2.5 Architecture and Organization

Computer architecture is a science for designing a computer system. The aim of a computer architect is to design a high performance system at a reasonable cost, fulfilling all other requirements. A computer's architecture provides various attributes to the computer system which are needed by a machine language programmer or a system software designer to develop a program. It is a conceptual model providing the following information:

1. Instruction set;
2. Instruction format;
3. Operation codes;
4. Operand types;
5. Operand addressing modes;
6. Registers;
7. Main memory space utilization (memory map);
8. I/O space allocation (I/O map);
9. Interrupts assignment and priority;
10. DMA channels assignment and priority;
11. I/O techniques used for various devices;
12. I/O controller command formats;
13. I/O controller status formats.

Figure 2.8 shows the computer architectural aspects for a machine language programmer. Computer organization gives an in-depth picture of its functional structure and logical

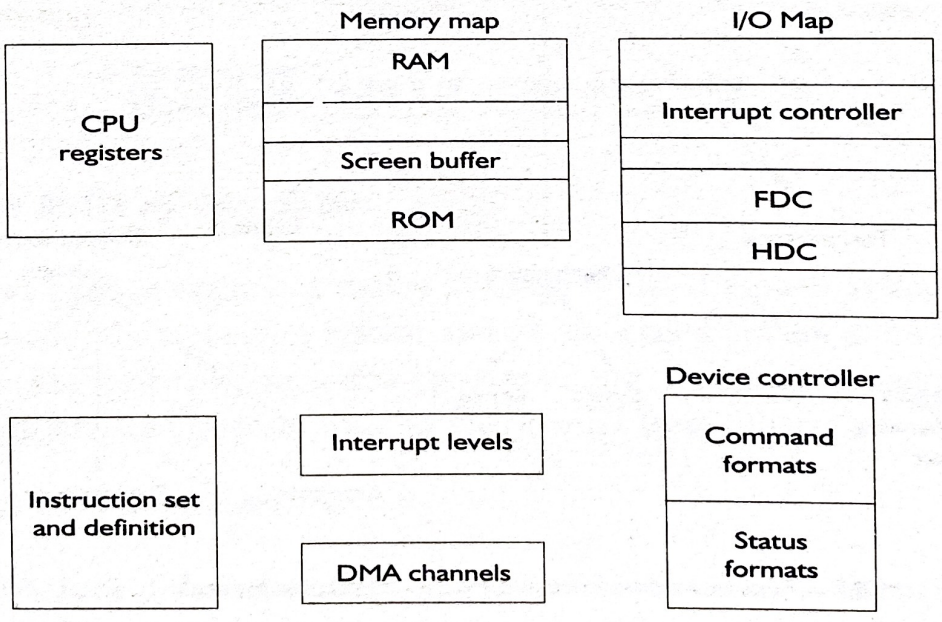


Fig. 2.8 Computer architecture for a programmer

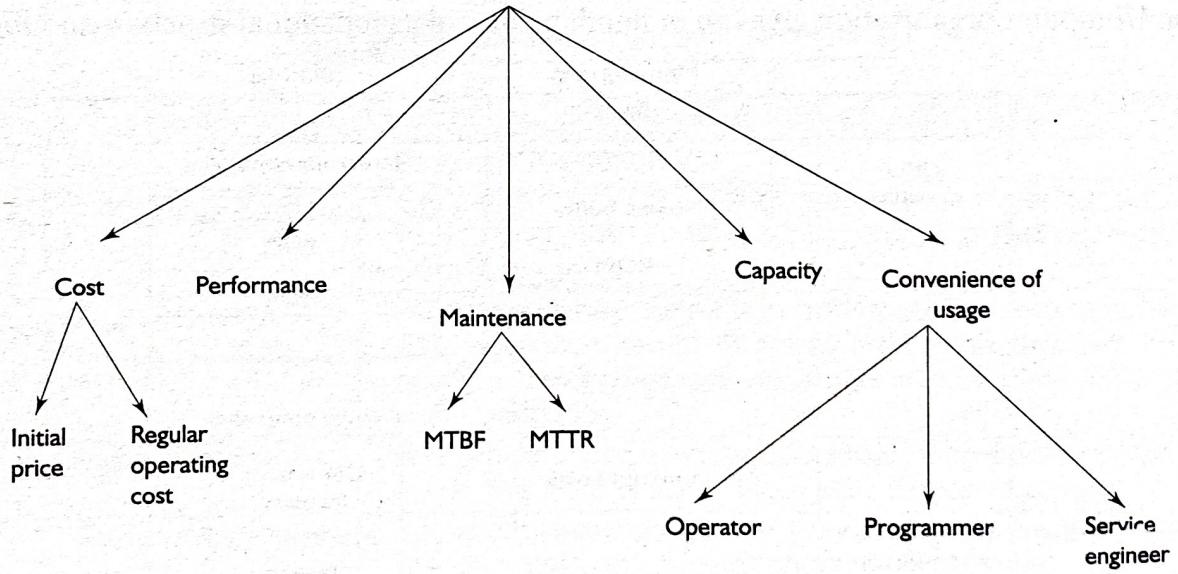
interconnections between the different units (functional blocks). Usually it includes the details of the hardware. Two computers with same architecture can have different organization. Similarly, two computers having same organization differ in their architecture. For designing a computer, its architecture is fixed first and then its organization is decided. The architecture indicates its hardware whereas the organization reveals its performance.

