CHAPTER

1

INTRODUCTION TO PARALLEL PROCESSING

Basic concepts of parallel processing on high-performance computers are introduced in this chapter. We will review the architectural evolution, examine various forms of concurrent activities in modern computer systems, and assess advanced applications of parallel processing computers, Parallel computer structures will be characterized as *pipelined computers, array processors, and multiprocessor systems.* Several new computing concepts, including data flow and VLSI approaches, will be introduced. The material presented in this introductory chapter will provide an overview of the field and pave the way to studying in subsequent chapters the details of theories of parallel computing, machine architectures, system controls, fast algorithms, and programming requirements.

1.1 EVOLUTION OF COMPUTER SYSTEMS

Over the past four decades the computer industry has experienced four generations of development, physically marked by the rapid changing of building blocks from relays and vacuum tubes (1940-1950s) to discrete diodes and transistors (1950-1960s), to small- and medium-scale integrated (SSI/MSI) circuits (1960-1970s), and to large- and very-large-scale integrated (LSI/VLSI) devices (1970s and beyond). Increases in device speed and reliability and reductions in hardware cost and physical size have greatly enhanced computer performance. However, better devices are not the sole factor contributing to high performance. Ever since the stored-program concept of von Neumann, the computer has been recognized as more than just a hardware organization problem. A modern computer system is really a composite of such items as processors, memories, functional units, interconnection networks, compilers, operating systems, peripheral devices, communication channels, and database banks.

To design a powerful and cost-effective computer system and to devise efficient programs to solve a computational problem, one must understand the underlying hardware and software system structures and the computing algorithms to be implemented on the machine with some user-oriented programming languages. These disciplines constitute the technical scope of *computer architecture*. Computer architecture is really a system concept integrating hardware, software, algorithms, and languages to perform large computations. A good computer architect should master all these disciplines. It is the revolutionary advances in integrated circuits and system architecture that have contributed most to the significant improvement of computer performance during the past 40 years. In this section, we review the generations of computer systems and indicate the general trends in the development of high performance computers.

1.1.1 Generations of Computer Systems

The division of computer systems into generations is determined by the device technology, system architecture, processing mode, and languages used. We consider each generation to have a time span of about 10 years. Adjacent generations may overlap in several years as demonstrated in Figure 1.1. The long time span is intended to cover both development and use of the machines in various parts of the world. We are currently in the fourth generation, while the fifth generation is not materialized yet.

The first generation (1938-1953): The introduction of the first electronic analog computer in 1938 and the first electronic digital computer, ENIAC (Electronic Numerical Integrator and Computer), in 1946 marked the beginning of the first generation of computers. Electromechanical relays were used as switching

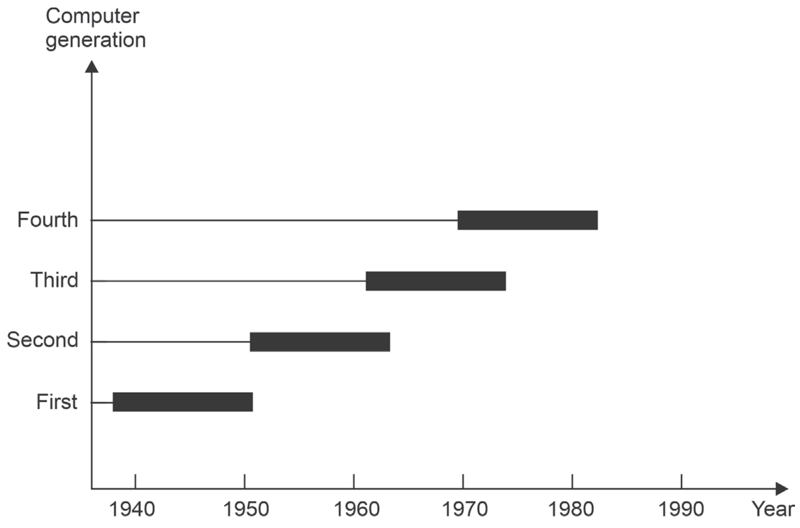

Figure 1.1: *The evolution of computer systems.*

devices in the 1940s, and vacuum tubes were used in the 1950s. These devices were interconnected by insulated wires. Hardware components were expensive then, which forced the CPU structure to be bit-serial: arithmetic is done on a bit-by-bit fixed-point basis, as in a ripple-carry addition which uses a single full adder and one bit of carry flag.

Only binary-coded machine language was used in early computers. In 1950, the first stored-program computer, EDVAC (Electronic Discrete Variable Automatic Computer), was developed. This marked the beginning of the use of system software to relieve the user's burden in low-level programming. However, it is not difficult to imagine that hardware costs predominated and software language features were rather primitive in the early computers. By 1952, IBM had announced its 701 electronic calculator. The system used Williams' tube memory, magnetic drums, and magnetic tape.

The second generation (1952-1963): Transistors were invented in 1948. The first transistorized digital computer, TRADIC, was built by Bell Laboratories in 1954. Discrete transistors and diodes were the building blocks: 800 transistors were used in TRADIC. Printed circuits appeared. By this time, coincident current magnetic core memory was developed and subsequently appeared in many machines. Assembly languages were used until the development of high-level languages, Fortran (formula translation) in 1956 and Algol *(algorithmic language)* in 1960.

