Storia dell'informatica - Prima parte

Luca Barra

Anno accademico 2023/2024

Indice

Capitolo 1

Storia delle architetture e dei sistemi di calcolo

Ci si potrebbe accontentare di una macchina che:

- azzera il contenuto di un registro;
- incrementare di uno il valore di un registro;
- saltare a un'istruzione specifica se due registri sono uguali;

Ma questo sarebbe insufficiente.

1.1 Sistemi di numerazione

Un *sistema di calcolo* manipola dei simboli seguendo determinate regole. I sistemi di numerazione sono stati un grande passo in avanti per l'umanità. Le quantità venivano inizialmente rappresentati da token che erano scomodi per numeri grandi. Allora, in Egitto (intorno al 3000 a. C), si sviluppa un sistema addizionale con simboli diversi per rappresentare quantità diverse. Tuttavia quel sistema non era posizionale, per cui la posizione non importa. Un altro sistema fu quello romano che era posizionale e sia additivo che sottrattivo. Intorno al 500 d. C. in india viene messo appunto un sistema addizionale in base 10 (le dita della mano) in cui il numero 0 assume importanza.

Un altro sistema fu quello sessagesimale (in base 60) inventato dai babilonesi. Si basava sulle falangi delle dita. Esso è ancora usato per gli angoli.

1.1.1 Il supporto al calcolo

La parola *zero* deriva dall'arabo *sifr*. Infatti, grazie a movimenti commerciali e invasioni i numeri passarono dall'india all'arabia all'europa. Nell'800 a. C. Mohammed ibn-Musa al-Khuwarizmi scrive alcuni trattati di algebra e aritmetica descrivendo e impiegando largamente il sistema di numerazione indiano. La diffusione europea si deve a *Fibonacci* che introduce il metodo "arabo" in Italia. Nasce così la necessita di rappresentare i numeri con l'abaco.

Nel 1600 a. C. si sviluppa l'industria bellica con la necessità di calcolare la traiettoria delle palle di cannone e nella navigazione transoceanica c'è bisogno di calcolare le carte nautiche. Allora *Nepero* sviluppa il concetto di *logaritmo* che trasforma una moltiplicazione molto grande in un addizione.

In inghilterra Edmund Gunter utilizza i logaritmi per inventare il regolo calcolatore. Nel 1623, Wilhelm Schikard progetta una macchina meccanica: l'orologio calcolatore, ma la macchina più famosa fu la *pascalina*. Inventata da Pascal, per aiutare il padre contabile. Leibniz concepi un prototipo di sistema di numerazione binario e fu un precursore della logica matematica moderna, ipotizzando che si potesse scrivere in maniera matematica il pensiero umano.

Nel 1725, Basile Bouchin utilizzo rotoli di carta perforati sui telai. Questo metodo fu migliorato da Falcon con delle schede di carta. Jacquard utilizzo delle schede metalliche. Quest'idea di usare schede perforate rimarrà nella programmazione fino agli anni '80.

Nel 1822 Thomas de Colmar costrui un aritmometro: una macchina calcolatrice portatile con le 4 operazioni con numeri fino a 12 cifre.

Figure 1.1: L'aritmometro

1.2 Charles Babbage

Charles Babbage (1792 - 1871), inglese, fu un'importante figura nella storia della matematica. Esso fu un membro di importanti società scientifiche e fu amico di Darwin e Pierre Simon de Laplace.

Tra il 1820 e il 1830 concepì una macchina (mai costruita) in grado di eseguire operazioni aritmetiche. Subito dopo concepì un'altra macchina più potenti (*analytical engine*) che utilizzava schede perforate. Per realizzarla dovette andare a cercare finanziamenti in Europa e nel 1840 venne a tenere un congresso a Torino[1], patrocinato da re Carlo Alberto di Savoia. L'intenzione di Babbage era quella di incontrare Giovanni Plana, uno scienziato molto influente. Plana mando un suo allievo, Menabrea ad ascoltare Babbage.

All'inizio del 1843 *Ada Augusta Byron*, contessa di Lovelace, traduce il lavoro di Menabrea e lo invia a Babbage. L'articolo, con le note di Ada, contiene molte idee che compariranno nelle moderne architetture. Sfortunatamente le idee di Babbage e Ada vennero dimenticate per oltre un secolo.

1.2.1 L'analytical engine

Ogni scheda perforata indicava la combinazione di colori. Utilizzava cicli per cui si potevano usare più volte le stesse schede. L'idea era quella che le operazioni venivano eseguite dal *mil* , un dispositivo complesso che funzionava come un "processore". Il mil usava *variabili*: cilindri di ottone in cui erano impilati un certo numero di dischi. Inoltre l'analytical engine poteva gestire calcoli in parallelo.

1.3 Boole e Hollerit

Nel 1854, George Boole pubblica un lavoro su quella che verrà chiamata *algebra booleana*. Fu considerato un lavoro puramente teorico fino agli anni '30 quando sarà riscoperto da Georgr Sitbitz.

Nel 1890, in occasione del censimento, venne chiesto di poter sfruttare un modo automatico. Herman Hollerit invento il sistema *elettrico di tabulazione*: una macchina in grado di elaborare rapidamente le informazioni di tutti i cittadini. Una scheda perforata conteneva tutte le informazioni relative a un singolo individuo tramite la presenza o l'assenza di fori in determinati punti. Queste schede venivano date al tabulatore che si occupava di leggere ogni scheda.

L'invenzione di Hollerit fu modificata e adattata ad altri contesti, per esempio le ferrovie, le assicurazioni, etc.

Nel 1896 Hollerit fonderà la Tabulating Machine Corporation che diverrà, nel 1924, la *International Business Machine Corporation*[2] che dominerà il mercato fino agli anni '70.