2.5.1. Implementation

The implementation of a computer's architecture (and organization) describes the form of hardware circuits grouped into functional blocks. Physical implementation of the computer design depends on its technology. Two computers of same organization designed at a course of period would have different component technology due to which their hardware circuit is different. Further, the architecture of these may or may not be same.

2.6 Dimensions of Computer Evolution

There are five dimension to measure the computer's excellence. These are Performance, Capacity, Cost, User friendliness and Maintainability. Figure 2.9 illustrates the important issues involved in these dimensions. These issues are used for designing the computer.



MTBF—Mean time between failures; MTTR—Mean time to repair

Fig. 2.9 Five dimensions of importance

2.6.1 Contributing Factors

As shown in Fig. 2.10, three factors which decide the superiority of the computer are as follows: Technology, Concepts and Techniques. The technology is constantly evolving in every aspect of the computer such as: CPU, memory, peripheral device, interconnecting path, software etc. The concepts followed to design a computer affects its performance and capacity. Computer architecture or designer uses latest technology for designing the computer. Usually more than one design technique are available for implementing a concept. For example, the virtual memory concept in the computer facilitates the user to run longer programs as compared to the physical memory capacity without any botheration (except using logical address instead of physical address). The CPU and the operating system handle the long programs by address translation and swapping. For this, the designer follows segmentation technique or paging technique.

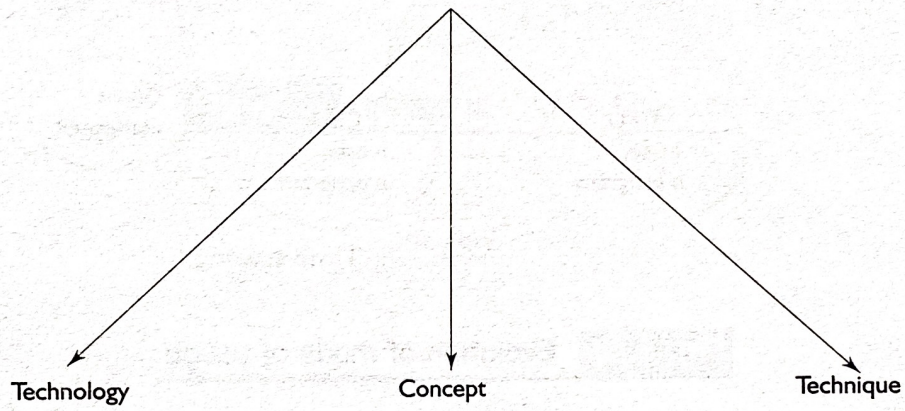


Fig. 2.10 Factors of design superiority

2.6.1.1 Usage Mode of Computer

Figure 2.11 shows various important modes of usage of the computer system at different times. In each mode, the operating system and its associated system software has some unique features. For example, in a multiprogramming system, the operating system ensures protection of each program without interference from others.