In 1959, Sperry Rand built the Larc system and IBM started the Stretch projeet. These were the first two computers attributable to architectural improvement. The Larc had an independent I/O processor which operated in parallel with one or two processing units. Stretch featured instruction lookahead and error correction, to be discussed in Section 1.2. The first IBM scientific, transistorized computer, IBM 1620, became available in 1960. Cobol (common business oriented language) was developed in 1959. Interchangeable disk packs were introduced in 1963. Batch processing was popular, providing sequential execution of user programs, one at a time until done.

The third generation (1962-1975) This generation was marked by the use of small-scale integrated (SSI) and medium-scale integrated (MSI) circuits as the basic building blocks. Multilayered printed circuits were used. Core memory was still used in CDC-6600 and other machines but by 1968, many fast computers, like CDC-7600, began to replace cores with solid-state memories. High-level languages were greatly enhanced with intelligent compilers during this period.

Multiprogramming was well developed to allow the simultaneous execution of many program segments interleaved with I/O operations. Many high-performance computers, like IBM 360/91, Illiac-IV, TI-ASC, Cyber-175, STAR-100, and C.mmp, and several vector processors were developed in the early seventies. Time-sharing operating systems became available in the late 1960s. Virtual memory was developed by using hierarchically structured memory systems.

The fourth generation (1972-present) The present generation computers emphasize the use of large-scale integrated (LSI) circuits for both logic and memory sections. High-density packaging has appeared. High-level languages are being extended to handle both scalar and vector data like the extended Fortran in many vector processors. Most operating systems are time-sharing, using virtual memories. Vectorizing compilers have appeared in the second generation of vector machines, like the Cray-1 (1976) and the Cyber-205 (1982). High-speed mainframes and supers appear in multiprocessor systems like the Univac 1100/80 (1976), Fujitsu M 382 (1981), the IBM 370/168 MP, the IBM 3081 (1980), the Burroughs 8-7800 (1978), and the Cray X-MP (1983). A high degree of pipe lining and multiprocessing is greatly emphasized in commercial supercomputers. A massively parallel processor (MPP) was custom-designed in In2. This MPP consisting of 16,384 bit-slice microprocessors, is under the control of one array controller for satellite image processing.

The future Computers to be used in the 1990s may be the next generation. Very-large-scale integrated (VLSI) chips will be used along with high-density modular design. Multiprocessors like the 16 processors in the S-I project at Lawrence Livermore National Laboratory and in the Denelcor's HEP will be required. Cray-2 is expected to have four processors, to be delivered in 1985. More than 1000 *mega float-point operations per second* (megaflops) are expected in these future supercomputcrs. We will study major existing systems and discuss possible future machines in subsequent chapters.

1.1.2 Trends Towards Parallel Processing

According to Sidney Fernbach: *"Today's large computers (mainframes) would have been considered 'supercomputers' 10 to 20 years ago. By the same token, today supercomputer will be considered 'state-of-art' standard equipment 10 to 20 years from now"* from an application point of view, the mainstream usage of computers is experiencing a trend of four ascending levels of sophistication:

- Data processing
- Information processing
- Knowledge processing
- Intelligence processing

The relationships between data, information, knowledge, and intelligence are demonstrated in Figure 1.2. The data space is the largest, including numeric numbers in various formats, character symbols, and multidimensional measures. Data objects are considered mutually unrelated in the space. Huge amounts of data are being generated daily in all walks of life, especially among the scientific. business, and government sectors. An information item is a collection of data objects that are related by some syntactic structure or relation. Therefore, information items form a subspace of the data space. Knowledge consists of information items plus some semantic meanings. Thus knowledge items form a

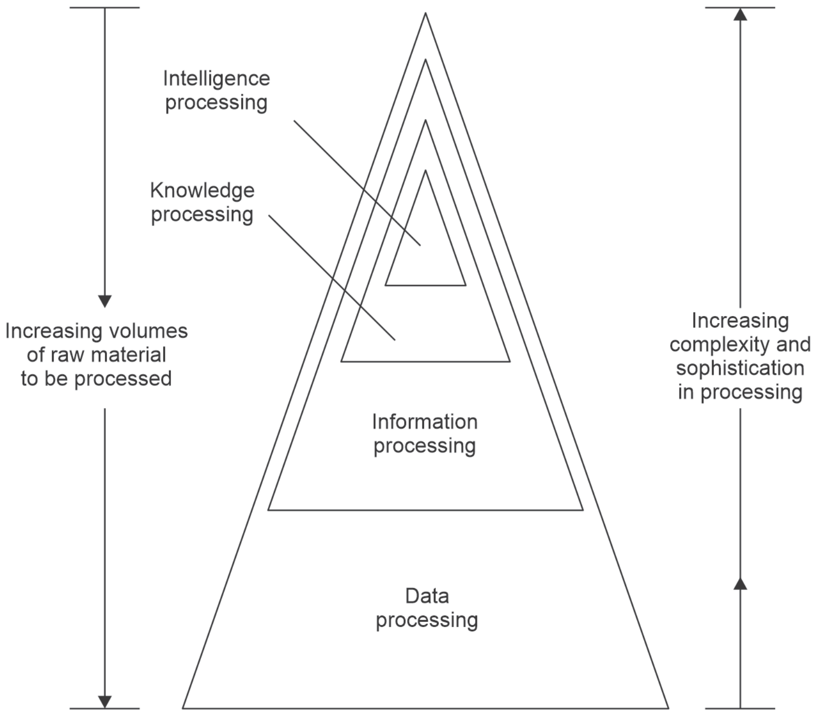
Intelligence processing
Knowledge processing
Increasing volumes of raw material to be processed
Information processing
Data processing
Increasing complexity and sophistication in processing

Figure 1.2: *The spaces of data, information, knowledge, and intelligence from the viewpoint of computer processing.*

subspace of the information space. Finally, intelligence is derived from a collection of knowledge items. The intelligence space is represented by the innermost and highest triangle in the Venn diagram.

