

Sommario

Prefazione	XVII
Prefazione all'edizione italiana	XXI
Capitolo 1 Introduzione	
1.1 Approccio strutturale	2
1.1.1 Linguaggi, livelli e macchine virtuali	2
1.1.2 Attuali macchine multilivello	4
1.1.3 Evoluzione delle macchine multilivello	8
1.2 Pietre miliari nell'architettura dei computer	13
1.2.1 Generazione zero – Computer meccanici (1642-1945)	13
– 1.2.2 Prima generazione – Valvole (1945-1955)	16
1.2.3 Seconda generazione – Transistor (1955-1965)	19
1.2.4 Terza generazione – Circuiti integrati (1965-1980)	22
1.2.5 Quarta generazione – Integrazione a grandissima scala (1980-?)	24
1.2.6 Quinta generazione – Computer a basso consumo e computer invisibili	27
1.3 Tipologie di computer	29
1.3.1 Forze tecnologiche ed economiche	29
1.3.2 Tipologie di computer	31
1.3.3 Computer usa e getta	32
1.3.4 Microcontrollori	34
1.3.5 Dispositivi mobili e da gioco	36
1.3.6 Personal computer	37
1.3.7 Server	37
1.3.8 Mainframe	40
1.4 Esempi di famiglie di computer	40
1.4.1 Introduzione all'architettura x86	41
1.4.2 Introduzione all'architettura ARM	46
1.4.3 Introduzione all'architettura AVR	49
1.5 Unità metriche	50
1.6 Organizzazione del libro	51
Capitolo 2 Organizzazione dei sistemi di calcolo	
2.1 Processori	55
2.1.1 Organizzazione della CPU	56

2.1.2 Esecuzione dell'istruzione	58
2.1.3 RISC contro CISC	62
2.1.4 Principi di progettazione dei calcolatori moderni	63
2.1.5 Parallelismo a livello d'istruzione	65
2.1.6 Parallelismo a livello di processore	69
2.2 Memoria principale	74
2.2.1 Bit	74
2.2.2 Indirizzi di memoria	75
2.2.3 Ordinamento dei byte	76
2.2.4 Codici correttori	78
2.2.5 Memoria cache	82
2.2.6 Assemblaggio e tipi di memoria	86
2.3 Memoria secondaria	87
2.3.1 Gerarchie di memoria	87
2.3.2 Dischi magnetici	88
2.3.3 Dischi IDE	92
2.3.4 Dischi SCSI	94
2.3.5 RAID	95
2.3.6 Dischi a stato solido	99
2.3.7 CD-ROM	101
2.3.8 CD-registrabili	105
2.3.9 CD-riscrivibili	108
2.3.10 DVD	108
2.3.11 Blu-Ray	111
2.4 Input/Output	111
2.4.1 Bus	111
2.4.2 Terminali	115
2.4.3 Mouse	121
2.4.4 Controller per videogiochi	123
2.4.5 Stampanti	125
2.4.5 Apparecchiature per le telecomunicazioni	130
2.4.6 Macchine fotografiche digitali	139
2.4.7 Codifica dei caratteri	141
2.5 Riepilogo	146

Capitolo 3 Livello logico digitale

3.1 Porte logiche e algebra di Boole	151
3.1.1 Porte logiche	152
3.1.2 Algebra di Boole	154
3.1.3 Implementazione delle funzioni booleane	156
3.1.4 Equivalenza di circuiti	158

3.2 Circuiti logici digitali elementari	162
3.2.1 Circuiti integrati	162
3.2.2 Reti combinatorie	163
3.2.3 Circuiti per l'aritmetica	169
3.2.4 Clock	173
3.3 Memoria	174
3.3.1 Latch	175
3.3.2 Flip-flop	177
3.3.3 Registri	180
3.3.4 Organizzazione della memoria	180
3.3.5 Chip di memoria	183
3.3.6 RAM e ROM	186
3.4 Chip della CPU e bus	191
3.4.1 Chip della CPU	192
3.4.2 Bus del calcolatore	194
3.4.3 Ampiezza del bus	196
3.4.4 Temporizzazione del bus	198
3.4.5 Arbitraggio del bus	202
3.4.6 Operazioni del bus	205
3.5 Esempi di chip della CPU	208
3.5.1 Intel Core i7	208
3.5.2 Texas Instruments OMAP4430	215
3.5.3 Il microcontrollore Atmel ATmega168	219
3.6 Esempi di bus	221
3.6.1 Bus PCI	222
3.6.2 PCI Express	230
3.6.3 Universal Serial Bus	235
3.7 Interfacce	239
3.7.1 Interfacce di I/O	239
3.7.2 Decodifica dell'indirizzo	240
3.8 Riepilogo	243

Capitolo 4 Livello di microarchitettura

4.1 Esempio di microarchitettura	249
4.1.1 Percorso dati	250
4.1.2 Microistruzioni	257
4.1.3 Unità di controllo microprogrammata: Mic-1	259
4.2 Esempio di ISA: IJVM	264
4.2.1 Stack	264
4.2.2 Modello della memoria di IJVM	267

4.2.3 Insieme d'istruzioni IJVM	268
4.2.4 Compilazione da Java a IJVM	273
4.3 Implementazione di esempio	274
4.3.1 Microistruzioni e notazione	274
4.3.2 Implementazione di IJVM con Mic-1	279
4.4 Progettazione del livello di microarchitettura	291
4.4.1 Velocità/costi	292
4.4.2 Riduzione della lunghezza del percorso di esecuzione	294
4.4.3 Architettura con prefetching: Mic-2	301
4.4.4 Architettura a pipeline: Mic-3	305
4.4.5 Pipeline a sette stadi: Mic-4	310
4.5 Miglioramento delle prestazioni	314
4.5.1 Memoria cache	314
4.5.2 Predizione dei salti	321
4.5.3 Esecuzione fuori sequenza e rinomina dei registri	326
4.5.4 Esecuzione speculativa	332
4.6 Esempi del livello di microarchitettura	334
4.6.1 Microarchitettura della CPU Core i7	335
4.6.2 Microarchitettura della CPU OMAP4430	341
4.6.3 Microarchitettura del microcontrollore ATmega168	346
4.7 Confronto tra i7, OMAP4430 e ATmega168	348
4.8 Riepilogo	349
Capitolo 5 Livello di architettura dell'insieme d'istruzioni	353
5.1 Panoramica del livello ISA	355
5.1.1 Proprietà del livello ISA	355
5.1.2 Modelli di memoria	357
5.1.3 Registri	359
5.1.4 Istruzioni	361
5.1.5 Panoramica del livello ISA del Core i7	361
5.1.6 Panoramica del livello ISA dell'OMAP4430 ARM	363
5.1.7 Panoramica del livello ISA dell'ATmega168 AVR	366
5.2 Tipi di dati	367
5.2.1 Tipi di dati numerici	368
5.2.2 Tipi di dati non numerici	369
5.2.3 Tipi di dati del Core i7	370
5.2.4 Tipi di dati dell'OMAP4430 ARM	370
5.2.5 Tipi di dati dell'ATmega168	371

5.3 Formati d'istruzione	371
5.3.1 Criteri progettuali per i formati d'istruzioni	372
5.3.2 Codice operativo espandibile	374
5.3.3 Formati delle istruzioni del Core i7	377
5.3.4 Formati delle istruzioni dell'OMAP4430 ARM	378
5.3.5 Formati delle istruzioni dell'ATmega168 AVR	380
5.4 Indirizzamento	381
5.4.1 Modalità d'indirizzamento	381
5.4.2 Indirizzamento immediato	381
5.4.3 Indirizzamento diretto	382
5.4.4 Indirizzamento a registro	382
5.4.5 Indirizzamento a registro indiretto	382
5.4.6 Indirizzamento indicizzato	384
5.4.7 Indirizzamento indicizzato esteso	385
5.4.8 Indirizzamento a stack	386
5.4.9 Modalità d'indirizzamento per istruzioni di salto	389
5.4.10 Ortogonalità dei codici operativi e delle modalità d'indirizzamento	390
5.4.11 Modalità d'indirizzamento del Core i7	392
5.4.12 Modalità d'indirizzamento dell'OMAP4430	394
5.4.13 Modalità d'indirizzamento dell'ATmega168 AVR	394
5.4.14 Analisi delle modalità d'indirizzamento	395
5.5 Tipi d'istruzioni	396
5.5.1 Istruzioni di trasferimento dati	396
5.5.2 Operazioni binarie	397
5.5.3 Operazioni unarie	398
5.5.4 Confronti e salti condizionati	400
5.5.5 Invocazione di procedura	402
5.5.6 Istruzioni di ciclo	403
5.5.7 Input/Output	404
5.5.8 Istruzioni del Core i7	407
5.5.9 Istruzioni della CPU ARM OMAP4430	410
5.5.10 Istruzioni dell'ATmega168 AVR	412
5.5.11 Insiemi d'istruzioni a confronto	414
5.6 Controllo del flusso	415
5.6.1 Flusso sequenziale e diramazioni	415
5.6.2 Procedure	416
5.6.3 Coroutine	421
5.6.4 Trap	423
5.6.5 Interrupt	424

5.7 Un esempio: le torri di Hanoi	428
5.7.1 Le torri di Hanoi nel linguaggio assemblativo del Core i7	428
5.7.2 Le torri di Hanoi nel linguaggio assemblativo dell'OMAP4430 ARM	430
5.8 Architettura IA-64 e Itanium 2	431
5.8.1 Il problema dell'ISA IA-32	432
5.8.2 Modello IA-64 e calcolo che utilizza il parallelismo esplicito	433
5.8.3 Riduzione degli accessi in memoria	433
5.8.4 Scheduling delle istruzioni	434
5.8.5 Riduzione dei salti condizionati: attribuzione di predicati	436
5.8.6 Caricamenti speculativi	439
5.9 Riepilogo	440
Capitolo 6 Livello macchina del sistema operativo	445
6.1 Memoria virtuale	446
6.1.1 Paginazione	447
6.1.2 Implementazione della paginazione	449
6.1.3 Paginazione a richiesta e working set	452
6.1.4 Politica di sostituzione delle pagine	454
6.1.5 Dimensione di pagina e frammentazione	456
6.1.6 Segmentazione	457
6.1.7 Implementazione della segmentazione	461
6.1.8 Memoria virtuale del Core i7	464
6.1.9 Memoria virtuale della CPU ARM OMAP4430	468
6.1.10 Memoria virtuale e caching	471
6.2 Virtualizzazione hardware	471
6.2.1 Virtualizzazione hardware nel Core i7	473
6.3 Istruzioni di I/O a livello OSM	473
6.3.1 File	474
6.3.2 Implementazione delle istruzioni di I/O a livello OSM	475
6.3.3 Istruzioni per la gestione di directory	479
6.4 Istruzioni per il calcolo parallelo a livello OSM	481
6.4.1 Creazione dei processi	481
6.4.2 Corsa critica	482
6.4.3 Sincronizzazione dei processi tramite semafori	486
6.5 Sistemi operativi di esempio	490
6.5.1 Introduzione	490
6.5.2 Esempi di memoria virtuale	496

6.5.3 Esempi di I/O a livello OS	500
6.5.4 Esempi di gestione dei processi	512
6.6 Riepilogo	518
Capitolo 7 Livello del linguaggio assemblativo	525
7.1 Introduzione al linguaggio assemblativo	526
7.1.1 Che cos'è un linguaggio assemblativo	526
7.1.2 Perché usare il linguaggio assemblativo	527
7.1.3 Formato delle istruzioni del linguaggio assemblativo	528
7.1.4 Pseudoistruzioni	530
7.2 Macroistruzioni	532
7.2.1 Definizione, chiamata ed espansione di macro	533
7.2.2 Macro con parametri	535
7.2.3 Caratteristiche avanzate	535
7.2.4 Implementazione delle funzionalità macro negli assemblatori	536
7.3 Processo di assemblaggio	537
7.3.1 Assemblatori a due passate	537
7.3.2 Prima passata	538
7.3.3 Seconda passata	542
7.3.4 Tabella dei simboli	544
7.4 Collegamento e caricamento	546
7.4.1 Compiti del linker	547
7.4.2 Struttura di un modulo oggetto	550
7.4.3 Rilocazione a tempo del binding e dinamica	551
7.4.4 Collegamento dinamico	554
7.5 Riepilogo	558
Capitolo 8 Architetture per il calcolo parallelo	561
8.1 Parallelismo nel chip	562
8.1.1 Parallelismo a livello delle istruzioni	563
8.1.2 Multithreading nel chip	570
8.1.3 Multiprocessori in un solo chip	576
8.2 Coprocessori	582
8.2.1 Processori di rete	583
8.2.2 Processori grafici	591
8.2.3 Crittoprocessori	594
8.3 Multiprocessori con memoria condivisa	594
8.3.1 Multiprocessori e multicompiler a confronto	594
8.3.2 Semantica della memoria	602

8.3.3 Architetture di multiprocessori simmetrici UMA	606
8.3.4 Multiprocessori NUMA	614
8.3.5 Multiprocessori COMA	623
8.4 Multicomputer a scambio di messaggi	625
8.4.1 Reti d'interconnessione	626
8.4.2 MPP: processori massicciamente paralleli	630
8.4.3 Cluster	640
8.4.4 Software di comunicazione per multicomputer	645
8.4.5 Scheduling	648
8.4.6 Memoria condivisa a livello applicativo	649
8.4.7 Prestazioni	657
8.5 Grid computing	663
8.6 Riepilogo	666
Capitolo 9 Bibliografia	671
Appendice A Aritmetica binaria	681
A.1 Numeri a precisione finita	681
A.2 Sistemi di numerazione in base fissa	683
A.3 Conversione tra basi	684
A.4 Numeri binari negativi	688
A.5 Aritmetica binaria	690
Appendice B Numeri in virgola mobile	693
B.1 Principi dell'aritmetica in virgola mobile	693
B.2 Standard in virgola mobile IEEE 754	696
Appendice C Programmazione in linguaggio assemblativo	703
C.1 Panoramica	704
—C.1.1 Linguaggio assemblativo	704
—C.1.2 Breve programma in linguaggio assemblativo	705
C.2 Processore 8088	706
C.2.1 Ciclo del processore	706
C.2.2 Registri d'uso generale	708
C.2.3 Registri puntatore	709
C.3 Memoria e indirizzamento	710
—C.3.1 Organizzazione di memoria e segmenti	711
—C.3.2 Indirizzamento	712

C.4 Istruzioni dell'8088	715
C.4.1 Trasferimento, copia e aritmetica	716
C.4.2 Operazioni logiche, su bit e di scorrimento	718
C.4.3 Cicli e operazioni iterative su stringhe	719
C.4.4 Istruzioni di salto e di chiamata	720
C.4.5 Chiamate di subroutine	722
C.4.6 Chiamate di sistema e subroutine di sistema	723
C.4.7 Osservazioni sull'insieme d'istruzioni	725
C.5 Assemblatore	726
C.5.1 Introduzione	726
C.5.2 as88, un assemblatore basato su ACK	727
C.5.3 Differenze tra assemblatori dell'8088	731
C.6 Tracer	732
C.6.1 Comandi del tracer	734
C.7 Installazione	735
C.8 Esempi	736
C.8.1 Esempio Hello World	737
C.8.2 Esempio con i registri d'uso generale	740
C.8.3 Istruzioni di chiamata e puntatori ai registri	741
C.8.4 Debugging di un programma per la stampa di array	744
C.8.5 Manipolazione di stringhe e istruzioni su stringhe	747
C.8.6 Tabella di salto	750
C.8.7 File: accesso diretto e accesso bufferizzato	752
Indice analitico	757

Prefazione

Le prime cinque edizioni di questo libro erano basate sull'idea che un computer può essere visto come una gerarchia di livelli, in cui ciascuno realizza una particolare e ben definita funzione. Questo fondamentale concetto è valido ancor oggi così come lo era quando venne pubblicata la prima edizione; per questo motivo è stato mantenuto come base anche di questa pubblicazione. Come nelle precedenti edizioni, sono trattati in modo dettagliato il livello logico digitale, il livello di micro-architettura, l'insieme delle istruzioni, il livello di architettura, il livello macchina del sistema operativo e quello del linguaggio assemblativo.

Pur mantenendo quindi la struttura di base, questa nuova edizione contiene molti cambiamenti, che permettono al testo di restare al passo con la rapida evoluzione dell'industria dei computer. Sono stati aggiornati anche i modelli utilizzati come casi di studio nel testo. Gli esempi presentati sono ora Intel Core i7, Texas Instrument OMAP4430 e Atmel ATmega168. Il Core i7 è un esempio di CPU largamente utilizzata su portatili, desktop e server, mentre l'OMAP4430 è una diffusissima CPU basata su ARM, molto utilizzata su smartphone e tablet.

Anche se probabilmente non avete mai sentito parlare del microcontrollore ATmega168, vi sarà spesso capitato di utilizzarne uno. L'ATmega168, basato su AVR, è presente in diversi sistemi integrati, a partire dalle radiosveglie digitali fino ai fornì a microonde. L'interesse per i sistemi integrati è in continua crescita e l'ATmega168 rappresenta una scelta ampiamente diffusa grazie al suo costo estremamente contenuto (qualche centesimo di euro), alla disponibilità di software e periferiche e al gran numero di programmatore specializzati. Il numero di microcontrollori ATmega168 esistenti supera di gran lunga il numero di processori Pentium e Core i3, i5 e i7. L'ATmega168 è anche utilizzato sulla piattaforma integrata a scheda singola (*single-board embedded computer*) Arduino, un popolare sistema per hobbisti progettato in Italia, e costa meno di una cena in pizzeria.

Nel corso degli anni, molti docenti mi hanno richiesto a più riprese materiale sulla programmazione in linguaggio assemblativo. Con la sesta edizione il materiale è reso disponibile alla pagina web di questo libro (indicata in seguito), dove può essere facilmente aggiornato ed espanso. Il linguaggio assemblativo scelto è quello dell'8088, che rappresenta una versione molto ridotta del diffusissimo insieme d'istruzioni iA32 utilizzato nei processori Core i7. Avremmo potuto utilizzare ARM, AVR o qualsiasi altra architettura della quale nessuno ha mai sentito parlare, ma l'8088, come strumento motivazionale, rappresenta la scelta migliore, perché molti studenti hanno a disposizione una CPU compatibile con l'8088. L'intero insieme di istruzioni del Core i7 è troppo

complesso per essere compreso dagli studenti in dettaglio: l'8088 è simile, ma molto più semplice. Il Core i7, descritto in dettaglio in questa edizione, è inoltre in grado di eseguire codice 8088. Tuttavia, poiché effettuare il debug del codice assemblativo è particolarmente difficile, ho messo a disposizione un insieme di strumenti per facilitare l'apprendimento della programmazione assemblativa, tra cui un assemblatore 8088, un simulatore e un *tracer*. Questi strumenti sono forniti per Windows, Unix e Linux e si trovano sul sito web del libro.

Nel corso degli anni il testo si è allungato (la prima edizione contava 443 pagine, questa ne conta 769). Una crescita inevitabile in quanto la materia continua a svilupparsi e vi è un numero sempre maggiore di cose da conoscere. Di conseguenza, quando il libro è utilizzato come testo di un corso semestrale, potrebbe non essere possibile terminarlo. Un possibile approccio potrebbe allora essere quello di studiare, come minimo indispensabile, i Capitoli 1, 2 e 3 interamente, la prima parte del Capitolo 4 (fino al Paragrafo 4.4 compreso) e il Capitolo 5. L'eventuale tempo rimanente potrebbe essere impiegato per il resto del Capitolo 4 e per parti dei Capitoli 6, 7 e 8 a seconda delle preferenze di docente e studenti.

Vediamo ora rapidamente le principali novità di questa nuova edizione. Il primo capitolo presenta ancora una panoramica storica delle architetture dei computer nella quale si evidenzia come si sia raggiunto lo stato attuale presentando le pietre miliari che hanno marcato il cammino. Molti studenti resteranno stupiti scoprendo che i computer più potenti degli anni '60 costavano milioni di dollari, ma avevano molto meno dell'uno per cento della potenza di calcolo dei loro smartphone. Nel corso del capitolo viene presentato l'ampio spettro di computer oggi esistenti, tra cui FPGA (*field-programmable gate arrays*), smartphone, tablet e console per videogiochi, e vengono introdotti i nostri tre esempi di architetture (Core i7, OMAP4430 e ATmega168).

Nel Capitolo 2 è stato ampliato il materiale sugli stili di elaborazione per includere i processori con parallelismo sui dati, tra cui le unità di elaborazione grafica (GPU). La panoramica sui dispositivi di memorizzazione è stata allargata per includere le sempre più diffuse memorie flash. È stato inoltre aggiunto nuovo materiale alla sezione Input/Output, con dettagli sui moderni controller di gioco, tra cui il Wiimote e il Kinect e sugli schermi interattivi utilizzati su smartphone e tablet.

Il Capitolo 3 è stato sottoposto a una profonda revisione. Tuttavia si parte ancora dal funzionamento dei transistor di modo che anche gli studenti totalmente privi di conoscenze hardware siano in grado di capire in linea di principio come funziona un moderno computer. Forniremo nuove informazioni sugli FPGA, strutture hardware programmabili che abbassano i costi di progetto a livello di porta logica a tal punto da renderli utilizzabili nelle scuole. In questo capitolo diamo una descrizione ad alto livello delle tre nuove architetture di esempio.

Il Capitolo 4, che è sempre stato molto apprezzato per la chiara spiegazione del reale funzionamento di un computer, non ha subito modifiche rispetto alla precedente edizione. Sono stati comunque aggiunti alcuni nuovi paragrafi che trattano il livello di microarchitettura di Core i7, OMAP4430 e ATmega168.

I Capitoli 5 e 6 sono stati aggiornati usando le nuove architetture d'esempio, in particolare con l'aggiunta di nuovi paragrafi che descrivono le istruzioni ARM e AVR. Il Capitolo 6 utilizza come esempio Windows 7 e non più Windows XP.

Il Capitolo 7, sulla programmazione in linguaggio assemblativo, resta in gran parte invariato rispetto al passato.

Per riflettere i nuovi sviluppi nell'ambito del calcolo parallelo, il Capitolo 8 ha subito una notevole revisione. Sono stati aggiunti nuovi particolari sull'architettura multiprocessore del Core i7 ed è descritta in dettaglio l'architettura della GPU NVIDIA Fermi. Infine, i paragrafi sui supercomputer BlueGene e Red Storm sono stati aggiornati per includere le più recenti modifiche apportate a queste enormi macchine.

Anche il Capitolo 9 è cambiato. Le letture suggerite sono state trasferite sulla pagina web e ora il capitolo contiene solo i riferimenti citati nel libro, molti dei quali sono nuovi: l'architettura dei calcolatori è un settore dinamico.

Le Appendici A e B non sono state modificate rispetto all'edizione precedente: negli ultimi anni la notazione binaria e in virgola mobile non è molto cambiata. L'Appendice C è stata scritta dal Dott. Evert Wattel della Vrije Universiteit di Amsterdam. Il Dott. Wattel ha un'esperienza pluriennale d'insegnamento mediante questi strumenti. A lui vanno i miei ringraziamenti per aver scritto questa appendice. Ci sono poche variazioni rispetto alla quinta edizione, ma gli strumenti a disposizione sono ora sul web e non più su un CD-ROM allegato.

