Chapter 10
CPUs: MicroCode, Protection, and Processor Modes

Douglas Comer. (2017). *Essentials of Computer Architecture* (2nd ed.). CRC Press.

M. Morris Mano, Charles R. Kime. (2015). *Logic and computer design fundamentals* (5th ed.). Pearson.

Contents

1. A Central Processor

- Early in the history of computers, centralization emerged as an important architectural approach – as much functionality as possible was collected into a single processor.
  - The processor, which became known as a *Central Processing Unit (CPU)*, controlled the entire computer, including both calculations and I/O.
- In contrast to early designs, a modern computer system follows a decentralized approach. The system contains multiple processors, many of which are dedicated to a specific function or a hardware subsystem.
  - For example, we will see that an I/O device, such as a disk, can include a processor that handles disk transfers.

2. CPU Complexity

- Because it must handle a wide variety of control and processing tasks, a modern CPU is extremely complex.
- Multi cores
  - Modern CPU chips contain multiple processors called cores. The cores all function in parallel, permitting multiple computations to proceed at the same time. Multicore designs are required for high performance because a single core cannot be clocked at arbitrarily high speeds.
- Multiple Roles
  - A CPU must fill several major roles: running application programs, running an operating system, handling external I/O devices, starting or stopping the computer, and managing memory. No single instruction set is optimal for all roles, so a CPU often includes many instructions.

2. CPU Complexity

- Protection And Privilege
  - Most computer systems incorporate a system of protection that gives some subsystems higher privilege than others.
    - For example, the hardware prevents an application program from directly interacting with I/O devices, and the operating system code is protected from inadvertent or deliberate change.
- Hardware Priorities
  - A CPU uses a priority scheme in which some actions are assigned higher priority than others.
    - For example, we will see that I/O devices operate at higher priority than application programs – if the CPU is running an application program when an I/O device needs service, the CPU must stop running the application and handle the device.
- Generality
  - A CPU is designed to support a wide variety of applications. Consequently, the CPU instruction set often contains instructions that are used for each type of application (i.e., a CISC design).
- High Speed.
  - The final, and perhaps most significant, source of CPU complexity arises from the desire for speed.
  - Parallelism is a fundamental technique used to create high-speed hardware.

3. Modes Of Execution

- In most CPUs, the hardware uses a set of parameters to handle the complexity and control operation.
- We say that the hardware has multiple modes of execution. At any given time, the current execution mode determines how the CPU operates.
- Items are associated with a CPU mode of execution
    - The subset of instructions that are valid
    - The size of data items
    - The region of memory that can be accessed
    - The functional units that are available
    - The amount of privilege

4. Backward Compatibility

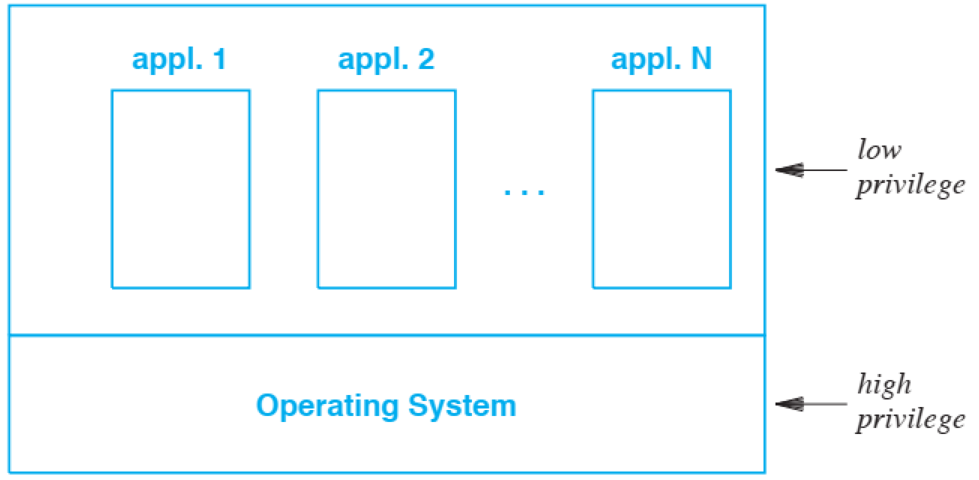
- Some CPUs have a mode that provides backward compatibility with a previous model.
- Backward compatibility allows a vendor to sell a CPU with new features, but also permits customers to use the CPU to run old software.
- When Intel first introduced a CPU that operated on thirty-two bit integers, the CPU included a compatibility mode that implemented the sixteen-bit instruction set from Intel's previous CPU.

5. Changing Modes

- There are two ways:
  - Automatic (initiated by hardware)
  - Manual (under program control)
- Automatic Mode Change
  - External hardware can change the mode of a CPU.
  - For example, when an I/O device requests service, the hardware informs the CPU. Hardware in the CPU changes mode (and jumps to the operating system code) automatically before servicing the device.
- Manual Mode Change
  - In essence, manual changes occur under control of a running program. Most often, the program is the operating system, which changes mode before it executes an application. However, some CPUs also provide multiple modes that applications can use, and allow an application to switch among the modes.

6. Multiple Levels Of Protection

- How many levels of privilege are needed, and what operations should be allowed at each level?