Computer usage started with *data processing*, which is still a major task o today's computers. With more and more data structures developed, many users are shifting to computer roles from pure data processing (mainly number crunching) to *information processing*. Most of today's computing is still confined within these two processing levels. A high degree of parallelism has been found at these levels. As the accumulated knowledge bases expanded rapidly in recent years, there grew a strong demand to use computers for *knowledge processing*. For example, the various expert computer systems listed in Table 1.1 are used for problem solving in specific areas where they can reach a level of performance comparable to that of human experts. It has been projected by some computer scientists that knowledge processing will be the main thrust of computer usage in the 1990s.

Today's computers can be made very knowlegeable but are far from being intelligent. Intelligence is very difficult to create; its processing even more so. Today's computers are very fast and obedient and have many reliable memory cells to be qualified for data-information-knowledge processing. But none of the

Table 1.1: *Some existing expert computer systems for knowledge processing*

System name	Expertise
AQ11	Diagnosis of plant diseases
Internist, casnet	Medical consulting
Dendral	Hypothesizing molecular structure from mass spectrograms
Dipmeter, advisor	Oil exploration
EL	Analyzing electrical circuits
Macsyma	Mathematical manipulation
Prospector	Mineral exploration
RI	Computer configuration
SPERIL	Earthquake damage estimation

existing computers can be considered a really intelligent thinking system. Computers are still unable to communicate with human beings in natural forms like speech and written languages, pictures and images, documents, and illustrations. Computers are far from being satisfactory in performing theorem proving, logical inference, and creative thinking. We are in an era which is promoting the use of computers not only for conventional data-information processing, but also toward the building of workable machine knowledge-intelligence systems to advance human civilization. Many computer scientists reel that the degree of parallelism exploitable at the two highest processing levels should be higher than that at the data-information processing levels.

From an operating system point of view, computer systems have improved chronologically in four phases:

- Batch processing
- Multiprogramming
- Time sharing
- Multiprocessing

In these four operating modes, the degree of parallelism increases sharply from phase to phase. The general trend is to emphasize parallel processing of information. In what follows, the term information is used with an extended meaning to include data, information, knowledge, and intelligence. We formally define parallel processing as follows:

> **Definition** Parallel processing is an efficient form of information processing which emphasizes the exploitation of concurrent events in the computing process. Concurrency implies parallelism, simultaneity, and pipelining. Parallel events may occur in multiple resources during the same time interval; simultaneous events may occur at the same time instant; and pipelined events may occur in overlapped time spans. These concurrent events are attainable in a computer system at various processing levels. Parallel processing demands concurrent execution of many programs in the computer. It is in

contrast to sequential processing. It is a cost-effective means to improve system performance through concurrent activities in the computer.

The highest level of parallel processing is conducted among multiple jobs or programs through multiprogramming, time sharing, and multiprocessing. This level requires the development of parallel processable algorithms. The implementation of parallel algorithms depends on the efficient allocation of limited hardware-software resources to multiple programs being used to solve a large computation problem. The next highest level of parallel processing is conducted among procedures or tasks (program segments) within the same program. This involves the decomposition of a program into multiple tasks. The third level is to exploit concurrency among multiple instructions. Data dependency analysis is often performed to reveal parallelism among instructions. Vectorization may be desired among scalar operations within DO loops. Finally, we may wish to have faster and concurrent operations within each instruction. To sum up, parallel processing can be challenged in four programmatic levels:

- Job or program level
- Task or procedure level
- Interinstruction level
- Intrainstruction level

The highest job level is often conducted algorithmically. The lowest intra-instruction level is often implemented directly by hardware means. Hardware roles increase from high to low levels. On the other hand, software implementations increase from low to high levels. The trade-off between hardware and software approaches to solve a problem is always a very controversial issue. As hardware cost declines and software cost increases, more and more hardware methods are replacing the conventional software approaches. The trend is also supported by the increasing demand for a faster real-time, resource-sharing, and fault-tolerant computing environment.

The above characteristics suggest that parallel processing is indeed a combined field of studies. It requires a broad knowledge of and experience with all aspects of algorithms, languages, software, hardware, performance evaluation, and computing alternatives. This book concentrates on parallel processing with centralized computing facilities. Distributed processing on physically dispersed and loosely coupled computer networks is beyond the scope of this book, though a high degree of concurrency is often exploitable in distributed systems.

Parallel processing and distributed processing are closely related. In some cases, we use certain distributed techniques to achieve parallelism. As data communications technology advances progressively, the distinction between parallel and distributed processing becomes smaller and smaller. In this extended sense, we may view distributed processing as a form of parallel processing in a special environment.

To achieve parallel processing requires the development of more capable and cost-effective computer systems. This book emphasizes the design and application of parallel processing computers, including various architectural configurations, functional capabilities, operating systems, algorithmic and programming requirements, and performance limitations of parallel-structured computers. The ultimate goal is to achieve high performance at lower cost in performing large scale scientific-engineering computing tasks in the various application areas to be introduced in Section 1.5.

