Vietnam National University HCMC

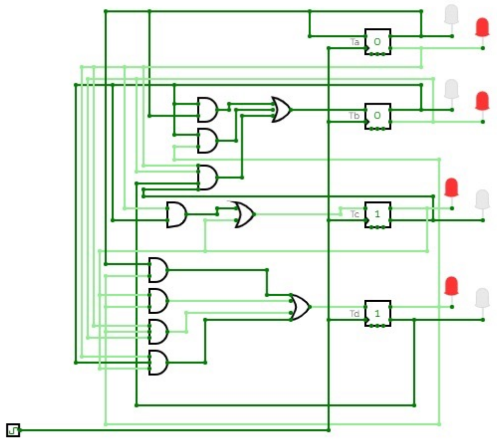
International University

School of Electrical Engineering

EE053IU

Digital Logic Design

Lecture 14: Data Processing and Control



INSTRUCTOR: Dr. Vuong Quoc Bao

1. The Computer System

- Acquire information from data sources.
- Process information by interpreting, evaluating, manipulating, converting, formatting, or otherwise working with acquired data
- Provide information in a meaningful form to data recipients.

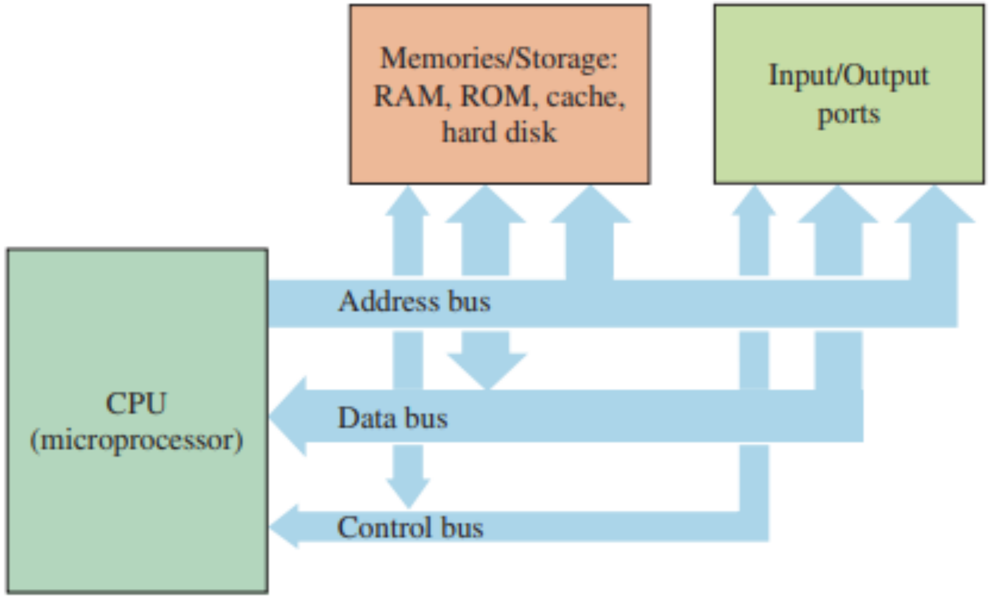


FIGURE 14–1 Basic computer block diagram.