2.6.1.2 Basic CPU Architecture

Figure 2.12 shows three different modes of CPU architecture. Firstly, in register architecture, the operands for the instructions are put into the CPU registers and hence they are

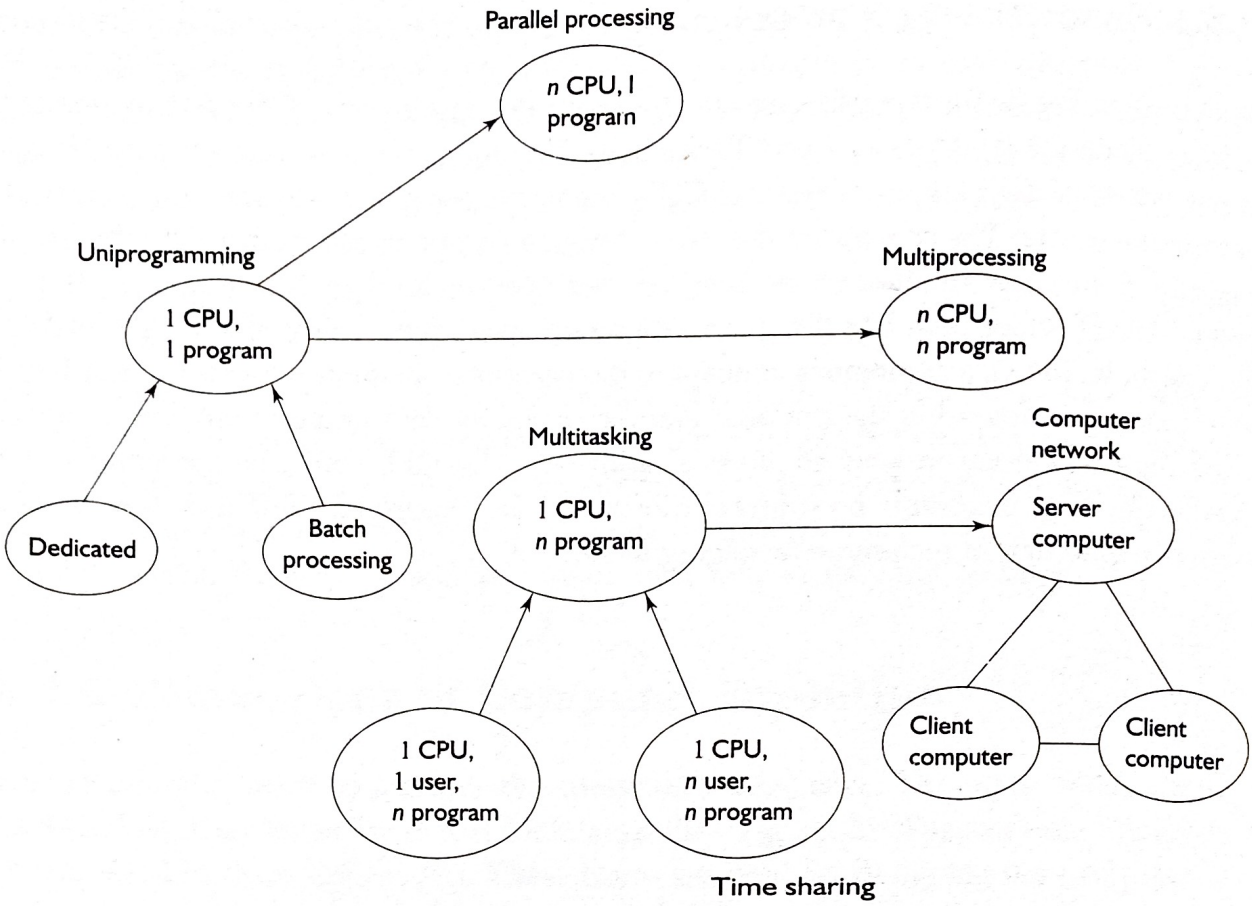


Fig. 2.11 Evolution of mode of usage

fetched fast during instruction cycle. In accumulator architecture which is the second type, the number of instructions in the program increases but the execution of the instruction is fast since one operand is in the accumulator itself. Lastly, in stack architecture, programming is very simple since any arithmetic operation is done on the item at the top of the stack.