Oltre agli strumenti per la programmazione in linguaggio assemblativo, il sito web contiene un simulatore grafico da utilizzare parallelamente allo studio del quarto capitolo. Gli studenti possono utilizzarlo come aiuto per meglio cogliere i principi ivi discussi. Il simulatore grafico è stato scritto dal Prof. Richard Salter dell'Oberlin College, e a lui vanno i nostri ringraziamenti.

Il sito web è raggiungibile all'indirizzo

<http://www.pearsonhighered.com/tanenbaum>

Da qui selezionate il collegamento a questo testo e scegliete la pagina che state cercando. Le risorse per gli studenti includono:

- assemblatore e tracer;
- simulatore grafico;
- letture suggerite.

Le risorse per i docenti includono:

- slide PowerPoint con le figure e le tabelle del testo;
- soluzioni degli esercizi di fine capitolo (in lingua inglese).

Per accedere a queste risorse, i docenti che adottano il testo dovranno registrarsi nell'Area Docenti sul sito web <http://hpe.pearson.it>.

Diverse persone hanno letto questo volume (o una sua parte) e fornito consigli utili, o sono state in altro modo d'aiuto. Vogliamo ringraziare, in particolare, Anna Austin, Mark Austin, Livio Bertacco, Valeria Bertacco, Debapriya Chatterjee, Jason Clemons, Andrew DeOrio, Joseph Greathouse e Andrea Pellegrini.

Ringraziamo le seguenti seguenti persone, che hanno rivisto il testo e suggerito alcuni cambiamenti: Jason D. Bakos (University of South Carolina), Bob Brown (Southern Polytechnic State University), Andrew Chen (Minnesota State University, Moorhead), J. Archer Harris (James Madison University), Susan Krucke (James Madison University), A. Yavuz Oruc (University of Maryland), Frances Marsh (Jamestown Community College), Kris Schindler (University at Buffalo).

Varie persone ci hanno aiutato nella creazione di nuovi esercizi: Byron A. Jeff (Clayton University), Laura W. McFall (DePaul University), Taghi M. Mostafavi (University of North Carolina at Charlotte), James Nystrom (Ferris State University). Abbiamo apprezzato, ancora una volta, il loro aiuto.

Il nostro editor, Tracy Johnson, ci è stato d'aiuto in tutti i modi ed è stato molto paziente. L'assistenza di Carole Snyder nel coordinamento delle persone coinvolte nel progetto è stata molto apprezzata. Bob Englehardt ha fatto un ottimo lavoro nella produzione.

Io (AST) vorrei ringraziare ancora una volta Suzanne per il suo amore e la sua pazienza che non ha mai fine, nemmeno dopo 21 libri. Barbara e Marvin sono da sempre fonte di gioia; ora sanno cosa fanno i professori per guadagnarsi da vivere. Aron appartiene alla nuova generazione: quella dei bambini che usano regolarmente i computer ancora prima di iniziare la scuola materna. Nathan non è ancora così avanti, ma quando avrà imparato a camminare l'iPad sarà a un passo.

Infine, io (TA) vorrei approfittare di questa opportunità per ringraziare mia suocera Roberta, che mi ha aiutato a ritagliare un po' di tempo di qualità per lavorare a questo libro. La sua tavola da pranzo a Bassano Del Grappa mi ha offerto la giusta quantità di solitudine, di accoglienza e di vino, per portare a termine questo compito importante.

ANDREW S. TANENBAUM
TODD AUSTIN

Prefazione all'edizione italiana

Due sono le chiavi di lettura di questa nuova edizione del *Tanenbaum*. La prima apre una finestra sull'affascinante mondo dei sistemi di calcolo. Mondo sempre più variegato e difforme. Un mondo che va dal milione di server nei data center di Google ai micro-controllori a 8 bit (che, acquistati in grandi quantità, possono costare meno di 10 centesimi). Un mondo nel quale anche il più accorto viaggiatore è destinato a perdere, a meno di non poter disporre di una solida e competente guida. E in effetti l'approccio strutturale del libro in questione ci accompagna passo passo nel viaggio che ci porta dal livello logico digitale a quello dei linguaggi orientati al tipo di problema; dal semplice (ma pionieristico) livello concettuale della macchina di Von Neumann agli Intel Core i7, alla piattaforma integrata di controllo Arduino. In questo ruolo, il *Tanenbaum* ricorda la tradizione di chiarezza e completezza delle precedenti edizioni e si riconferma un utilissimo strumento didattico.

Ma c'è un altro aspetto che spesso tende a sfuggire agli utenti informatici. Aspetto che è bene illustrare con le stesse parole dell'autore: "Il messaggio da cogliere è che il modello di calcolo più diffuso in una data epoca dipende molto dalla tecnologia, dall'economia e dalle applicazioni disponibili al momento e può cambiare nel momento in cui cambiano questi fattori". Se Babbage avesse avuto il silicio...

Quindi, e giustamente, questo testo non è soltanto un "libro di testo", ma ci informa anche su aspetti hardware che hanno potenzialità rivoluzionarie. Un esempio su tutti: la trattazione delle memorie flash: prestazioni in salita e costi in discesa.

Cosa aspettarci dalla prossima edizione del *Tanenbaum*?

Prof. Ottavio M. D'Antona
Dipartimento di Informatica
Università degli Studi di Milano



Introduzione

Un calcolatore digitale è una macchina in grado di svolgere dei compiti per le persone eseguendo le istruzioni che le vengono assegnate. Con il termine **programma** si intende una sequenza d'istruzioni che descrive come portare a termine un dato compito. I circuiti elettronici dei computer possono riconoscere ed eseguire direttamente soltanto un insieme limitato d'istruzioni semplici in cui tutti i programmi devono essere convertiti prima di poter essere eseguiti. Raramente queste istruzioni elementari sono molto più complicate delle seguenti:

- sommare due numeri;
- controllare se un numero vale zero;
- copiare una porzione di dati da una parte all'altra della memoria.

L'insieme di queste istruzioni primitive forma un linguaggio, chiamato **linguaggio macchina**, attraverso il quale è possibile comunicare con il computer. Chi progetta un calcolatore deve decidere quali istruzioni costituiranno il suo linguaggio macchina. Per ridurre la complessità e il costo dei dispositivi elettronici generalmente si cerca di progettare le più semplici istruzioni primitive che siano compatibili con il tipo di utilizzo e i requisiti prestazionali in base ai quali si progetta il computer. Dato che quasi tutti i linguaggi macchina sono semplici ed elementari, risulta difficile e noioso utilizzarli.

Nel corso del tempo, questa semplice osservazione ha portato a strutturare i computer come una serie di livelli di astrazione, costruiti l'uno sull'altro. In questo modo la complessità è gestibile più agevolmente e i computer possono essere progettati in modo sistematico e organizzato. Chiamiamo **approccio strutturale** questo modo di concepire l'architettura dei computer: il concetto sarà descritto in modo più preciso nel paragrafo successivo. Successivamente prenderemo in considerazione alcuni sviluppi storici, lo stato dell'arte e alcuni importanti esempi.

1.1 Approccio strutturale

Come abbiamo appena accennato esiste una gran differenza tra ciò che è adatto agli utenti e ciò che lo è per i computer. Gli utenti vogliono fare *X*, ma i computer possono fare soltanto *Y*: l'obiettivo del libro è spiegare in che modo sia possibile risolvere questo problema.

1.1.1 Linguaggi, livelli e macchine virtuali

Il problema può essere affrontato in due modi distinti: entrambi prevedono la definizione di un nuovo insieme d'istruzioni che sia più comodo da utilizzare rispetto alle istruzioni macchina predefinite. Considerate globalmente, anche queste nuove istruzioni formano un linguaggio, che chiamiamo L1, allo stesso modo in cui le istruzioni macchina formavano a loro volta un linguaggio, che chiamiamo L0. I due approcci differiscono nel modo in cui i programmi scritti in L1 possono essere eseguiti da un computer, in grado di eseguire soltanto quelli scritti nel proprio linguaggio, L0.

Un metodo per eseguire un programma scritto in L1 consiste nel sostituire, in una fase iniziale, ogni sua istruzione con un'equivalente sequenza di istruzioni in L0. Il programma che ne risulta è costituito interamente da istruzioni di L0 e può essere eseguito dal computer al posto del programma L1 originale. Questa tecnica è chiamata traduzione.

L'altra tecnica consiste invece nello scrivere un programma in L0 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 L0. Questa tecnica, che non richiede la generazione preventiva di un nuovo programma L0, è chiamata interpretazione e il programma che la esegue è detto interprete.

Traduzione e interpretazione sono simili e in entrambi i casi il computer può trattare istruzioni L1 eseguendo le equivalenti sequenze di istruzioni L0. La differenza è che, nel caso della traduzione, il programma L1 viene, all'inizio, convertito interamente in un programma L0. Il programma L1 può quindi essere gettato via, mentre il programma L0 viene caricato nella memoria del computer per essere eseguito. Durante l'esecuzione il computer ha il controllo del nuovo programma L0.

Nell'interpretazione invece ciascuna istruzione L1 viene esaminata, decodificata ed eseguita direttamente, cioè senza alcuna traduzione. In questo caso il computer ha il controllo del programma interprete, mentre il programma L1 è visto come un insieme di dati. Entrambi i metodi, oltre a una sempre più frequente combinazione dei due, sono ampiamente utilizzati.

Invece di ragionare in termini di traduzione e interpretazione, spesso è più semplice immaginare l'esistenza di un ipotetico computer o **macchina virtuale**, il cui linguaggio macchina sia L1. Chiamiamo M1 questa macchina virtuale M1 (e M0 quella corrispondente al linguaggio L0). Se una tale macchina potesse essere costruita in modo sufficientemente economico, non ci sarebbe affatto bisogno del linguaggio L0 né di una macchina capace di eseguire programmi L0. Gli utenti potrebbero semplicemente scrivere i loro programmi in L1, che verrebbero eseguiti direttamente dal computer. Anche se una macchina virtuale con linguaggio macchina L1 è troppo costosa o complicata da

essere costruita realmente, si possono tuttavia scrivere programmi dedicati. Questi programmi possono essere interpretati oppure tradotti da un programma scritto in L0, eseguibile direttamente sui computer esistenti. In altre parole, si possono scrivere programmi per macchine virtuali come se queste esistessero veramente.

Per rendere la traduzione o l'interpretazione utilizzabili in pratica, i linguaggi L0 e L1 non devono essere "troppo" diversi fra loro. Per la maggior parte delle applicazioni questo vincolo fa sì che L1, pur essendo migliore di L0, sia spesso ancora lontano dal linguaggio ideale. Questo risultato può sembrare forse scoraggiano rispetto alle intenzioni iniziali, secondo cui la creazione di L1 avrebbe dovuto sollevare il programmatore dall'onere di esprimere algoritmi in un linguaggio più adatto alle macchine che agli utenti. In ogni caso la situazione non è senza speranza.

L'approccio più ovvio consiste nell'inventare un nuovo insieme d'istruzioni che sia, rispetto a L1, maggiormente orientato agli utenti piuttosto che alle macchine. Anche questo terzo insieme forma a sua volta un linguaggio, che chiamiamo L2 (e la corrispondente macchina virtuale sarà M2). Si possono scrivere direttamente programmi in L2 come se esistesse realmente una macchina virtuale dotata di tale linguaggio macchina. Questi programmi possono essere tradotti in L1 oppure eseguiti da un interprete scritto in L1.

La definizione di una serie di linguaggi, ciascuno dei quali più pratico da utilizzare rispetto al precedente, può continuare indefinitamente finché non se ne ottenga uno sufficientemente adeguato. Ciascun linguaggio utilizza il precedente come base. Quindi un computer che usa questa tecnica può essere immaginato come una serie di strati o livelli disposti l'uno sopra l'altro, come mostrato nella Figura 1.1. Il livello, o linguaggio, che si trova più in basso è il più semplice e quello più in alto è il più sofisticato.

Tra linguaggi e macchine virtuali sussiste un'importante relazione. Una macchina ha un proprio linguaggio macchina, costituito da tutte le istruzioni che è in grado di eseguire. Di fatto, una macchina definisce un linguaggio. Nello stesso tempo, un linguaggio definisce una macchina: quella che è in grado di eseguire tutti i programmi scritti in quel linguaggio. Anche se una macchina definita da un linguaggio arbitrario potrebbe ovviamente essere estremamente complicata e costosa da realizzare, è tuttavia possibile immaginarla. Una macchina dotata di C, C++ o Java come linguaggio macchina risulterebbe sicuramente complessa ma, grazie alla tecnologia odierna, sarebbe in ogni caso realizzabile. Vi è tuttavia una buona ragione per non percorrere questa via: tale computer risulterebbe più costoso rispetto all'utilizzo di altre tecniche. Il semplice fatto che sia realizzabile non è sufficiente, dato che un progetto, per essere fattibile, deve anche essere economicamente conveniente.

In un certo senso, un computer con n livelli può essere interpretato come n distinte macchine virtuali. Osserviamo che, come succede per molti termini del linguaggio informatico, l'espressione "macchina virtuale" assume anche altri significati, uno dei quali sarà trattato più avanti in questo volume. Solo i programmi scritti nel linguaggio L0 possono essere eseguiti direttamente dalla componentistica digitale senza far ricorso alla traduzione o all'interpretazione, mentre i programmi scritti in L1, L2, ..., Ln devono essere interpretati da un interprete eseguito a un livello inferiore oppure devono essere tradotti in un altro linguaggio, corrispondente a un livello più basso.

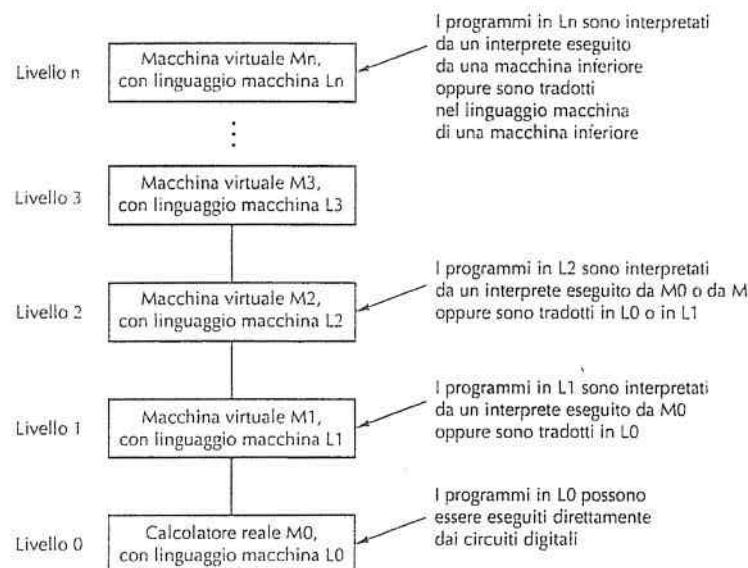


Figura 1.1 Macchina multilivello.

Se si scrivono programmi utilizzando la macchina di un dato livello non ci si deve preoccupare degli interpreti o dei traduttori sottostanti, dato che la struttura della macchina assicura che questi programmi saranno, in un modo o nell'altro, eseguiti. Non è di alcun interesse pratico il fatto che siano portati a termine da un interprete, eseguito a sua volta da un altro interprete, oppure che siano eseguiti direttamente dai circuiti digitali. In entrambi i casi si ottiene lo stesso risultato: i programmi vengono eseguiti.

La maggior parte dei programmatore che usa una macchina di un dato livello è interessata solamente al livello superiore, quello che meno assomiglia al linguaggio macchina del livello più basso. Tuttavia chi è interessato a comprendere il reale funzionamento di un computer li deve studiare tutti. Anche chi progetta nuovi computer o nuovi livelli deve avere la stessa familiarità con tutti i livelli e non solo con quello superiore. Il tema centrale di questo libro è costituito da concetti e tecniche inerenti la costruzione di macchine come gerarchie di livelli, nonché dai dettagli di quest'ultimi.

1.1.2 Attuali macchine multilivello

La maggior parte dei moderni computer consiste di due o più livelli. Come mostrato nella Figura 1.2 esistono anche macchine costituite da sei livelli. Il livello 0, che si trova alla base, rappresenta il vero e proprio hardware della macchina, i cui circuiti eseguono i programmi scritti nel linguaggio macchina del livello 1. Per completezza occorre dire che esiste un ulteriore livello al di sotto di quello che noi consideriamo livello 0. Questo livello è chiamato **livello dei dispositivi** e non è mostrato nella Figura 1.2, dato che

ricade nel campo dell'ingegneria elettronica (al di fuori quindi dell'ambito di questo libro). A questo livello i progettisti hanno a che fare con i singoli transistor, gli elementi primitivi nella progettazione dei computer. Se si vuole studiare il funzionamento interno dei transistor si deve entrare nel campo della fisica dello stato solido.

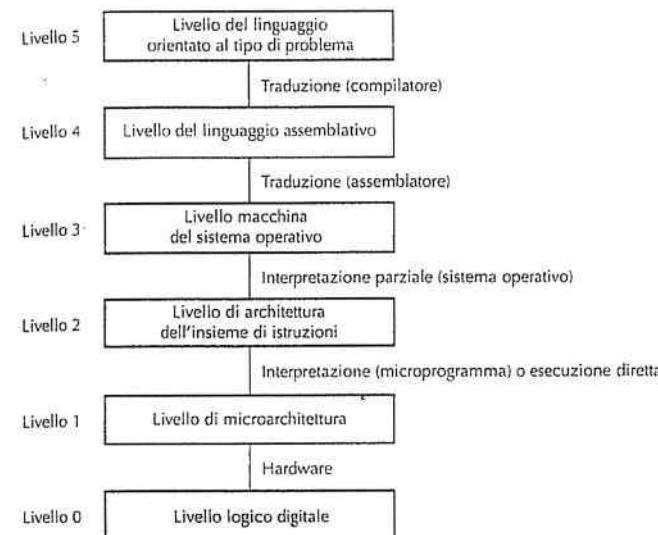


Figura 1.2 Computer a sei livelli. Sotto ciascun livello è indicato il metodo di supporto (oltre al nome del programma corrispondente).

Gli oggetti che analizzeremo nel livello più basso della nostra trattazione, il **livello logico digitale**, sono le **porte (gate)**. Queste, pur essendo costruite utilizzando componenti analogici come i transistor, possono essere correttamente modellate come dispositivi digitali. Ciascuna porta, costituita al più da una manciata di transistor, è dotata di uno o più input digitali (segnali corrispondenti ai valori 0 o 1) e calcola in output una semplice funzione dei valori d'ingresso, per esempio AND od OR. È possibile combinare un piccolo numero di porte per formare una memoria a 1 bit, che può memorizzare i valori 0 e 1. Combinando le memorie a 1 bit in gruppi, per esempio di 16, 32 o 64, è possibile formare quelli che vengono chiamati registri. Ciascun **registro** può contenere un numero il cui valore può variare fino a un certo limite. È inoltre possibile combinare le porte per formare lo stesso motore computazionale principale. Nel Capitolo 3 esamineremo in dettaglio le porte e il livello logico digitale.

Subito dopo troviamo il **livello di micro-architettura**. 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*, unità aritmetico-logica), capace di effettuare semplici operazioni

ni aritmetiche. I registri sono connessi alla ALU per formare un percorso dati lungo il quale questi ultimi si spostano. L'operazione base del percorso dati consiste nel selezionare uno o due registri, permettere alla ALU di operare su di loro (per esempio sommandoli) e memorizzare infine il risultato in uno dei registri. In alcune macchine le operazioni del percorso dati sono controllate da un programma chiamato **micropogramma**, mentre su altre il percorso dati è controllato direttamente dall'hardware. Nelle prime edizioni di questo libro abbiamo chiamato tale livello il "livello di microprogrammazione", dato che in passato era quasi sempre rappresentato da un interprete software. Attualmente invece il percorso dati è spesso controllato in modo diretto dall'hardware (interamente o parzialmente). Per rispecchiare questo cambiamento abbiamo modificato il nome in "livello di micro-architettura".

Nelle macchine in cui il controllo del percorso dati avviene via software il micropogramma è un interprete per le istruzioni del livello 2, che utilizza il percorso dati per prelevare, esaminare ed eseguire un'istruzione alla volta. Nel caso per esempio di un'istruzione ADD, questa viene prelevata, i suoi operandi localizzati e portati nei registri, la somma calcolata dalla ALU e infine il risultato viene ricopiatato nel registro appropriato. Passi analoghi si verificano anche in una macchina con controllo via hardware del percorso dati, senza però avere un programma memorizzato e dedicato all'interpretazione delle istruzioni del livello 2.

Il livello 2 è costituito da quello che chiamiamo il **livello ISA** (*Instruction Set Architecture Level*, livello di architettura dell'insieme d'istruzioni). Ogni produttore di computer pubblica un manuale per ciascuno dei propri modelli, un "Manuale di riferimento del linguaggio macchina" o "Principi di funzionamento del computer Western Wombat Model 100X", o qualcosa di simile. Questi manuali trattano il livello ISA, ma non quelli sottostanti; quando descrivono l'insieme d'istruzioni della macchina, presentano in realtà le istruzioni eseguite in modo interpretato dal micropogramma o dai circuiti elettronici. Se un produttore di computer dotasse una delle proprie macchine di due diversi interpreti, dovrebbe fornire due distinti manuali del "linguaggio macchina" dato che i due interpreti definirebbero due livelli ISA differenti.

Il livello successivo è generalmente un **livello ibrido**. La maggior parte delle istruzioni nel suo linguaggio fa parte anche del livello ISA (non vi è ragione per cui un'istruzione che appaia a un livello non possa essere presente anche in altri). Inoltre, vi è un insieme di nuove istruzioni, una diversa organizzazione della memoria e la capacità di eseguire programmi in modo concorrente, oltre a varie altre funzionalità. Fra i diversi modi di progettare il livello 3 vi è una maggior varietà rispetto a quanto non si riscontri nei livelli 1 e 2.

I nuovi servizi aggiunti nel livello 3 sono realizzati da un interprete eseguito al livello 2, storicamente chiamato **sistema operativo**. Le istruzioni del livello 3, identiche a quelle del livello 2, sono eseguite direttamente dal micropogramma (o dai circuiti elettronici) e non dal sistema operativo. In altre parole, alcune delle istruzioni del livello 3 sono interpretate dal sistema operativo, mentre altre sono interpretate in modo diretto dal micropogramma; per questo motivo tale livello viene detto "ibrido". Nel corso del libro ci si riferirà a questo livello come al **livello macchina del sistema operativo**.

Tra i livelli 3 e 4 vi è una spaccatura fondamentale. I tre livelli inferiori non sono progettati per essere utilizzati dal programmatore medio, ma concepiti principalmente per eseguire interpreti e traduttori necessari come supporto per i livelli più alti. Questi interpreti e traduttori sono scritti da professionisti chiamati **programmatori di sistema**, specializzati nella progettazione e nell'implementazione di nuove macchine virtuali. Il livello 4 e quelli superiori sono invece pensati per i programmati che devono risolvere problemi applicativi.