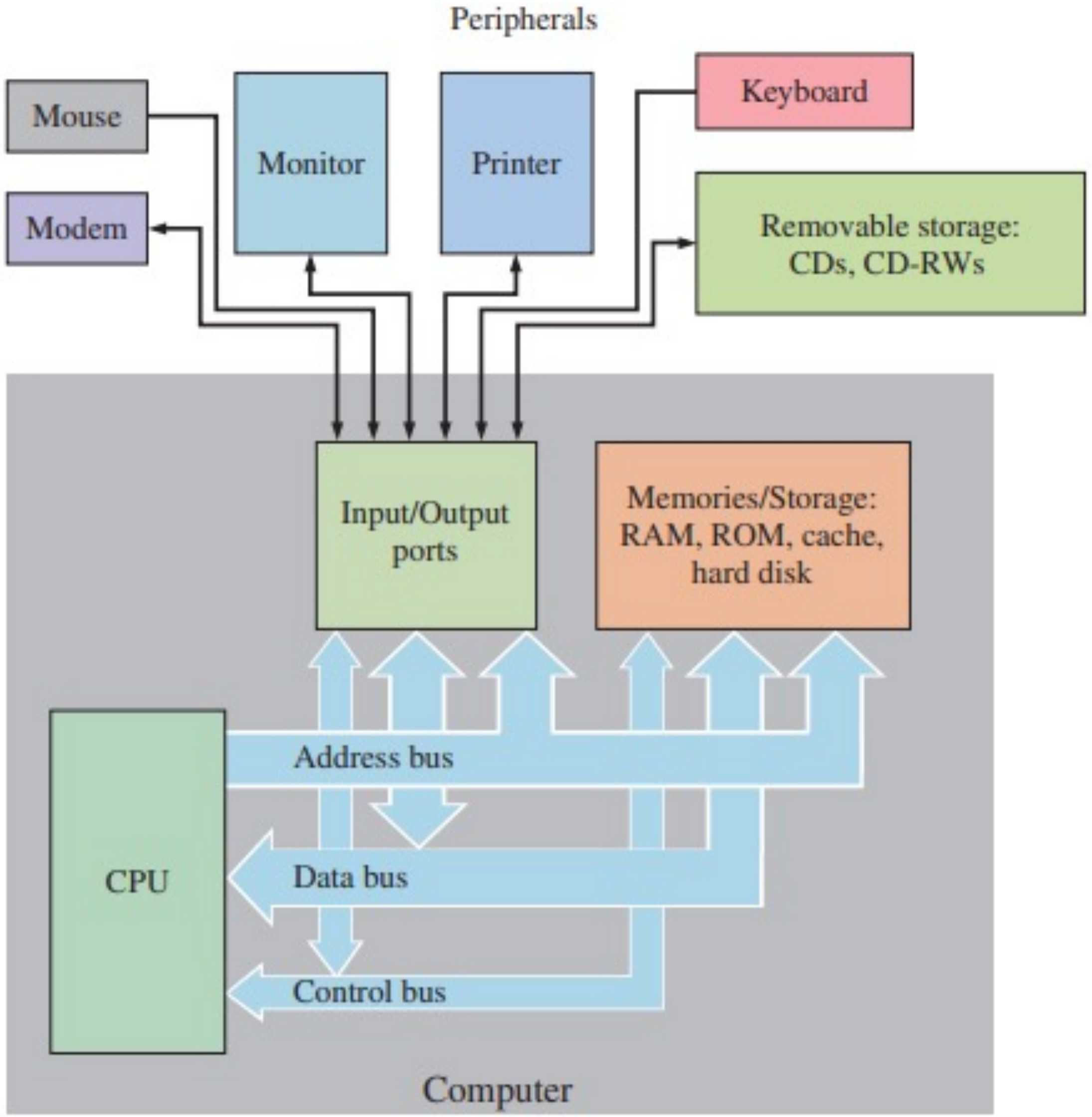


FIGURE 14–2 Basic block diagram of a typical computer system including common peripherals. The computer itself is shown in the gray block.

