3.2 CISC vs RISC

There are two popular concepts associated with CPU design and instruction set:

- Complex Instruction Set Computing (CISC)
- Reduced Instruction Set Computing (RISC)

All relatively older systems (main frame, mini or micro) have CISC systems. Though today's systems comprise both types. RISC systems are more popular today due to their performance level as compared to CISC systems. However, due to high cost, RISC systems are used only when a special need for speed, reliability etc. arises.

3.2.1 CISC TRENDS

From the early days, one of the parameters for computer evaluation was the instruction set. There are two critical aspects: total number of instructions, and their capabilities. Generally, the instruction set of a CISC system is made efficient by incorporating a large number of powerful instructions. The goal is to reduce the size of the compiled program (machine language) with limited instructions. Basically, a powerful instruction is equivalent to three or four simple instructions put together. Because of the limited size of the compiled program, the requirement for the main memory is also small. In olden days, the main memory was based on magnetic core memory that was costly. Thus, including powerful instructions in the instruction set of a CPU reduces the main memory size and the cost. Another added advantage of having powerful (complex) instructions is, the lesser the number of instructions in a (compiled) program, the lesser is the time spent by the CPU for fetching instructions. This decreases execution time considerably since the core memory (the olden day memory) is slow with access time of 1 to 2 microseconds. Hence, we have dual benefits for having powerful instructions in the instruction set: reduced system price

and reduced program execution time. However, a highly efficient compiler is required to use the powerful instructions more frequently while translating the high level language program into a machine language program. Hence, the system software (compiler) becomes huge in order to generate a small object code. This does not affect the users as the compilation of an application program is a one time affair. Figure 3.1 illustrates the CISC scenario. Currently, computers use semiconductor memory as the main memory (and cache memory). It is cheaper and faster. Hence, the points discussed in the favor of complex instruction set are not relevant any more in present times.

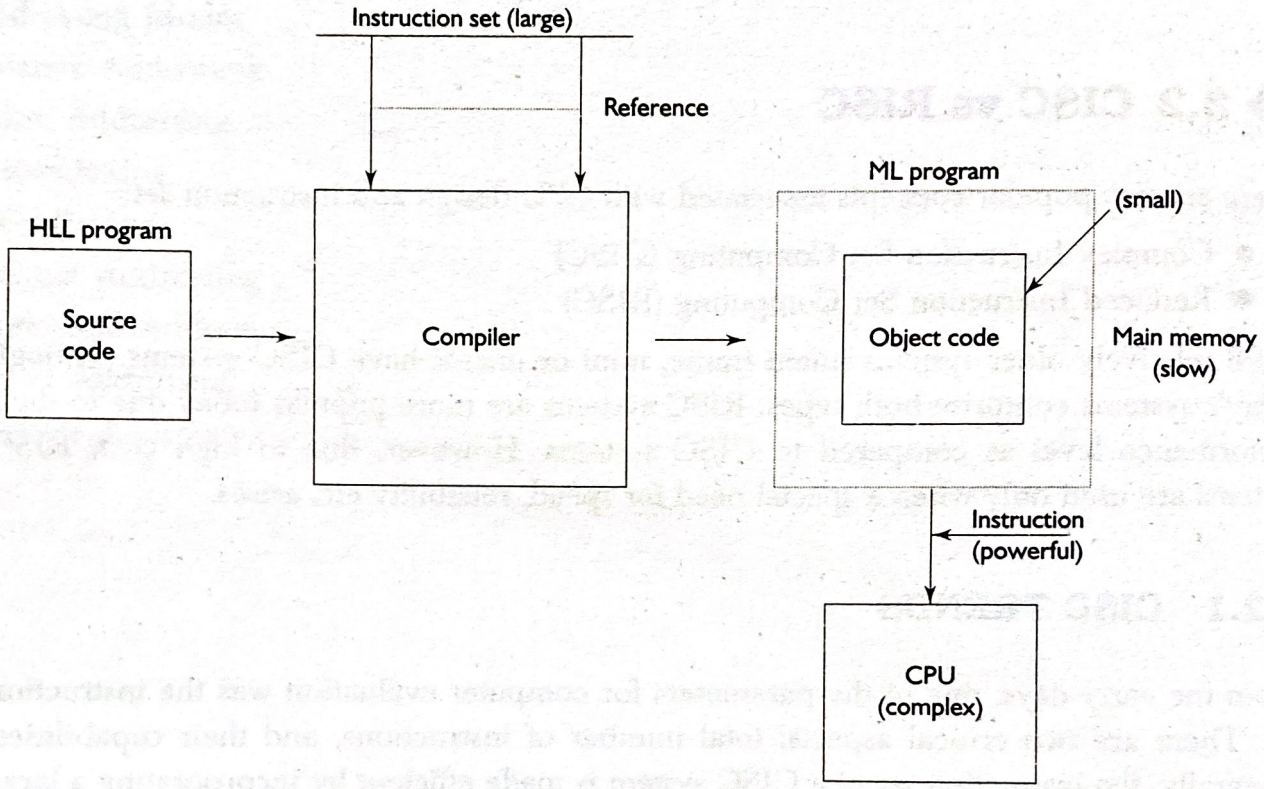

Fig. 3.1 CISC scenario

3.2.2 CISC Drawbacks

The CISC systems have the following drawbacks:

1. *CPU complexity*: The control unit design (mainly instruction decoding) becomes complex since the instruction set is large with heavily encoded instructions.
2. *System size and cost*: There is a lot of hardware circuitry due to complexity of the CPU. This increases the hardware cost of the system and also the power requirement.