2.6.1.3 Storage Styles

Figure 2.13 shows different types of storage in a computer. The registers within CPU also serve as storage but they store only the operands and not the instructions. Its capacity is also limited. Hence, the registers are not included while talking about the memory of a computer. The capacity of the main memory is important since it decides the size of the active program at any given time. Since the main memory is costly, the designers provide cheaper secondary storage devices which function as an extension of the main memory for

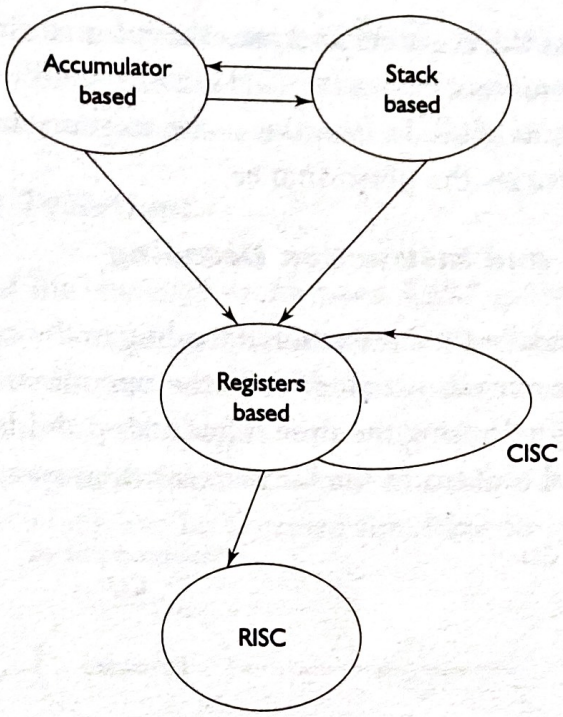


Fig. 2.12 Modes of CPU architecture

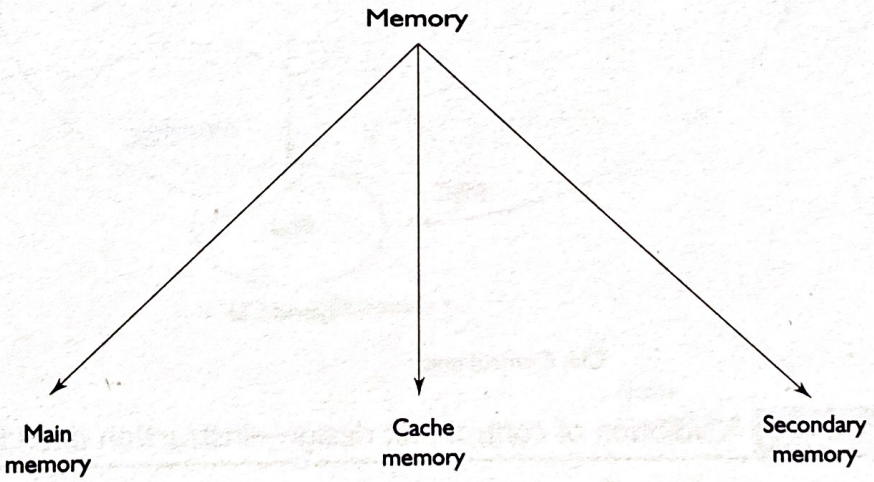


Fig. 2.13 Storage types in computers

storing programs and data. This forms a two level storage. A large capacity of secondary memory (disk, tape etc.) is provided. The access time of secondary memory is more than that of main memory. In a three level storage, a small but high speed cache memory is used as a temporary buffer between the CPU and main memory. It is used to store and supply the contents of frequently accessed main memory locations.

The software recognises the registers and uses them for storing the operands of instructions. Whereas, the cache memory is an internal feature of CPU and the software does not consider it as a memory unit at all. In fact, the cache memory is not an additional storage; it is the buffer which increases the performance.

2.6.1.4 *Control Unit and Instruction Decoding*

Figure 2.14 shows the trends in the instruction decoding in the control unit. The hardwired control unit has hardware circuits which decode the opcode in the instruction and generate appropriate control signals using the time references provided by the clock signal. The microprogrammed control unit stores the bit patterns corresponding to the control signals