Un altro cambiamento che si riscontra a partire dal livello 4 è il modo in cui vengono supportati i livelli superiori. Mentre i livelli 2 e 3 sono sempre interpretati, i livelli 4, 5 e quelli superiori sono generalmente supportati mediante traduzione.

Un'ulteriore differenza tra i primi tre livelli e gli altri riguarda la natura del linguaggio corrispondente. I linguaggi macchina dei livelli 1, 2 e 3 sono numerici; i loro programmi consistono quindi in lunghe serie di numeri; certamente più adatte alle macchine che agli esseri umani. A partire dal livello 4 i linguaggi contengono invece parole e abbreviazioni umanamente comprensibili.

Il livello 4, quello del linguaggio assemblativo, è in realtà una forma simbolica di uno dei linguaggi sottostanti. Questo livello fornisce ai programmati un modo per scrivere programmi per i livelli 1, 2 e 3 in una forma meno difficoltosa rispetto a quella dei linguaggi delle rispettive macchine virtuali. I programmi in linguaggio assemblativo sono inizialmente tradotti nei linguaggi* dei livelli 1, 2 o 3 e successivamente interpretati dall'appropriata macchina virtuale o reale. Il programma che esegue la traduzione è chiamato **assemblatore**.

Il livello 5 consiste generalmente di linguaggi, spesso chiamati **linguaggi ad alto livello**, definiti per essere utilizzati dai programmati di applicazioni. Ne esistono letteralmente a centinaia; alcuni fra i più conosciuti sono C, C++, Java, Perl, Python e PHP. I programmi scritti in questi linguaggi sono generalmente tradotti al livello 3 o al livello 4 da un traduttore detto **compilatore**; in casi meno frequenti è anche possibile che essi siano interpretati. I programmi in Java, per esempio, di solito sono tradotti inizialmente in un linguaggio di tipo ISA chiamato *Java byte code*, che viene successivamente interpretato.

In alcuni casi, il livello 5 consiste di un interprete specifico per un determinato campo di applicazione, per esempio la matematica simbolica. Fornisce dati e operazioni per risolvere problemi in quell'area in termini facilmente comprensibili agli esperti di quel campo.

Riassumendo, il concetto chiave da ricordare è che i computer sono progettati come una serie di livelli, ciascuno costruito su quelli che lo precedono. Ciascun livello rappresenta una diversa astrazione, caratterizzata dalla presenza di oggetti e operazioni diversi. Progettando e analizzando i computer in questo modo è possibile tralasciare temporaneamente i dettagli irrilevanti, riducendo di conseguenza un soggetto complesso in qualcosa di più facile comprensione.

L'insieme dei tipi di dati, delle operazioni e delle funzionalità di ciascun livello è chiamato **architettura**. Questa tratta gli aspetti visibili agli utenti di quel determinato livello. Fanno quindi parte dell'architettura le caratteristiche che un programmatore può vedere, come la quantità di memoria disponibile, mentre non ne fanno parte gli aspetti

implementativi, come la tecnologia con cui è realizzata la memoria. Lo studio di come progettare le parti di un computer visibili ai programmatore è chiamato **architettura degli elaboratori**. Nella pratica comune architettura dei calcolatori e organizzazione dei computer significano essenzialmente la stessa cosa.

1.1.3 Evoluzione delle macchine multilivello

Per fornire una panoramica delle macchine multilivello, ne esamineremo brevemente gli sviluppi storici, mostrando come il numero e la natura dei livelli siano evoluti nel corso degli anni. I programmi scritti in un vero linguaggio macchina (livello 1) possono essere eseguiti direttamente dai circuiti del computer (livello 0), senza far ricorso ad alcuna traduzione o interpretazione. Questi circuiti formano, insieme alla memoria e ai dispositivi di input/output, l'**hardware** del computer. L'hardware consiste di oggetti tangibili (circuiti integrati, schede con circuiti stampati, cavi, trasformatori, memorie e stampanti) piuttosto che di idee astratte, algoritmi o istruzioni.

Al contrario, il **software** consiste di **algoritmi** (istruzioni dettagliate su come realizzare un determinato compito) e delle loro rappresentazioni per i computer, vale a dire i programmi. I programmi possono essere memorizzati su hard disk, floppy disk, CD-ROM o altri supporti, ma l'essenza del software è l'insieme delle istruzioni che costituiscono il programma, non il supporto fisico su cui sono registrate.

Nei primissimi computer, il confine tra hardware e software era chiaro e cristallino. Nel corso del tempo, man mano che i computer si sono evoluti, quel confine si è però sfocato in modo considerevole per via dell'aggiunta, della rimozione e dell'unione di livelli, tanto che oggi è spesso difficile tenerli separati (Vahid, 2003). E proprio per questo, un tema centrale di questo libro è:

hardware e software sono logicamente equivalenti.

Ogni operazione eseguita via software può anche essere costruita direttamente in hardware, preferibilmente dopo averne compreso in modo sufficientemente approfondito il funzionamento. Come ha affermato Karen Panetta Lentz: "L'hardware non è che software pietrificato". Ovviamente è anche vero il contrario: ogni istruzione eseguita dall'hardware può essere simulata via software. La scelta di collocare certe funzioni nell'hardware e altre nel software è basata su fattori quali il costo, la velocità, l'affidabilità e la frequenza di modifiche che ci si aspetta. Esistono poche regole ferree nello stabilire se X deve andare nell'hardware e Y deve essere programmato esplicitamente. Queste scelte cambiano a seconda del mercato delle tecnologie, della domanda e del modo di usare il computer.

Invenzione della microprogrammazione

I primi computer digitali, risalenti agli anni '40, avevano solamente due livelli: il livello ISA, in cui erano realizzati tutti i programmi, e il livello logico digitale, che li eseguiva. I circuiti del livello logico digitale erano complessi, difficili da comprendere e da realizzare e, oltretutto, inaffidabili.

Nel 1951 Maurice Wilkes, un ricercatore della Cambridge University, propose di progettare un computer a tre livelli, al fine di semplificare drasticamente l'hardware e ridurre così il numero di (poco affidabili) valvole necessarie (Wilkes, 1951). Questa macchina avrebbe dovuto avere un interprete predefinito e non modificabile (il microprogramma), la cui funzione sarebbe stata quella di eseguire programmi al livello ISA mediante interpretazione. Sarebbe stato necessario un minor numero di circuiti elettronici, dato che l'hardware avrebbe dovuto eseguire microprogrammi costituiti da un insieme limitato di istruzioni invece di programmi a livello ISA, basati su un insieme di istruzioni molto più ampio. Dato che i circuiti di allora erano costituiti da valvole¹, una tale semplificazione prometteva di ridurne il numero migliorando di conseguenza l'affidabilità (cioè diminuendo il numero giornaliero di guasti).

Durante gli anni '50 vennero costruite poche di queste macchine a tre livelli, mentre lo furono di più negli anni '60. A partire dal 1970 l'idea di avere un livello ISA interpretato da un microprogramma divenne dominante e tutte le principali macchine dell'epoca la implementarono.

Invenzione del sistema operativo

In questi primi anni la maggior parte dei computer era di tipo "fai da te", il che significava che il programmatore doveva far funzionare la macchina personalmente. Vicino a ogni macchina vi era un foglio su cui ogni programmatore prenotava la fascia oraria durante la quale desiderava far eseguire il proprio programma, per esempio dalle 3 alle 5 del mattino (molti programmatore preferivano lavorare quando vi era silenzio in sala macchine). Quando arrivava il momento, il programmatore si dirigeva verso la sala macchine con un mazzo di schede perforate da 80 colonne (uno dei primi supporti di input) in una mano e una matita appuntita nell'altra. Una volta arrivato nella stanza del computer spingeva gentilmente il precedente programmatore fuori dalla porta e prendeva il controllo del computer.

Se il programmatore desiderava eseguire un programma FORTRAN si verificavano i seguenti passi.

1. Guardava nell'armadietto dove erano tenuti tutti i programmi, estraeva il grande mazzo di schede verdi etichettato "compilatore FORTRAN", lo inseriva nel lettore delle schede perforate e premeva il pulsante START.
2. Inseriva il suo programma nel lettore delle schede e premeva il pulsante CONTINUE. Il programma veniva letto.
3. Quando il computer si arrestava, il programmatore faceva leggere il suo programma FORTRAN una seconda volta. Anche se alcuni compilatori richiedevano una sola passata sull'input, molti altri ne richiedevano due o più. A ogni passata doveva essere letto un gran numero di schede.
4. Infine la traduzione si avvicinava al termine. Spesso in questa fase il programmatore diventava nervoso, poiché, nel caso in cui il compilatore avesse trovato un errore nel

¹ I cosiddetti tubi a vuoto (N.d.R.).

Anno	Nome	Realizzato da	Commento
1834	Analytical Engine	Babbage	Primo tentativo di costruire un computer digitale
1936	Z1	Zuse	Prima macchina calcolatrice funzionante basata su relé
1943	COLOSSUS	Governo britannico	Primo computer elettronico
1944	Mark I	Aiken	Primo computer americano di uso generale
1946	ENIAC I	Eckert/Mauchley	Inizio della moderna storia dei computer
1949	EDSAC	Wilkes	Primo computer a programma memorizzato
1951	Whirlwind I	M.I.T.	Primo computer con elaborazione in tempo reale
1952	IAS	Von Neumann	Progetto su cui si basa la maggior parte delle macchine odierne
1960	PDP-1	DEC	Primo minicomputer (venduto in 50 unità)
1961	1401	IBM	Piccola macchina aziendale di grande diffusione
1962	7094	IBM	Dominò il campo della computazione scientifica nei primi anni '60
1963	B5000	Burroughs	Prima macchina progettata per un linguaggio ad alto livello
1964	360	IBM	Prima linea di prodotti progettata come una famiglia
1964	6600	CDC	Primo supercomputer per calcoli scientifici
1965	PDP-8	DEC	Primo minicomputer con un mercato di massa (venduto in 50.000 unità)
1970	PDP-11	DEC	Dominò il mondo dei minicomputer negli anni '70
1974	8080	Intel	Primo computer a 8 bit di uso generale su un chip
1974	CRAY-1	Cray	Primo supercomputer vettoriale
1978	VAX	DEC	Primo superminicomputer a 32 bit
1981	IBM PC	IBM	Inizio dell'era moderna dei personal computer
1981	Osborne-1	Osborne	Primo computer portatile
1983	Lisa	Apple	Primo personal computer dotato di una GUI
1985	386	Intel	Progenitore a 32 bit della linea Pentium
1985	MIPS	MIPS	Prima macchina RISC commerciale
1985	XC2064	Xilinx	Primo FPGA
1987	SPARC	Sun	Prima workstation RISC basata su SPARC
1989	GridPad	Grid Systems	Primo tablet computer commerciale
1990	RS6000	IBM	Prima macchina superscalare
1992	Alpha	DEC	Primo personal computer a 64 bit
1992	Simon	IBM	Primo smartphone
1993	Newton	Apple	Primo computer palmare (PDA)
2001	POWER4	IBM	Primo multiprocessore dual-core

Figura 1.4 Alcune pietre miliari nello sviluppo del computer digitale moderno.

Per i successivi 150 anni non successe niente di significativo finché un professore di matematica della Cambridge University, Charles Babbage (1792-1871), inventore tra l'altro del tachimetro, progettò e costruì una macchina chiamata *difference engine* ("macchina differenziale"). Questo dispositivo meccanico, capace come quello di Pascal soltanto di sommare e sottrarre, fu progettato per calcolare tabelle di numeri utili per la navigazione. L'intera costruzione della macchina fu pensata per eseguire un solo algoritmo, il metodo matematico delle differenze finite. L'aspetto più interessante della *difference engine* fu però la sua forma di output: i risultati venivano incisi mediante una punta d'acciaio su una lastra di rame, anticipando quindi i futuri supporti a singola scrittura quali le schede perforate e i CD-ROM.

Sebbene la *difference engine* funzionasse in modo abbastanza soddisfacente, Babbage si stancò presto di una macchina capace di eseguire un solo algoritmo. Egli cominciò a profondere una sempre maggior quantità di tempo e di beni familiari (per non parlare delle 17.000 sterline di fondi governativi) nella progettazione e costruzione di una macchina che ne rappresentasse l'evoluzione: la *analytical engine* ("macchina analitica"). La *analytical engine* era composta da quattro componenti: il magazzino (la memoria), il mulino (l'unità computazionale), il dispositivo di input (le schede perforate) e il dispositivo di output (output stampato e perforato). Il magazzino era composto da 1000 parole da 50 cifre, utilizzate per memorizzare dati e risultati. Il mulino poteva prelevare gli operandi dal magazzino per sommargli, sottrargli, moltiplicarli o dividerli e infine memorizzarne nuovamente il risultato nel magazzino. Anche questa macchina, al pari della *difference engine*, era interamente meccanica.

Il grande avanzamento rappresentato dall'*analytical engine* consisteva nel fatto di essere di uso generale. Leggeva le istruzioni dalle schede perforate e le eseguiva. Alcune istruzioni comandavano la macchina in modo che prelevasse due numeri dal magazzino, li portasse nel mulino, eseguisse su di loro una data operazione (per esempio la somma) e ne rispedisse il risultato al magazzino. Altre istruzioni erano in grado di analizzare un numero e ramificare l'esecuzione del programma in base al suo segno. Perforando un diverso programma sulle schede di input era quindi possibile far eseguire all'*analytical engine* diverse computazioni, cosa che era invece impossibile con la *difference engine*.

Dato che la *analytical engine* era programmabile mediante un semplice linguaggio assemblativo, era necessario produrre il software. A tal fine, Babbage assunse Ada Augusta Lovelace, la giovane figlia del famoso poeta inglese, Lord Byron. Ada Lovelace fu quindi la prima programmatrice della storia e in suo onore fu chiamato il linguaggio di programmazione Ada.

Sfortunatamente, come avviene nel caso di molti progettisti moderni, Babbage non riuscì mai a ottenere un hardware completamente privo di errori. Il problema nasceva dalla necessità di avere migliaia e migliaia di rotelle e ingranaggi per ottenere un grado di precisione che la tecnologia del diciannovesimo secolo non era ancora in grado di fornire. Ciononostante le sue idee erano in largo anticipo sui tempi, al punto che la maggior parte dei computer moderni presenta oggi una struttura molto simile all'*analytical engine*. Per questo motivo è giusto affermare che Babbage fu il padre (o meglio, il nonno) degli odierni calcolatori.

Il successivo e significativo avanzamento avvenne verso la fine degli anni '30, quando uno studente tedesco di ingegneria, Konrad Zuse, costruì una serie di macchine calcolatrici automatiche utilizzando dei relé eletromagnetici. Una volta scoppiata la guerra, non riuscì a ottenere finanziamenti dal governo, poiché i burocrati si aspettavano di vincere il conflitto in così breve tempo da ritenere che la nuova macchina non sarebbe stata pronta prima della fine della guerra. Zuse non conosceva il lavoro di Babbage e le sue macchine furono distrutte nel 1944 durante un bombardamento degli Alleati su Berlino. Anche se per questi motivi il suo lavoro non poté influenzare lo sviluppo delle macchine successive, Zuse va comunque considerato come uno dei pionieri in questo campo.

Non molto tempo dopo, negli Stati Uniti altri due ricercatori, John Atanasoff dell'Iowa State College e George Stibitz dei Bell Labs, si dedicarono alla progettazione di calcolatori. La macchina di Atanasoff era incredibilmente avanzata per l'epoca: era basata sull'aritmetica binaria e utilizzava dei condensatori per la memoria. Affinché la loro carica non si disperdesse, la memoria veniva periodicamente aggiornata, secondo un processo che Atanasoff chiamò *jogging the memory*. Le moderne memorie dinamiche (DRAM) funzionano secondo lo stesso principio. Sfortunatamente la macchina non divenne mai realmente operativa; in un certo senso Atanasoff fu come Babbage, cioè un precursore che non ebbe successo a causa dell'inadeguatezza tecnologica del proprio tempo.

Al contrario il computer di Stibitz, sebbene meno evoluto di quello di Atanasoff, riuscì a funzionare e il suo inventore ne diede una dimostrazione pubblica nel 1940 durante una conferenza al Dartmouth College. Tra il pubblico c'era John Mauchley, a quel tempo ancora sconosciuto, ma di cui in seguito si sarebbe sentito parlare nel mondo dei computer. Mentre Zuse, Stibitz e Atanasoff stavano progettando calcolatori automatici, un giovane di nome Howard Aiken si sforzava di svolgere a mano tediosi calcoli numerici, per la sua tesi di dottorato, alla Harvard University. Dopo la laurea, Aiken comprese l'importanza di poter svolgere calcoli per mezzo di una macchina. Frequentando una biblioteca, scoprì il lavoro di Babbage e decise di costruire per mezzo di relé il computer di uso generale che Babbage a suo tempo non riuscì a realizzare mediante ingranaggi.

Nel 1944, alla Harvard University, Aiken completò la sua prima macchina, il Mark I. Era composta da 72 parole di 23 cifre decimali, aveva un tempo d'istruzione di 6 secondi e usava un nastro di carta perforato per l'input e l'output. Dal momento in cui Aiken realizzò il suo successore, il Mark II, i computer basati su relé divennero obsoleti; era iniziata infatti l'era dell'elettronica.

1.2.2 Prima generazione – Valvole (1945-1955)

Lo stimolo per lo sviluppo dei computer elettronici venne dalla Seconda Guerra Mondiale. Nella prima parte della guerra i sottomarini tedeschi stavano decimando la flotta britannica. Da Berlino gli ammiragli tedeschi spedivano i comandi via radio ai sottomarini; questi ordini potevano quindi essere intercettati dai britannici. Il problema era che i messaggi venivano codificati per mezzo di un dispositivo chiamato ENIGMA; il precursore di questa macchina era stato progettato da Thomas Jefferson che, oltre a essere stato il terzo presidente degli Stati Uniti d'America, fu anche un inventore dilettante.

All'inizio della guerra lo spionaggio britannico riuscì a procurarsi una macchina ENIGMA grazie all'aiuto dei servizi segreti polacchi che erano riusciti a rubarla ai tedeschi. Tuttavia, per poter decifrare un messaggio codificato occorreva svolgere un'enorme quantità di calcoli, ed era necessario poterlo fare molto velocemente, non appena il comando veniva intercettato, affinché ciò potesse essere di una qualche utilità. Per decodificare questi messaggi il governo britannico creò un laboratorio segretissimo per la costruzione di un computer chiamato COLOSSUS. Il famoso matematico inglese Alan Turing diede il suo aiuto alla progettazione di questa macchina. COLOSSUS divenne operativo nel 1943 ma, dato che il governo britannico teneva sotto segreto militare per 30 anni praticamente ogni aspetto del progetto, lo sviluppo di COLOSSUS non ebbe alcun seguito. Vale la pena ricordarlo semplicemente perché fu il primo elaboratore digitale al mondo.

Oltre a distruggere le macchine di Zuse e incentivare la costruzione di COLOSSUS, la guerra condizionò il campo dei computer anche negli Stati Uniti. L'esercito aveva bisogno di tabelle per impostare la mira dell'artiglieria pesante e a tal scopo vennero assunte centinaia di donne (in quanto ritenute più precise degli uomini) affinché producessero in serie queste tabelle per mezzo di calcolatori manuali. Tuttavia la procedura richiedeva molto tempo, e vi era sempre il rischio dell'errore umano.

John Mauchley, che aveva studiato sia i lavori di Atanasoff sia quelli di Stibitz, venne a sapere che l'esercito era interessato alla costruzione di calcolatori meccanici. Come molti altri ricercatori dopo di lui, avanzò all'esercito una richiesta di sovvenzione per finanziare la costruzione di un calcolatore elettronico. Nel 1943 la richiesta fu accettata e Mauchley cominciò a costruire, insieme al suo studente J. Presper Eckert, un computer elettronico che chiamarono ENIAC (*Electronic Numerical Integrator And Computer*). L'elaboratore ENIAC era costituito da 18.000 valvole termoioniche e 1500 relé, pesava 30 tonnellate e consumava 140 KW di energia. Dal punto di vista dell'architettura, la macchina era dotata di 20 registri, ciascuno dei quali in grado di memorizzare un numero decimale a 10 cifre. Un registro decimale è una memoria molto piccola che può mantenere un valore fino a un massimo numero di cifre digitali, analogamente al contachilometri che tiene traccia di quanta strada una macchina ha percorso nella sua vita. ENIAC veniva programmato regolando 6000 interruttori multi-posizione e connettendo una moltitudine di prese con una vera e propria foresta di cavi.

La macchina fu terminata solo nel 1946, quando ormai non poteva più essere utile per lo scopo originario. Tuttavia, dato che la guerra era finita, fu concesso a Mauchley e Eckert di organizzare una scuola estiva per presentare il loro lavoro ai colleghi. Questo evento segnò l'inizio di un'esplosione di interesse nella costruzione di grandi computer digitali.

Dopo quella scuola estiva molti altri ricercatori intrapresero la costruzione di computer elettronici. Il primo a essere operativo fu l'elaboratore EDSAC (1949), costruito presso la Cambridge University da Maurice Wilkes. Fra gli altri progetti, vi furono JOHNIAC alla Rand Corporation, ILLIAC alla Illinois University, MANIAC presso il Los Alamos Laboratory e WEIZAC al Weizmann Institute in Israele.

Eckert e Mauchley cominciarono presto a lavorare su un successore della loro macchina, chiamato EDVAC (*Electronic Discrete Variable Automatic Computer*). Quel-

progetto morì però nel momento in cui decisero di lasciare la Pennsylvania University per formare a Philadelphia (la Silicon Valley non era ancora stata inventata) una propria società, la Eckert-Mauchley Computer Corporation. Dopo una serie di fusioni questa compagnia è diventata la moderna Unisys Corporation.

Facendo una piccola digressione in ambito legale è opportuno ricordare che Eckert e Mauchley registrarono un brevetto in cui sostenevano di essere gli inventori del computer digitale. A posteriori si può affermare che non sarebbe stato male possedere un tale brevetto. Dopo anni di controversie legali i tribunali decisero che il brevetto Eckert-Mauchley non era valido e che l'inventore del computer digitale, pur non avendolo mai brevettato, era stato John Atanasoff, che aveva effettivamente reso l'invenzione di dominio pubblico.

Mentre Eckert e Mauchley erano impegnati a lavorare su EDVAC, un membro del progetto ENIAC, John von Neumann, andò all'Institute of Advanced Studies di Princeton per costruire la propria versione dell'EDVAC, la macchina IAS. Von Neumann era un genio del livello di Leonardo da Vinci. Parlava molte lingue, era un esperto di fisica e matematica e aveva la capacità di ricordare perfettamente ogni cosa avesse sentito, visto o letto; era inoltre in grado di citare a memoria e alla lettera libri che aveva letto anni prima. Nel momento in cui cominciò a interessarsi ai computer era già il più importante matematico al mondo.

