tecnico, schematico

Descrizione generata automaticamente

Per un dato bit il riporto in uscita viene utilizzato come riporto in entrata per il bit alla sua sinistra. Il riporto in entrata del bit all’estremità destra è collegato al valore 0. Questo tipo di sommatore è chiamato sommatore a propagazione di riporto, dato che nel caso peggiore, sommando 1 a 111...111 (binario), la somma non può essere completata finché il riporto non si sia propagato lungo tutta la parola, dal bit all’estremità destra fino a quello all’estremità sinistra.

**Unità aritmetico logiche**

La maggior parte dei calcolatori contiene un unico circuito in grado di effettuare le operazioni

AND, OR e somma di due parole. Di solito un circuito di questo tipo funzionante su parole a *n* bit è costruito a partire da *n* identici circuiti per le singole posizioni dei bit. La Figura 3.19 è un esempio di tale circuito, chiamato unità aritmetico logica o ALU (Arithmetic Logic Unit); esso può calcolare una qualsiasi delle quattro funzioni, A AND B, A OR B, oppure A + B, definito dalle linee F*0* e F*1* preposte alla selezione della funzione aritmetica.

Immagine che contiene diagramma, Piano, Disegno tecnico, schematico

Descrizione generata automaticamente

**Latch**

Per creare una memoria a 1 bit è necessario disporre di un circuito che in qualche modo “ricordi” i precedenti valori di input. La Figura 3.22(a) mostra come sia possibile costruire un circuito di questo tipo utilizzando due porte NOR.

Il circuito della Figura 3.22(a) è chiamato Latch SR e ha due input: S, per impostare

(Setting) il valore del latch e R per azzerarlo (Resetting). Il circuito ha anche due output, Q e , che sono complementari l’uno rispetto all’altro.

L’output di un latch non è quindi determinato unicamente dai valori di input correnti.

Per vedere come ciò avviene, assumiamo, come si verifica nella maggior parte dei casi,

che sia S sia R valgano 0. Ai fini della spiegazione assumiamo inoltre che Q = 0.

Immagine che contiene diagramma, schizzo, Piano, Disegno tecnico

Descrizione generata automaticamente

**Latch SR temporizzato**

Spesso è preferibile impedire che un latch cambi di stato se non in specifici momenti. Un

circuito che gode di questa caratteristica è detto latch SR temporizzato; per costruirlo occorre aggiungere un input, il clock, il cui valore è generalmente 0. Quando il clock vale 0 (strobe) entrambe le porte AND generano in output il valore 0, indipendentemente dai valori di S e R, impedendo quindi al latch di cambiare di stato. Quando il clock vale 1 (enable) le porte AND non bloccano più i segnali S e R che possono dunque tornare a pilotare lo stato del latch.

Immagine che contiene diagramma, Carattere, Piano, schizzo

Descrizione generata automaticamente

**Latch D temporizzato**

Un buon modo per risolvere l’ambiguità dei latch SR (causata dalla situazione S = R = 1)

è evitare che si verifichi.

La Figura 3.24 mostra un circuito che ha un solo input, D. Dato che l’input della porta AND rappresentata in basso è sempre il complemento dell’input di quella superiore, non può mai accadere che entrambi gli input valgono 1. Quando D = 1 e il clock vale 1, il latch viene portato nello stato Q = 1, mentre, quando D = 0 e il clock vale 1, il latch passa nello stato Q = 0.

In altre parole quando il clock vale 1 il valore corrente di D viene campionato e memorizzato nel latch. Questo circuito, chiamato latch D temporizzato, è una vera e propria memoria a 1 bit, in cui il valore memorizzato è sempre disponibile sulla linea Q. Per caricare in memoria il valore corrente di D occorre spedire un impulso positivo sulla linea dei clock.