[1] Alcuni suoi disegni sono conservati all'accademia delle scienze

[2] IBM

Figure 1.2: Un operatrice

1.4 Le schede perforate

Le *schede perforate* verranno ampiamente adottate per gran parte del XX secolo. Oltre che per memorizzare dati vennero usati anche per scrivere i codici dei programmi. Infatti i programmatori usavano pacchetti di schede perforate in cui una scheda conteneva un'istruzione. Ogni pacchetto veniva letto da una macchina stampante. Per fare ciò si consegnava il pacchetto all'operatore del centro di calcolo che si occupava della gestione. Questo processo poteva durare ore o giorni. Se il programma "crashava" veniva prodotto un *core dump* con le informazioni sul contenuto di tutti i registri al momento del blocco.

1.5 I calcolatori analogici

I *calcolatori analogici*, alla fine del '800 e all'inizio del '900, devono lavorare con grandezze reali, non finite. Essi sfruttavano e combinavano diverse proprietà della materia (livelli di tensione, ingranaggi) per eseguire calcoli complessi. Tuttavia non hanno mai preso piede, in quanto erano difficili da programmare.

Il primo calcolatore analogico generale fu inventato da *Vannevar Bush*[3].

1.6 La nascita dei calcolatori digitali

Tra il 1930 e il 1950 si iniziarono a sviluppare i moderni computer.

Un *circuito digitale* (e un computer) si basa su interruttori a comando elettrico. Il primo esempio di interruttore è il *relè* che funziona come una calamita.

A essere fondamentale fu anche l'invenzione del *triodo*: un componente elettronico in grado di amplificare un segnale elettrico. Se si applica una tensione variabile, la quantità di corrente che raggiunge l'anodo sarà amplificato. Il triodo si può anche comportare come un interruttore. Inoltre, essendo il triodo principalmente elettrico è di base superiore al relè, in quanto consuma meno corrente e si rompe di meno.

Attualmente i triodi sono usati in campo musicale (più nello specifcico HiFi).

1.6.1 I primi computer

I primi computer si basarono sul relè poichè più economico. Nel 1937, George Sitbitz, comprende che i principi della logica booleana possono essere usati per la costruzione dei circuiti digitali. Con i relè costruisce il *model K*, un addizionatore a due cifre, e successivamente il *Complex Number Calculator*, che si occupava delle quattro operazioni. Nello stesso periodo, Claude Shannon[4] discusse la sua tesi di laurea proprio su i relè e la logica booleana.

[3] Inventore degli ipertesti

[4] Il padre della teoria dell'informazione

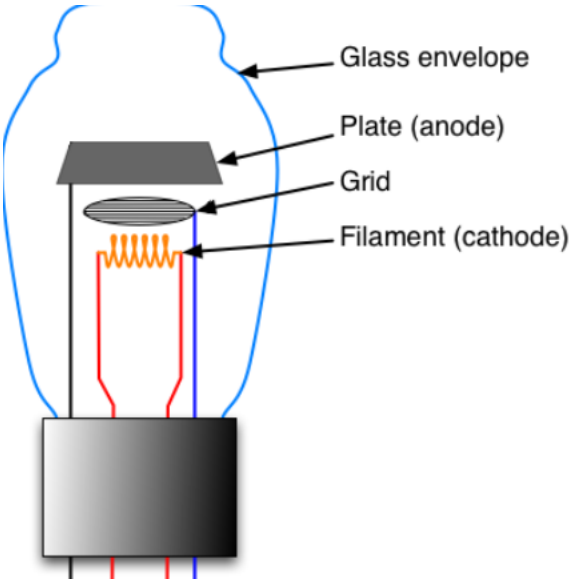


Figure 1.3: Un triodo

Konrad Zuse, un ingegnere civile, progetto tra il 1936 e il 1938 una macchina meccanica chiamata *V1*, il cui nome sarà modificato in *Z1*.

Nella Z1 iniziarono a emergere alcuni concetti:

- Memoria;
- AU: unità aritmetica (precursore della ALU);
- Lettore di schede perforate;
- CU: control unit (precursore della CPU);
- Selettore di memoria.

Helmut Schreyer dal 1939 ricevette un finanziamento e con Zuse realizzò lo Z2 (a cui seguira uno Z£) che utilizzava le valvole termoioniche.

Howard Aiken, nel 1937, volle costruire una versione dell'analitycal engine, usando i relè, creando il *Mark I* (finito nel 1943, ma quasi subito obsoleto) che rimase in funzione fino al 1959. Il Mark I usava l'*architettura Harvard* (contrapposta all'*architettura Von Neumann*). Nell'architettura Harvard il programma è memorizzato in una memoria diversa da quella dei dati.

Il Mark I fu seguito dal Mark II. Grace Murray Hopper, programmatrice del Mark II, introdusse la nozione di *bug informatico*. Il nome "bug" deriva dal fatto che Grace trovo una cimice bruciata nel Mark II che ne comprometteva il funzionamento.

L'*Atanasoff-Berry Computer* (ABC) fu il primo computer realizzato interamente con componenti elettronici. Era alimentato a tensione alternata a 60 Hz, pesava poco e occupava poco spazio, tuttavia non era un computer *Turing completo*. Per memorizzare i dati usava la carica di un condensatore[5].

Definizione 1.6.1: Turing completezza

Un computer è Turing completo se è in grado di eseguire qualunque operazione possibile a livello teorico

1.7 Intermezzo: la seconda guerra mondiale

Lo scoppio della guerra ostacola alcuni ricercatori, ma ne agevola altri. Per esempio, la Gran Bretagna aveva necessità di decifrare i messaggi criptati usati dall'esercito tedesco. I tedeschi usavano la macchina *enigma* per