Una delle cose che gli apparvero subito evidenti fu che la programmazione dei computer mediante un gran numero di interruttori e cavi era lenta, tediosa e poco flessibile. Egli comprese che anche i programmi, al pari dei dati, potevano essere rappresentati in forma numerica all'interno della memoria del computer. Inoltre si rese conto che la contorta aritmetica decimale seriale usata nell'elaboratore ENIAC, in cui ogni cifra era rappresentata da 10 valvole (1 accesa e 9 spente), poteva essere sostituita da un'aritmetica binaria parallela, in modo simile a quanto aveva realizzato Atanasoff anni prima.

Il primo progetto elementare che descrisse è conosciuto al giorno d'oggi come macchina di von Neumann. Fu utilizzato nel computer EDSAC, il primo che memorizzava il programma in memoria, ed è ancora oggi, a ben mezzo secolo di distanza, alla base di quasi tutti i computer digitali. Questo progetto, così come la macchina IAS costruita in collaborazione con Herman Goldstine, ha avuto una tale influenza che vale la pena descriverlo brevemente. Nonostante si assocì sempre il nome di Von Neumann a questo progetto, anche Goldstine e altri ricercatori diedero un contributo sostanziale. Uno schema dell'architettura è mostrato nella Figura 1.5.

La macchina di von Neumann era composta da cinque componenti principali: la memoria, l'unità aritmetico-logica, l'unità di controllo e i dispositivi di input e output. La memoria era costituita da 4096 locazioni, ciascuna delle quali conteneva 40 bit. Ciascuna parola poteva mantenere due istruzioni da 20 bit oppure un numero da 40 bit. Le istruzioni erano composte da 8 bit, che definivano il tipo d'istruzione, e dai restanti 12 bit, che specificavano una delle 4096 parole di memoria. L'unità aritmetico-logica e l'unità di controllo formavano insieme il "cervello" del computer; negli elaboratori moderni esse sono combinate in un unico chip chiamato CPU (*Central Processing Unit*, "unità centrale di calcolo").

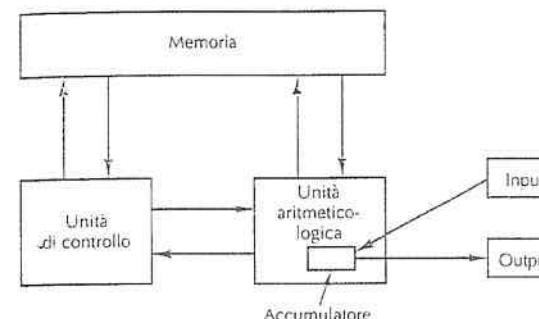


Figura 1.5 La macchina originale di von Neumann.

All'interno dell'unità aritmetico-logica vi era uno speciale registro da 40 bit, chiamato accumulatore. La tipica istruzione sommava una parola di memoria all'accumulatore oppure ne copiava il contenuto in memoria. La macchina non aveva un'aritmetica in virgola mobile dato che von Neumann riteneva che ogni matematico che si rispetti dovesse essere in grado di tener traccia a mente della posizione della virgola (per essere precisi, della virgola binaria).

Praticamente nello stesso periodo in cui von Neumann realizzava la macchina IAS, altri ricercatori del M.I.T. stavano costruendo un computer. Diversamente da IAS, ENIAC e altre macchine dello stesso tipo, dotate di lunghe parole e pensate per poter elaborare numeri molto grandi, la macchina del M.I.T., chiamata Whirlwind I, aveva parole di soli 16 bit ed era progettata per il controllo in tempo reale. Grazie a questo progetto si deve l'invenzione della memoria a nuclei magnetici da parte di Jay Forrester e, infine, la nascita del primo minicomputer commerciale.

Mentre avveniva tutto questo, IBM era una piccola società impegnata nel commercio di schede perforate e di macchine per il loro ordinamento. Inizialmente IBM non era fortemente interessata ai computer, sebbene avesse in parte finanziato il lavoro di Aiken. Cominciò a diventarlo nel 1953 dopo la produzione del modello 701 e quindi molto tempo dopo che la compagnia di Eckert e Mauchley era diventata, grazie al computer UNIVAC, il numero uno sul mercato. Il modello 701 era dotato di 2048 parole di 36 bit, con due istruzioni per parola, e fu la prima di una serie di macchine scientifiche che in una decade cominciarono a dominare il mercato. Tre anni dopo fu il momento del modello 704, che inizialmente aveva 4096 parole di memoria centrale, istruzioni a 36 bit e un nuovo, innovativo, hardware in virgola mobile. Nel 1958 IBM cominciò la produzione del modello 709, la sua ultima macchina che utilizzava le valvole, in pratica una miglioria del computer 704.

1.2.3 Seconda generazione – Transistor (1955-1965)

Il transistor è stato inventato nel 1948 presso i Bell Labs da John Bardeen, Walter Brattain e William Shockley, per la cui scoperta ricevettero nel 1956 il premio Nobel per la

fsica. In 10 anni il transistor rivoluzionò i computer al punto che nei tardi anni '50 i computer a valvole divennero obsoleti. Il primo computer a transistor fu costruito presso i Lincoln Laboratory del M.I.T. Si trattava di una macchina a 16 bit che seguiva la linea del Whirlwind I. Fu chiamato TX-0 (*Transistorized eXperimental computer 0*), ideato semplicemente come dispositivo per testare il più evoluto TX-2.

Il computer TX-2 non prese mai il volo, ma nel 1957 Kenneth Olsen, uno degli ingegneri che lavoravano al Lincoln Laboratory, fondò una società, la Digital Equipment Corporation (DEC), per produrre una macchina commerciale molto simile al modello TX-0. Passarono ben quattro anni prima di veder apparire questa macchina, chiamata PDP-1; ciò fu dovuto principalmente al fatto che i finanziatori che avevano fondato la DEC erano fermamente convinti che non vi fosse mercato per i computer. Dopo tutto anche T.J. Watson, ex presidente di IBM, una volta ebbe a dire che il mercato mondiale di computer era all'incirca di sole quattro o cinque unità. Al contrario DEC si affermò principalmente per la vendita di schede di circuiti di piccole dimensioni alle imprese, da integrare nei loro prodotti.

Nel 1961, quando finalmente apparve, l'elaboratore PDP-1 era dotato di 4096 parole di memoria da 18 bit e poteva eseguire 200.000 istruzioni al secondo. Queste prestazioni corrispondevano alla metà di quelle dell'IBM 7090, successore a transistor del modello 709, nonché il più veloce computer al mondo dell'epoca. Il PDP-1 costava però 120.000 dollari, mentre l'IBM 7090 ne costava milioni. DEC vendette decine di PDP-1 e questo rappresentò la nascita dell'industria dei microcomputer.

Uno dei primi PDP-1 fu donato al M.I.T., dove attrasse rapidamente l'attenzione di alcuni dei giovani geni in erba, così comuni in quella università. Una delle principali innovazioni del PDP-1 era un display visuale in grado di disegnare punti in qualsiasi zona dello schermo da 512 per 512 pixel. Da lì a breve gli studenti programmarono la macchina PDP-1 per poter giocare a *spacewar*, facendo così conoscere al mondo il primo videogame della storia.

Pochi anni dopo DEC introdusse il modello PDP-8, una macchina a 12 bit, molto più economica del PDP-1 (16.000 dollari). La principale innovazione introdotta dal PDP-8 fu quella di avere un unico bus, chiamato *omnibus*, mostrato nella Figura 1.6. Un bus è un insieme di cavi paralleli utilizzati per connettere i diversi componenti di un computer. Questa architettura fu un grande passo in avanti rispetto alla macchina IAS, che era centralizzata rispetto alla memoria, e fu adottata da quasi tutti i successivi computer di piccole dimensioni. DEC riuscì a vendere 50.000 PDP-8, affermandosi così leader nel mercato dei microcomputer.

Nel frattempo la reazione di IBM all'avvento del transistor fu la costruzione del modello 7090, una versione a transistor del computer 709, citato precedentemente, e in seguito del modello 7094. Il computer 7094 aveva un ciclo di clock di 2 microsecondi e possedeva una memoria centrale di 32.536 parole da 36 bit. I modelli 7090 e 7094 segnarono la fine delle macchine in stile ENIAC e dominarono il mondo dell'elaborazione scientifica durante gli anni '60.

Nello stesso momento in cui, grazie al modello 7094, IBM diventava una delle principali forze del mercato, otteneva enormi profitti vendendo una piccola macchina per le aziende, chiamata 1401. Questa macchina era in grado di leggere e scrivere nastri

magnetici, leggere e perforare schede e stampare output a una velocità paragonabile a quella del modello 7094. Il tutto a una frazione del prezzo. Era quindi perfetta per la gestione dei dati aziendali, ma aveva pessime prestazioni nei calcoli scientifici.

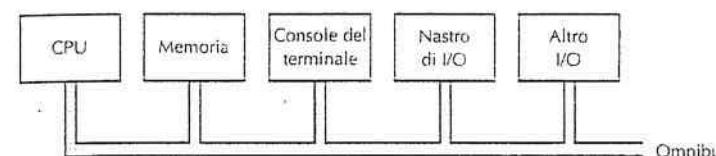


Figura 1.6 L'omnibus del computer PDP-8.

L'architettura del modello 1401 era insolita, dato che non aveva alcun registro e neppure una lunghezza fissa di parola. La sua memoria era composta da 4000 byte da 8 bit, sebbene modelli successivi supportassero fino a 16.000 byte, una dimensione che all'epoca era stupefacente. Ciascun byte conteneva un carattere da 6 bit, un bit di amministrazione e un bit usato per segnalare la fine di una parola. Per fare un esempio, un'istruzione MOVE specificava un indirizzo d'origine e uno di destinazione e iniziava a spostare byte dall'origine alla destinazione finché non ne incontrava uno il cui bit di fine parola era impostato a 1.

Nel 1964 una piccola e sconosciuta società, la Control Data Corporation (CDC), introdusse il modello 6600, una macchina che era quasi un ordine di grandezza più veloce dell'imponente 7094 e di ogni altra macchina dell'epoca. Per i matematici che si occupavano di analisi numerica, spesso chiamati *number cruncher*, "masticatori di numeri", fu amore a prima vista e il computer CDC fu lanciato verso il successo. Il segreto della sua velocità, e il motivo per cui fosse molto più del modello 7094, stava nel fatto che la CPU era, al suo interno, una macchina altamente parallela. Era dotata di varie unità funzionali che potevano lavorare contemporaneamente: alcune erano destinate a svolgere addizioni, altre moltiplicazioni e altre ancora divisioni. Anche se per ottenere il maggior guadagno possibile era richiesta un'attenta programmazione, con un po' di lavoro si riusciva a far eseguire contemporaneamente anche 10 istruzioni.

Come se non bastasse, il modello 6600 aveva al suo interno un certo numero di piccoli computer che lo aiutavano; un po' come i Sette Nani con Biancaneve, questo voleva dire che la CPU poteva spendere tutto il suo tempo a macinare numeri lasciando ai piccoli computer tutti i dettagli della gestione dei programmi e dell'input/output. A posteriori si può affermare che il 6600 era in anticipo di decenni sui tempi; molte delle idee chiave proprie dei computer moderni possono infatti essere ricondotte a questa macchina.

Il progettista del computer 6600, Seymour Cray, fu una figura leggendaria del livello di von Neumann. Dedicò la sua intera vita a costruire macchine sempre più veloci, ora chiamate *supercomputer*, tra cui gli elaboratori 6600, 7600 e Cray-1. Inventò inoltre un famoso algoritmo per l'acquisto di un'autovettura: si va dal rivenditore più vicino a casa propria, si indica la macchina più vicina alla porta e si dice: "Voglio questa". L'al-

goritmo impiega il minor tempo possibile per le cose non importanti (come comprare una macchina) per lasciare il maggior tempo possibile alle cose importanti (come progettare supercomputer).

Nella stessa epoca ci furono molti altri computer, ma uno si distinse per un motivo particolare e merita per questo di essere menzionato: il Burroughs B5000. I progettisti di macchine come i modelli PDP-1, 7094 e 6600 si concentravano esclusivamente sull'hardware, sia nel tentativo di renderlo economico (DEC) sia in quello di renderlo veloce (IBM e CDC), mentre ignoravano quasi completamente il software. I progettisti del B5000 adottarono invece un approccio diverso. Costruirono una macchina con il preciso intento di programmarla in Algol 60, un precursore di C e Java, e inclusero nell'hardware molte caratteristiche finalizzate a facilitare il compito del compilatore. Nacque così l'idea che anche il software ha la sua importanza; purtroppo questa considerazione fu quasi subito abbandonata.

1.2.4 Terza generazione – Circuiti integrati (1965-1980)

Nel 1958 l'invenzione dei circuiti integrati su silicio da parte di Jack Kilby e Robert Noyce (che lavorarono in maniera indipendente) permise di realizzare su un unico chip decine di transistor. Questo metodo di assemblaggio rese possibile la costruzione di computer più piccoli, più veloci e più economici rispetto ai loro predecessori basati su transistor. Alcuni dei computer più significativi di questa generazione sono descritti in seguito.

Fin dal 1964 IBM era la società leader nel mondo dei computer e si trovava di fronte a un grande problema che riguardava le sue due macchine di maggior successo e redditività. I modelli 7094 e 1401 erano totalmente incompatibili. Uno era un "macinatore di numeri" a grande velocità che usava aritmetica binaria parallela su registri a 36 bit, mentre l'altro era un diffuso processore di input/output che usava aritmetica decimale su parole di memoria a lunghezza variabile. Molti fra le società clienti di IBM avevano comprato entrambe le macchine e non amavano l'idea di dover avere due dipartimenti di programmazione separati senza nulla in comune.

Quando arrivò il momento di sostituire questi modelli, IBM fece un cambio radicale. Introdusse un'unica linea di prodotti, il System/360, basata su circuiti integrati e progettata sia per i calcoli scientifici sia per quelli commerciali. Il System/360 presentava molteplici innovazioni, ma la più importante fu che si trattava di una famiglia di circa mezza dozzina di macchine dotate dello stesso linguaggio assemblativo, ma di dimensione e potenza crescenti. I clienti potevano quindi sostituire i loro 1401 con i 360 Model 30 e i loro 7094 con i 360 Model 75. Il Model 75 era il più grande e il più veloce (e anche il più caro), ma il software scritto per uno di questi modelli poteva, in linea di principio, essere eseguito anche su uno differente. In pratica il software scritto per un piccolo modello funzionava senza problemi su un modello più grande, ma non valeva il viceversa: quando ci si spostava verso un modello di dimensioni inferiori il programma poteva non entrare in memoria. In ogni caso ciò rappresentava un grande avanzamento rispetto alla situazione delle macchine 7094 e 1401. L'idea di avere una *famiglia di computer* fu subito colta anche da molti altri produttori che, in pochi anni, misero sul mercato linee di macchine che ricoprivano un grande spettro di prezzi e prestazioni. La Figura 1.7

mostra alcune caratteristiche della famiglia 360 originaria (successivamente vennero introdotti anche altri modelli).

Un'altra grande innovazione della serie 360 fu la multiprogrammazione, grazie a cui è possibile avere più programmi in memoria allo stesso tempo, di modo che quando si è in attesa di completare un'operazione di input/output si possono eseguire dei calcoli. Il risultato che si ottiene con questa tecnica è un maggior utilizzo della CPU.

Il 360 fu anche il primo a essere capace di emulare (simulare) altri computer. Il modello più piccolo poteva emulare il computer 1401, mentre quelli più grandi erano in grado di emulare il modello 7094, di modo che, durante il passaggio ai computer 360, i clienti potevano continuare a utilizzare i loro vecchi programmi (binari) senza modificarli. Alcuni modelli eseguivano i programmi scritti per il 1401 in modo talmente più veloce che molti clienti non convertirono mai il proprio software.

Proprietà	Model 30	Model 40	Model 50	Model 65
Prestazioni relative	1	3,5	10	21
Ciclo di clock (in miliardesimi di secondo)	1000	625	500	250
Quantità massima di memoria (in byte)	65.536	262.144	262.144	524.288
Numero di byte prelevati per ciclo di clock	1	2	4	16
Massimo numero di canali per i dati	3	3	4	6

Figura 1.7 L'offerta iniziale della linea di prodotti IBM 360.

Sul 360 l'emulazione risultava facile, in quanto tutti i modelli iniziali, e molti di quelli successivi, erano microprogrammati. Tutto quello che IBM doveva fare era scrivere tre microprogrammi, uno per l'insieme d'istruzioni native del modello 360, uno per quelle del modello 1401 e uno per quelle del modello 7094. Questa flessibilità fu una delle ragioni principali per cui fu introdotta la microprogrammazione nel 360. La motivazione di Wilkes sulla riduzione del numero di valvole non contava più, ovviamente, perché il 360 non aveva valvole.

Il modello 360, grazie a un compromesso, risolse anche il dilemma dell'aritmetica binaria parallela e di quella decimale seriale: la macchina aveva 16 registri da 32 bit per l'aritmetica binaria, ma la sua memoria era orientata al byte, come quella del modello 1401. Era inoltre dotata d'istruzioni seriali simili a quelle del modello 1401 per lo spostamento in memoria di blocchi di dimensioni variabili.

Un'altra caratteristica saliente del computer 360 era un enorme (per l'epoca) spazio degli indirizzi di $2^{24}=16.777.216$ byte. All'epoca, una tale quantità di memoria appariva praticamente come infinita, dato che il costo della memoria era di qualche dollaro al byte. Sfortunatamente la serie 360 fu seguita dalle serie 370, 4300, 3080, 3090, 390 e dalla serie z, tutte basate essenzialmente sulla stessa architettura. A metà degli anni '80 il limite della memoria divenne un problema reale e IBM fu costretta ad abbandonare in parte la compatibilità quando introdusse gli indirizzi a 32 bit, necessari per indirizzare una nuova memoria di 2^{32} byte.

Con il senso di poi si potrebbe discutere sul perché non si fosse pensato fin dall'inizio all'utilizzo di indirizzi a 32 bit, dato che le macchine erano state dotate di registri e parole a 32 bit; all'epoca però nessuno avrebbe mai potuto immaginare una macchina con 16 milioni di byte di memoria. Anche se per IBM il passaggio a indirizzi di 32 bit ebbe successo, si trattava ancora di una soluzione temporanea al problema dell'indirizzamento della memoria, perché i sistemi di calcolo avrebbero presto richiesto la capacità di indirizzare più di 2^{32} (4.294.967.296) byte di memoria. In pochi anni sarebbero apparsi sulla scena computer con indirizzi di 64 bit.

Nella terza generazione, anche il mondo dei microcomputer fece un grande passo in avanti con l'introduzione da parte di DEC della serie PDP-11, un successore a 16 bit del modello PDP-8. Sotto molti punti di vista la serie PDP-11 era una sorta di piccolo fratello della serie 360 così come il computer PDP-1 lo era stato per il modello 7094. Entrambi i computer 360 e PDP-11 avevano i registri orientati alla parola e la memoria orientata al byte, e tutti e due vennero prodotti in una gamma in cui il rapporto prezzo/prestazioni variava. Il modello PDP-11 ebbe un enorme successo specialmente presso le università, permettendo a DEC di mantenere la leadership sugli altri produttori di microcomputer².

1.2.5 Quarta generazione – Integrazione a grandissima scala (1980-?)

Negli anni '80 la tecnologia VLSI (*Very Large Scale Integration*, "integrazione a larghissima scala") permise di inserire in un unico chip prima decine di migliaia, poi centinaia di migliaia e infine milioni di transistor. Questo sviluppo portò velocemente alla realizzazione di computer più piccoli e più veloci. Prima del PDP-1 i computer erano talmente grandi e costosi che, per farli funzionare, aziende e università dovevano disporre di speciali uffici chiamati **centri di calcolo**, mentre, con l'avvento dei microcomputer, ogni dipartimento poteva dotarsi di un proprio elaboratore. Dal 1980 i prezzi crollarono al punto che anche i privati potevano permettersi di possedere un computer: ebbe così inizio l'era dei personal computer.

I personal computer furono utilizzati in modo molto diverso rispetto ai computer di grandi dimensioni. Vennero usati per l'elaborazione di testi, per l'amministrazione domestica (con i ben noti *spreadsheet*) e per numerose altre applicazioni fortemente interattive (come i videogiochi) che computer di dimensioni maggiori non potevano trattare in modo altrettanto pratico.

I primi personal computer venivano generalmente venduti in scatole di montaggio. Ciascun kit conteneva una scheda con circuito stampato, vari chip, tra cui di solito vi era un Intel 8080, alcuni cavi, un trasformatore e talvolta un floppy disk da 8 pollici. Era compito dell'acquirente montare le parti per costruire il computer, e il software non era incluso: se si voleva un qualsiasi programma occorreva scriverlo. Successivamente si diffuse per i processori 8080, il sistema operativo CP/M, scritto da Gary Kildall. Si trattava di un vero sistema operativo (su floppy disk), con un file system e una serie

d'istruzioni che l'utente doveva scrivere con la tastiera per farle leggere da un interprete di comandi (*shell*).

Un altro dei primi personal computer fu l'Apple e in seguito l'Apple II, progettati da Steve Jobs e Steve Wozniak nel loro famoso garage. Questa macchina ebbe un'enorme diffusione tra i privati e nelle scuole, rendendo la Apple dall'oggi al domani un serio concorrente sul mercato.

IBM, la forza principale dell'industria dei computer, dopo aver osservato a lungo e considerato a fondo che cosa stavano facendo le aziende concorrenti, decise infine di entrare nel mercato dei personal computer. Dato che progettare da zero una nuova macchina usando solo componenti proprietarie, costruite con transistor proprietari fatti di materiali di proprietà, avrebbe richiesto troppo tempo, IBM scelse una strada alternativa e piuttosto inusuale. Diede a un suo dirigente, Philip Estridge, una grande borsa carica di soldi e gli affidò l'incarico di andare a costruire un personal computer lontano dalle interferenze dei burocrati della sede centrale di Armonk, nello stato di New York. Estridge iniziò a fare acquisti a 2000 Km di distanza, a Boca Raton (Florida), scelse come CPU l'Intel 8088 e costruì il Personal Computer IBM utilizzando componenti commerciali. Il PC IBM fu introdotto nel mercato nel 1981 e divenne subito il computer più venduto nella storia. Quando il PC ha compiuto i 30 anni sono apparsi diversi articoli sulla sua storia, tra cui quelli a firma Bradley (2011), Goth (2011), Bride (2011) e Singh (2011).

IBM fece anche qualcos'altro d'inusuale di cui in seguito si sarebbe pentita. Diversamente dal solito, invece di mantenere completamente segreto il progetto della macchina (o almeno di proteggerlo con un muro enorme e impenetrabile di brevetti) pubblicò, in un libro venduto a soli 49 dollari, tutti gli schemi al completo, compresi i diagrammi dei circuiti. L'idea era quella di permettere ad altre società di realizzare delle schede aggiuntive per il PC IBM, in modo da aumentarne la flessibilità e la popolarità. Sfortunatamente per IBM numerose altre società cominciarono a realizzare dei **cloni** del PC, dato che il progetto era completamente pubblico e i componenti facilmente reperibili sul mercato. Spesso questi computer erano molto più economici di quelli IBM; in questo modo prese il via una nuova industria.

Nonostante altre compagnie, tra cui Commodore, Apple e Atari, realizzassero personal computer senza utilizzare le CPU Intel, la forza del mercato dei PC IBM era talmente grande che gli altri ne vennero schiacciati. Solo in pochi sopravvissero e soltanto in mercati di nicchia.