Immagine che contiene diagramma, schizzo, linea, Carattere

Descrizione generata automaticamente

**Flip-flop**

In molti circuiti è necessario campionare il valore di una certa linea in un particolare istante e memorizzarlo. In questi circuiti, chiamati flip-flop, la transizione di stato non si verifica quando il clock vale 1, ma durante la transizione del clock da 0 a 1 (fronte di salita) oppure da 1 a 0 (fronte di discesa).

In questa situazione la lunghezza dell’impulso del clock non ha alcuna importanza, purché le transizioni si verifichino con sufficiente velocità.

Immagine che contiene diagramma, linea, Piano, Disegno tecnico

Descrizione generata automaticamente

**Registri**

Tra le svariate configurazioni in cui i flip-flop sono disponibili, la Figura 3.28(a) ne mostra una particolarmente semplice consistente in due flip-flop dotati dei segnali Clear e Preset. I due flip-flop non sono in relazione l’uno con l’altro.

Immagine che contiene diagramma, testo, Piano, schematico

Descrizione generata automaticamente

La Figura 3.28(b) mostra invece una configurazione diversa: un flip-flop ottale.

In questo caso gli otto flip-flop D non solo sono sprovvisti di e delle linee di reimpostazione, ma inoltre tutte le linee di clock sono collegate fra loro e pilotate dal pin 11.

**RAM e ROM**

Le RAM(Random Access Memories) sono memorie che hanno lo stesso tempo di accesso indipendentemente dalla posizione della cella. Esistono due tipi di RAM: statiche e dinamiche.

Le RAM statiche (SRAM) sono costruite utilizzando circuiti simili ai flip-flop D e hanno la proprietà di mantenere il proprio contenuto fintanto che vi è alimentazione. Sono molto veloci e i loro tempi di accesso sono usualmente dell’ordine dei nanosecondi.

le RAM dinamiche (DRAM) non usano flip-flop, ma sono composte da un array di celle, ciascuna delle quali contiene un transistor e un piccolo condensatore. Il condensatore può essere caricato o scaricato per memorizzare i valori 0 oppure 1.

**Chip di memoria non volatile**

le ROM (Read-Only Memory, “memoria di sola lettura”) che non possono essere modificate o cancellate, né intenzionalmente né accidentalmente. I dati sono inseriti durante la sua fabbricazione.

le PROM (Programmable ROM) ROM programmabile una sola volta bruciando dei fusibili collocati nelle intersezioni della matrice.

la EPROM (Erasable PROM, PROM cancellabile), i cui campi non solo possono essere programmati, ma anche cancellati. Quando si espone la piccola lente al quarzo che si trova nella EPROM a un’intensa luce ultravioletta per 15 minuti, tutti i bit assumono il valore 1. In genere le EPROM hanno la stessa organizzazione delle RAM statiche.

la EEPROM (la prima E sta per elettricamente) che possono essere cancellate applicando impulsi elettrici invece di dover inserire il chip in una camera speciale per l’esposizione alla luce ultravioletta.

Immagine che contiene testo, ricevuta, numero, schermata

Descrizione generata automaticamente

**Bus del calcolatore**

Un bus è un collegamento elettrico che unisce diversi dispositivi, ma possono essere anche interni alla CPU. Sono classificati in base alla loro funzione e ciascun tipo di bus soddisfa certi requisiti e gode di proprietà specifiche.

Immagine che contiene diagramma, Piano, Disegno tecnico, schematico

Descrizione generata automaticamente

Esempi più famosi sono: il bus ISA, il bus EISA, il bus PCI, il bus SCSI, l’Universal Serial Bus.

Immagine che contiene testo, Carattere, numero, ricevuta

Descrizione generata automaticamente

**Ampiezza del bus**

Nella progettazione dei bus maggiore è il numero di linee d’indirizzo di un bus, maggiore sarà la quantità di memoria che la CPU potrà indirizzare direttamente. Se un bus ha *n* linee di indirizzi, una CPU può indirizzare 2*n* diverse locazioni di memoria.

Il problema principale di bus larghi è che questi richiedono più spazio, a causa della maggiore quantità di fili, e quindi sono più costosi.