[5]La DRAM è implementata con la stessa tecnica

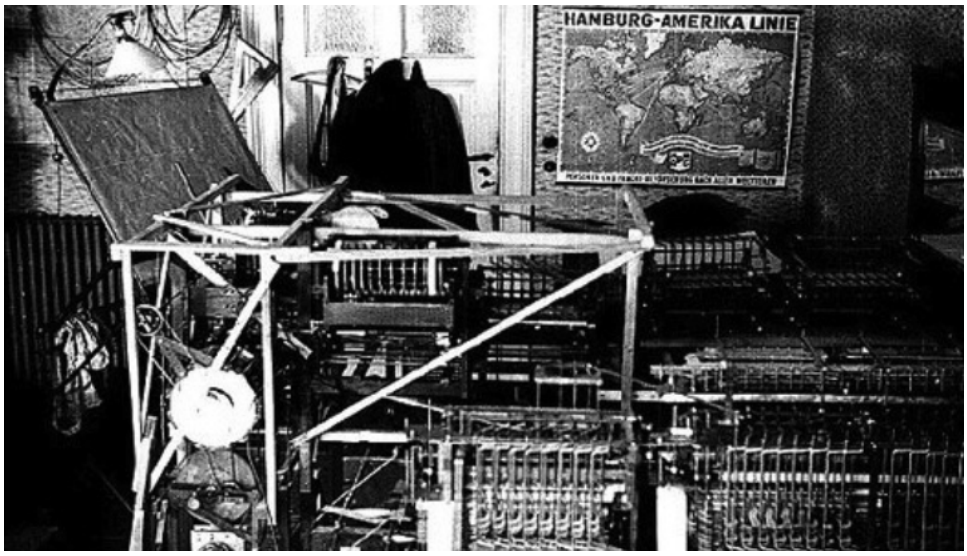
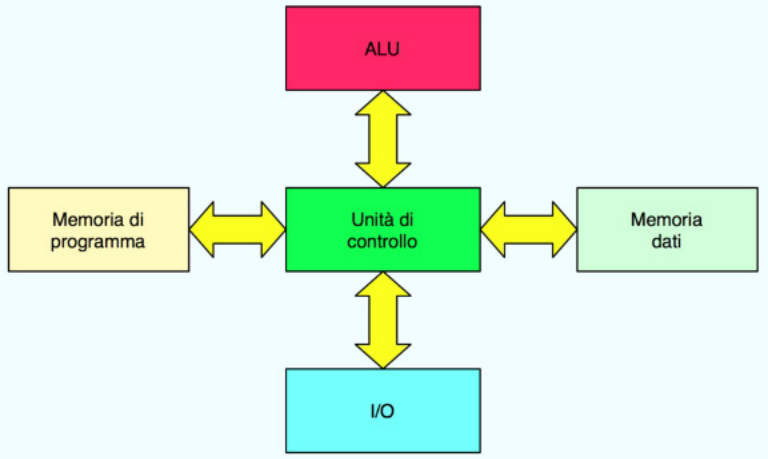


Figure 1.4: La Z1



cifrare i messaggi: aveva centinaia di migliaia di possibili cifrature, quindi non si poteva trovare il messaggio originale in tempo utile. I polacchi miserò a punto le "*bombe*" per decriptare i messaggi di enigma, mai i tedeschi aggiornarono enigma per renderlo più sicuro.

Successivamente, *Alan Turing* e Welcham miserò a punto una versione aggiornata delle bombe in grado di decriptare la nuova enigma. I tedeschi misero a punto una nuova cifratura[6]. Per decifrarla, *Tommy Flower*, mise a punto il *Colossus Mark 1.* Al Mark 1 seguirà il *Mark 2* che era 5 volte più veloce del suo precedente.

C'era però un altro problema sul fronte: calcolare la traiettoria delle palle di cannone. Si calcolavano usando molti parametri e delle tavole balistiche, diverse per ogni cannone. L'esercito statunitense utilizzava delle persone dette "computers" per effettuare questi calcoli usando delle calcolatrici e dei calcolatori analogici. Ogni tavola balistica conteneva 3000 traiettorie (per ogni singola traiettoria occorrevano 750 calcoli).

Herman Goldstine, un professore di 29 anni, collaboro con *John Mauchly* e *John Eckert* stipulando un contratto con l'esercito perlo svilupp di un Elettronic Numerical Integrator, successivamente *Elettronic Numerical Integrator and Computer* (ENIAC) che non sarà utilizzato nella seconda guerra mondiale in quanto fu finito dopo.

1.7.1 L'ENIAC

ENIAC era molto grande e presentava guasti frequenti essendo interamente elettrico. Esso non utilizzava programmi memorizzati, ma eseguiva somme (in base 10) controllate da cambi elettrici (il cui risultato era memorizzato in degli *accumulatori*) e switch che andava configurato manualmente. Sia l'input che l'output era sotto forma di schede perforate. Nel complesso ENIAC era molto veloce e, se configurato in maniera opportuna, poteva eseguire anche moltiplicazioni.

L'ENIAC aveva un ciclo di clock molto veloce (200 microsecondi) ed era programmato interamente a mano. Era estremamente complicato configurare ENIAC, poichè bisognava formalizzare, scomporre e collegare correttamente gli accumulatori. Per cui, una volta programmato si cercava di farlo lavorare con tutti i dati possibili prima di riconfigurarlo.

Eckert e Mauchly lavorarono a un altro progetto: *Electronic Discrete Variable Automatic Computer* (EDVAC) che poteva memorizzare programmi. Inoltre Eckert teorizzo un nuovo tipo di memoria: la Mercury Delay Line (MDL) per memorizzare sia dati che programmi. Era una memoria dinamica, ma sequenziale che utilizzava i suoni per memorizzare i bit.

1.7.2 L'EDVAC e John Von Neumann

Von Neumann scrisse un famoso report intitolato "First Draft of a Report on the EDVAC" in cui si ha la prima descrizione di un sistema con programmi memorizzati (l'EDVAC, appunto). Ed è proprio da questo articolo che deriva il termine "architettura di Von Neumann". Questo articolo suscito sia interesse che polemiche[7]. Il fatto che fosse firmato solo da Von Neumann, che aveva solo formalizzato matematicamente il concetto, adombro i tre inventori di EDVAC. Successivamente, in un'intervista, Goldstine disse che era stato lui a scrivere l'articolo.