Uno dei sopravvissuti, seppur a mala pena, fu il Macintosh di Apple. Il Macintosh era stato introdotto nel 1984 come successore dello sfortunato Apple Lisa, il primo computer dotato di una GUI (*Graphical User Interface*, "interfaccia grafica per l'utente"), simile alla ormai popolare interfaccia di Windows. Lisa fallì a causa del costo troppo elevato, mentre il più economico Macintosh, introdotto l'anno successivo, riscosse un enorme successo e ispirò amore e passione nei suoi numerosi ammiratori.

Il giovane mercato dei personal computer fece nascere anche il desiderio, fino ad allora senza precedenti, di avere computer portatili. A quel tempo un computer portatile aveva senso quanto può averne oggi un frigorifero portatile. Il primo vero personal computer portatile fu l'Osborne-1, che con i suoi 11 Kg fu più che altro un "computer da

² Questi modelli erano generalmente chiamati *minicomputer* (N.d.R.).

bagagliaio". Eppure, dimostrò che era possibile realizzare computer portatili. L'Osborne-I ebbe un successo commerciale modesto, ma un anno dopo Compaq produsse il suo primo clone di PC IBM portatile e divenne rapidamente il leader di tale mercato.

La versione iniziale del PC IBM veniva venduta insieme al sistema operativo MS-DOS, fornito dalla Microsoft Corporation, una società che all'epoca era ancora piccola. Visto che Intel riusciva a produrre CPU sempre più potenti, IBM e Microsoft poterono sviluppare un successore di MS-DOS, chiamato OS/2, dotato di un'interfaccia utente grafica simile a quella dell'Apple Macintosh. Allo stesso tempo Microsoft, pensando all'ipotesi che OS/2 potesse non prendere piede, cominciò a sviluppare un suo proprio sistema operativo, Windows, eseguito sopra MS-DOS. Per riassumere questa lunga vicenda basti dire che OS/2 non ebbe successo, che fra IBM e Microsoft vi fu un'accesa e pubblica disputa e che Microsoft continuò per la propria strada, facendo di Windows un enorme successo. Come la piccola Intel e la ancora più piccola Microsoft abbiano potuto detronizzare IBM, una delle più grandi, ricche e potenti società nella storia del mondo, è senza dubbio una lezione che viene attentamente studiata in tutte le scuole di economia del mondo.

Dopo il successo dell'8088, Intel continuò a realizzarne versioni via via migliori e più potenti. Particolarmenre degno di nota fu il processore a 32 bit 80386, introdotto nel 1985, che fu seguito da una versione potenziata, chiamata naturalmente 80486. Le versioni successive sono andate sotto il nome di Pentium e Core e sono quelle utilizzate in quasi tutti i moderni PC. Il termine utilizzato da molte persone per descrivere l'architettura di tutti questi processori è x86. Il nome x86 è anche utilizzato per indicare i chip compatibili prodotti da AMD.

Alla metà degli anni '80 iniziò ad affermarsi un nuovo tipo di progettazione chiamata RISC, che consisteva nel sostituire le complicate architetture esistenti con altre molto più semplici (e più veloci). Queste macchine erano in grado di eseguire più istruzioni allo stesso tempo, spesso in un ordine diverso da quello in cui apparivano nel programma. I concetti di CISC, RISC e superscalare saranno introdotti nel Capitolo 2 e verranno analizzati approfonditamente nel corso del libro.

Sempre alla metà degli anni '80, Ross Freeman sviluppò con alcuni colleghi di Xilinx un approccio intelligente per costruire circuiti integrati senza bisogno di montagne di soldi né di un impianto di produzione di silicio. Questo nuovo tipo di circuito, chiamato FPGA (*field-programmable gate array*), conteneva una grande quantità di porte logiche generiche che potevano essere "programmate" in un qualsiasi circuito incorporabile nel dispositivo. Questo nuovo straordinario approccio al progetto di circuiti rese l'hardware FPGA malleabile come il software. Con l'utilizzo di un FPGA, al costo variabile da poche decine ad alcune centinaia di dollari, divenne possibile costruire sistemi informatici specializzati per applicazioni uniche al servizio di un numero ristretto di utenti. Fortunatamente, le imprese di fabbricazione di silicio potevano ancora produrre chip più veloci, con consumo ridotto e meno costosi per le applicazioni che richiedevano milioni di chip. Ma per applicazioni con pochi utenti, come la prototipazione, le applicazioni di progettazione a bassi volumi e l'istruzione, i componenti FPGA rimangono uno strumento importante per la costruzione di hardware.

I computer restarono a 8, 16 o 32 bit fino al 1992, quando DEC presentò il rivoluzionario Alpha, una vera macchina a 64 bit di tipo RISC che surclassava di gran lunga le prestazioni degli altri computer. La macchina ebbe un successo limitato, ma dovette passare un decennio prima che le macchine a 64 bit cominciassero ad affermarsi su grande scala, soprattutto fra i server di fascia alta.

Per tutti gli anni '90 i sistemi di calcolo sono diventati sempre più veloci grazie a una varietà di ottimizzazioni micro-architetturali, molte delle quali saranno esaminate in questo libro. Gli utenti di questi sistemi sono stati coccolati dai produttori di computer, perché ogni nuovo sistema comprato sarebbe stato in grado di eseguire i loro programmi molto più velocemente del loro vecchio sistema. Tuttavia, verso la fine degli anni '90 questa tendenza cominciava a svanire a causa di due importanti ostacoli di progetto: gli architetti erano a corto di trucchi per rendere i programmi più veloci e il raffreddamento dei processori stava diventando troppo costoso.

Nel disperato tentativo di continuare a costruire processori più veloci, la maggior parte delle società di computer ha iniziato a virare verso architetture parallele per ricavare maggiori prestazioni dai propri chip. Nel 2001 IBM ha introdotto l'architettura dual-core POWER4, il primo caso di CPU che incorporava due processori sullo stesso chip. Oggi la maggior parte dei processori di classe desktop e server, e anche alcuni processori integrati, incorporano più processori su un singolo chip. Le prestazioni di questi multiprocessori si sono rivelate purtroppo tutt'altro che stellari per l'utente medio, perché (come vedremo nei capitoli successivi) macchine parallele richiedono ai programmati di parallelizzare esplicitamente i programmi, un compito difficile e soggetto a errori.

1.2.6 Quinta generazione – Computer a basso consumo e computer invisibili

Nel 1981 il governo giapponese annunciò di voler stanziare 500 milioni di dollari per aiutare le società locali nella costruzione della quinta generazione di computer, che si sarebbe basata sull'intelligenza artificiale e che avrebbe rappresentato un enorme balzo in avanti rispetto alla "stupida" quarta generazione. I produttori di computer americani ed europei, avendo visto le compagnie giapponesi prendere piede in molti mercati, come quello delle telecamere, degli stereo e dei televisori, piombarono improvvisamente nel panico più totale e cominciarono a chiedere, tra le altre cose, finanziamenti ai rispettivi governi. Il progetto giapponese di quinta generazione, nonostante tanta enfasi, sostanzialmente fallì e venne abbandonato nel silenzio. In un certo senso fu come per l'*analytical engine* di Babbage; si trattò di un'idea visionaria talmente avanti rispetto ai tempi che la tecnologia necessaria non era neanche prevedibile.

Quella che può essere definita la quinta generazione di computer vide comunque la luce, seppur in un modo inaspettato: i computer si rimpicciolirono. Nel 1989 la Grid Systems rilasciò il primo tablet computer, chiamato GridPad. Si trattava di un piccolo schermo su cui gli utenti potevano scrivere con una speciale penna. Sistemi come il GridPad hanno mostrato che i computer non avevano bisogno di essere allocati su una scrivania o in una sala macchine, ma potevano essere inseriti in un supporto facile da trasportare, con touchscreen e riconoscimento della scrittura per dargli un valore aggiunto.

L'Apple Newton, prodotto nel 1993, mostrò che un computer poteva essere costruito con dimensioni non più grandi di quelle di un lettore di cassette audio. Come il Grid Pad, anche il Newton utilizzava come dispositivo di input la scrittura a mano libera, e questo rappresentò un grande ostacolo al suo successo: in seguito però altre macchine della stessa tipologia, ora chiamate PDA (*Personal Digital Assistants*, "assistente digitale personale"), hanno migliorato l'interfaccia utente e hanno avuto un'enorme diffusione. L'evoluzione di questi dispositivi ha portato agli attuali smartphone.

L'interfaccia di scrittura del PDA fu messa a punto da Jeff Hawkins, che aveva fondato una società denominata Palm per sviluppare PDA economici rivolti al mercato di massa. Hawkins era un ingegnere elettronico di formazione, ma aveva un vivo interesse nel campo delle neuroscienze, la disciplina che studia il cervello umano. Si rese conto che il riconoscimento della scrittura a mano libera poteva essere reso più affidabile insegnando agli utenti a scrivere in un modo più facilmente leggibile dai computer, una tecnica di input che Hawkins ha chiamato "Graffiti". Era necessario un breve periodo di formazione per l'utente, ma alla fine la scrittura diventava più veloce e affidabile. Il primo PDA della Palm, chiamato Palm Pilot, è stato un enorme successo. Graffiti è uno dei più grandi successi nel settore informatico, avendo mostrato la potenza della mente umana che... trae vantaggio dalla potenza della mente umana.

Gli utenti di PDA avevano piena fiducia nei loro dispositivi e li utilizzavano religiosamente per gestire appuntamenti e contatti. Quando i telefoni cellulari iniziarono a guadagnare popolarità nei primi anni '90, IBM colse al volo l'opportunità di integrare il telefono cellulare con il PDA, creando così lo "smartphone". Il primo smartphone, chiamato Simon, utilizzava un touchscreen per la gestione dell'input e offriva all'utente tutte le funzionalità di un PDA più il telefono, i giochi e l'email. Le dimensioni sempre più piccole dei componenti e il costo sempre più contenuto hanno portato a una grande diffusione degli smartphone, rappresentati dai popolari Apple iPhone e dalle piattaforme Google Android.

Tuttavia, neanche i PDA e gli smartphone possono essere considerati veramente rivoluzionari. Molto più importanti sono invece i computer cosiddetti "invisibili", poiché integrati in elettrodomestici, orologi, carte di credito e numerosi altri dispositivi (Bechini et al., 2004). In un ampio spettro di applicazioni questi processori garantiscono grandi funzionalità a basso costo. Anche se si può discutere sul fatto che questi processori siano veramente una nuova generazione (dato che sono in circolazione dagli anni '70), è indubbio che stiano rivoluzionando il funzionamento di migliaia di elettrodomestici e altri dispositivi. Stanno già cominciando ad avere un forte impatto sul mondo e nei prossimi anni la loro influenza crescerà rapidamente. Un aspetto non comune di questi computer integrati è che l'hardware e il software sono spesso coprogettati (Henkel et al., 2003). Più avanti nel libro torneremo a occuparci di loro.

Se la prima generazione è rappresentata dalle macchine a valvole (per esempio ENIAC), la seconda dalle macchine a transistor (per esempio, l'IBM 7094), la terza dalle macchine a circuiti integrati (per esempio, l'IBM 360) e la quarta dai personal computer (per esempio, le CPU Intel), la quinta generazione è in realtà caratterizzata più da un cambiamento di modello che da una specifica nuova architettura. In futuro i computer si troveranno ovunque e saranno integrati in ogni oggetto, in poche parole essi saranno

invisibili: faranno parte dei comuni gesti della vita di ogni giorno, come aprire porte, accendere luci, spendere denaro e migliaia di altre azioni. Questo modello, ideato dallo scomparso Mark Weiser, venne originariamente chiamato *ubiquitous computing* ("computazione onnipresente"), anche se spesso ci si riferisce a esso con il termine *pervasive computing* ("computazione pervasiva") (Weiser, 2002); cambierà profondamente il mondo, così come lo fece la rivoluzione industriale. Nel libro non tratteremo ulteriormente l'argomento, ma per avere maggiori informazioni si consultino (Lyytinen e Yoo, 2002; Saha e Mukherjee, 2003; Sakamura, 2002).

1.3 Tipologie di computer

Mentre nel paragrafo precedente abbiamo affrontato brevemente la storia dei sistemi di elaborazione, in questo guarderemo al presente e rivolgeremo anche uno sguardo al futuro. Sebbene i personal computer siano gli elaboratori più conosciuti, al giorno d'oggi esistono anche altri tipi di macchine: per questo vale la pena dare un'occhiata a quanto c'è intorno a noi.

1.3.1 Forze tecnologiche ed economiche

L'industria dei computer avanza più velocemente di tutte le altre. La principale forza trainante è la capacità, che i produttori di circuiti hanno, di integrare di giorno in giorno sempre più transistor all'interno dei chip. Avere più transistor, cioè piccoli interruttori elettronici, significa avere memorie più grandi e processori più potenti. Gordon Moore, cofondatore ed ex presidente di Intel, un giorno scherzando disse che, se la tecnologia dell'aviazione fosse cresciuta velocemente quanto quella dei computer, ora un aeroplano costerebbe 500 euro e farebbe il giro della terra in soli 20 minuti con 20 litri di carburante. Per di più un tale aeroplano avrebbe le dimensioni di una scatola da scarpe. In particolare Moore, durante la preparazione di un discorso dinanzi a un gruppo industriale, sottolineò che ogni nuova generazione di chip veniva introdotta 3 anni dopo quella precedente. Dato che ogni nuova generazione ha una quantità di memoria quattro volte maggiore della precedente, comprese che il numero di transistor in un chip aumentava a una velocità costante e predisse che questa crescita sarebbe continuata allo stesso modo per decenni. Questa osservazione è conosciuta come la legge di Moore.

Attualmente con il termine "legge di Moore" si intende il fatto che il numero di transistor raddoppia ogni 18 mesi; questo è equivalente a dire che il numero di transistor aumenterebbe circa del 60% ogni anno. La Figura 1.8, che riporta le dimensioni di chip di memoria e la loro data di apparizione, conferma che la legge di Moore è rimasta valida per oltre tre decenni.

Ovviamente la legge di Moore non è una vera e propria legge, ma un'osservazione empirica sulla velocità con cui i fisici dello stato solido e gli ingegneri hanno fatto avanzare lo stato dell'arte, e una previsione che ciò continuerà anche in futuro. Alcuni osservatori del mondo dell'industria si aspettano che la legge di Moore continui a valere per almeno un altro decennio. Altri osservatori si aspettano che la dissipazione di energia, la dispersione di corrente e altri effetti si manifestino prima e causino gravi problemi che dovranno essere risolti (Bose, 2004, Kim et al., 2003).

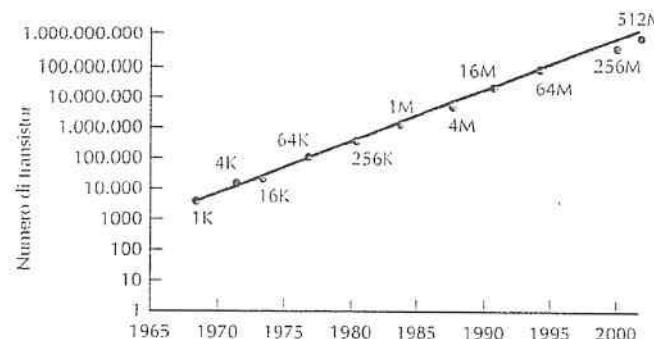


Figura 1.8 La legge di Moore prevede un aumento annuale del 60% del numero di transistor che può essere inserito su un chip. I valori indicati al di sopra e al di sotto della linea rappresentano le dimensioni della memoria, in bit.

La realtà sulla miniaturizzazione dei transistor è che il loro spessore sarà presto solo di pochi atomi. A quel punto i transistor saranno composti da troppo pochi atomi per essere affidabili, o più semplicemente si raggiungerà il punto in cui ulteriori diminuzioni di dimensione richiederanno l'utilizzo di componenti subatomiche. (Un buon consiglio a tutti coloro che lavorano in una fabbrica di produzione di silicio: si raccomanda di prendere un giorno libero quando decideranno di scomporre il transistor costituito da un solo atomo!)

Nonostante le molte sfide da affrontare per estendere l'andamento della legge di Moore, ci sono tecnologie promettenti all'orizzonte, tra le quali gli sviluppi in informatica quantistica (Oskin et al., 2002) e i nanotubi di carbonio (Heinze et al., 2002) che possono creare un'opportunità per spingere l'elettronica oltre i limiti imposti dal silicio.

La legge di Moore ha creato quello che gli economisti chiamano un ciclo virtuoso. Gli avanzamenti tecnologici (transistor/chip) portano a prodotti migliori e allo stesso tempo più economici. Prezzi inferiori portano a loro volta alla creazione di nuove applicazioni (nessuno programmava videogiochi per computer quando un elaboratore costava 10 milioni di dollari, anche se quando il prezzo scese a 120.000 dollari alcuni studenti del M.I.T. hanno raccolto la sfida). Nuove applicazioni creano nuovi mercati e fanno sorgere nuove società che cercano di trarne vantaggio. L'esistenza di tutte queste società genera competizione, che, a sua volta, crea la domanda economica per avere tecnologie migliori grazie a cui sconfiggere la concorrenza. In questo modo il cerchio si chiude.

Un altro fattore che guida lo sviluppo tecnologico è la prima legge del software di Nathan (dovuta a Nathan Myhrvold, un ex dirigente di Microsoft). Egli affermò che: "Il software è come un gas. Si espande fino a riempire il recipiente in cui è contenuto". Tornando agli anni '80 l'elaborazione di testi era fatta mediante programmi come *troff*. *Troff* occupa in memoria uno spazio dell'ordine dei KB, mentre i moderni elaboratori di testo occupano vari MB; senza alcun dubbio quelli futuri richiederanno GB di memoria (in prima approssimazione i prefissi K, M e G significano rispettivamente migliaia,

milioni e miliardi. si veda comunque il Paragrafo 1.5 per maggiori dettagli). Il software, continuando ad acquisire sempre più funzionalità (come crostacei che si attaccano via via alla chiglia di una barca), crea una richiesta costantemente crescente di processori più veloci, memorie più grandi e supporti di I/O di maggiore capacità.

Se nel corso degli anni l'aumento del numero di transistor per chip è stato sensazionale, i progressi delle altre tecnologie legate ai computer non sono stati da meno. Per fare un esempio, il PC IBM/XT introdotto nel 1982 aveva un hard disk da 10 MB, mentre a distanza di trent'anni i suoi successori sono comunemente dotati di hard disk da 1 TB. Questo aumento della capacità di cinque ordini di grandezza in 30 anni rappresenta un incremento annuale di circa il 50%. Misurare i progressi dei dischi è però un compito complesso, dato che oltre alla dimensione vi sono altri parametri da considerare, come la velocità di trasferimento, il tempo di ricerca e il prezzo. In ogni caso con quasi tutte queste metriche si verifica che dal 1982 il rapporto prezzo/prestazioni è migliorato ogni anno almeno del 50%. Questo enorme miglioramento delle prestazioni dei dischi, associato al fatto che il volume economico degli hard disk prodotti nella Silicon Valley ha superato quello delle CPU, ha spinto Al Hoagland ad affermare che il nome della famosa valle è sbagliato: andrebbe infatti chiamata Iron Oxide Valley (essendo questo il materiale – un ossido di ferro – su cui vengono registrati i dati nei dischi). La tendenza sta però lentamente ritornando a favore del silicio, dato che le memorie flash basate su silicio stanno rimpiazzando in molti sistemi i tradizionali dischi meccanici.

Un'altra area ad aver conosciuto uno spettacolare progresso è stata quella delle telecomunicazioni e delle reti. In meno di due decenni siamo passati dai modem a 300 bit/s, ai modem analogici a 56.000 bit/s fino alle fibre ottiche a 10^{12} bit/s. I cavi telefonici transatlantici in fibra ottica, come il TAT-12/13, costano circa 700 milioni di euro, durano 10 anni, e possono trasportare 300.000 chiamate simultanee; questi dati corrispondono a meno di un centesimo ogni 10 minuti di chiamata intercontinentale. È già stato dimostrato che si possono realizzare sistemi ottici di comunicazione a 10^{12} bit/s su distanze superiori a 100 km e senza l'utilizzo di amplificatori. La crescita esponenziale di Internet non richiede ulteriori commenti.

1.3.2 Tipologie di computer

Richard Hamming, ex-ricercatore dei Bell Labs, una volta osservò che un cambiamento della quantità di un ordine di grandezza produce un cambiamento della qualità. Secondo questa osservazione una macchina da corsa che può raggiungere i 1000 Km/h nel deserto del Nevada è un tipo di vettura fondamentalmente diverso da una comune automobile che va a 100 Km/h su un'autostrada. Analogamente un grattacielo da 100 piani non è semplicemente una versione ingrandita di un condominio da 10 piani. Nel caso dei computer non si parla di un fattore 10, ma addirittura di un fattore 1 milione nell'arco di tre decenni.

Il guadagno fornito dalla legge di Moore può essere sfruttato dai produttori dei chip in modi diversi. Uno di loro consiste nel costruire computer sempre più potenti allo stesso prezzo; l'altro nel costruire di anno in anno sempre lo stesso computer, ma a un prezzo inferiore. L'industria dei computer ha seguito entrambi gli approcci, e altri an-

ra, al punto che oggi esiste un'ampia varietà di computer. La Figura 1.9 mostra una classificazione sommaria dei computer attuali.

Nei paragrafi seguenti esamineremo ciascuna di queste categorie e ne considereremo brevemente le proprietà.

Tipo	Prezzo (\$)	Esempio di applicazione
Computer usa e getta	0.5	Cartoline d'auguri
Microcontrollore	5	Orologi, automobili, elettrodomestici
Dispositivi mobili e da gioco	50	Videogiochi e smartphone
Personal computer	500	Computer desktop o portatili
Server	5K	Server di rete
Raggruppamento di workstation	50-500K	Minisupercomputer di un dipartimento
Mainframe	5M	Elaborazione batch dei dati in una banca

Figura 1.9 L'attuale spettro di computer disponibili. I prezzi vanno interpretati "cum grano salis" (o meglio ancora, "cum saxo salis").

1.3.3 Computer usa e getta

Al gradino più basso troviamo i chip inseriti all'interno di cartoline d'auguri, che emettono la melodia di "Happy Birthday to You" o qualche altro motivetto altrettanto popolare. Gli autori non hanno ancora trovato una cartolina di condoglianze che suoni un canto funebre, ma ora che hanno reso pubblica questa idea si aspettano di vederne apparire presto una sul mercato. Per chi è cresciuto con grandi computer da milioni di dollari l'idea di un computer usa e getta ha senso quanto può averne quella di un aereo con lo stesso utilizzo.

Tuttavia i computer usa e getta fanno parte del nostro mondo e continueranno a farne parte anche in futuro. Probabilmente il più importante sviluppo nell'area di questi tipi di computer è rappresentato dai chip RFID (*Radio Frequency Identification*, "identificazione a radiofrequenza"). Oggi è possibile costruire, per pochi centesimi, chip RFID senza batteria, larghi meno di 0,5 mm, caratterizzati da un numero a 128 bit predefinito e univoco e contenenti al loro interno un piccolo radiotrasmettitore. Quando ricevono impulsi radio da un'antenna esterna i chip RFID si caricano grazie al segnale entrante sufficientemente a lungo da riuscire a ritrasmettere all'antenna il proprio numero identificativo. Anche se questi chip sono piccoli, le loro possibili implicazioni non lo sono affatto.