**Temporizzazione del bus**

I bus possono essere separati in due categorie in base alla loro temporizzazione.

**Bus sincroni**

Un bus sincrono ha una linea pilotata da un oscillatore a cristalli (clock), sulla quale un segnale consiste in un’onda quadra con frequenza compresa tra 5 e 133 MHz. Il clock viene distribuito a tutte le unità funzionali collegate al bus. Il segnale del clock scandisce le varie transizioni dei segnali e il passaggio da un ciclo di bus al ciclo successivo. Tutte le operazioni sul bus richiedono un numero intero di questi cicli, chiamati cicli di bus.

Quando la CPU vuole leggere dalla memoria:

- la CPU, nel primo ciclo di clock, fornisce l’indirizzo della parola sulle linee di indirizzo;

- dopo che le linee d’indirizzo si sono stabilizzate sui nuovi valori, vengono asserite (indica che si accede alla memoria) e (asserita per le letture e negato per le scritture);

- se la memoria impiega tempo supplementare a rispondere asserisce per segnalare alla CPU di non aspettarla. Questa azione inserisce alcuni stati di attesa (cicli di bus addizionali) finché la memoria non completi l’operazione;

- una volta che la memoria è pronta nega e mette i dati sulle linee apposite;

- la CPU legge le linee dati, memorizzando il valore in un registro;

- dovendo leggere i dati la CPU nega e .

**Bus asincroni**

Un bus asincrono non ha invece un orologio principale e quindi i cicli di bus possono avere una qualsiasi lunghezza e non devono essere necessariamente uguali nella comunicazione tra dispositivi.

Infatti, uno dei problemi del bus sincrono è proprio che qualsiasi operazione sul bus si svolge in tempi multipli del clock del bus.

**Arbitraggio del bus**

Nel caso in cui due dispositivi vogliano accedere al bus, si può incorrere in un meccanismo di arbitraggio, che può essere centralizzato o decentralizzato.

**Arbitraggio centralizzato**

In questo schema un singolo arbitro del bus determina chi sarà il prossimo master. In molte CPU l’arbitro è integrato nel chip stesso della CPU, mentre in altre è necessario utilizzare un chip separato. Il bus contiene un’unica linea di richiesta OR-cablata che in qualsiasi momento può essere asserita da uno o più dispositivi.

a differenza dei dispositivi che hanno bisogno di un accesso immediato pena la perdita di dati. In molti calcolatori questo problema viene risolto mettendo la memoria su un bus separato rispetto ai dispositivi di I/O, in modo che la CPU non debba competere con questi per l’accesso al bus.

**Arbitraggio decentralizzato**

Un calcolatore potrebbe avere per esempio 16 linee di richiesta, ciascuna con la propria priorità. Quando un dispositivo vuole utilizzare il bus asserisce la linea di richiesta. Tutti i dispositivi monitorano tutte le linee di richiesta in modo che alla fine di ciascun ciclo del bus ognuno di loro può sapere se era il richiedente con priorità più elevata e se quindi ha diritto a utilizzare il bus durante il ciclo successivo.

**Esempi di bus**

I bus sono il collante che tiene insieme i sistemi di calcolatori. In questo paragrafo daremo uno sguardo dettagliato ad alcuni modelli molto diffusi.

**Bus ISA**

Il bus del PC IBM è stato lo standard de facto dei sistemi basati su 8088.

Questo bus aveva 62 linee di segnali, tra cui 20 per gli indirizzi della memoria, 8 per i dati e una per asserire rispettivamente lettura e scrittura in memoria e lettura e scrittura sulle periferiche. Erano presenti inoltre segnali per richiedere e concedere gli interrupt e per utilizzare il DMA.

**Bus PCI**

Nel 1990 furono necessari bus sempre più performanti per il trasferimento di grandi quantità di dati in poco tempo, ed Intel ne approfittò progettando un bus con una larghezza di banda molto superiore all’EISA e lo chiamò PCI (Peripheral Component Interconnect).