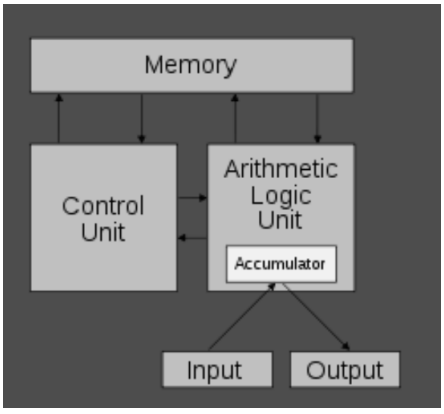


Figure 1.5: L'architettura Von Neumann

[6]Cifratura di Lorenz

[7]Il rivelare al pubblico l'ENVAC ne rese impossibile il brevetto

L'EDVAC:

- Rende più veloce l'esecuzione dei programmi;
- Rende più facile ricompilare i programmi;
- Rende più facile e veloce l'esecuzione delle istruzioni di controllo;
- Tuttavia è necessaria una memoria spaziosa ed efficiente e si ha il rischio di modificare inavvertitamente il programma stesso.

Alcune caratteristiche dell'EDVAC:

- Ogni istruzione da 44 bit (non esisteva ancora il byte) era suddivisa in 5 campi:
  - 4 bit per l'operazione;
  - 10 bit X 3 per l'indirizzo dove reperire gli operandi e l'indirizzo dove salvare in memoria il risultato;
  - 10 bit per l'indirizzo della prossima istruzione.
- Erano disponibili solo 12 istruzioni:
  - le 4 operazioni aritmetiche (più moltiplicazione e divisione con arrotondamento);
  - il salto incondizionato;
  - lo shift;
  - lettura e scrittura sul nastro perforato;
  - lettura da console;
  - stop.
- Non esisteva il concetto di memoria secondaria permanente (si usavano le Delay lines).

Note:-

L'EDVAC rimase operativo per circa 10 anni.

1.7.3 L'eredità dell'EDVAC

- L'EDVAC fu il primo computer a usare la memoria centrale per memorizzare programmi e dati;
- L'EDVAC fu il primo computer a usare il concetto di programma memorizzato;
- Si ha la necessità di mettere a punto una memoria adeguata alle necessità dei circuiti logici;
- Si capisce che l'efficienza dei computer non dipende solo dalla velocità di calcolo, ma anche dalla velocità di accesso alla memoria.

1.8 Il post-EDVAC

Il primo computer a programma memorizzato fu il Manchester Small Scale Experimental Machine (SSEM). Esso utilizza i Williams-Kilburn Tube, ossia una sorta di memoria ad accesso diretto (RAM). Dato che era necessario un refreshing era più propriamente una DRAM (come le Mercury Delay Lines). Il Williams-Kilburn Tubes era formato da un tubo catodico (come i vecchi televisori).

Caratteristiche del SSEM:

- 32 parole da 32 bit;
- Un registro da 32 bit conteneva l'istruzione in esecuzione;
- Un registro da 32 bit veniva usato come accumulatore;
- Pesava solo una tonnellata;
- La ALU eseguiva solo sottrazioni e negazioni (le altre operazioni erano implementate via software).

Per questa macchina furono scritti solo 3 programmi (di cui uno da Turing).

Successivamente, nel 1949, venne costruito il Manchester Mark 1 che era un computer a tutti gli effetti. Esso fu commercializzato come Ferranti Mark 1.

Un'altra macchina, figlia dell'EDVAC, fu il Whirlwind, sviluppato al MIT. Nel Whirlwind si hanno due principali innovazioni:

- fu il primo computer a operare in parallelo su 16 bit (usando opportune unità logiche in parallelo). Questo lo rendeva più veloce anche grazie all'introduzione della memoria a nucleo magnetico (core memory);
- il microprogramma, ossia un programma che controlla il funzionamento del computer. Inoltre si intoducevano micro-istruzioni condizionali.

Dal 1946 al 1952 si ebbe un rapido sviluppo dei computer, tramite conferenze e pubblicazioni. In questi anni il grande pubblico iniziò a conoscere i computer, anche tramite articoli del New York Times. In uno di questi articoli, Turing, osservò che era necessario comprendere le potenzialità di queste macchine.

Impatti culturali:

- Nel 1952 viene usato UNIVAC I per predire l'esito delle elezioni presidenziali;
- Si espande la consapevolezza dei computer.

1.8.1 I transistor

Il 23 Dicembre 1947, William Shockley, John Bardeen e Walter Brattain, inventarono il transistor. I transistor rappresentano un'evoluzione delle valvole termoioniche. I transistor sono più piccoli, più veloci e più affidabili.

Un transistor è costruito con un semiconduttore (silicio o germanio) che può essere drogato con impurità per aumentarne la conducibilità (P se con cariche positive, N se con cariche negative). Un transistor è composto da tre strati drogati in modo alternato (PNP o NPN). Ai tre strati viene applicata una tensione che permette di controllare la corrente che passa tra due strati.

Caratteristiche dei transistor:

- Possono essere usati come interruttori;
- La corrente fluisce tra emettitore e collettore (due dei tre strati);
- Se usati come switch, riescono a commutare più velocemente delle valvole termoioniche;
- Funzionano a frequenze più alte;
- Consumano meno energia.

Note:-

Velocemente sostituirono le valvole termoioniche.

1.8.2 Le memorie negli anni '50

La MDL e i WKT si rivelerono inadeguati per via della loro lentezza. Per cui, nei primi anni '50, si svilupparono nuove memorie:

⇒ *Nastri magnetici*;

⇒ *Rotating drum memory*;

⇒ *Magnetic core memory.*

E nella seconda metà degli anni '50 si svilupparono gli hard disk (HDD).

Note:-
Ma doveva ancora emergere la distinzione tra memoria primaria e secondaria.

L'UNIVAC I (UNIversal Automatic Computer I), progettato da Eckert e Mauchly, fu il primo computer commerciale. Esso fu venduto a numerose aziende e agenzie governative (in tutto 46 unità). Come memoria principale usava una MDL, ma come memoria di massa adottava un nastro magnetico (UNISERVO).

Nel 1952 inizia la commercializzazione dei Mainframe IBM. Dal 1956 questi computer verranno costruiti con la tecnologia a transistor (domineranno il mercato fino alla metà degli anni '70).