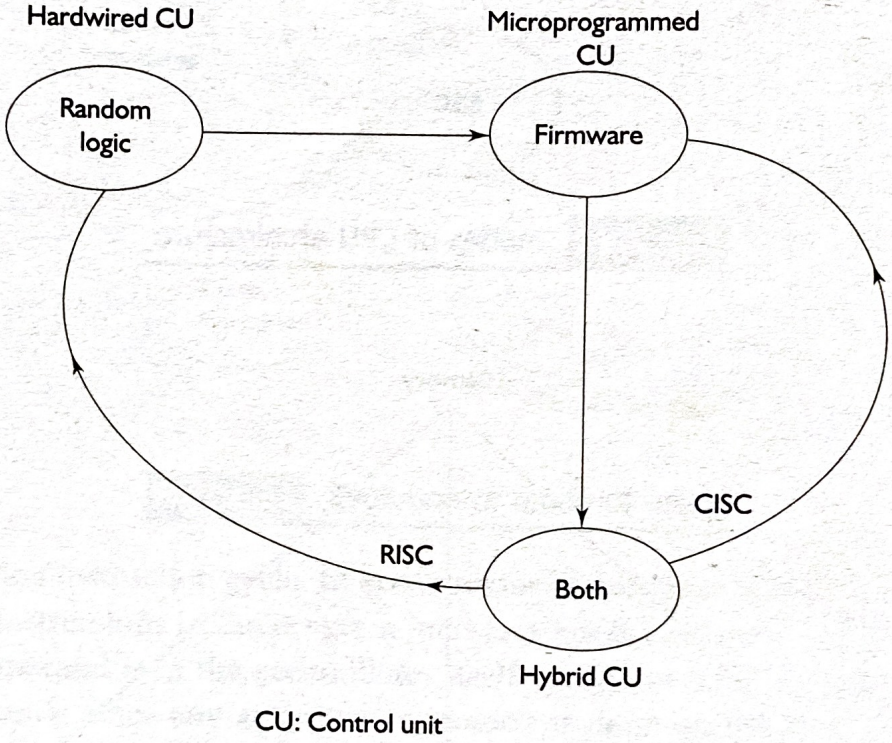


Fig. 2.14 Evolution of control unit design—instruction decoding

in several microinstructions for each instruction. A read only memory is used inside the control unit wherein the designer stores the microprograms (sequence of microinstructions) for each instruction. During program execution, the control unit fetches the microprogram corresponding to the opcode. Thus, the control unit serves as a 'stored microprogram' sequencer.

The development of microprograms is easier than designing the circuits for a hardwired control unit. But its speed is slower since the bit patterns have to be fetched from the

control memory. The hybrid control unit uses a mixture of both ideas. Part of the control unit which is time critical is hardwired and the remaining part is microprogrammed. The RISC system has to be as fast as possible and hence usually its control unit is hardwired.

2.6.1.5 Main Memory Techniques

Figure 2.15 shows some of the memory techniques. CPU speed is constantly improving due to advanced component technology but the memory technology has not evolved that rapidly. Hence, the main aiin was to get better performance from the same given technology for the memory. Interleaving is a concept of dividing the memory into two parts: odd addressed and even addressed locations. In other words, adjacent locations are placed in separate modules which can be accessed simultaneously. This reduces the overall access time.