Most computer manufacturers started with the development of systems with a single central processor, called a uniprocessor system. We will reveal various means to promote concurrency in uniprocessor systems in Section 1.2. Uniprocessor systems have their limit in achieving high performance. The computing power in a uniprocessor can be further upgraded by allowing the use of multiple processing elements under one controller. One can also extend the computer structure to include multiple processors with shared memory space and peripherals under the control of one integrated operating system. Such a computer is called a multi *processor system.*

As far as parallel processing is concerned, the general architectural trend is being shifted away from conventional uniprocessor systems to multiprocessor systems or to an array of processing elements controlled by one uniprocessor. In all cases, a high degree of pipe lining is being incorporated into the various system levels. We will introduce these parallel computer structures in Section 1.3. After learning the parallelism in both uniprocessor and multiprocessor systems, we will then study several architectural classification schemes based on the machine structures and operation modes.

1.2 PARALLELISM IN UNIPROCESSOR SYSTEMS

Most general-purpose uniprocessor systems have the same basic structure. In this section, we will briefly review the architecture of uniprocessor systems. The development of parallelism in uniprocessors will then be introduced categorically, It is assumed that readers have had at least one basic course in the past on conventional computer organization. Therefore, we will provide only concise specifications of the architectural features of two popular commercial computers. Parallel-processing mechanisms and methods to balance subsystem bandwidths will then be described for a typical uniprocessor system. Details of these structures, mechanisms, and methods can be found in references suggested in the bibliographic notes.

1.2.1 Basic Uniprocessor Architecture

A typical uniprocessor computer consists of three major components: the main memory, the *central processing unit* (CPU), and the input-output (I/O) subsystem. The architectures of two commercially available uniprocessor computers are given below to show the possible interconnection of structures among the three subsystems. We will examine major components in the CPU and in the I/O subsystem.

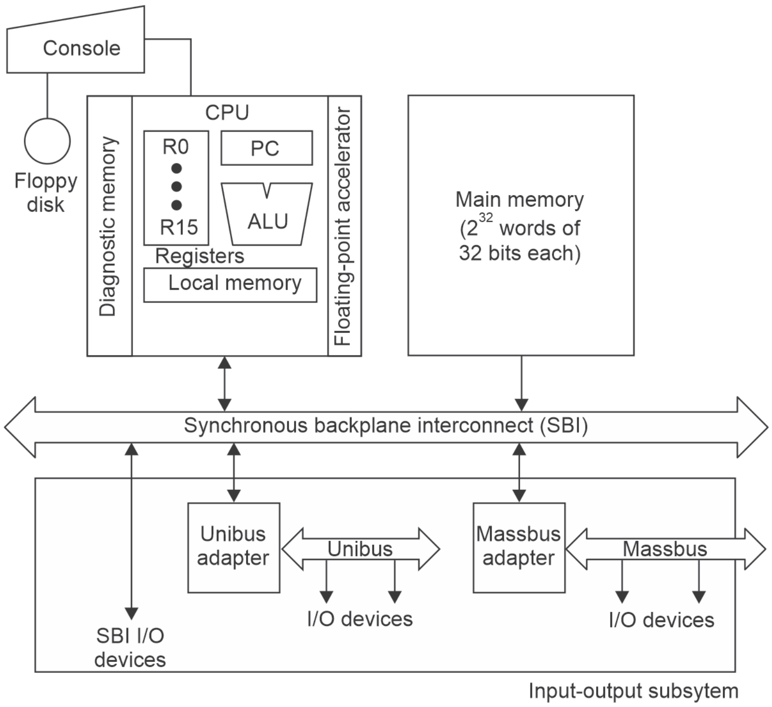

Figure 1.3: *The system architecture of the supermini VAX-11/780 uniprocessor system (Courtesy of Digital Equipment Corporation)*

Figure 1.3 shows the architectural components of the super minicomputer VAX-11/780, manufactured by Digital Equipment Company. The CPU contains the master controller of the VAX system. There are sixteen 32-bit general-purpose registers, one of which serves as the *program counter* (PC). There is also a special *CPU status register* containing information about the current state of the processor and of the program being executed. The CPU contains an *arithmetic and logic unit* (ALU) with an optional *floating-point accelerator,* and some local *cache memory* with an optional diagnostic memory. The CPU can be intervened by the operator through the console connected to a floppy disk.

The CPU, the main memory (2^{32} words of 32 bits each), and the I/O subsystems are all connected to a common bus, the synchronous backplane interconnect (SBI). Through this bus, all I/O devices can communicate with each other, with the CPU, or with the memory. Peripheral storage or I/O devices can be connected directly to the SBI through the *unibus* and its controller (which can be connected to PDP-11 series minicomputers), or through a *massbus* and its controller.

Another representative commercial system is the mainframe computer IBM System 370/Model 168 uniprocessor, shown in Figure 1.4. The CPU contains the