A partire dal Pentium, praticamente ogni calcolatore basato su Intel, così come molti altri computer, aveva un bus PCI.

banda per l’I/O sta creando confusione nell’architettura dei PC.

Sono state proposte varie soluzioni ma quella che sembra avere più possibilità di successo è chiamata PCI Express. Questo bus non ha nulla a che fare con il precedente bus PCI, ma è stato chiamato così solo per una questione di marketing e tra l’altro non è nemmeno un vero e proprio bus.

**Architettura di PCI Express**

La caratteristica principale del bus PCI Express è la rinuncia al bus parallelo e quindi l’architettura basata sul master-slave. Con questo bus vengono fatte delle connessioni punto-punto ad alta velocità rompendo radicalmente con la tradizione ISA/PCI e adotta molte idee proveniente dal mondo delle rete locali e in particolare dalla commutazione Ethernet.

Quello che fa PCI Express è fornire un commutatore di uso generale per connettere i chip mediante collegamenti seriali.

Universal Serial Bus (USB)

* I bus Pci e Pci express sono ottimizzati per la connessione delle periferiche ad alta velocità.
* Viene standardizzato nel 98.
* Si compone di un hub principale connesso al bus di sistema dove, a sua volta si possono collegare le periferiche o altri hub.
* Hub principale quando rileva una connessione genera un interruzione per il SO che: Interroga il dispositivo per sapere che tipo di periferica, se la larghezza banda è sufficiente assegna un numero (1-127), ora è pronto per funzionare. Questo collegamento di chiama punto-punto.

Frame

* Controllo
* Isocroni
* Bulk
* Interrupt

Pacchetti

* Token
* Dati
* Handshake
* Speciali

Interfacce di I/O

* UART ( Universal Asynchronous Receiver Trasmitter) : Legge un byte dal bus e trasmette un bit alla volta su una linea seriale.
* USART ( Universal Synchronous Asynchronous Receiver Trasmitter), aggiungono alle UART la possibilità di effettuare trasmissioni sincrone
* PIO (Parallel I/O) chip per il collegamento di un disp I/O con comunicazione parallela.

Indirizzamento I/O

* Port-mapped I/O (o I/O isolato ): è necessario una linea del control bus che distingue se l’operazione dev essere eseguita in memoria oppure su I/O e sfrutta istruzioni specifiche.
* Memory-mapped I/O: Riserva uno spazio in memoria destinato all I/O e le operaz di lettura e scrittura in memoria eseguite in quello spazio di indirizzamento saranno dirottate sull I/O.

MICROARCHITTETURA

Il livello di microarchitettura fornisce una visione astratta per il livello ISA e dipende anche dagli obiettivi di costo e prestazioni della macchina.

Il microprogramma si compone di una sequenza di microistruzione che utilizzano variabili che costituiscono lo stato della macchina

Il program counter (PC) contiene il riferimento della prossima microistruzione da eseguire ed ognuno di essa si compone di:

* Codice operativo (Opcode) che identifica il tipo di istruzione
* Operando/i si cui si applicherà tale istruzione.

Il modello di esecuzione è basato sul fetch-decodifica-esecuzione.

MODALITà INDIRIZZAMENTO:

* Immediato
* Diretto
* Indiretto
* A registro
* Indiretto a registro
* Indicizzato
* A registro base
* A stack

TIPOLOGIA ISTRUZIONE

* Unarie
* Binarie
* Trasferimento dati
* Selezione e confronto
* Iterazione
* Chiamata di una procedura
* Input/Output