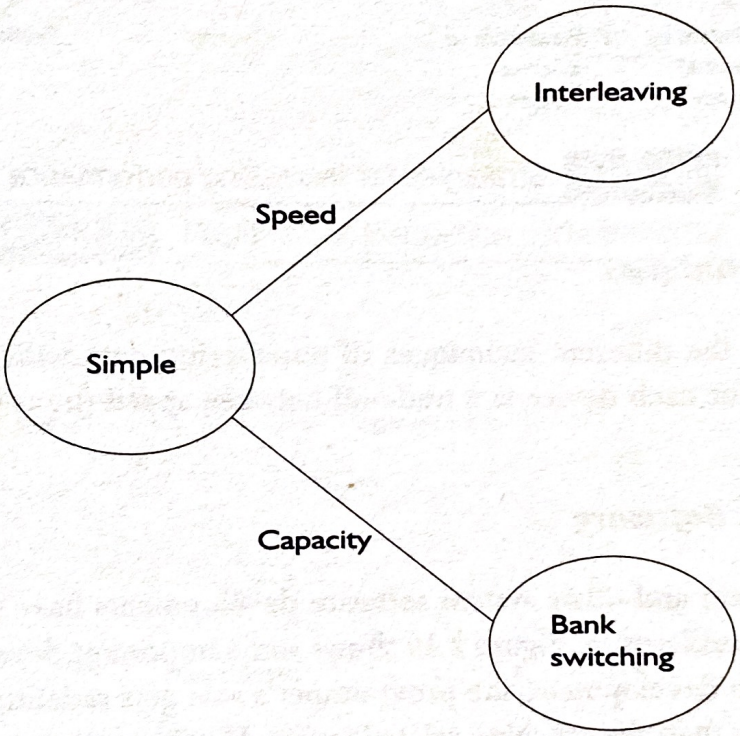


Fig. 2.15 Evolution of main memory concepts

For a given CPU, the memory capacity is limited by the number of address bits. The bank switching is a concept of breaking this limitation without the knowledge of CPU in collaboration with the operating system. Multiple banks of same capacity is used and at a time one of them is selected.

2.6.1.6 *Instruction Cycle Handling*

In general, the performance of a subsystem can be increased by multiple techniques as shown in Fig. 2.16. These are discussed in following chapters. Figure 2.17 shows different techniques of instruction cycle handling in order to increase the number of instructions processed per second.

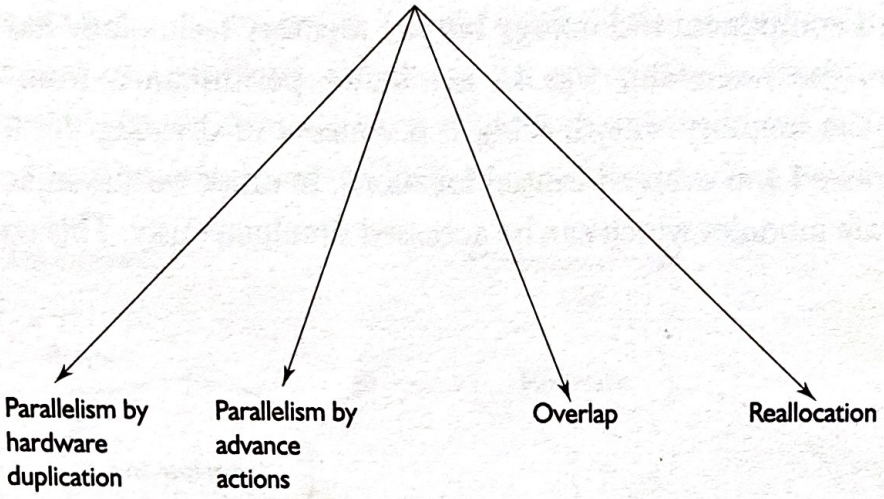


Fig. 2.16 Strategies for increasing performance

2.6.1.7 *I/O Techniques*

Figure 2.18 shows the different techniques of transferring data with peripheral devices. The choice made for each device is a trade-off between speed (transfer rate desired) and cost (hardware).

2.6.1.8 *System Software*

The operating system and other system software developments have grown parallel with the hardware and architecture. Figure 2.19 shows some important developments in system software. With each development, the programmer's role gets redefined. He concentrates more on algorithms than the machine related issues. However, some amount of overhead is added to program execution time with each new development. This time has become negligible due to the increase in CPU speed and other techniques of parallelism and overlap. In embedded system, the entire system software gets embedded in hardware permanently. This is mainly based on 'microcontroller' which is more than a microprocessor in view of on-chip ROM, RAM and I/O ports. Its instruction set is usually limited and less powerful. Still it is more 'complete' than a microprocessor and is eligible to be termed as 'microcomputer' chip.