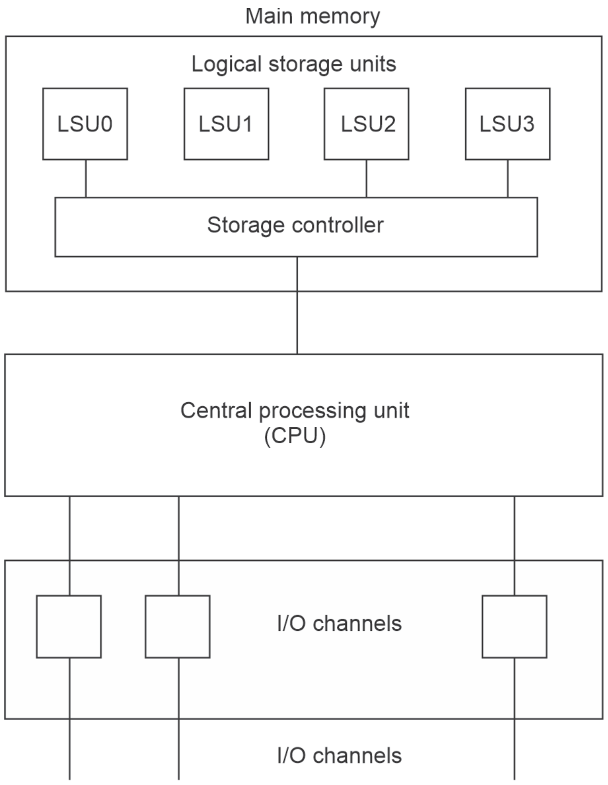

Figure 1.4: *The system architecture of the mainframe IBM System 370/Model 168 uniprocessor computer (Courtesy of International Business Machines Corp.).*

instruction decoding and execution units as well as a cache. Main memory is divided into four units, referred to as *logical storage units* (LSU), that are four-way interleaved. The storage controller provides multiport connections between the CPU and the four LSUs. Peripherals are connected to the system via high-speed I/O channels which operate asynchronously with the CPU. In Chapter 9, we will show that this uniprocessor can be modified to assume some multiprocessor configurations.

Hardware and software means to promote parallelism in uniprocessor systems are introduced in the next three subsections. We begin with hardware approaches which emphasize resource multiplicity and time overlapping. It is necessary to balance the processing rates of various subsystems in order to avoid bottlenecks and to increase total system throughput, which is the number of instructions (or basic computations) performed per unit time. Finally, we study operating system software approaches to achieve parallel processing with better utilization of the system resources.

1.2.2 Parallel Processing Mechanisms

A number of parallel processing mechanisms have been developed in uniprocessor computers. We identify them in the following six categories:

- Multiplicity of functional units
- Parallelism and pipelining within the CPU
- Overlapped CPU and I/O operations
- Use of a hierarchical memory system
- Balancing of subsystem bandwidths
- Multiprogramming and time sharing

We will describe below the first four techniques and discuss the remaining two approaches in the subsections to follow.

Multiplicity of functional units: The early computer had only one arithmetic and logic unit in its CPU. Furthermore, the ALU could only perform one function at a time, a rather slow process for executing a long sequence of arithmetic logic instructions. In practice, many of the functions of the ALU can be distributed to multiple and specialized functional units which can operate in parallel. The CDC-6600 (designed in 1964) has 10 functional units built into its CPU (Figure 1.5). These 10 units are independent of each other and may operate simultaneously. A scoreboard is used to keep track of the availability of the functional units and registers being demanded. With 10 functional units and 24 registers available, the instruction issue rate can be significantly increased.

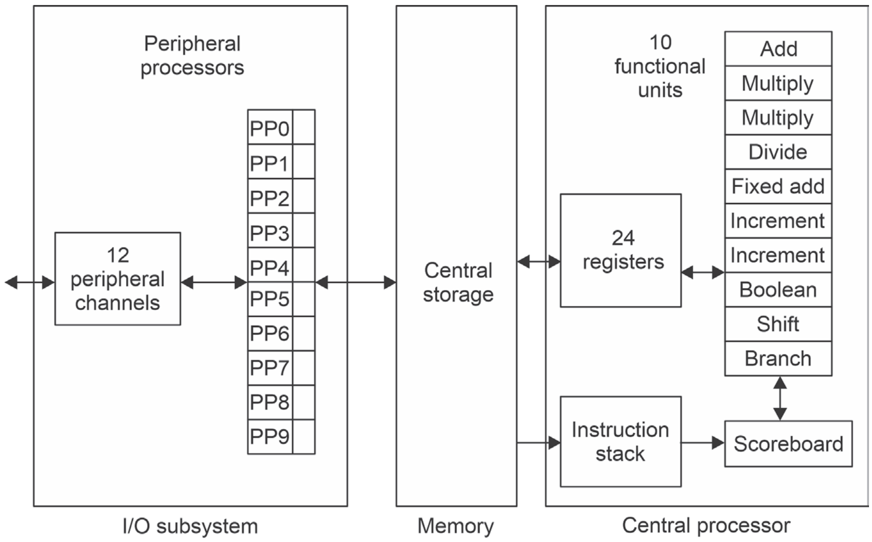

Figure 1.5: *The system architecture of the CDC-6600 computer (Courtesy of Control Data Corp.).*

Another good example of a multifunction uniprocessor is the IBM 360/91 (1968), which has two parallel *execution units* (E units): one for fixed-point arithmetic, and the other for floating-point arithmetic. Within the floating-point E unit are two functional units: one for floating-point add-subtract and the other for floating-point multiply-divide. IBM 360/91 is a highly pipelined, multifunction, scientific uniprocessor. We will study 360/91 in detail in Chapter 3. Almost all modern computers and attached processors are equipped with multiple functional units to perform parallel or simultaneous arithmetic logic operations. This practice of functional specialization and distribution can be extended to array processors and multiprocessors, to be discussed in subsequent chapters.

Parallelism and pipelining within the CPU: Parallel adders, using such techniques as carry-lookahead and carry-save, are now built into almost all ALUs. This is in contrast to the bit-serial adders used in the first-generation machines. High-speed multiplier recoding and convergence division are techniques for exploring parallelism and the sharing of hardware resources for the functions of multiply and divide (to be described in Section 3.2.2). The use of multiple functional units is a form of parallelism with the CPU.