3. *Clock speed*: Due to increased circuits, the propagation delays are more and the CPU cycle time is large and hence, the effective clock speed is reduced.
4. *Reliability*: The heavy hardware is prone to frequent failures.
5. *Maintainability*: Troubleshooting and detecting a fault is a big task since there are a large number of huge circuits. The invention of microprogramming has reduced this burden to some extent. In-built diagnostic microcodes were also provided in many CISC systems, giving a helping hand to the hardware engineer in case of system failures.

3.2.3 RISC Concept

The term 'KISS' is often used for RISC concept meaning 'Keep it short and simple'. The layman concept of RISC is this: 'being simple' helps doing 'more simple things' easily and reliably without using resources and efforts, which in turn results in better efficiency. Technically speaking, it means that the CPU's instruction set should have only less number of simple instructions. Figure 3.2 depicts the RISC scenario.

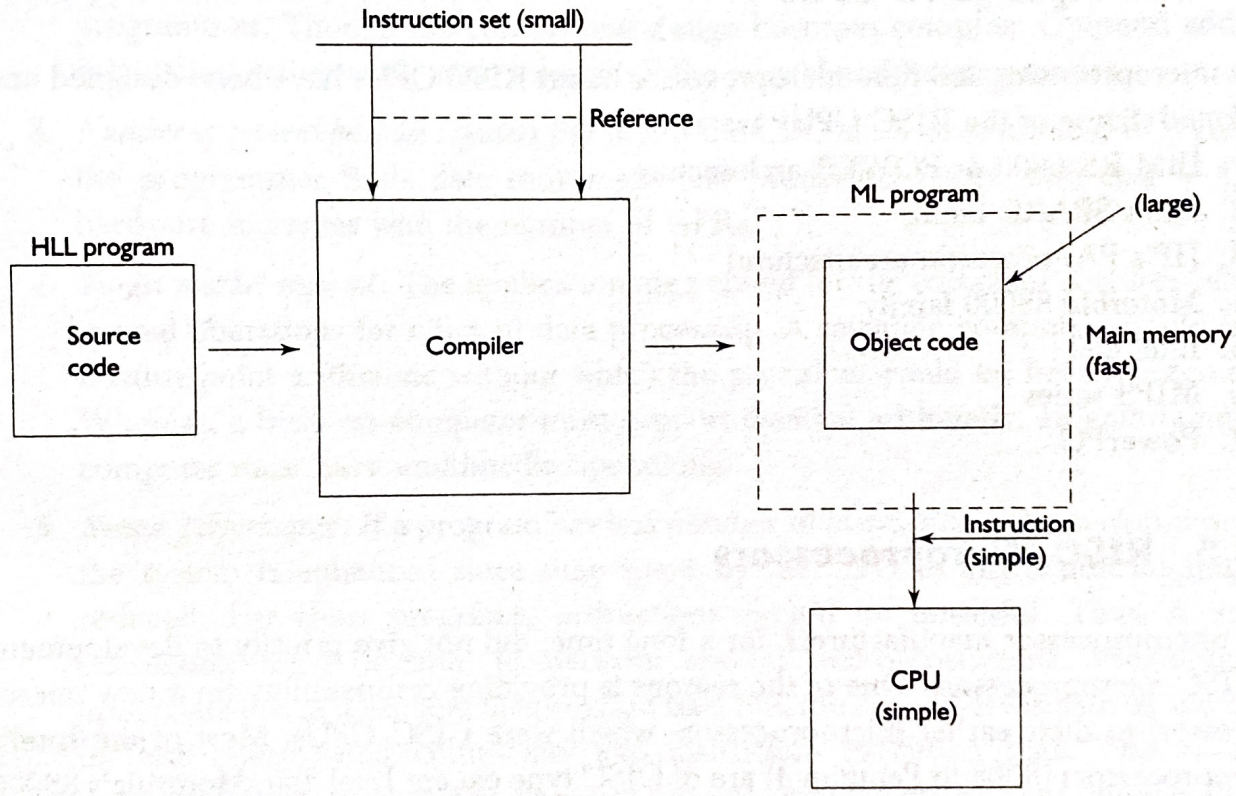

Fig. 3.2 RISC scenario

The RISC architecture is covered under the following features:

1. Only simple instructions
2. Small instruction set
3. Equal instruction length for all instructions
4. Large number of registers for storing operands
5. Load/store architecture: The operand for an arithmetic instruction such as 'ADD' is available in a register and not in memory. Similarly the result of an 'ADD' instruction is stored in a register and not in memory. Accordingly 'LOAD' instruction should precede an 'ADD' instruction and 'STORE' instruction should follow 'ADD' instruction, if necessary. Hence, the compiler will generate a lot of 'LOAD' and 'STORE' instructions.
6. Faster instruction execution at a rate of one instruction/clock cycle takes place. Instruction pipelining, built-in cache memory and superscalar architecture are included in the CPU so that on an average, one instruction comes out of the pipeline for every clock.

3.2.4 Sample RISC CPUs

Both microprocessor and non-microprocessor based RISC CPUs have been designed and marketed. Some of the RISC CPUs are as follows:

1. IBM RS/6000 or POWER architecture
2. Sun's SPARC family
3. HP's PA (precision architecture)
4. Motorola 88000 family
5. Intel 860
6. MIPS series
7. PowerPC

3.2.5 RISC Microprocessors

The microprocessor manufacturers, for a long time, did not give priority to development of RISC microprocessors. One of the reasons is providing compatibility (in a new microprocessor) to their earlier microprocessors which were CISC CPUs. Most of the Intel's microprocessors (8008 to Pentium 4) are of CISC type except Intel 860. Motorola's 88000 family belongs to RISC. The PowerPC developed jointly by IBM, Motorola and Apple is a RISC CPU. In fact, it was designed to provide a low cost RISC microprocessor to the user.