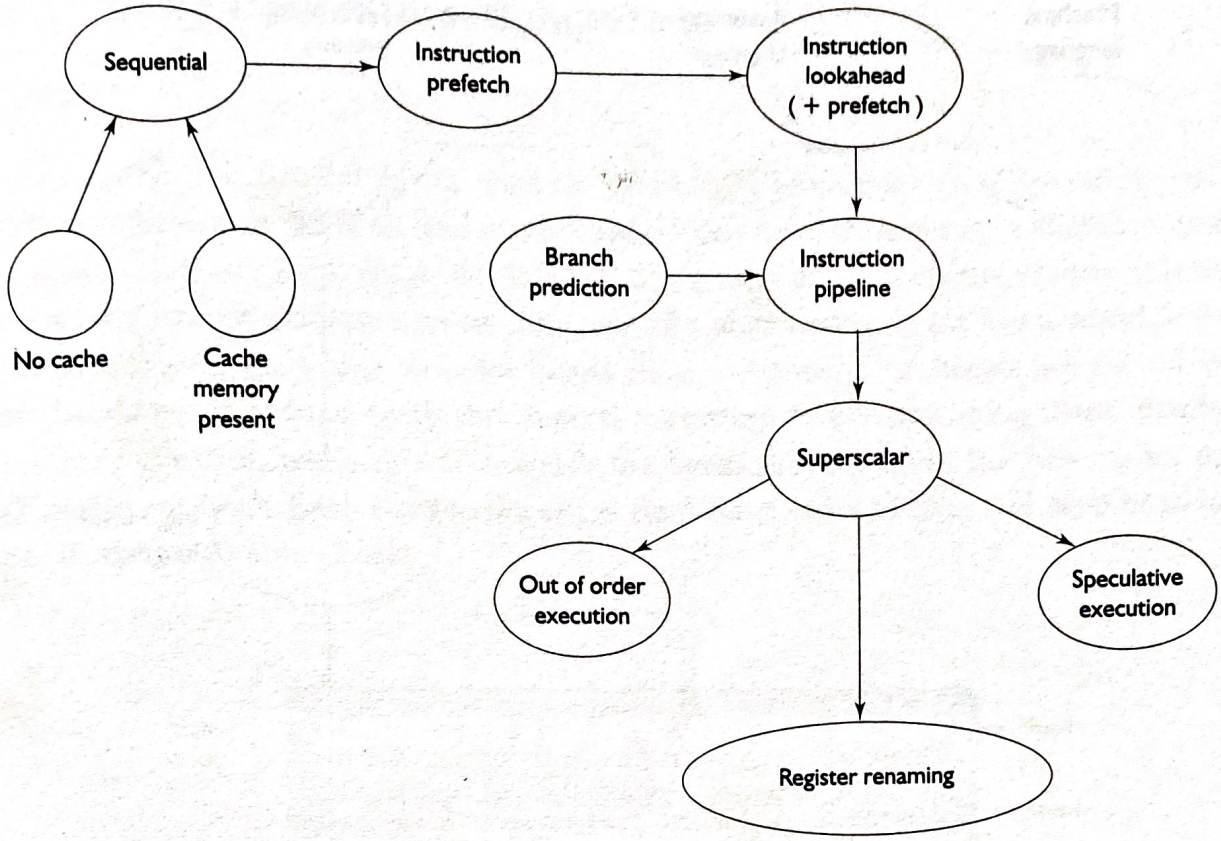


Fig. 2.17 Evolution of instruction cycle handling

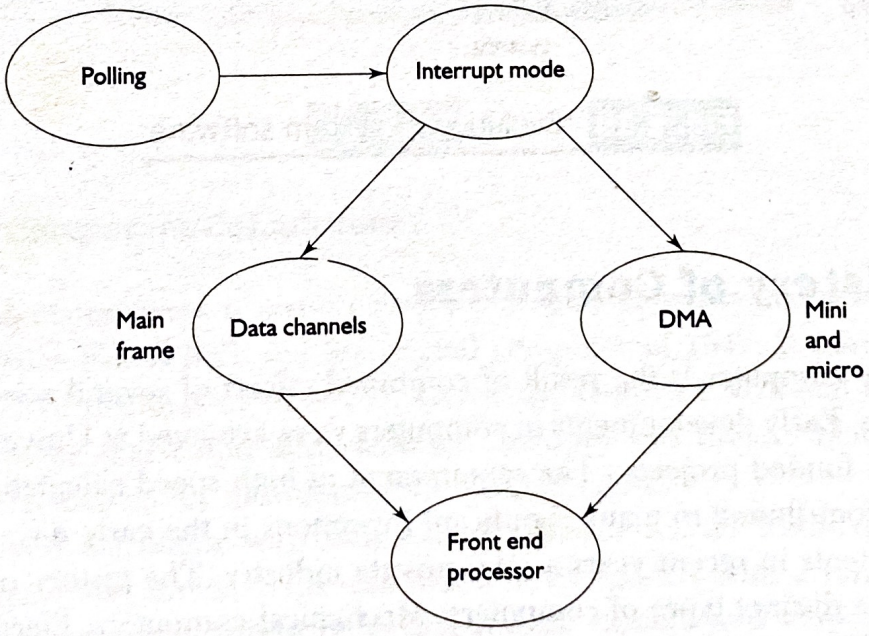