Various phases of instruction executions are now pipelined, including instruction fetch, decode, operand fetch, arithmetic logic execution, and store result. To facilitate overlapped instruction executions through the pipe, instruction prefetch and data buffering techniques have been developed. Instruction and arithmetic pipeline designs will be covered in Chapters 3 and 4. Most commercial uniprocessor systems are now pipelined in their CPU with a clock rate between 10 and 500 ns.

Overlapped CPU and I/O operations: I/O operations can be performed simultaneously with the CPU computations by using separate I/O controllers, channels, or I/O processors. The direct-memory-access (DMA) channel can be used to provide direct information transfer between the I/O devices and the main memory. The DMA is conducted on a cycle-stealing basis, which is apparent to the CPU. Furthermore, 1:0 multiprocessing, such as the use of the 10 I/O processors in CDC-6600 (Figure 1.5), can speed up data transfer between the CPU (or memory) and the outside world. I/O subsystems for supporting parallel processing will be described in Section 2.5. Back-end database machines can be used to manage large databases stored on disks.

Use of hierarchical memory system Usually, the CPU is about 1000 times faster than memory access. A hierarchical memory system can be used to close up the speed gap. Computer memory hierarchy is conceptually illustrated in Figure 1.6. The innermost level is the register files directly addressable by ALU. Cache memory can be used to serve as a buffer between the CPU and the main memory. Block access of the main memory can be achieved through multiway interleaving across parallel memory modules (Figure 1.4). Virtual memory space can be established with the use of disks and tape units at the outer levels.

Details of memory subsystems for both uniprocessor and multiprocessor computers are given in Chapter 2. Various interleaved memory organizations are

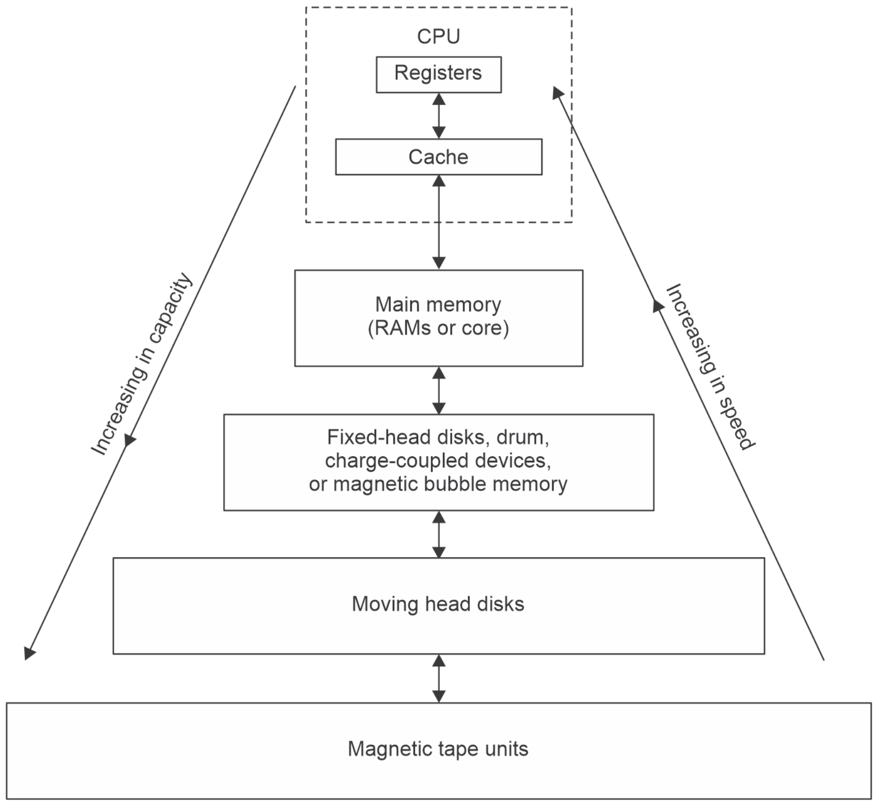
CPU
Registers
Cache
Main memory (RAMs or core)
Fixed-head disks, drum, charge-coupled devices, or magnetic bubble memory
Moving head disks
Magnetic tape units
Increasing in capacity
Increasing in speed

Figure 1.6: *The classical memory hierarchy*.

given in Section 3.1.4. Parallel memories for array processors are treated in Section 6.2.4, along with the description of the Burroughs Scientific Processor (1978). Multiprocessor memory and cache coherence problems will be treated in Section 7.3. All these techniques are intended to broaden the memory bandwidth to match that of the CPU.

1.2.3 Balancing of Subsystem Bandwidth

In general, the CPU is the fastest unit in a computer, with a processor cycle t_p of tens of nanoseconds: the main memory has a cycle time t_m of hundreds of nanoseconds; and the I/O devices are the slowest with an average access time t_d of a few milliseconds. It is thus observed that

$$t_p > t_m < t_p \tag{1.1}$$

For example, the IBM 370/l68 has t_m = 5 ms (disk), t_m = 320 ns, and t_p = 80 ns. With these speed gaps between the subsystems, we need to match their processing bandwidths in order to avoid a system bottleneck problem.