Note:-
Il termine Mainframe deriva dal fatto che questi computer erano così grandi che occupavano un'intera stanza.

I Mainframe IBM 700/700 erano divisi in base all'impiego:

- Applicazioni scientifiche e ingegneristiche;
- Applicazioni commerciali: manipolavano principalmente stringhe.

Queste due categorie erano tra di loro incompatibili (avevano diversi ISA).

1.8.3 Memorie a nucleo magnetico

Le memorie a nucleo magnetico (core memory) erano molto più veloci delle MDL e dei WKT. Furono usate tra il 1955 e il 1975. Erano costituite da un insieme di anelli di materiale ferromagnetico (ferrite) che potevano essere magnetizzati in due direzioni. L'orientamento del campo indicava il valore del bit. Tuttavia erano molto costose e non erano adatte per l'archiviazione di grandi quantità di dati. Il "core dump" era un'operazione che permetteva di leggere il contenuto della memoria dopo un crash.

1.8.4 IBM 650

L'IBM 650 fu il primo computer a essere venduto in grandi quantità (2000 unità). Era un computer relativamente economico (200.000 dollari) e fu usato per applicazioni commerciali. Venne anche venduto alle università.

Questo computer era "general purpose" e ebbe un utilizzo didattico per l'insegnamento della programmazione. Viene spesso annoverato come l'antenato del PC. Oltre a questo era accessibile direttamente dall'utente (non era necessario un operatore).

Funzionava a tubi catodici con memoria a tamburo rotante. Nel 1959 venne commercializzato la versione a transistor (IBM 650-T).

Un caratteristica peculiare di questo computer era che i numeri erano rappresentati in forma bi-quinaria: ci vogliono 6 bit per rappresentare una cifra tra 0 e 9. Le istruzioni macchina avevano la forma xx yyyy zzzz:

- xx: operazione;
- yyyy: indirizzo della locazione di memoria;
- zzzz: indirizzo della prossima istruzione.

Capitolo 2

Storia dei linguaggi di programmazione

2.1 Introduzione

L'idea di "linguaggio di programmazione" emerge per far fare al compilatore determinati compiti. Ogni computers, per quanto sofisticato comprende solo il concetto di bit, ma per gli esseri umani è difficile esprimersi in quei termini. Per questo motivo si è pensato di creare un linguaggio che fosse più vicino alla nostra comprensione. Con il tempo si sono succedute molte idee e c'è stata una selezione naturale dei linguaggi. Alla fine i linguaggi veramente importanti sono poco più di una decina.

Note:-

Ai giorni nostri si da per scontata l'idea di linguaggio, ma neggli anni '40, Von Neumann affermava di non vederne l'utilità.

All'inizio si riteneva che nessuno avrebbe deciso di scrivere un programma in un linguaggio perchè sarebbe stato troppo lento rispetto allo scrivere un programma in assembler, ma oggi quest'idea è superata per via dei sempre più efficienti compilatori.

Ripercorrendo la storia dei sistemi di calcolo si ha una prima idea con l'analytical engine di Babbage, con Ada Lovelace che scrive il primo programma per questa macchina. Successivamente con ENIAC si ha il primo computer elettronico (che poteva essere riconfigurato), ma il primo linguaggio di programmazione è il Plankalkul di Konrad Zuse, che però non è mai stato implementato. Nel Mark I Di Aiken le istruzioni erano codificate su nastro perforato. Tra il '43 e il '45 Goldstine e Von Neumann sviluppano il concetto di "flow chart" in cui si ha "=" interpretato come assegnamento. Nel '48 il Manchester SSEM usa a 32 switch per stabilire il valore di un bit. Nell'EDVAC, i bit che componevano il programma (scritto in linguaggio macchina) e i dati, tutti rappresentati in binario, erano inseriti uno a uno nelle Mercury Delay Lines.

Definizione 2.1.1: Initial order

Nel EDSAC nasce per la prima volta l'idea "initial order": il codice di una determinata istruzione era associato a una lettera assegnata in modo mnemonico (per esempio "S" significava subtract). Un istruzione macchina era costituita da un tre caratteri di 5 bit ciascuno.

Note:-

Nell'initial order il programma era codificato su un nastro perforato. Era una sorta di antenato del linguaggio assembler.

In sostanza:

- Alla fine degli anni '40 c'erano poche decine di programmatori e meno di 20 computers;
- Non esistevano corsi/tutorial di programmazione;
- Erano stati scritti pochi programmi.

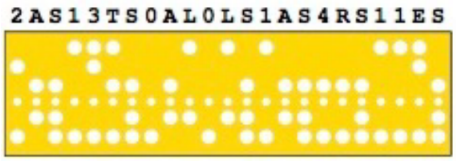


Figure 2.1: Initial order

Definizione 2.1.2: Short code

Lo short code viene implementato nel '49 da Maurice Wilkes. Si tratta di un linguaggio che permette di scrivere programmi in modo più semplice. Veniva assegnato uno specifico numero a 6 bit per indicare variabili e simboli di un'equazione. Si scrisse anche un programma per il calcolo delle equazioni da calcolare (una sorta di primitivo interprete).

Tra il 1948 e il 1950 Haskell Curry sviluppa la programmazione strutturata. Tuttavia Curry non considerava la fase di analisi sintattica

2.2 Gli anni '50

Definizione 2.2.1: For

Il costrutto for viene introdotto nel '51 da Rutishaur. La sua idea permetteva di generare codice rilocabile.

Nel 1950 l'italiano Corrado Böhm concepisce un linguaggio di alto livello e un metodo di traduzione in linguaggio macchina. Nel suo lavoro:

- Vengono introdotti "if then else" e "goto";
- Gli statement di assegnamento;
- Le subroutine;
- Un compilatore scritto nello stesso linguaggio dei programmi che deve tradurre.

Il compilatore di Böhm genera codice proporzionalmente al numero di passi da eseguire. Inoltre il linguaggio di Böhm è universale (tuttavia è solo su carta).