A CPU that runs applications needs at least two levels of protection: the operating system must run with absolute privilege, but application programs can run with limited privilege.

Figure 8.2 Illustration of a CPU that offers two levels of protection. The operating system executes with highest privilege, and application programs execute with less privilege.

7. Microcoded Instructions

- Instead of implementing the instruction set directly with digital circuits, a CPU is built in two pieces.
    - First, a hardware architect builds a fast, but small processor known as a *microcontroller*.
    - Second, to implement the CPU instruction set (called a *macro instruction set*), the architect writes software for the microcontroller. The software that runs on the microcontroller is known as *microcode*.
- For example: Further suppose that the microcontroller only offers sixteen-bit arithmetic. To implement a thirty-two-bit addition, the microcode must add sixteen bits at a time, and must add the carry from the low-order bits into the high-order bits.

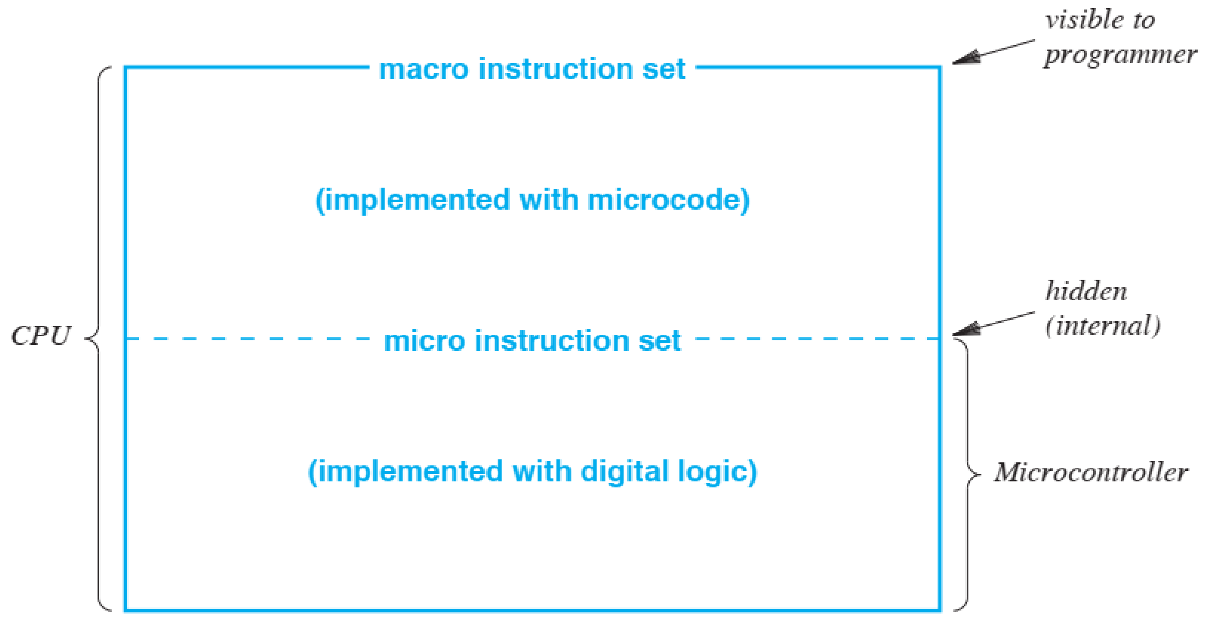


Figure 8.3 Illustration of a CPU implemented with a microcontroller. The macro instruction set that the CPU provides is implemented with microcode.

8. The Advantage Of Microcode

- There are three motivations.
  - 1. Because microcode offers a higher level of abstraction, building microcode is less prone to errors than building hardware circuits.
  - 2. Building microcode takes less time than building circuits.
  - 3. Because changing microcode is easier than changing hardware circuits, new versions of a CPU can be created faster.
- Microcode does have some disadvantages that balance the advantages:
  - 1. Microcode has more overhead than a hardware implementation.
  - 2. Because it executes multiple micro instructions for each macro instruction, the microcontroller must run at much higher speed than the CPU.
  - 3. The cost of a macro instruction depends on the micro instruction set.

9. FPGAs And Changes To The Instruction Set

- The microcontroller and microcode typically reside on the integrated circuit along with the rest of the CPU, and are only used internally.
- Only the macro instruction set is available to programmers.
- Interestingly, some CPUs have been designed that make the microcode dynamic and accessible to customers who purchase the CPU.
    - That is, the CPU contains facilities that allow the underlying hardware to be changed after the chip has been manufactured.

9. FPGAs And Changes To The Instruction Set

- Why would customers want to change a CPU?
- The motivations are flexibility and performance.
  - For example, a company that sells video games might add macro instructions to manipulate graphics images, and a company that makes networking equipment might create macro instructions to process packet headers. Using the underlying hardware directly (e.g., with microcode) can result in higher performance.
- One technology that allows modification has become especially popular. Known as *Field Programmable Gate Array (FPGA)*, the technology permits gates to be altered after a chip has been manufactured.
  - Technologies like dynamic microcode and FPGAs allow a CPU instruction set to be modified or extended after the CPU has been purchased. The motivations are flexibility and higher performance.