The *bandwidth* of a system is defined as the number of operations performed per unit time. In the case of a main memory system, the memory bandwidth is measured by the number of memory words that can be accessed (either fetch or store) per unit time. Let W be the number of words deliven, xi per memory cycle t_m. Then the maximum memory bandwidth B_m, is equal to

$$B_m = \frac{W}{t_m} \text{ (words/s or bytes/s)} \tag{1.2}$$

For example, the IBM 3033 uniprocessor has a processor cycle t_p = 57 ns, Eight double words (8 bytes each) can be requested from an eight-way interleaved memory system (with eight LSEs in Figure 1.7) per each memory cycle 1m = 456 ns, Thus, the maximum memory bandwidth of the 3033 is B_m, = 8 × 8 bytes/456 ns = 140 megabytes/s. Memory access conflicts may cause delayed access of some of the processor requests. In practice, the utilized memory bandwidth B^u_m is usually lower than B_m; that is, $B^u_m \leq B_m$. A rough measure of B^u_m has been suggested as

$$B^u_m = \frac{B_m}{\sqrt{M}} \tag{1.3}$$

where M is the number of interleaved memory modules in the memory system (to be described in Section 3.1.4). For the IBM 3033 uniprocessor, we thus have an approximate $B^u_m = 140/\sqrt{8}$ = 49.5 megabytes/s.

For external memory and I/O devices, the concept of bandwidth is more involved because of the sequential-access nature of magnetic disks and tape units. Considering the latencies and rotational delays, the data transfer rate may vary. In general, we refer to the average data transfer rate Ed as the bandwidth of a disk unit A typical modern disk may have a data rate of I megabyte/s. With multiple disk drives, the data rate can increase to 10 megabytes/s, say for 10 drives per channel controller. A modern magnetic tape unit has a data transfer rate around 1,5 megabytes/s. Other peripheral devices, like line printers, readers/punch, and CRT terminals, are much slower due to mechanical motions.

The bandwidth of a processor is measured as the maximum CPU computation rate B_p' as in 160 megaflops for the Cray-l and 12.5 *million instructions* per second (MIPS) for IBM 370/168. These are all peak values obtained by $1/t_p$' = 1/12.5 ns and 1/80 ns respectively. In practice, the utilized CPU rate is $B^u_p \leq B_p$. The utilized CPU rate B^u_p is based on measuring the number of output results (in words) per second:

$$B^u_p = \frac{R_w}{T_p} \text{ (words/s)} \tag{1.4}$$

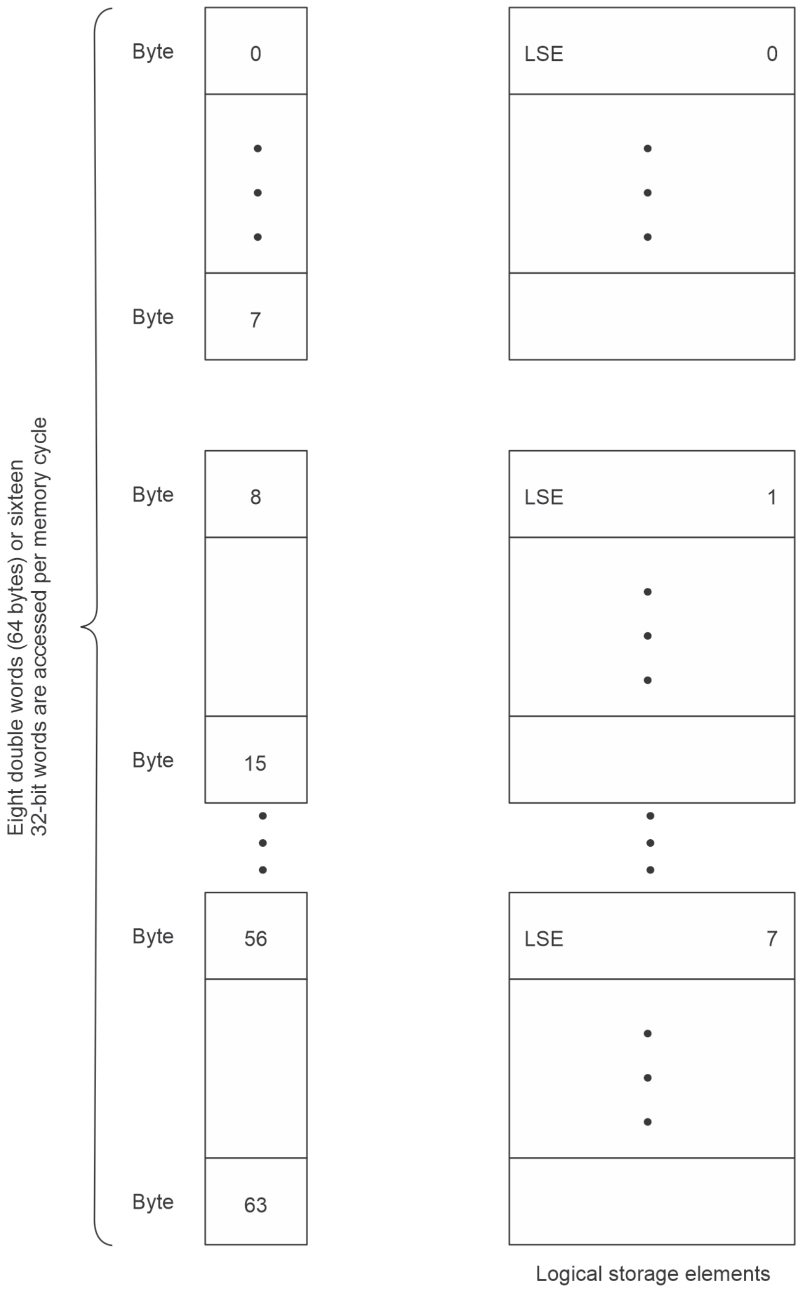
Eight double words (64 bytes) or sixteen 32-bit words are accessed per memory cycle
Byte 0
Byte 7
Byte 8
Byte 15
Byte 56
Byte 63
LSE 0
LSE 1
LSE 7
Logical storage elements

Figure 1.7: *The interleaved memory structure in IBM 3033 uniprocessor*.

where R_w is the number of word results and T_p is the total CPU time required to generate the R_w results. For a machine with variable word length, the rate will vary. For example, the CDC Cyber-205 has a peak CPU rate of 200 megaflops for 32-bit results and only 100 megaflops for 64-bit results (one vector processor is assumed).