2.2.1 AUTOCODE

Definizione 2.2.2: AUTOCODE

Il primo compilatore degno di questo nome viene svippato nel 1953 da Alick Glennie che noto che si poteva usare lo stesso computer per tradurre un programma e far girare il programma sullo stesso computer. AUTOCODE era complicato e il suo compilatore era composto da 750 istruzioni.

AUTOCODE:

- Rendeva più veloce e semplice la programmazione;
- Costituiva una perdita di efficienza del 10% rispetto al linguaggio macchina.

Ma il lavoro di Glennie non ebbe il successo sperato poichè in quegli anni il focus non era tanto sullo scrivere un programma ma sul fatto che i calcolatori si rompessero continuamente.

Negli anni '50 si sviluppa l'idea di pseudo-codice e si parlava di automatic coding invece che di compilatori.

2.2.2 Gli anni dal '54 al '56

In queglu anni si stava tenendo il SIMATIC (Symposium on the Mechanization of Thought Processes) in cui si discuteva delle proprie scoperte.

- Nel 1954 si sviluppa il primo assembler moderno, il SOAP;
- Tra il 1954 e il 1956 alla Boeing Airplane Company di Seattle viene sviluppato il sistema BACAIC, in cui espressioni algebriche vengono tradotte in subroutine di linguaggio macchina;
- Enton Elsworth lavora ad un sistema per la traduzione di equazioni algebriche in linguaggio macchina (dell'IBM 701), e chiama il suo sistema Kompiler;
- Viene messo a punto ADES, il primo linguaggio di programmazione imperativo;
- Nel 1956 , per IBM 650, viene sviluppato il compilatore IT.

Definizione 2.2.3: IT

IT funzionava in due fasi:

1. Veniva generato codice assembler intermedio;
2. Da quel codice veniva generato codice macchina

IT permetteva di scrivere in un linguaggio semplice con un'implementazione efficiente.

2.2.3 Il FORTRAN

Definizione 2.2.4: FORTRAN

Nel 1957 il FORTRAN fa la sua comparsa. All'inizio del 1954 John Backus, Harlan Herrick e Irving Ziller iniziano a lavorare su un linguaggio di programmazione. In quegli anni i programmi venivano scritti o in assembler o in linguaggio macchina, per cui non si pensava che il FORTRAN potesse avere successo.

Specifiche:

- Avrebbe dovuto essere facile scrivere programmi in FORTRAN e sarebbe dovuto essere efficiente;
- Avrebbe dovuto essere facile da imparare;
- Non doveva essere svincolato dall'hardware (in quanto linguaggio di IBM).

Il manuale del FORTRAN aveva una grafica professionale, ma era pieno di errori e incompleto. Tuttavia il FORTRAN influenzo la maggior parte dei linguaggi successivi e fino agli anni '70 fu utilizzato come standard per applicazioni scientifiche.

2.2.4 LISP

Definizione 2.2.5: LISP

Nel 1958 fa il suo debutto il LISP. Fu il primo dei linguaggi funzionali. Nasce per la manipolazione di espessioni simboliche con l'uso del concetto di "ricorsione". Inoltre il LISP usava sia liste che alberi. Offriva un garbage collector e un sistema di tipi dinamico. Permetteva la meta-programmazione.

- Viene usata una notazione polacca (prefissa);
- Alcuni operatori potevano venire implementati direttamente in linguaggio macchina.

Note:-

Alcune parti di LISP erano codificate in FORTRAN

In LISP tutto veniva rappresentato da liste concatenate. La lista concatenata a destra è la rappresentazione interna dei dati.

2.2.5 COBOL

Definizione 2.2.6: COBOL

Il COBOL (COmmon Business Oriented Language) fu concepito per applicazioni di tipo amministrativo e commerciale. Il COBOL aveva una sintassi "english-like", ciò lo rendeva facilmente comprensibile. Esempio:

ADD A TO B GIVING C

Il COBOL venne sviluppato dal CODASYL (Conference on Data Systems Languages) e il DoD (Department of Defense) USA obbligo le compagnie produttrici di computer a fornire il COBOL nelle loro installazioni.

Corollario 2.2.1 La sintassi del COBOL

La sintassi del COBOL fu il risultato di un processo decisionale influenzato da FLOW-MATIC (di Grace Hopper) e dall'IBM COMTRAN (di Bob Bemer)[a].

[a] Molto marginalmente.

Gli obiettivi:

- doveva essere portatile;
- doveva essere efficiente;
- doveva essere facilmente comprensibile e utilizzabile.

Ma il COBOL non fu recepito bene dalla comunità informatica: non si consideravano i programmatori COBOL dei veri programmatori. Inoltre il COBOL non era adatto per applicazioni scientifiche. Ai giorni nostri il COBOL è quasi del tutto scomparso.

2.3 Gli anni '60

2.3.1 ALGOL 60

Definizione 2.3.1: L'ALGOL 60

L'ALGOL 60 (ALGOrithmic Language 1960) fu frutto di una collaborazione tra l'ACM (Association for Computing Machinery) e l'IFIP (International Federation for Information Processing). L'ALGOL 60 era molto legato all'hardware sottostante.

Note:-

Tony Hoare[a] noto che l'ALGOL 60 era molto più avanti rispetto ai linguaggi dell'epoca.

[a] Inventore del Quicksort e del puntatore NULL

ALGOL 60 introduce:

- **call by value** (per i parametri);
- **call by name** (per le variabili);
- **call by reference** (per i parametri, passaggio di un puntatore);

L'ALGOL è il primo linguaggio a richiedere esplicitamente il tipo delle variabili, Inoltre viene introdotta la distinzione tra assegnamento e uguaglianza. Oltre a qeusto si introducono le prime primitive di controllo strutturate: if then else, while loop e for. Si introducono anche i blocchi di istruzioni:

BEGIN ... END

L'ALGOL per via del pesante utilizzo di GOTO consentiva programmazione non strutturata. Ciò rende il codice meno leggibile.

Definizione 2.3.2: Spaghetti code

Spaghetti code indica codice non strutturato e "ingarbugliato", difficilmente leggibile.