Fig. 2.18 Evolution of I/O techniques

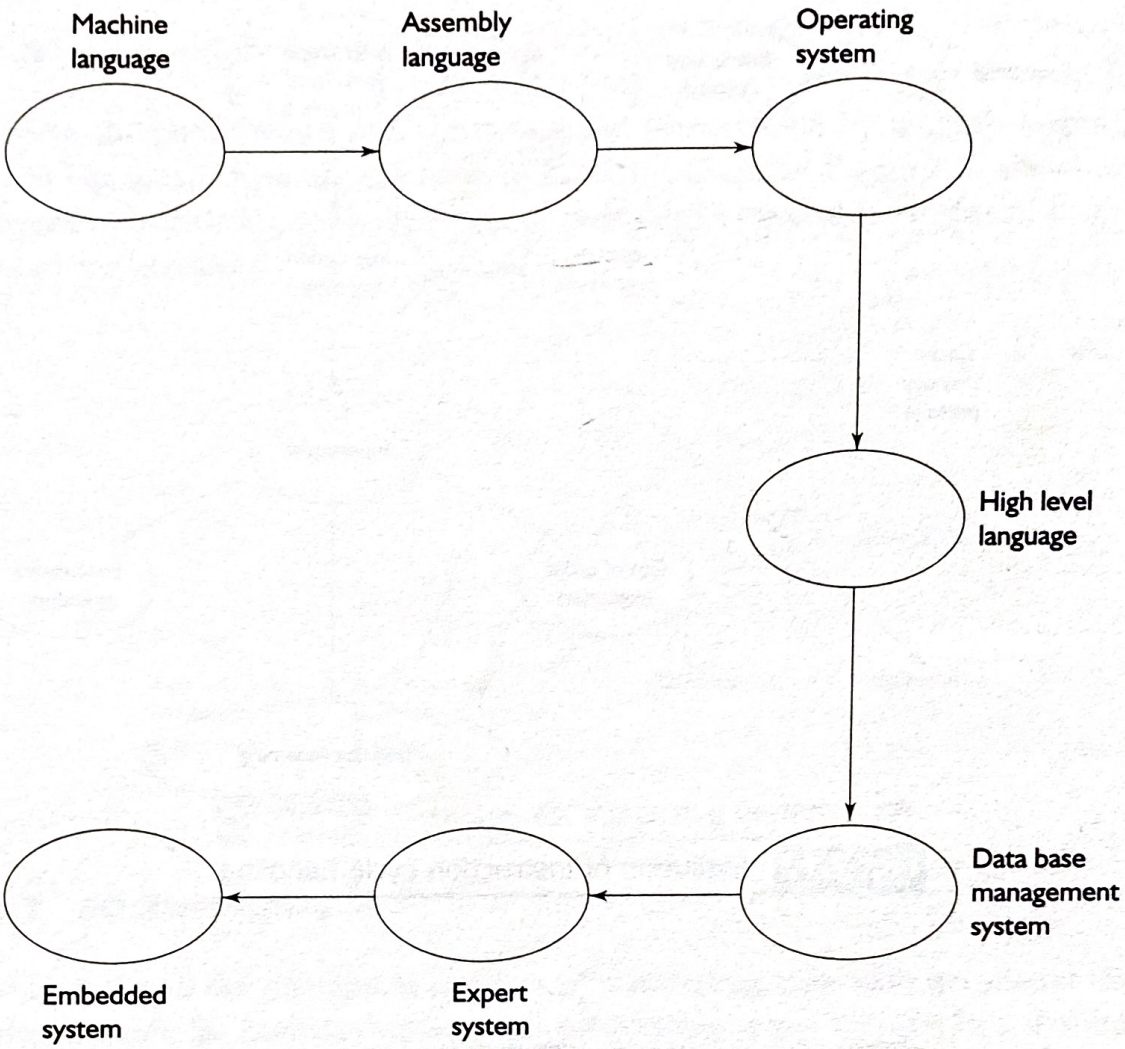


Fig. 2.19 Evolution of system software