Based on current technology (1983), the following relationships have been observed between the bandwidths of the major subsystems in a high-performance uniprocessor:

$$B_{\underline{m}} \geq B^u{}_m \geq B_p \geq B^u{}_p \geq B_d \tag{1.5}$$

This implies that the main memory has the highest bandwidth. since it must be updated by both the CPU and the I/O devices, as illustrated in Figure 1.8. Due to the unbalanced speeds (Eq. 1.1), we need to match the processing power of the three subsystems. Two major approaches are described below.

Bandwidth balancing between CPU and memory: The speed gap between the CPU and the main memory can be closed up by using fast cache memory between them. The cache should have an access time $t_e = t_p$. A block of memory words is moved from the main memory into the cache (such as 16 words/block for the IBM 3033) so that immediate instructions/data can be available most of the time from the cache. The cache serves as a data/instruction buffer. Detailed descriptions of cache memories will be given in Sections 2.4 and 7.3

Bandwidth balancing between memory and I/O device: Input-output channels with different speeds can be used between the slow I/O devices and the main memory. These I/O channels perform buffering and multiplexing functions to transfer the data from multiple disks into the main memory by stealing cycles from the CPU. Furthermore, intelligent disk controllers or database machines can be used to filter out the irrelevant data just off the tracks of the disk. This filtering will alleviate the I/O channel saturation problem. The combined buffering, multiplexing, and filtering operations thus can provide a faster, more effective data transfer rate, matching that of the memory.

In the ideal case, we wish to achieve a totally balanced system, in which the entire memory bandwidth matches the bandwidth sum of the processor and I/O devices; that is,

$$B^u{}_p + Bd = B^u{}_p \tag{1.6}$$

where $B^u{}_p = Bp$ and $B^u{}_m = B_m$ are both maximized. Achieving this total balance requires tremendous hardware and software supports beyond any of the existing systems.

1.2.4 Multiprogramming and Time Sharing

Even when there is only one CPU in a uniprocessor system, we can still achieve a high degree of resource sharing among many user programs. We will briefly review the concepts of multiprogramming and time sharing in this subsection. These are software approaches to achieve concurrency in a uniprocessor system.

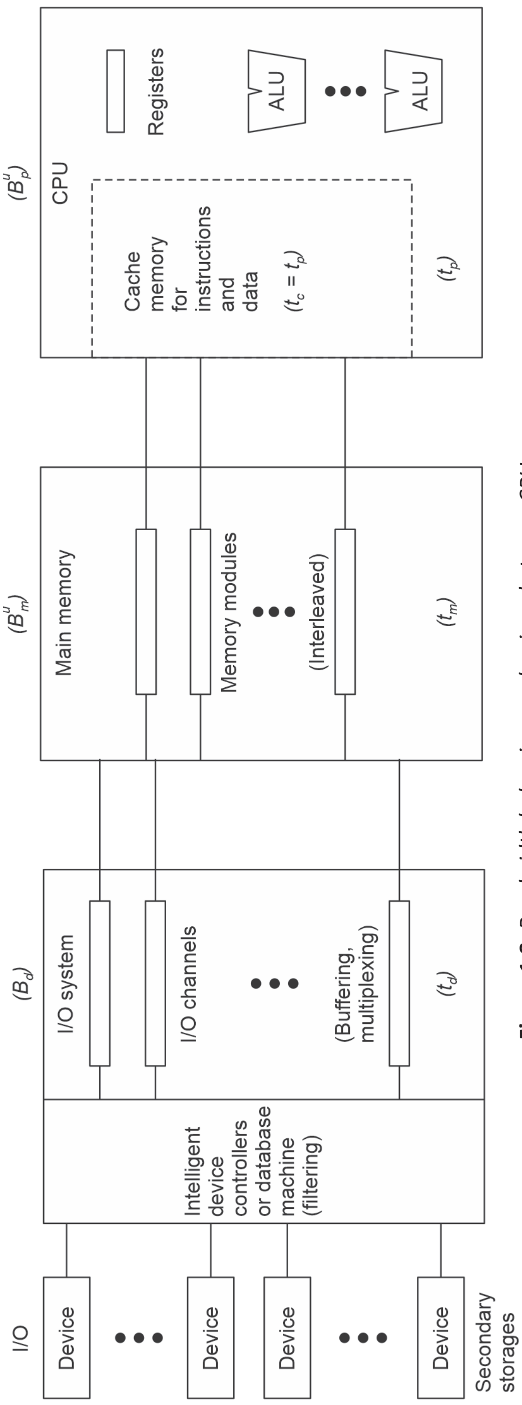

Figure 1.8: *Bandwidth balancing mechanisms between CPU, memory, and I/O subsystem in a uniprocessor computer.*