2.3.2 La Backus-Naur Form (BNF)

Definizione 2.3.3: Backus-Naur Form (BNF)

La BNF è una notazione per grammatiche context free. Viene usata per descrivere la sintassi dei linguaggi di programmazione e, essendo un metà linguaggio, può essere usata per descrivere se stessa.

Note:-

Tutto ciò viene spiegato nel corso "Linguaggi formali e traduttori"

Definizione 2.3.4: Panini Backus Form

La Panini Backus Form è una versione ridotta della BNF.

Corollario 2.3.1 | parsificatori

I parsificatori sono programmi che leggono un testo e lo analizzano per determinare la sua struttura grammaticale. ALGOL porto alla creazione di un parsificatore (LL). Nel 1965, Donald Knuth sviluppa un altro parsificatore (LR), ma solo alla fine degli anni '60 verranno messi a punto dei parsificatori efficienti.

2.3.3 BASIC

Definizione 2.3.5: Basic (Beginner's All-purpose Symbolic Instruction Code)

Il BASIC nasce nel 1964 da un progetto di John Kemeny e Thomas Kurtz. Il BASIC era un linguaggio di programmazione semplice e facile da imparare. Il suo obiettivo era quello di consentire a studenti, non di informatica, di scrivere programmi.

Note:-

Il BASIC inizia a diffondersi negli anni '70 con la diffusione dei "micro-computers". In alcuni casi il BASIC diventava l'ambiente di lavoro del PC (un po' come la shell).

Il BASIC era sufficientemente ad alto livello da poter essere utilizzato da chiunque ed ebbe diffusione come linguaggio didattico. Ma anche il BASIC fu guardato con sufficienza da molti informatici: Dijstra sosteneva che fosse inutile insegnare a programmare a gente che era stata esposta al COBOL o al BASIC.

Definizione 2.3.6: Visual BASIC

Il visual BASIC fu introdotto da Microsoft nel 1991 sebbene fu utilizzato molto poc a causa della diffusione di linguaggi più potenti.

Altre varianti furono:

- Altair BASIC;
- Game BASIC (poi ridenominato Integer BASIC);
- Applesoft BASIC.

2.3.4 La questione del GOTO

Teorema 2.3.1 Bohm-Jacopini

Il teorema ha diverse formulazioni equivalenti, ma sostanzialmente asserisce che qualsiasi funzione computabile può essere calcolata da un programma costituito esclusivamente da una combinazione di:

- Sequenze di istruzioni eseguite una dopo l'altra;
- Istruzioni di selezione (del tipo if then else);
- Istruzioni di iterazione (del tipo while do).

Note:-

In sostanza il teorema dimostrava che il GOTO poteva essere messo da parte.

Nel 1968, Dijstra critica l'uso del GOTO in una lettera alle Communication of the ACM. Successivamente, nel 1974, Knuth mostra che in alcuni casi il GOTO era la scelta migliore. Ne manuale di Brian Kernigham e Dennis Ritchie osservano che il GOTO era necessario per gestire alcuni casi di errore.

2.3.5 Il Simula

Definizione 2.3.7: Simula I

Il Simula I nasce nel 1966 da un progetto di Kristen Nygaard e Ole-Johan Dahl. Il Simula I è il primo linguaggio ad oggetti. Il suo scopo principale era quello di gestire simulazioni.

Note:-

Dahl e Nygaard si ispirarono al concetto di record class introdotto da Hoare nell'ALGOL 60.

Definizione 2.3.8: Simula 67

Il Simula 67 è la seconda versione del Simula. Il Simula 67 è il primo linguaggio ad oggetti a essere usato in ambito commerciale. Il Simula 67 introduce il concetto di classe e di sottoclasse. Le istanzee di una classe sono dette oggetti e l'operazione new permette di creare un oggetto. In Simula 67 erano anche presenti i puntatori (chiamati ref) ai record di attivazione (oggetti).

Note:-

Per questi motivi si introduce un meccanismo di garbage collection.

Il Simula 67 introduce anche la data abstraction e l'ereditarietà. Uno degli utilizzi del simula era la modellazione di sistemi concorrenti.

2.4 Gli anni '70

2.4.1 Pascal

Definizione 2.4.1: Pascal

Il Pascal nasce nel 1970 da un progetto di Niklaus Wirth. Può essere considerato un successore dell'ALGOL. Vengono introdotti i puntatori e il tipo CHAR. Il Pascal incoraggia la costruzione di procedure chiare, usa strutture dati dinamiche, call by value e call b reference. Un'altra caratteristica era il suo sistema di tipi molto ricco, ma molto rigido[a].

[a] Con type checking in fase di compilazione

Obiettivi:

- Semplicità;
- Chiarezza;
- Efficienza;
- Scrittura di programmi complessi.

Il compilatore del Pascal non generava direttamente codice macchina, ma generava un codice intermedio: il P-Code. Il P-Code era un linguaggio assembly portabile. Il P-Code veniva interpretato da una macchina virtuale. Verso la fine degli anni '70 il Pascal era ormai in uso e insegnato nelle università.

L'Educational Testing Service (ETS), rese il Pascal il linguaggio standard per i test AP (Advanced Placement). Nel 1983 Hejlsberf sviluppa il Turbo Pascal, un compilatore per PC. Il Turbo Pascal era un compilatore molto efficiente e permetteva di scrivere programmi in Pascal in modo molto veloce (diventò lo standard dei PC).

Apple utilizzo il Pascal come linguaggio di riferimetno per il suo sistema operativo così come Microsoft. Tuttavia il Pascal declino in favore di C e di UNIX.

2.4.2 Il Prolog

Definizione 2.4.2: Il Prolog

Il Prolog (PROgrammation en LOGique) nasce nel 1972 da un progetto di Alain Colmerauer e Philippe Roussel. Il Prolog è un linguaggio dichiarativo basato sulla logica del primo ordine. Il Prolog è un dimostratore automatico di teoremi.