Partiamo da un'applicazione di uso mondano: eliminare il codice a barre dai prodotti. Sono già stati realizzati test sperimentali in cui i prodotti venduti in un negozio hanno, al posto del codice a barre, un chip RFID applicato dal produttore. Il cliente sceglie i prodotti desiderati, li ripone in un carrello della spesa e lo spinge semplicemente fuori dal negozio, evitando le casse. All'uscita un lettore dotato d'antenna spedisce un segnale richiedendo a ciascun prodotto di identificarsi, cosa che viene fatta attraverso una breve trasmissione radio. Anche il cliente viene identificato per mezzo di un chip

presente sul proprio bancomat o carta di credito, così che alla fine del mese il negozio potrà inviargli una dettagliata ricevuta di tutti gli acquisti effettuati. Se il cliente non ha un conto in una banca o una carta di credito che supporta RFID, scatta invece un allarme. Questo sistema non solo può sostituire i cassieri ed eliminare le relative attese in coda, ma può anche servire da sistema antifurto, dato che non è di alcuna utilità nascondere un prodotto in tasca o nello zainetto.

Una proprietà interessante di questo sistema è che, mentre il codice a barre identifica solo il tipo di prodotto, i chip RFID, grazie ai 128 bit a disposizione, possono identificare anche il prodotto stesso. Ne consegue che, per esempio, ogni pacchetto di aspirine presente sullo scaffale di un supermercato potrebbe avere un diverso codice RFID. Questo significa che se un produttore farmaceutico dovesse scoprire un difetto in un lotto di aspirine già distribuito, si potrebbe richiedere ai supermercati di tutto il mondo di far suonare un allarme se un cliente compra un pacchetto il cui numero RFID fa parte di tale gruppo. Ciò sarebbe attuabile anche se l'acquisto fosse effettuato in un paese lontano e a distanza di mesi; le aspirine non appartenenti al lotto difettato non provocherebbero invece alcun allarme.

Ma etichettare pacchetti di aspirine, biscotti o cibo per cani è soltanto l'inizio. Perché fermarsi a etichettare i biscotti per cani quando si può etichettare il cane stesso? Alcuni padroni stanno già chiedendo ai veterinari di impiantare nei loro animali un chip RFID, in modo da poterli rintracciare nel caso venissero rubati o si perdessero. Anche gli allevatori vogliono etichettare il loro bestiame. Il passo successivo che ci si aspetta è quello di genitori nervosi che richiedono al pediatra di impiantare un chip RFID nei loro figli, per poterli ritrovare nel caso venissero rapiti o si perdessero. Già che ci siamo, perché non obbligare gli ospedali a impiantarli in tutti i neonati, in modo da evitare gli scambi di bebè? Senza dubbio i governi e la polizia possono trovare ottime ragioni per identificare e tracciare i movimenti di tutti i cittadini in qualsiasi momento. Detto questo, le "implicazioni" dei chip RFID a cui si alludeva precedentemente risultano un po' più chiare.

Un'altra applicazione (leggermente meno controversa) dei chip RFID è l'identificazione di veicoli. Quando un treno, con chip RFID integrati, passa vicino a un lettore, il computer che vi è collegato ottiene una lista delle vetture che sono transitate. Questo sistema permette di localizzare agevolmente tutti i vagoni, il che può essere d'aiuto a fornitori, clienti e ferrovie. Uno schema simile sarebbe applicabile anche ai camion, mentre per le automobili l'idea è già utilizzata per il pagamento dei pedaggi autostradali per via elettronica (per esempio nel sistema E-Z Pass).

Anche la gestione dei bagagli delle compagnie aeree, così come molti altri sistemi di consegna, potrebbero trarre vantaggio dai chip RFID. All'aeroporto di Heathrow, a Londra, è stato condotto un esperimento per sollevare i passeggeri in arrivo dal ritiro dei loro bagagli. Le borse dei passeggeri che avevano acquistato questo servizio venivano marcate mediante chip RFID, poi spedite separatamente all'interno dell'aeroporto e infine consegnate direttamente nei rispettivi hotel. Fra le svariate possibilità di utilizzo dei chip RFID si possono immaginare macchine che, giunte alla stazione di verniciatura di una linea di assemblaggio, comunicano il colore che deve essere loro applicato; ulteriori possibili impieghi comprendono lo studio delle migrazioni di animali, la pro-

duzione di vestiti che comunicano alla lavatrice la temperatura da utilizzare e molto altro ancora. Alcuni chip potrebbero essere integrati con sensori in modo che i bit inferiori del loro numero identificativo potrebbero specificare temperatura corrente, pressione, umidità o altre variabili ambientali.

Chip RFID più avanzati sono in grado di memorizzare dati in modo permanente. Questa possibilità ha spinto la Banca Centrale Europea a decidere di inserire tali chip nelle banconote di euro dei prossimi anni. I chip sarebbero in grado di memorizzare il loro percorso. Ciò non solo renderebbe virtualmente impossibile contraffare le banconote di euro, ma permetterebbe anche di individuare i riscatti pagati ai rapitori e i bottini delle rapine; inoltre sarebbe facile tener traccia del denaro sporco e invalidarne le banconote. In futuro, quando il denaro cesserà di essere anonimo, una procedura standard di polizia potrebbe essere quella di controllare dove sia stato di recente il denaro di un sospetto. Chi avrà bisogno di impiantare chip nelle persone quando i loro portafogli ne saranno già pieni? Molto probabilmente, quando l'opinione pubblica capirà che cosa permettono di fare i chip RFID, nasceranno dibattiti pubblici in materia.

La tecnologia utilizzata nei chip RFID si sta sviluppando rapidamente. I più piccoli sono passivi (non alimentati internamente) e capaci soltanto di trasmettere il loro numero univoco quando richiesto. Quelli più grandi invece sono attivi, possono contenere una piccola batteria e un computer elementare, e sono in grado di compiere alcuni calcoli. Le piccole carte utilizzate per le transazioni economiche rientrano in questa categoria.

I chip RFID si distinguono non solo per essere attivi o passivi, ma anche in base allo spettro di radiofrequenze a cui possono rispondere. Quelli che operano a basse frequenze hanno una velocità di trasferimento dati limitata, ma possono essere captati da un'antenna anche a grande distanza. Quelli che operano ad alte frequenze hanno invece una più alta velocità di trasferimento dati, ma un raggio d'azione più ristretto. I chip differiscono anche sotto altri aspetti e continuano a essere migliorati. Internet è piena d'informazioni riguardanti i chip RFID; un buon punto di partenza è il sito www.rfid.org.

1.3.4 Microcontrollori

Al gradino successivo troviamo i computer integrati in apparecchiature che non sono vendute come elaboratori. I computer integrati in un dispositivo, a volte chiamati **microcontrollori**, lo comandano e ne gestiscono l'interfaccia utente. I microcontrollori sono presenti in un gran numero di apparecchi molto diversi fra loro; in seguito sono elencate alcune di queste categorie:

1. elettrodomestici (radiosveglie, lavatrici, asciugatrici, forni a microonde, impianti antifurto);
2. dispositivi per la comunicazione (telefoni senza fili, cellulari, fax, cercapersone);
3. periferiche del computer (stampanti, scanner, modem, lettori CD-ROM);
4. apparecchi per l'intrattenimento (videoregistratori, DVD, stereo, lettori MP3, decoder video);
5. dispositivi per le immagini (TV, macchine fotografiche digitali, videocamere digitali, obiettivi, fotocopiatrici);

6. strumenti medicali (raggi X, MRI, elettrocardiogrammi, termometri digitali);
7. armi (missili, torpedini);
8. apparecchiature per gli acquisti (macchinette per la distribuzione automatica, bancomat, registratori di cassa);
9. giochi (bambole paranti, console per videogame, auto o barche telecomandate).

Un'automobile può facilmente contenere 50 microcontrollori che comandano diversi sottosistemi tra cui l'antibloccaggio dei freni, l'iniezione della benzina, la radio e il GPS. Un jet ne può contenere anche più di 200, mentre una famiglia può facilmente possederne centinaia senza neanche saperlo. Fra pochi anni praticamente ogni dispositivo elettrico conterrà un microcontrollore. Il numero di microcontrollori venduti ogni anno supera di vari ordini di grandezza le vendite di tutti gli altri tipi di computer, fatta eccezione per quelli usa e getta.

Se i chip RFID sono sistemi minimi, i microcontrollori sono invece sistemi completi, anche se di piccole dimensioni. Ciascun microcontrollore è dotato di un processore, di una memoria e di capacità di I/O. Queste ultime di solito comprendono la gestione di pulsanti e interruttori del dispositivo e il controllo di luci, display e audio. Nella maggior parte dei casi il software è integrato nel chip sotto forma di una memoria di sola lettura creata durante la fabbricazione del microcontrollore. Generalmente i microcontrollori possono essere di due tipi: a uso generale, o specifico per un'applicazione. I primi non sono altro che piccoli, ma comuni, computer; mentre gli ultimi hanno un'architettura e un insieme d'istruzioni progettati specificatamente per una particolare applicazione, per esempio multimediale. I microcontrollori esistono in versioni da 4, 8, 16 e 32 bit.

Tuttavia anche i microcontrollori di uso generale differiscono per vari aspetti dai PC standard. Prima di tutto sono estremamente critici dal punto di vista dei costi. Un'azienda che deve comprare milioni di microcontrollori potrebbe basare la propria scelta su una differenza di prezzo di un centesimo per unità. Questo vincolo porta i produttori di microcontrollori a basare le proprie scelte architettoniche molto più sui costi di produzione, rispetto ai produttori di chip da centinaia di euro. Il prezzo dei microcontrollori varia tuttavia in modo rilevante a seconda del loro numero di bit, della grandezza e tipo della memoria di cui sono dotati e di altri fattori; per farsi un'idea, un microcontrollore a 8 bit acquistato in grandi quantità può costare meno di 10 centesimi. Questo prezzo rende possibile l'inserimento di un computer all'interno di una radiosveglia da 9,95 euro.

In secondo luogo, praticamente tutti i microcontrollori lavorano in tempo reale; essi ricevono uno stimolo e ci si aspetta che diano una risposta immediata. Una comune situazione è l'accensione di una luce quando l'utente preme un pulsante; in tal caso non ci deve essere alcun ritardo tra il momento in cui il pulsante viene premuto e quello in cui la luce si accende. Spesso la necessità di lavorare in tempo reale influisce sul tipo di architettura.

In terzo luogo, i sistemi integrati sono generalmente soggetti a vincoli fisici legati a dimensioni, peso, consumo della batteria e ad altri limiti elettrici e meccanici. I microcontrollori al loro interno devono essere progettati tenendo ben presenti queste restrizioni.

Un esempio particolarmente piacevole di microcontrollore è la piattaforma integrata di controllo Arduino, progettata a Ivrea da Massimo Banzi e David Cuartielles. L'obiet-

tivo del progetto era di produrre una piattaforma di calcolo integrata che costasse meno di una pizza quattro stagioni, rendendo così l'hardware facilmente accessibile a studenti e a hobbisti. È stato un compito difficile, perché in Italia le pizze costano veramente poco! Comunque lo scopo è stato raggiunto: un sistema completo Arduino costa meno di 20 dollari!

Questo sistema è un progetto hardware open-source, il che significa che tutti i dettagli sono pubblicati e disponibili gratuitamente in modo che chiunque può costruire (e anche vendere) un sistema Arduino. Arduino è basato sul microcontrollore Atmel AVR 8-bit RISC e la maggior parte delle schede include anche il supporto I / O di base. La scheda è programmata con un linguaggio di programmazione integrato, chiamato Wiring, dotato di tutte le funzioni necessarie per controllare dispositivi real-time. Ciò che rende la piattaforma Arduino divertente da usare è la sua comunità di sviluppo attiva e di grandi dimensioni. Ci sono migliaia di progetti pubblicati che utilizzano Arduino e che vanno da un rilevatore elettronico d'inquinamento ambientale, a una giacca per ciclisti dotata di indicatori di direzione, a un rilevatore di umidità che invia un'email quando una pianta ha bisogno di essere innaffiata, a un aeroplano senza equipaggio. Per saperne di più su Arduino e darvi da fare con progetti personali, visitate www.arduino.cc.

1.3.5 Dispositivi mobili e da gioco

A un gradino più in alto si trovano i dispositivi mobili e le console per videogiochi. Si tratta di normali computer spesso dotati di speciali capacità grafiche e sonore, ma poco espandibili e forniti di software limitato. Inizialmente erano basati su CPU a basso costo per gestire la telefonia e per poter giocare, tramite il televisore, a semplici giochi come il ping pong. Negli anni si sono evoluti in sistemi molto più potenti, rivaleggiando in alcuni aspetti con i personal computer e superandone talvolta le prestazioni.

Per avere un'idea di che cosa ci sia all'interno di questi sistemi consideriamo le specifiche tecniche di tre prodotti diffusi. Come primo esempio, esaminiamo la PlayStation 3 di Sony. È dotata di una CPU multicore proprietaria a 3,2 GHz (chiamata Cell) basata sulla CPU RISC PowerPC e di sette SPE (*Synergistic Processing Elements*, "elementi di elaborazione sinergici") a 128 bit. La PlayStation 3 dispone inoltre di 512 MB di RAM, un chip grafico dedicato Nvidia a 550 MHz e un lettore Blu-ray. Come secondo esempio, consideriamo la console Microsoft Xbox 360. Quest'ultima è dotata di una CPU IBM PowerPC triple-core a 3,2 GHz, di 512 Mb di RAM, di un chip grafico dedicato ATI a 500 MHz, di un lettore DVD e di un hard disk. Il terzo esempio è il tablet Samsung Galaxy. All'interno di questo dispositivo vi sono 2 core ARM a 1 GHz e un processore grafico (integrati sul SoC, *System-on-a-chip*, Nvidia Tegra 2), 1 GB di RAM, una doppia telecamera, un giroscopio a 3 assi e una memoria flash.

Anche se queste macchine non sono potenti quanto i personal computer di fascia alta prodotti nello stesso periodo, esse non sono molto distanti da questi ultimi e sotto alcuni aspetti li superano (un esempio è l'SPE a 128 bit della PlayStation 3, più evoluta della CPU di un qualsiasi PC). La differenza principale tra queste macchine e un PC non risiede tanto nella CPU quanto nel fatto che le prime sono sistemi chiusi. Anche se di solito le console hanno interfacce USB e FireWire, gli utenti non possono espanderle attraverso schede aggiuntive. Inoltre, e forse questo è l'aspetto più importante, queste

piattaforme sono attentamente ottimizzate per un particolare tipo di applicazioni, quelle altamente interattive con grafica 3D e output multimediale. Tutto il resto è secondario. Queste restrizioni hardware e software, la scarsa possibilità di espansione, le memorie di piccole dimensioni, l'assenza di un monitor ad alta risoluzione e un disco fisso piccolo o del tutto assente fanno sì che queste macchine possano essere vendute a prezzi più bassi rispetto ai personal computer. Nonostante queste restrizioni, nel mondo sono stati venduti milioni di questi dispositivi e i volumi di vendita sono in continua crescita.

I dispositivi mobili hanno il requisito aggiuntivo di dover consumare meno energia possibile per funzionare: meno energia usano, più a lungo dura la loro batteria. Questo requisito costituisce un'importante sfida in fase di progetto, perché tablet e smartphone devono essere parsimoniosi nel consumo di energia, ma, allo stesso tempo, soddisfare gli utenti che si aspettano elevate prestazioni, come la grafica 3D, l'elaborazione multimediale in alta definizione è il gioco.

1.3.6 Personal computer

Continuando a salire i gradini della scala degli elaboratori arriviamo ai personal computer, cioè quelle macchine a cui perlopiù si pensa quando si sente pronunciare il termine "computer". Essi comprendono i modelli desktop e quelli portatili. Generalmente sono dotati di alcuni gigabyte di memoria, di un disco fisso capace di immagazzinare terabyte di dati, di un lettore CD-ROM/DVD/Blu-ray, di una scheda audio, di un'interfaccia di rete, di un monitor ad alta risoluzione e di altre periferiche. Sono dotati di sistemi operativi sofisticati; hanno molte possibilità di espansione e dispongono di un'ampia gamma di software disponibile.

Il cuore di ogni personal computer è costituito da una scheda di circuiti stampati che contiene la CPU, la memoria, vari dispositivi di I/O (come un chip audio e talvolta un modem), oltre alle interfacce per tastiera, mouse, disco, rete e ad alcuni slot di espansione. La Figura 1.10 mostra un'immagine di una di queste schede.

I computer portatili sono fondamentalmente dei PC realizzati in dimensioni ridotte. Usano le stesse componenti hardware, seppur costruite in dimensioni inferiori, e possono eseguire lo stesso software dei PC da scrivania. Poiché la maggior parte dei lettori avrà probabilmente una certa familiarità con PC e notebook, non è necessario ulteriore materiale introduttivo.

Ancora un'altra variante di questo tema sono i tablet, come il ben noto iPad. Questi dispositivi non sono altro che normali PC in un corpo più piccolo, con un disco allo stato solido al posto di un disco tradizionale, un touchscreen e una CPU diversa dalle x86. Ma dal punto di vista architettonico, i tablet sono solo notebook con un diverso fattore di forma.

1.3.7 Server

Spesso vengono impiegate versioni potenziate dei personal computer o delle workstation come server di rete, sia per reti locali (di solito all'interno di un'azienda) sia per Internet.

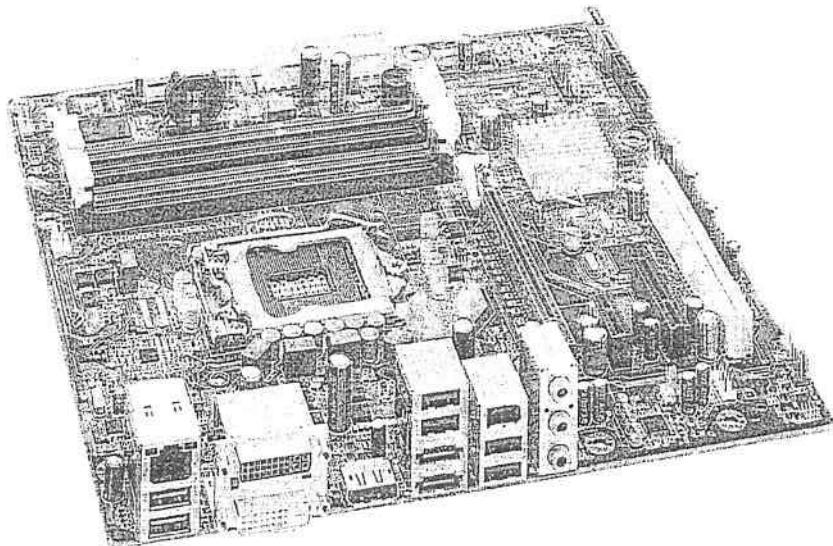


Figura 1.10 Nel cuore di ogni personal computer c'è una scheda di circuiti stampati. Questa immagine è una fotografia della scheda Intel DQ67SV. La fotografia è di proprietà di Intel Corporation (copyright 2011) e pubblicata con autorizzazione.

Esistono configurazioni mono o multiprocessore, dotate di gigabyte di memoria, terabyte di spazio su hard disk e connessioni di rete ad alta velocità. Alcuni server sono in grado di gestire migliaia di transazioni al secondo.

Dal punto di vista dell'architettura un server monoprocessoresso non è però molto diverso da un personal computer monoprocessoresso. È semplicemente più veloce e più grande, ha una maggiore memoria di massa e generalmente ha una connessione alla rete più veloce. I server eseguono gli stessi sistemi operativi dei personal computer, di solito uno dei dialetti di UNIX oppure di Windows.

Cluster

Dato che il rapporto prezzo/prestazioni dei server continua progressivamente a migliorare, negli ultimi anni i progettisti di sistemi hanno cominciato a connettere fra loro un gran numero di queste macchine per formare i cosiddetti cluster. Essi consistono in più sistemi di classe server connessi fra loro mediante reti la cui velocità è dell'ordine dei gigabyte al secondo. Queste macchine eseguono software speciale che permette loro di lavorare in modo congiunto su uno stesso problema, spesso di tipo aziendale, scientifico o ingegneristico. Generalmente le macchine utilizzate vengono chiamate COTS (*Commodity Off The Shelf*, "prodotti pronti per l'uso") cioè elaboratori che chiunque può acquistare in un normale negozio di computer. La principale aggiunta che viene fatta è

una connessione di rete ad alta velocità, anche se talvolta si utilizzano normali schede di rete commerciali.

In genere i cluster di grandi dimensioni si trovano in appositi locali o edifici dedicati denominati data center. Questi impianti possono comprendere, da una decina di macchine fino a 100.000, o più. Solitamente, la limitazione è imposta dalla quantità di denaro disponibile. Grazie ai bassi costi, le singole società possono ora possedere cluster per uso interno. Spesso i termini "cluster" e "data center" sono usati in maniera intercambiabile, anche se tecnicamente il primo indica l'insieme dei server, mentre il secondo indica il locale o l'edificio che li ospita.

Il tipico utilizzo di un cluster è quello di Internet Web server. Quando un sito web riceve migliaia di richieste al secondo per le sue pagine, la soluzione più economica è spesso quella di costruire un data center con centinaia, o addirittura migliaia, di server. Le richieste in arrivo vengono suddivise tra i server per consentirne l'elaborazione parallela. Per esempio, Google possiede data center sparsi in tutto il mondo per gestire le funzioni di ricerca; la più grande, a The Dalles in Oregon, è una struttura grande come due campi di calcio. La posizione è stata scelta perché i data center consumano grandi quantità di energia elettrica e The Dalles è sede di una centrale idroelettrica da 2 GW sul fiume Columbia, che può fornire una tale quantità di energia. Si pensa che Google disponga complessivamente di più di 1.000.000 di server nei suoi data center.

Il mercato dei computer è dinamico, con cose che cambiano continuamente. Nel 1960, l'informatica era dominata da giganteschi mainframe (vedi successivamente) che costavano decine di milioni di dollari, a cui gli utenti si collegavano mediante piccoli terminali remoti. Questo modello era fortemente centralizzato. Più avanti, negli anni '80, quando sono comparsi sulla scena i personal computer, milioni di persone ne hanno acquistato uno e il calcolo è stato decentralizzato.

Con l'avvento dei data center, stiamo iniziando a rivivere il passato sotto forma del cloud computing, il mainframe V2.0. L'idea è che tutti avranno uno o più dispositivi semplici (come PC, notebook, tablet e smartphone) che fungono essenzialmente da interfaccia al cloud (cioè, al data center) in cui sono memorizzate tutte le foto, i video, la musica e altri dati dell'utente. In questo modello, i dati sono accessibili da diversi dispositivi ovunque e in qualsiasi momento senza che l'utente debba preoccuparsi di dove siano. Qui, un data center pieno di server ha sostituito un unico grande computer centrale, ma il paradigma è ritornato di nuovo quello di un tempo: gli utenti dispongono di terminali semplici, mentre i dati e la potenza di calcolo sono centralizzati altrove.