STORIA DEI SO

* 1° Genereazione (1945-55) valvole termoioniche: La programmazione era in linguaggio macchina oppure attraverso cablaggio complessi e nel 1950 nasce il primo supporto cartaceo per la memorizzazione ed il caricamento dei programmi
* 2° Generazione (1955-65) Transistor e sistemi batch: Metà anni 50 nasce il transistor, i primi mainframe IBM occupano intere stanze, costosi ed utilizzati solo da Università e grandi gruppi industriali.
* 3° Generazione (1965-1980) Circuiti integrati e multiprogram: Nasce IBM system 360 che utilizza i circuiti integrati ed il sistema operativo OS 360 che permette la multiprogrammazione (la zSeries è un suo discendente), adottavano la tecnica spooling per la lettura e nasce il primo sistema timesharing.
* 4° Generazione (1980-oggi) Personal Computer: IBM progetta il primo PC con il DOS, Apple progetta LISA e Machintosh il primo ambiente user-friendly, Microsoft progetta Windows e di conseguenza Apple MacOS

PROCESSI:

Il processo è l’esecuzione di un programma e ha associato:

* Uno spazio degli indirizzi dove sono memorizzati il programma eseguibile, i dati e lo stack
* Un insieme di risorse: registri, file aperti, elenco di processi correlati, allarmi in sospeso….
* Una riga all’interno della tabella dei processi

I processi possono cooperare per raggiungere un obiettivo comune ed hanno bisogno di sincronizzarsi attraverso comunicazione IPC

Ai processi possono essere inviati dei segnali in analogia agli interrupt hardware.

L’utente che ha avviato un processo ne è il proprietario.

Un processo può creare processi figli.

FILE:

I file sono il modello stratto per raggruppare insieme dati nascondendo le peculiarità dell’hardware che li memorizza

Il file system è l’ambiente che gestisce i file ed è organizzato in modo gerarchico.

L’elenco delle directory che occorre attraversare dalla radice(root) per raggiungere il file è detto percorso o path del file.

PROTEZIONE:

La protezione dei dati è una caratteristica importante dei SO, i file sono protetti mediante l’assegnazione di un codice di protezione a 9 bit costituito da tre ambiti:

* Il proprietario
* Utenti del medesimo gruppo del proprietario
* Altri utenti

Ogni ambito ha 3 bit (rws bits) che definiscono le operazioni:

* Lettura
* Scrittura
* Esecuzione

SHELL

Il SO è il codice che realizza le chiamate di Sistema

Gli editor, compilatori, assemblatori, linker e interprete non fanno parte del SO

La GUI è un applicazione che sta sopra al SO proprio come l’interprete dei comandi, ma non far parte del SO

SYSTEM CALL

I SO hanno due funzione principali:

* Forniscono astrazioni ai programmi utente
* Gestiscono le risorse del computer

Una chiamata di system è una procedura speciale che viene eseguita in modalità kernel ad esempio

Per la gestione dei processi:

* In POSIX la chiamata fork() permette di creare una replica del processo
* In UNIX ad ogni processo è assegnato uno spazio di indirizi suddivisi in tre segmenti disgiunti

Per la gestione delle directory:

* Chiamate mkdir() e rmdir() servono rispettivamente a creare una directory e cancellarne una vuota
* Link() permette di creare un riferimento ad un file o una directory
* Mount() permette di fondere insieme due file system

MICROKERNEL

Con l’approccio a strati i progettisti potevano scegliere dove finire il confine tra kernel e utente, inizialmente tutti i livelli erano nel kernel. La difettosità di un codice dipende dalla dimensione del modulo, dalla sua età e altri fattori, statisticamente 10 bug in 1000 righe di codice, l’idea di base della struttura microkernel è di incrementare l’affidabilità suddividendo il sistema operativo in piccoli moduli collaudati e di avere un solo modulo eseguito in modalità kernel

I sistemi operativi a microkernel sono utilizzati in quelle applicazioni mission critical che hanno requisiti di alta affidabilità.

CLIENT SERVER

* I server, ciascuno dei quali fornisce un servizio;
* I Client, che utilizzano questi servizi.

Questo modello è noto in letteratura come modello client-server. Spesso il livello più basso è un microkernel, ma ciò non è necessario, la loro comunicazione avviene tramite messaggi, per ottenere un servizio, un processo client costruisce un messaggio indicando la propria richiesta e lo invia al servizio appropriato.

1. [↑](#footnote-ref-1)