Note:-

I dimostratori automatici di teoremi sono utilizzati nel corso di "Metodi formali per l'informatica" e nella parte da +3 CFU del corso di "Linguaggi e paradigmi".

2.5 Il C

2.5.1 La nascita del C++

Definizione 2.5.1: Il C++

Il C++ nasce nel 1979 da un progetto di Bjarne Stroustrup. Si tratta di un'evoluzione del C orientata all'object-oriented.

Note:-

Nel 2001 viene rilasciato il D, un'evoluzione del C++ influenzata da Java, Ruby e Python.

2.5.2 Objective-C e C-Sharp

Definizione 2.5.2: Objective-C

L'Objective-C nasce nel 1983 da un progetto di Brad Cox e Tom Love. Si tratta di un'evoluzione del C orientata all'object-oriented.

Definizione 2.5.3: C-Sharp

Il C-Sharp nasce nel 2000 da un progetto di Microsoft. Si tratta di un'evoluzione del C orientata all'object-oriented. È stato sviluppato per la piattaforma .NET.

2.6 Gli anni '90: il Web e Java

Nel 1993 avvenne un evento che cambiò il mondo dell'informatica: il World Wide Web. Al NCSA (National Center for Supercomputing Applications) viene sviluppato il Mosaic, il primo browser grafico. Il Mosaic rendeva le pagine web più accessibili e più facili da creare. Questo browser metteva a disposizione degli utenti una enorme quantità di informazioni.Negli anni '90 nascono anche le applet: piccoli programmi che venivano eseguiti all'interno del browser il cui linguaggio era WORA (Write Once Run Anywhere). Questo linguaggio diventerà poi il Java.

Definizione 2.6.1: Java

Il Java nasce a partire dal 1990 da un progetto di Sun Microsystems[a]. Fu rilasciato nel 1995 insieme al browser HotJava (per eseguire le applet). Successivamente tutti gli altri browser implementarono una Java Virtual Machine.

[a] Azienda che sviluppò Solaris e NFS.

2.7 Verso il 21esimo secolo : i linguaggi di scripting

Definizione 2.7.1: Linguaggio di scripting

Un linguaggio di scripting è un linguaggio di programmazione interpretato che viene utilizzato per scrivere script. Gli script sono programmi che vengono eseguiti da un interprete. Per esempio il linguaggio bash è un linguaggio di scripting.

Note:-

In linea di principio qualsiasi linguaggio può essere usato come linguaggio di scripting, ma in pratica richiederebbero un interprete.

Linguaggi di scripting general purpose:

- coniugano le caratteristiche dei classici linguaggo di programmazione;
- prototipazione veloce.

Definizione 2.7.2: Perl

Il Perl nasce nel 1987 da un progetto di Larry Wall come linguaggio di scripting per l'elaborazione di testi e file. È usato per scrivere script per il web, per l'amministrazione di sistemi e per l'elaborazione di testi.

Note:-

Di recente, il Perl, viene insegnato nel corso di "Bioinformatica" alla magistrale.

2.8 Digressione: altri linguaggi

Definizione 2.8.1: Python

Il Python nasce nel 1991 da un progetto di Guido van Rossum. Possiede caratteristiche object-oriented e funzionali e ha una sintassi semplice e chiara.

Definizione 2.8.2: PHP

Il PHP (PHP: Hypertext Preprocessor) nasce nel 1994 da un programma di Rasmus Lerdorf. È un linguaggio di scripting per la programmazione web.

Definizione 2.8.3: JavaScript

Il JavaScript nasce nel 1995 (sviluppato in 10 giorni) da un progetto di Brendan Eich. È un linguaggio di scripting per la programmazione web. Con JavaScript nasce una guerra tra Netscape e Internet Explorer.

Definizione 2.8.4: Ruby

Il Ruby nasce nel 1995 da un progetto di Yukihiro Matsumoto. È un linguaggio di scripting orientato agli oggetti che doveva inglobare le caratteristiche migliori:

- la praticità del Perl;
- orientato agli oggetti cone lo Smalltalk;
- la semplicità del Lisp.

Capitolo 3

Storia dei sistemi operativi

3.1 Prima dei sistemi operativi

Il concetto di programma negli anni '40 era appenna accennato e molto diverso da quello attuale. Per esempio, l'ENIAC, il primo computer elettronico, era programmato tramite cablaggi elettrici. Il concetto di programma memorizzato in memoria centrale è stato introdotto da John von Neumann nel 1945 (con l?EDVAC e le istruzioni macchina). Ma questa era una procedura manuale, a quei tempi non si pensava minimamente al farsi aiutare da un programma per inserire un programma nel computer.

Definizione 3.1.1: Sistema operativo

Un sistema operativo è un programma che aiuta a "far girare" gli altri programmi.

Le cose iniziaro a cambiare dal 1955. Con il tempo i sistemi operativi sono diventati sempre più complessi e sofisticati. Uno dei compiti storici degli S. O. era quello di rendere semplice agli utenti l'utilizzo di un computer.

Storicamente molti dei concetti ancora usati nei moderni S. O. (paginazione della memoria, memoria virtuale, etc.) nascono dalla scarsità di risorse dei primi computer.

3.2 Fase 1 : Job by Job / Open Shop

- I computer erano usati solo dai loro progettisti e dai loro collaboratori;
- Scrivere un programma veniva svolto in più passi:
  1. Scrivere il programma su carta;
  2. Trasferire il programma su schede perforate;
  3. Nello slot prenotato il programmatore si recava nella sala del computer;
  4. Inseriva le schede perforate nel computer (se il computer non era guasto);
  5. Faceva partire il programma;
  6. Potevano essere lette le celle di memoria per seguire l'esecuzione del programma;
  7. L'output poteva essere stampato o convertito in schede perforate
- Se un programma andava storto era difficile il debugging;
- Si introduce la figura professionale dell'operatore addetto alla sala macchine che stampava la fotografia della memoria in caso di comportamenti anomali;
- Si caricava un solo programma alla volta;
- A metà degli anni '50 nascono i primi assembler.

3.3 Fase 2 : Sistemi batch

-