Chi può sapere per quanto tempo questo modello sarà valido? Tra una decina d'anni potrebbe facilmente accadere che talmente tante persone avranno memorizzato così tante canzoni, foto e video nel cloud da far diventare l'infrastruttura (wireless) per la comunicazione un vero pantano. Questo potrebbe portare a una nuova rivoluzione: personal computer, dove la gente memorizza i propri dati nelle proprie macchine a livello locale, evitando così il traffico via etere.

Il messaggio da cogliere è che il modello di calcolo più diffuso in una data epoca dipende molto dalla tecnologia, dall'economia e dalle applicazioni disponibili al momento e può cambiare nel momento in cui cambiano questi fattori.

1.3.8 Mainframe

In cima alla scala di computer troviamo i mainframe, computer grandi come una stanza che rievocano gli elaboratori degli anni '60. In molti casi queste macchine sono i diretti discendenti dei mainframe IBM 360. La maggior parte di loro ha una velocità che non è molto più elevata rispetto a quella dei server più potenti, ma tutti i mainframe hanno capacità di I/O maggiori e sono spesso equipaggiati con un vasto numero di dischi, per la memorizzazione di migliaia di gigabyte di dati. Pur essendo costosi vengono ancora utilizzati per via dell'enorme investimento che rappresentano in termini di software, dati, procedure operative e personale specializzato. Molte società ritengono che sia più conveniente pagare una volta ogni tanto alcuni milioni di euro per comprarne uno nuovo, piuttosto che prendere in considerazione lo sforzo necessario a riprogrammare tutte le loro applicazioni per macchine più piccole.

È questa classe di computer che ha portato all'ormai famoso bug del millennio, provocato dal fatto che i programmati (principalmente in COBOL) degli anni '60 e '70 usavano (per risparmiare memoria) solo due cifre decimali per rappresentare gli anni. Non avrebbero mai pensato che il loro software sarebbe durato trenta o quarant'anni. Grazie all'enorme lavoro profuso per risolvere il problema, il disastro previsto non si è verificato; ciononostante molte società sono ricadute nello stesso errore aggiungendo agli anni altre due cifre decimali. Gli autori prevedono quindi che la fine della civiltà così come l'abbiamo conosciuta avverrà il 31 dicembre 9999, quando l'equivalente di 8000 anni di vecchio codice COBOL smetterà simultaneamente di funzionare.

Negli ultimi anni, Internet ha dato nuova linfa ai mainframe, fino ad allora usati principalmente per l'esecuzione di software vecchio di 40 anni. Queste macchine hanno trovato una nuova nicchia come potenti server Internet gestendo per esempio un massiccio numero di transazioni di commercio elettronico al secondo, in particolare laddove sono richieste enormi basi di dati.

Fino a poco tempo fa esisteva anche un'ulteriore categoria di computer, ancora più potente dei mainframe: i supercomputer. Avevano CPU incredibilmente veloci, molti gigabyte di memoria principale e dischi e reti ad alta velocità. Queste macchine venivano utilizzate per imponenti calcoli scientifici e ingegneristici come la simulazione della collisione fra galassie, la sintesi di nuove medicine o la simulazione del flusso di aria attorno alle ali degli aerei. Negli ultimi anni i data center costituiti da componenti commerciali hanno però offerto la stessa potenza computazionale a un costo decisamente inferiore, e i veri supercomputer sono diventati una razza in via d'estinzione.

1.4 Esempi di famiglie di computer

In questo libro focalizzeremo l'attenzione su tre ben note tipologie di architetture: x86, ARM e AVR. La prima è presente in quasi tutti i personal computer (inclusi i PC Windows e Linux, e i Mac) e suoi sistemi server. L'interesse verso i primi è motivato dal fatto che ogni lettore ne ha senza dubbio utilizzato uno, mentre quello verso i server è dovuto al fatto che questi eseguono tutti i servizi di Internet. L'architettura ARM domina il mercato mobile; per esempio, la maggior parte degli smartphone e dei tablet si basa su

questi processori. Infine, l'architettura AVR è diffusa nei microcontrollori economici presenti in molti computer integrati. I computer integrati sono invisibili agli utenti, ma controllano macchine, televisori, forni a microonde, lavatrici e praticamente qualsiasi altro dispositivo che costa più di 50 euro.

In questo paragrafo introdurremo brevemente le tre ISA (architetture dell'insieme d'istruzioni) che verranno usate come esempio nel resto di questo libro.

1.4.1 Introduzione all'architettura x86

Nel 1968 Robert Noyce, inventore dell'integrazione dei circuiti su silicio, Gordon Moore, famoso per la legge omonima, e Arthur Rock, un imprenditore di San Francisco, fondarono la Intel Corporation per produrre chip di memoria. Durante il primo anno di attività il fatturato Intel fu di soli 3000 dollari, ma da allora gli introiti sono decollati (Intel è ora il più grande produttore di CPU al mondo).

Nei tardi anni '60 i calcolatori erano grandi macchine elettromeccaniche della grandezza di una moderna stampante laser e del peso di 20 Kg. Nel settembre del 1969 una società giapponese, la Busicom, si rivolse a Intel per richiedere la produzione di 12 chip appositamente progettati per un nuovo calcolatore elettronico. Ted Hoff, l'ingegnere di Intel cui fu affidato l'incarico, analizzò il progetto e si rese conto che, per realizzare le stesse operazioni, avrebbe potuto mettere su un singolo chip una CPU a 4 bit di uso generale e che questa sarebbe stata allo stesso tempo più semplice ed economica. Così nel 1971 nacque il processore 4004, la prima CPU su un chip costituita da 2300 transistor (Faggin et al., 1996).

Occorre notare che né Intel né Busicom si resero subito conto di che cosa avessero appena realizzato. Quando Intel decise che valeva la pena provare a utilizzare il 4004 in altri progetti, propose a Busicom di ricomprare tutti i diritti, restituendole i 60.000 dollari che la società giapponese aveva pagato per lo sviluppo della CPU. L'offerta fu subito accettata e da quel momento Intel poté cominciare a lavorare su una versione a 8 bit del processore, l'8008, introdotto nel 1972. La famiglia Intel, iniziata con il 4004 e l'8008 è illustrata nella Figura 1.11 che riporta per ogni chip la data di rilascio, la frequenza di clock, il numero di transistor e la dimensione della memoria.

Intel non si aspettava una grande richiesta per l'8008 e quindi impostò una linea di produzione su un basso volume. Fra lo stupore generale, il chip riscosse un enorme interesse, tale da spingere Intel a progettare una nuova CPU che aggirasse il principale problema dell'8008, cioè il limite di 16 KB di memoria (dovuto al numero di contatti esterni del chip). Il progetto terminò nel 1974 con la nascita dell'8080, una piccola CPU di uso generale. Esattamente come avvenne per il PDP-8, questo prodotto conquistò di colpo il mercato e divenne improvvisamente un articolo con un mercato di massa. La differenza è che, invece di venderne migliaia come fece DEC, Intel riuscì a venderne milioni.

Nel 1978 comparve l'8086, un'autentica CPU a 16 bit su un solo chip. L'8086 fu progettato per essere simile all'8080, pur senza la completa compatibilità. L'8086 fu seguito dall'8088, che aveva la sua stessa architettura e poteva eseguire gli stessi programmi, ma era dotato di un bus a 8 invece che a 16 bit, rendendolo più lento e allo stesso tempo più economico dell'8086. Quando IBM scelse l'8088 come CPU per il

primo PC IBM, questo processore divenne rapidamente lo standard nel mercato dei personal computer.

Chip	Data	MHz	N. transistor	Memoria	Descrizione
4004	4/1971	0,108	2300	640	Primo microprocessore su un solo chip
8008	4/1972	0,108	3500	16 KB	Primo microprocessore a 8 bit
8080	4/1974	2	6000	64 KB	Prima CPU di uso generale su un solo chip
8086	6/1978	5-10	29.000	1 MB	Prima CPU a 16 bit su un solo chip
8088	6/1979	5-8	29.000	1 MB	Usato nel PC IBM
80286	2/1982	8-12	134.000	16 MB	Introduzione della modalità protetta
80386	10/1985	16-33	275.000	4 GB	Prima CPU a 32 bit
80486	4/1989	25-100	1,2 M	4 GB	Memoria cache da 8 KB integrata
Pentium	3/1993	60-233	3,1 M	4 GB	Due pipeline; istruzioni MMX nei modelli successivi
Pentium Pro	3/1995	150-200	5,5 M	4 GB	Cache integrata a due livelli
Pentium II	5/1997	233-450	7,5 M	4 GB	Pentium Pro con istruzioni MMX
Pentium III	2/1999	650-1400	9,5 M	4 GB	Istruzioni SSE per la grafica 3D
Pentium 4	11/2000	1300-3800	42 M	4 GB	Hyperthreading; ulteriori istruzioni SSE
Core Duo	1/2006	1600-3200	152 M	2 GB	Due core in un singolo circuito stampato
Core	7/2006	1200-3200	410 M	64 GB	Architettura quad-core a 64 bit
Core i7	1/2011	1100-3300	1160 M	24 GB	Processore grafico integrato

Figura 1.11 I membri principali della famiglia di CPU Intel. Le velocità del clock sono misurate in MHz (megahertz) dove 1 MHz corrisponde a 1 milione di cicli/s.

Né l'8088 né l'8086 potevano indirizzare più di 1 MB di memoria, il che nei primi anni '80 divenne via via un serio problema, al punto che Intel decise di progettare l'80286, una versione più potente, ma ancora compatibile con l'8086. L'insieme base d'istruzioni era essenzialmente lo stesso dell'8086 e dell'8088, ma l'organizzazione della memoria era molto differente e piuttosto complessa, a causa dei requisiti di compatibilità con gli altri processori. L'80286 fu utilizzato nel PC IBM/AT e nei modelli PS/2 di gamma media. Come l'8088 fu anch'esso un grande successo, soprattutto perché era in genere considerato come un 8088 più veloce.

Il successivo passo fu logicamente l'80386, una vera CPU a 32 bit su un chip che vide la luce nel 1985. Come il precedente, questo processore era una versione più o meno compatibile con tutto ciò che risaliva all'8080. Essendo retrocompatibile fu una benedizione per chi aveva necessità di eseguire vecchio software, ma un fastidio per chi avrebbe

preferito un'architettura moderna, semplice, ben progettata e non appesantita dagli errori e dalla tecnologia degli anni precedenti.

Quattro anni dopo apparve l'80486. Era in sostanza una versione più veloce dell'80386 con in più un'unità in virgola mobile e una memoria cache di 8 KB sul chip. La memoria cache era usata per conservare, all'interno o vicino alla CPU, le parole di memoria utilizzate con più frequenza, di modo da evitare l'accesso (lento) alla memoria centrale. L'80486 aveva anche un supporto multiprocessore predefinito per permettere ai produttori di costruire sistemi composti da più CPU che condividessero una memoria comune.

A quel punto Intel scoprì nel modo peggiore (perdendo una causa legale per violazione di diritti) che i numeri (come 80486) non potevano essere registrati come marchio, e per questo alla generazione successiva fu dato un nome: Pentium (dal termine greco "cinque", ΠΕΝΤΑ). Diversamente dall'80486, che aveva una sola pipeline interna, il Pentium ne aveva due, il che lo aiutò a essere due volte più veloce (le pipeline verranno illustrate in dettaglio nel Capitolo 2).

In seguito, durante la sua corsa produttiva, Intel aggiunse l'insieme d'istruzioni speciali MMX (*MultiMedia eXtension*, "estensioni multimediali"). Queste istruzioni furono ideate per velocizzare i calcoli necessari per l'elaborazione di audio e video, rendendo così superflua la presenza di coprocessori multimediali.

Quando apparve la generazione successiva, la gente che sperava di vedere il Sexium (*sex* significa sei in Latino) rimase delusa. Ormai il nome Pentium era talmente conosciuto che gli addetti al marketing lo vollero mantenere e il nuovo chip prese il nome di Pentium Pro.

Nonostante il piccolo cambiamento nel nome, questo processore rappresentò una maggiore discontinuità rispetto al passato. Invece di avere due o più pipeline il Pentium Pro aveva un'organizzazione interna molto diversa e poteva eseguire fino a cinque istruzioni allo stesso tempo.

Un'altra innovazione introdotta dal Pentium Pro fu la memoria cache a due livelli. Nel chip stesso del processore vi erano 8 KB di memoria veloce contenenti le istruzioni usate più di frequente e altri 8 KB dedicati in modo analogo ai dati. All'interno della stessa alloggiamento del Pentium Pro (ma non nel chip stesso) vi era una seconda memoria cache di 256 KB.

Anche se il Pentium Pro era dotato di una cache di grandi dimensioni, mancavano le istruzioni MMX (dato che Intel non riuscì a produrre un così grande chip a un costo accettabile). Quando i progressi tecnologici permisero di avere contemporaneamente sullo stesso chip sia le istruzioni MMX sia la cache, il risultato fu la nascita del Pentium II. Successivamente, per migliorare le prestazioni con la grafica 3D, furono aggiunte ulteriori istruzioni multimediali, chiamate SSE (*Streaming SIMD Extensions*, "estensioni per stream SIMD") (Raman et al., 2000). Al nuovo chip fu dato il nome di Pentium III, anche se internamente era in sostanza un Pentium II.

Il Pentium successivo, rilasciato nel novembre 2000, era basato su un'architettura interna differente, ma aveva lo stesso insieme di istruzioni dei precedenti Pentium. Per celebrare l'evento Intel passò dalla numerazione in numeri romani a quella in numeri arabi e lo chiamò Pentium 4. Come al solito il Pentium 4 era più veloce di tutti i suoi

predecessori; la versione a 3.06 GHz introdusse in più una nuova e interessante caratteristica: l'*hyperthreading*. Questa potenzialità permetteva ai programmi di dividere il loro lavoro in due flussi di controllo che il Pentium 4 poteva eseguire in parallelo, accelerandone l'esecuzione. Inoltre vennero aggiunte altre istruzioni SSE per velocizzare ulteriormente l'elaborazione audio e video.

Nel 2006 Intel ha cambiato il marchio da Pentium a Core e ha lanciato un chip dual core, il **Core 2 Duo**. Quando Intel decise di volere anche una versione a core singolo del chip, più economica, iniziò semplicemente a vendere i Core 2 Duo con un core disabilitato, perché sprecare un po' di silicio su ogni chip prodotto era in ultima analisi più economico rispetto a sostenere enormi spese per il progetto e il test di un nuovo chip prodotto da zero. La serie Core ha continuato a evolversi, con l'i3, l'i5, e l'i7 come varianti per i computer di bassa, media e alta fascia, rispettivamente, ai quali seguiranno sicuramente altri modelli. La Figura 1.12 mostra una foto del Core i7. Questo chip contiene, in realtà, otto core, ma, fatta eccezione per la versione Xeon, ne sono abilitati soltanto sei. Grazie a questo approccio un chip con uno o due core difettosi può essere ancora venduto disabilitando gli eventuali core malfunzionanti. Ogni core ha la propria cache di primo e di secondo livello, ma vi è anche una cache condivisa di livello 3 (L3), utilizzata da tutti i core. Parleremo della cache in dettaglio più avanti nel libro.

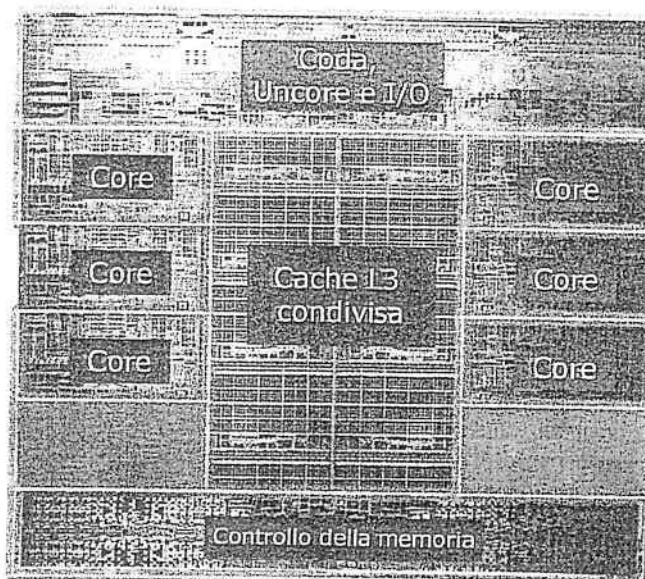


Figura 1.12 Il microprocessore Intel Core i7-3960X, © 2011 Intel Corporation. Il chip misura 21×21 mm² e contiene 2,27 miliardi di transistor (con "Uncore" si intendono le funzionalità esterne al core).

Intel, oltre alla linea principale di CPU per computer desktop di cui abbiamo parlato sinora, ha prodotto varianti di alcuni processori Pentium dedicate a mercati specifici. Nel 1998 ha introdotto una nuova linea di prodotti chiamata Celeron, in sostanza una versione più economica e con prestazioni ridotte del Pentium II, pensata per i PC di fascia bassa. Dato che l'architettura del Celeron è la stessa del Pentium II, non verrà trattata nel corso del libro. Nel giugno del 1998 Intel ha introdotto anche una versione speciale del Pentium II dedicata al mercato professionale. Questo processore, chiamato Xeon, aveva una cache più grande, un bus più veloce e un miglior supporto multiprocessore; a parte questi miglioramenti si trattava di un normale Pentium II, quindi neanche esso verrà trattato specificatamente. Anche del Pentium III è stata prodotta una versione Xeon, come per i chip più recenti. Sugli ultimi processori una delle funzionalità aggiuntive dello Xeon è la presenza di più core.

Nel 2003 Intel ha messo in commercio il Pentium M (M per Mobile), progettato per i computer portatili. Questo chip faceva parte dell'architettura Centrino, i cui obiettivi erano la riduzione del consumo energetico per prolungare la durata delle batterie, la produzione di computer più piccoli e leggeri e la predisposizione alla connessione di rete senza fili secondo lo standard IEEE 802.11 (WiFi). Il Pentium M consumava molto meno ed era più piccolo del Pentium 4, due caratteristiche che avrebbero presto permesso a lui e ai suoi successori di soppiantare la micro-architettura Pentium 4 nei futuri progetti Intel.

Tutti i processori Intel sono retròcompatibili con i loro predecessori fino all'8086. In altre parole un Pentium 4 o un Core possono eseguire programmi dell'8086 senza alcuna modifica. Per Intel questa compatibilità è sempre stata un requisito di progettazione, al fine di non far perdere agli utenti i loro investimenti in software. Il Core è però quattro ordini di grandezza più complesso dell'8086 ed è quindi naturale che possa fare molte cose in più. L'aggiunta graduale di tutte queste estensioni ha portato a un'architettura che non è così elegante come quella che si otterrebbe dando a qualcuno 42 milioni di transistor e il compito di progettarla ex novo.

È interessante notare che la legge di Moore, sebbene sia stata a lungo associata al numero di bit di memoria, risulta valida anche per i chip della CPU. Il grafico nella Figura 1.13 rappresenta in scala semi-logaritmica il numero di transistor di un chip (Figura 1.8) rispetto alla sua data di apparizione e mostra che la legge di Moore viene rispettata anche in base a questi dati.

Anche se con ogni probabilità la legge di Moore continuerà a essere valida per alcuni anni a venire, cominciano a profilarsi delle nuvole all'orizzonte per via dei problemi legati alla dissipazione del calore. L'uso di transistor più piccoli permette di raggiungere frequenze più alte, ma allo stesso tempo richiede una tensione più elevata. L'energia consumata e il calore dissipato sono proporzionali al quadrato della tensione, ragion per cui raggiungere velocità più elevate significa avere più calore da smaltire. Un processore Pentium 4 a 3,6 GHz consuma 115 W, il che significa che diventa caldo all'incirca quanto una lampadina da 100 W. Aumentare la frequenza di clock peggiora il problema.

Nel novembre del 2004 Intel ha dovuto rinunciare al Pentium 4 a 4 GHz a causa dei problemi dipendenti dalla dissipazione del calore. L'uso di ventole di grandi dimensioni può aiutare, ma il loro rumore infastidisce gli utenti e il raffreddamento ad acqua, usato nei mainframe di grandi dimensioni, non è un'opzione attualmente praticabile per i desktop (e ancor meno per quelli portatili).

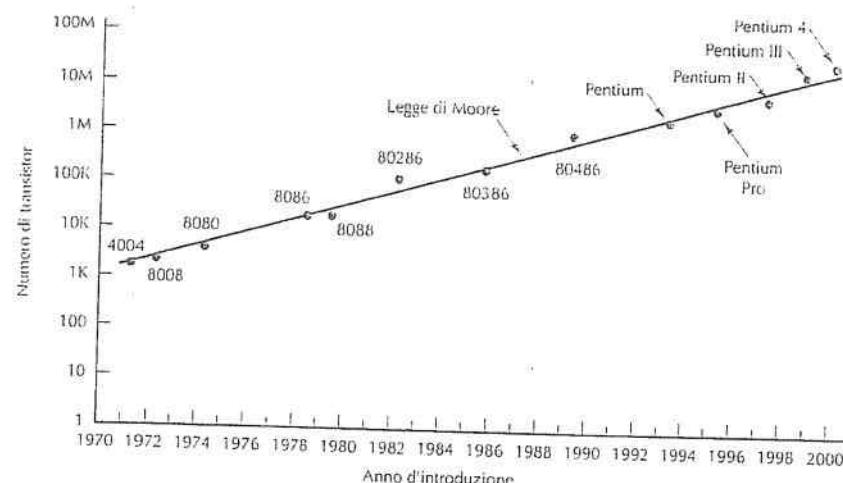


Figura 1.13 La legge di Moore per i chip della CPU (Intel).

Per questo motivo la finora inarrestabile marcia delle velocità di clock potrebbe essersi conclusa, almeno finché gli ingegneri di Intel non escogiteranno un modo per liberarsi efficientemente del calore generato. Nei progetti attuali di Intel vengono collocate due o più CPU su un singolo chip, insieme a una grande cache condivisa. Per via della relazione tra tensione e frequenza di clock, due CPU sullo stesso chip consumano molta meno potenza di una sola CPU con velocità doppia. Ne consegue che in futuro il guadagno fornito dalla legge di Moore potrebbe essere via via sfruttato per inserire nel chip un numero maggiore di core e cache sempre più grandi, piuttosto che per raggiungere velocità di clock sempre più elevate. Lo sfruttamento delle potenzialità dei multiprocessori mette i programmati di fronte a grandi sfide, perché a differenza delle sofisticate micro-architetture monoprocessoressi del passato che permettevano di ottenere migliori prestazioni dai programmi esistenti, i multiprocessori richiedono al programmatore di orchestrare in modo esplicito l'esecuzione parallela, utilizzando thread, semafori, memoria condivisa e altre tecnologie portatrici di bug... e di mal di testa.

1.4.2 Introduzione all'architettura ARM

Nei primi anni '80, la società britannica Acorn Computer, facendo seguito al successo ottenuto dal loro personal computer a 8 bit BBC Micro, iniziò a lavorare su una seconda macchina con la speranza di competere con il PC IBM che da poco era apparso sul mercato. La BBC Micro era basata sul processore a 8-bit 6502. Steve Furber e i suoi colleghi della Acorn ritenevano che il 6502 non aveva abbastanza muscoli per competere con il PC IBM a 16 bit 8086; iniziarono così a guardare le opzioni offerte dal mercato e decisero che erano troppo limitate.

Ispirati dal progetto Berkeley RISC, in cui un piccolo team aveva progettato un processore molto veloce (che alla fine portò all'architettura SPARC), decisero di costruire per il progetto una propria CPU e lo chiamarono Acorn RISC Machine (o ARM, che sarà poi ribattezzata Advanced RISC machine dopo la separazione di ARM da Acorn). Il progetto fu completato nel 1985, aveva istruzioni e dati a 32 bit e uno spazio di indirizzamento a 26 bit, ed era prodotto dalla VLSI Technology.

La prima architettura ARM (chiamata ARM2) fece la sua apparizione nel personal computer Acorn Archimedes. Questo PC era molto veloce ed economico per i suoi tempi, viaggiava a 2 MIPS (milioni di istruzioni al secondo) e costava solo 899 sterline inglese al momento del lancio. La macchina ebbe una grande diffusione nel Regno Unito, in Irlanda, in Australia e in Nuova Zelanda, in particolare nelle scuole.

Dopo il successo di Archimedes, Apple contattò Acorn per sviluppare un processore ARM per l'imminente progetto Apple Newton, il primo computer palmare. Per focalizzarsi meglio sul progetto, il team di ARM lasciò Acorn per creare una nuova società, denominata Advanced RISC Machines (ARM). Il nuovo processore ARM fu chiamato ARM 610 e alimentava l'Apple Newton nel momento del suo lancio, nel 1993. A differenza del progetto ARM originale, questo nuovo processore ARM integrava 4 KB di memoria cache, migliorando significativamente le prestazioni. Sebbene l'Apple Newton non sia stato un grande successo, l'ARM 610 ha visto altre applicazioni vincenti, tra cui il PC RISC Acorn.

Nella metà degli anni '90, ARM collaborò con Digital Equipment Corporation per sviluppare una versione di ARM ad alta velocità e basso consumo, destinata ad applicazioni mobili che non potevano permettersi alti consumi, come i PDA. Insieme realizzarono il progetto StrongARM, che fin dalla sua prima apparizione si mise in evidenza nel settore per la sua alta velocità (233 MHz) e per la bassissima potenza richiesta (1 Watt). Questa efficienza è stata ottenuta attraverso un progetto semplice ed elegante che includeva due cache da 16 KB per istruzioni e dati. Il processore StrongARM e i suoi successori DEC hanno avuto un discreto successo sul mercato, trovando applicazione in un buon numero di PDA, set-top box, dispositivi multimediali e router.

Il progetto ARM più degno di attenzione è probabilmente l'ARM7, rilasciato da ARM nel 1994 e ancora oggi largamente utilizzato. Il progetto utilizzava cache separate per istruzioni e dati, e incorporava l'insieme di istruzioni a 16-bit denominato Thumb. Questo insieme di istruzioni è una versione ristretta dell'intero insieme di istruzioni ARM a 32 bit che consente ai programmati di codificare molte delle operazioni più comuni in piccole istruzioni a 16 bit, riducendo in modo significativo la quantità di memoria utilizzata dai programmi. Il processore ha ben figurato in una vasta gamma di applicazioni embedded di bassa-media fascia, quali tostapane e controllo motori, ma anche nella console portatile Nintendo Game Boy Advance.

A differenza di molte società di computer, ARM non produce nessuno dei suoi microprocessori, ma crea piuttosto progetti, librerie e strumenti di sviluppo per ARM, che concede in licenza a progettisti di sistemi e produttori di chip. Per esempio, la CPU utilizzata nel tablet Android Samsung Galaxy Tab è un processore basato su ARM. Il Galaxy Tab utilizza il processore system-on-chip Tegra 2, che contiene due processori ARM Cortex-A9 e una unità di elaborazione grafica Nvidia GeForce. I core del Tegra 2

sono stati progettati da ARM, integrati in un SoC di progettazione Nvidia e prodotti dalla Taiwan Semiconductor Manufacturing Company (TSMC). Si tratta di una impressionante collaborazione fra società di paesi diversi in cui tutte le aziende hanno contribuito al valore finale del prodotto.

La Figura 1.14 mostra una foto dello stampo del SoC Tegra 2 di Nvidia. Il progetto contiene tre processori ARM: due core ARM Cortex-A9 a 1.2-GHz più un core ARM7. I Cortex-A9 sono core a doppia emissione, con esecuzione fuori sequenza, dotati di 1 MB di cache L2 e di supporto per multiprocessing con memoria condivisa. Abbiamo citato molte parole chiave che incontreremo nei prossimi capitoli. Per ora, è sufficiente sapere che queste caratteristiche rendono il design molto veloce! L'ARM7 è un core ARM più vecchio e più piccolo utilizzato per la configurazione del sistema e la gestione del consumo energetico. La grafica è gestita da un'unità di elaborazione grafica (GPU) GeForce a 333 MHz ottimizzata per il funzionamento a bassa potenza. Sono inoltre inclusi nel chip Tegra 2 encoder/decoder video, un processore audio e un interfaccia video HDMI.

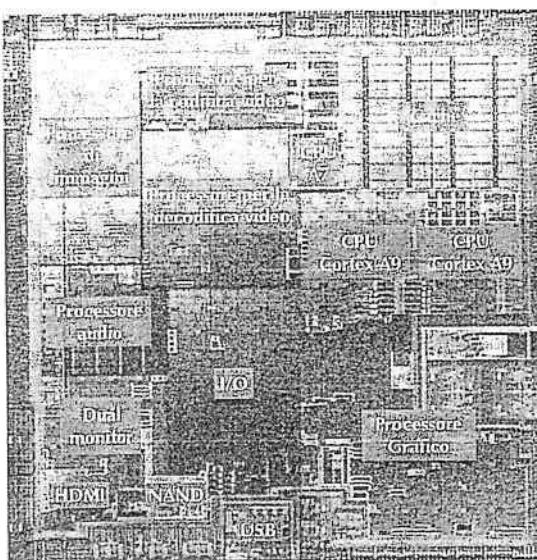


Figura 1.14 Il SoC Nvidia Tegra 2. Copyright 2011, Nvidia Corporation.
Fotografia pubblicata con autorizzazione.

L'architettura ARM ha avuto grande successo nel mercato dei dispositivi a basso consumo, mobili e integrati. Nel gennaio 2011, ARM ha annunciato di aver venduto 15 miliardi di processori dall'inizio della sua attività e ha comunicato che le vendite continuavano a crescere. Anche se creata su misura per i mercati di fascia bassa, l'architettura

ARM ha le capacità computazionali necessarie per fare bene in qualunque mercato e ci sono indizi che i suoi orizzonti si stiano espandendo. Per esempio, nel mese di ottobre 2011 è stato annunciato un processore ARM a 64-bit, mentre nel gennaio 2011 Nvidia ha annunciato il "Progetto Denver", un system-on-a-chip basato su ARM in fase di sviluppo destinato ai server e ad altri mercati. Il progetto utilizzerà più processori ARM a 64 bit e una GPU a uso generale (GPGPU). Gli aspetti del progetto rivolti al basso consumo contribuiranno a ridurre le esigenze di raffreddamento in *server farm* "fattoria di server" e data center.

1.4.3 Introduzione all'architettura AVR

Il nostro terzo esempio è molto diverso dal primo (l'architettura x86, utilizzata in personal computer e server) e dal secondo (l'architettura ARM, utilizzati in PDA e smartphone). Si tratta dell'architettura AVR ed è utilizzata in sistemi integrati di fascia bassa. La storia di AVR iniziò nel 1996 al Norwegian Institute of Technology, dove gli studenti Alf Egil-Bogen e Vegard Wollan progettarono una CPU RISC a 8-bit chiamata AVR. È stato riportato che il nome deriva da "(A)lf e (V)egard (R)ISC processor". Poco dopo che il progetto fu completato, Atmel lo acquistò e diede vita alla Atmel Norway, dove due architetti continuarono a perfezionare la progettazione del processore. Atmel lanciò il suo primo microcontrollore AVR, l'AT90S1200, nel 1997. Per facilitare la sua adozione da parte dei progettisti di sistemi, fecero in modo che la piedinatura (*pinout*) fosse esattamente la stessa di quella del processore Intel 8051, uno dei microcontrollori più popolari del momento. Oggi c'è un grande interesse per l'architettura AVR, perché è il cuore della diffusa piattaforma integrata open source Arduino.

L'architettura AVR viene realizzata in tre classi di microcontrollori, come mostra la Figura 1.15. La classe più bassa, la tinyAVR, è progettata per le applicazioni con maggiori vincoli in termini di spazio, potenza e costi. Essa comprende una CPU a 8-bit, il supporto di base per I/O digitale e il supporto per un ingresso analogico (per esempio, la lettura della temperatura da un termostato). Il tinyAVR è così piccolo che i suoi pin devono assolvere un doppio compito; possono cioè essere riprogrammati in fase di esecuzione per una qualsiasi delle funzioni digitali o analogiche supportate dal microcontrollore. Il megaAVR, che si trova nel sistema open-source Arduino, è dotato anche di supporto per l'I/O seriale, di orologi interni e di uscite analogiche programmabili. Il top della gamma, in questa fascia bassa, è il microcontrollore AVR XMEGA, che incorpora anche un acceleratore per operazioni di crittografia e il supporto integrato per interfacce USB.

Chip	Flash	EEPROM	RAM	Pin	Caratteristiche
tinyAVR	0,5–16 KB	0–512B	32–512B	6–32	Molto piccolo, I/O digitale, ingresso analogico
megaAVR	8–256 KB	0,5–4 KB	0,25–8 KB	28–100	Molte periferiche, uscita analogica
AVR XMEGA	16–256 KB	1–4 KB	2–16 KB	44–100	Crittografia accelerata, USB

Figura 1.15 Classi di microcontrollori della famiglia AVR.

Insieme a varie periferiche aggiuntive, ogni classe di processori AVR include varie risorse di memoria. I microcontrollori montano generalmente tre tipi di memoria: flash, EEPROM e RAM. La memoria flash è programmabile con l'ausilio di un'interfaccia esterna e di alte tensioni: è qui dove il codice del programma e i dati sono memorizzati. La RAM Flash è non volatile, quindi mantiene ciò che in essa è stato scritto anche se il sistema viene spento. Come la memoria flash, anche la EEPROM è non volatile, ma a differenza della RAM flash, può essere modificata dal programma durante l'esecuzione. In questa memoria un sistema integrato mantiene le informazioni di configurazione dell'utente, per esempio l'impostazione della visualizzazione dell'ora in formato 12 o 24 ore. Infine, la RAM è la memoria in cui sono mantenute le variabili dei programmi in esecuzione. Questa memoria è volatile, quindi ogni valore memorizzato viene perso quando si toglie l'alimentazione al sistema. Studieremo in dettaglio le tipologie di RAM, volatili e non volatili, nel prossimo capitolo.

La ricetta per il successo nel business dei microcontrollori è di stipare nel chip tutto quello di cui ci potrebbe essere bisogno (anche il lavello della cucina, se si riuscisse a ridurlo a un millimetro quadrato) e poi inserirlo in un supporto economico, di piccole dimensioni e con pochi pin. Integrando molte funzioni nel microcontrollore sarà possibile farlo lavorare su diverse applicazioni, rendendolo piccolo ed economico potrà essere utilizzato in diversi contesti. Per avere un'idea di quante cose vengono cablate in un microcontrollore moderno, diamo uno sguardo alle periferiche incluse nell'Atmel megaAVR-168.

1. Tre timer (due a 8 bit e uno a 16).
2. Orologio in tempo reale con oscillatore.
3. Sei canali di modulazione in ampiezza di impulsi usati, per esempio, per controllare l'intensità luminosa o la velocità del motore.
4. Otto canali di conversione analogico/digitale utilizzati per leggere i livelli di tensione.
5. Ricetrasmettitore seriale universale.
6. Interfaccia seriale I2C, uno standard comune per l'interfacciamento ai sensori.
7. Timer "Watchdog" programmabile che rileva quando il sistema è bloccato.
8. Comparatore analogico integrato che confronta due tensioni di ingresso.
9. Rilevatore di sbalzi di tensione che interrompe il sistema quando viene a mancare l'alimentazione.
10. Oscillatore interno programmabile per pilotare il clock della CPU.

1.5 Unità metriche

Per evitare ogni tipo di confusione, vale la pena precisare che in questo libro, come di solito avviene nell'informatica, si usano le unità metriche e non le tradizionali unità di misura inglesi.

Nella Figura 1.16 sono elencati i principali prefissi metrici. Di solito sono abbreviati utilizzando le loro lettere iniziali e si usano lettere maiuscole per i multipli, come KB,

MB, e così via, e minuscole per i sottomultipli. Quindi, per esempio, una linea di comunicazione a 1 Mbps trasmette 10^6 bit/s e un clock a 100 ps ha un tic ogni 10^{-10} secondi. Dato che sia *milli* sia *micro* iniziano con la lettera "m" si utilizza "m" per milli e " μ " (la lettera greca mu) per micro.

Occorre inoltre sottolineare che, nella pratica comune, queste unità hanno un significato leggermente diverso quando vengono utilizzate come misure della dimensione di memorie, dischi, file e database. In questi casi, dato che le dimensioni delle memorie sono sempre potenze di due, K assume il significato di 2^{10} cioè 1024 invece che di $10^3 = 1000$. Quindi una memoria da 1 KB contiene 1024 e non 1000 byte. Analogamente una memoria da 1 MB contiene $2^{20} = 1.048.576$ byte, una memoria da 1 GB ne contiene $2^{30} = 1.073.741.824$, e un database da 1 Tbyte contiene $2^{40} = 1.099.511.627.776$ byte. Tuttavia, una linea di comunicazione a 1 Kbps può trasmettere 1000 bit al secondo e una LAN a 10 Mbps funziona a 10.000.000 bit/s, dato che questi tassi non sono potenze di due. Sfortunatamente molti tendono a confondere questi due sistemi, specialmente per le dimensioni dei dischi. Per eliminare queste ambiguità le organizzazioni di standardizzazione hanno introdotto i nuovi termini kibibyte per 2^{10} byte, mebibyte per 2^{20} byte, gibibyte per 2^{30} byte e tebibyte per 2^{40} byte, ma l'industria sta rallentando il processo di adozione di questa terminologia. Noi pensiamo che fino a quando queste nuove parole non diventano di uso comune sia meglio continuare a usare i simboli KB, MB, GB e TB per indicare rispettivamente 2^{10} , 2^{20} , 2^{30} e 2^{40} byte e i simboli Kbps, Mbps, Gbps e Tbps per indicare rispettivamente 10^3 , 10^6 , 10^9 e 10^{12} bit/s.

Esp.	Valore esplicito	Prefisso	Esp.	Valore esplicito	Prefisso
10^{-3}	0.001	milli	10^3	1.000	Kilo
10^{-6}	0.000001	micro	10^6	1.000.000	Mega
10^{-9}	0.000000001	nano	10^9	1.000.000.000	Giga
10^{-12}	0.00000000001	pico	10^{12}	1.000.000.000.000.000	Tera
10^{-15}	0.0000000000001	femto	10^{15}	1.000.000.000.000.000.000	Peta
10^{-18}	0.000000000000001	atto	10^{18}	1.000.000.000.000.000.000.000	Exa
10^{-21}	0.00000000000000001	zepto	10^{21}	1.000.000.000.000.000.000.000.000	Zetta
10^{-24}	0.0000000000000000001	yocto	10^{24}	1.000.000.000.000.000.000.000.000.000	Yotta

Figura 1.16 Principali prefissi metrici.

1.6 Organizzazione del libro

Questo libro tratta dei computer multilivello (che comprendono praticamente tutti i computer moderni) e del modo in cui sono organizzati. Verranno esaminati in notevole dettaglio quattro livelli: il livello logico digitale, il livello di micro-architettura, il livello ISA e il livello macchina del sistema operativo. Alcuni degli aspetti che esamineremo saranno la progettazione complessiva del livello (e il motivo per cui è stato progettato in quel modo), i tipi d'istruzioni e dati disponibili, l'organizzazione e l'indirizzamento della memoria e il metodo tramite il quale il livello è stato implementato. Lo studio di questi argomenti, e altri simili, è chiamato organizzazione, o architettura, del computer.

Ci preoccuperemo principalmente dei concetti piuttosto che dei dettagli e degli aspetti matematici. Per questa ragione alcuni degli esempi saranno fortemente semplificati, al fine di porre l'enfasi sulle idee centrali.

Nel corso del libro useremo le architetture x86, ARM e AVR come esempi pratici, in modo da far capire come i principi presentati nel testo possono essere, e sono, messi in pratica nella realtà. Questi tre esempi sono stati scelti in base a molteplici ragioni. In primo luogo sono tutti ampiamente utilizzati e con ogni probabilità il lettore può avere accesso ad almeno uno di loro. In secondo luogo, ciascuno ha una sua propria e determinata architettura, il che pone le basi per effettuare confronti e incoraggia un approccio basato su domande come: "Quali sono le alternative?". Spesso i libri che trattano di una sola macchina lasciano al lettore la sensazione che gli sia stata svelata "l'unica e corretta progettazione di un computer"; ciò è assurdo alla luce dei molti compromessi e delle decisioni arbitrarie che un progettista è obbligato a compiere. Si incoraggia il lettore a studiare questi computer, ed eventualmente altri, con un occhio critico; lo si spinge a cercare di capire il motivo per cui le cose sono in un determinato modo, oltre che a chiedersi come sarebbero potute essere diversamente, invece di accettarle passivamente.

Occorre chiarire fin dall'inizio che questo libro non spiega come programmare le architetture x86, ARM o AVR. Quando sarà necessario, queste macchine saranno usate a scopo dimostrativo, senza alcuna pretesa di essere completi. I lettori interessati a un'introduzione esauriente a una di queste macchine dovrebbero consultare le pubblicazioni specifiche.

Il Capitolo 2 è un'introduzione ai componenti base dei computer: processori, memorie e dispositivi di input/output. Si intende dare una panoramica dell'architettura di un sistema oltre a un'introduzione ai capitoli successivi.

I Capitoli da 3 a 6 trattano individualmente i livelli mostrati nella Figura 1.2. La trattazione sarà effettuata dal basso verso l'alto, dato che le macchine sono state tradizionalmente progettate in questo modo.

La progettazione di un generico livello è in gran parte determinata dalle proprietà del livello sottostante e per questo è difficile capire un livello a meno di non avere prima compreso a fondo i precedenti. Inoltre sembra più istruttivo procedere dai livelli più semplici fino a quelli più complessi, e non viceversa.

Il Capitolo 3 riguarda il livello logico digitale, il vero e proprio hardware della macchina. Si spiega che cosa sono le porte logiche e come possono essere combinate per formare circuiti funzionali. Si introduce anche l'algebra booleana, uno strumento per l'analisi dei circuiti digitali. Vengono spiegati inoltre i bus del computer e in particolare il popolare bus PCI. Nel capitolo vengono presentati numerosi esempi tratti dal mondo reale, tra cui i tre casi di studio sopracitati.

Il Capitolo 4 introduce l'architettura e il controllo del livello di micro-architettura. Ricorrendo anche a vari esempi ci si concentrerà sulla funzione principale di questo livello, che consiste nell'interpretare le istruzioni del livello 2 per quello superiore. Il capitolo contiene anche una trattazione del livello di micro-architettura di alcune macchine reali.

Il Capitolo 5 esamina il livello ISA, quello che molti rivenditori di computer definiscono come il linguaggio macchina. In questo capitolo analizzeremo in dettaglio le tre macchine d'esempio.

Il Capitolo 6 si occupa delle istruzioni, dell'organizzazione della memoria e dei meccanismi di controllo presenti nel livello macchina del sistema operativo. Gli esempi utilizzati sono Windows (diffuso sui sistemi desktop basati su x86) e UNIX, usato su molti sistemi basati su x86 e ARM.

Il Capitolo 7 riguarda il livello del linguaggio assemblativo e tratta sia del linguaggio sia del processo di assemblaggio. Viene inoltre introdotto l'argomento del collegamento (*linking*).

Il Capitolo 8 tratta dei computer paralleli, un argomento la cui importanza al giorno d'oggi è sempre più grande. Alcuni di questi computer hanno più CPU che condividono una memoria comune. Altri invece sono costituiti da varie CPU, ma senza condivisione di memoria.

Alcuni sono supercomputer, altri sono sistemi su singolo chip, mentre altri ancora sono cluster.

Il Capitolo 9 contiene un elenco alfabetico di riferimenti bibliografici. Le letture consigliate sono riportate sul sito web del libro, all'indirizzo: www.prenhall.com/tanenbaum.

PROBLEMI

1. Si spieghino con parole proprie i seguenti termini:
 - a. traduttore (compilatore)
 - b. interprete
 - c. macchina virtuale.
2. Ha significato che un compilatore genera output per il livello di micro-architettura invece che per il livello ISA? Si analizzino i pro e i contro di tale ipotesi.
3. È possibile immaginare un computer multilivello in cui il livello dei dispositivi e i livelli logico digitali non siano i livelli più bassi? Si motivi la risposta.
4. Si consideri un computer in cui tutti i livelli siano diversi. Ciascun livello possiede istruzioni che sono m volte più potenti di quelle del livello sottostante; cioè un'istruzione del livello r può compiere il lavoro di m istruzioni del livello $r - 1$. Se un programma del livello l richiede k secondi per essere eseguito, quanto tempo impiegheranno gli equivalenti programmi dei livelli 2, 3 e 4, assumendo che siano necessarie n istruzioni del livello r per interpretare una singola istruzione del livello $r + 1$?
5. Alcune istruzioni del livello macchina del sistema operativo sono identiche alle istruzioni del linguaggio ISA. Queste istruzioni sono eseguite direttamente dal microprogramma o dall'hardware invece che dal sistema operativo. Alla luce della risposta data al problema precedente, perché questo avviene?
6. Si consideri un computer con interpreti identici ai livelli 1, 2 e 3. Un interprete impiega n istruzioni per prelevare, esaminare ed eseguire un'istruzione. Se un'istruzione del livello l richiede k nanosecondi per essere eseguita, quanto tempo impiega un'istruzione ai livelli 2, 3 e 4?
7. In che senso l'hardware e il software sono equivalenti? E in che senso non lo sono?
8. La *difference engine* di Babbage aveva un unico programma, fisso, che non poteva essere modificato. È essenzialmente la stessa cosa di un moderno CD-ROM il cui contenuto non può essere cambiato? Si motivi la risposta.
9. Una delle conseguenze dell'idea di von Neumann di memorizzare i programmi in memoria è che anch'essi possono essere modificati, esattamente come i dati. È possibile immaginare un esempio in cui questa funzione potrebbe essere utile? (Suggerimento: si pensi a operazioni aritmetiche sugli array).