2. Practical Computer System Considerations

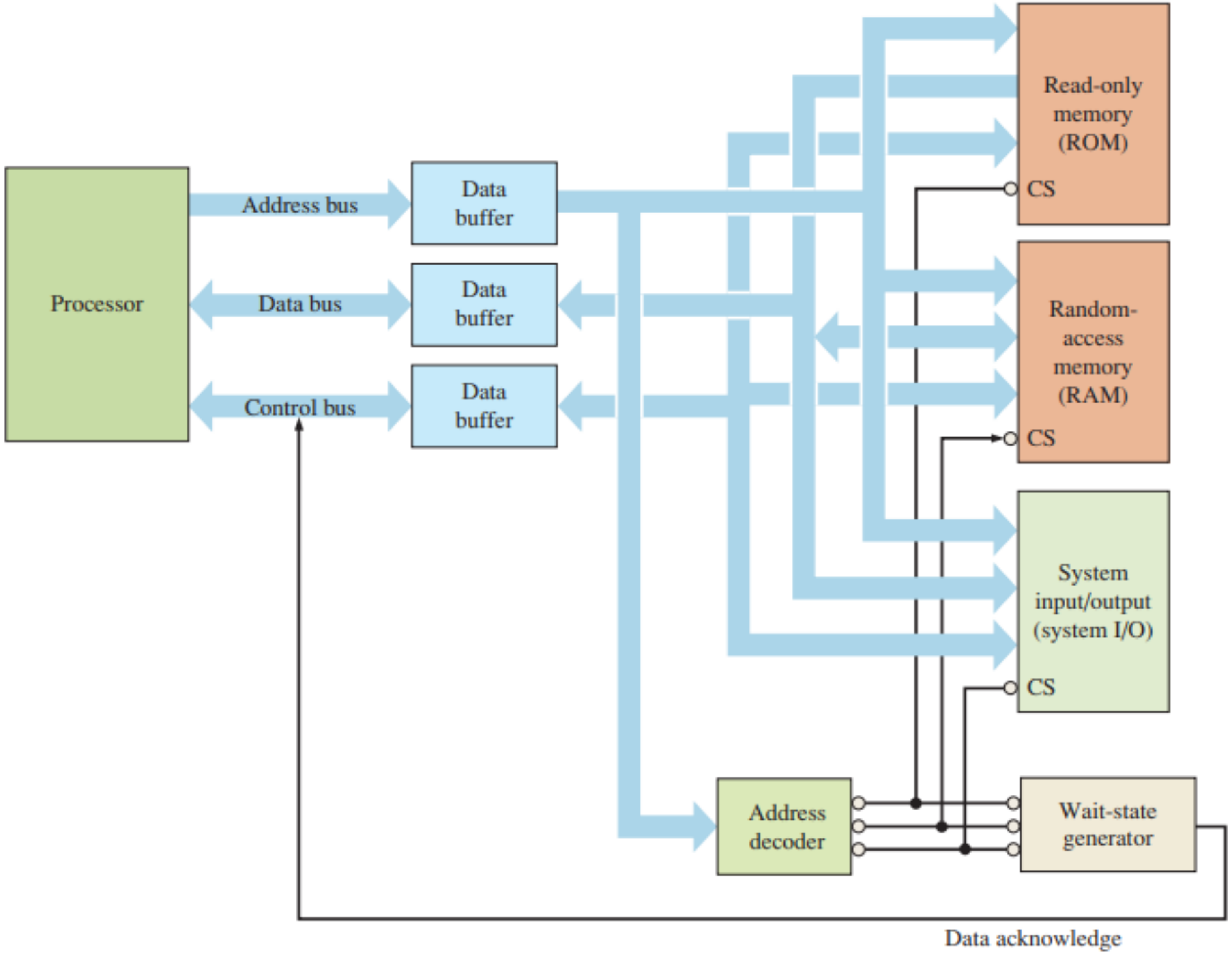


FIGURE 14–3 Block diagram of a practical computer system.

Shared Signal Lines

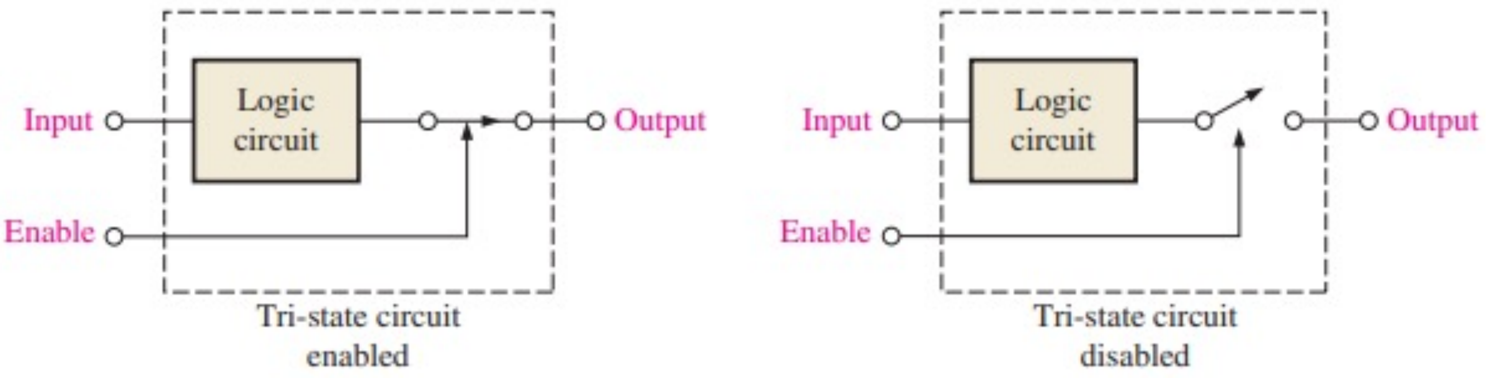


FIGURE 14–4 Logic devices with tri-state outputs.

System Timing

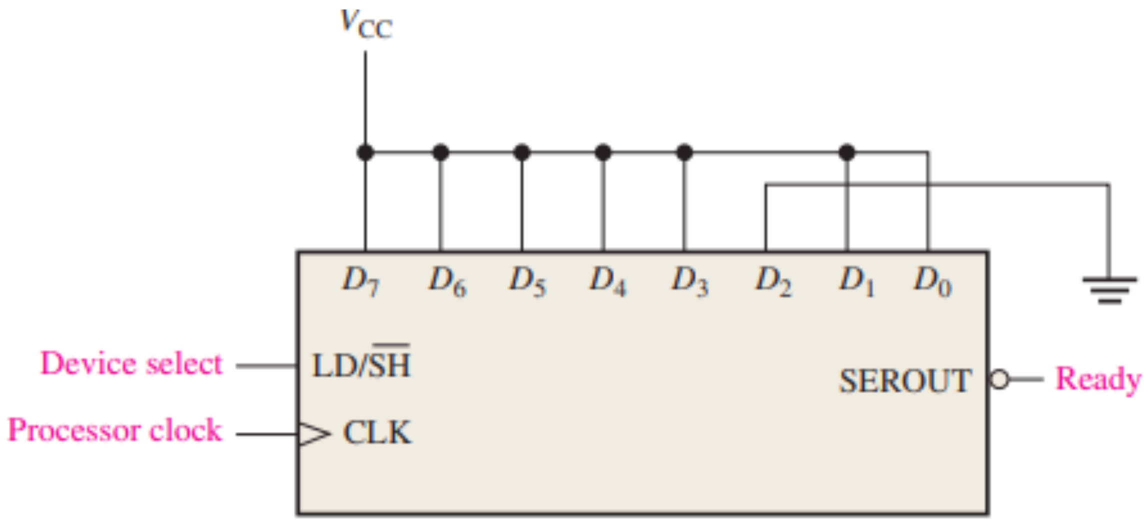
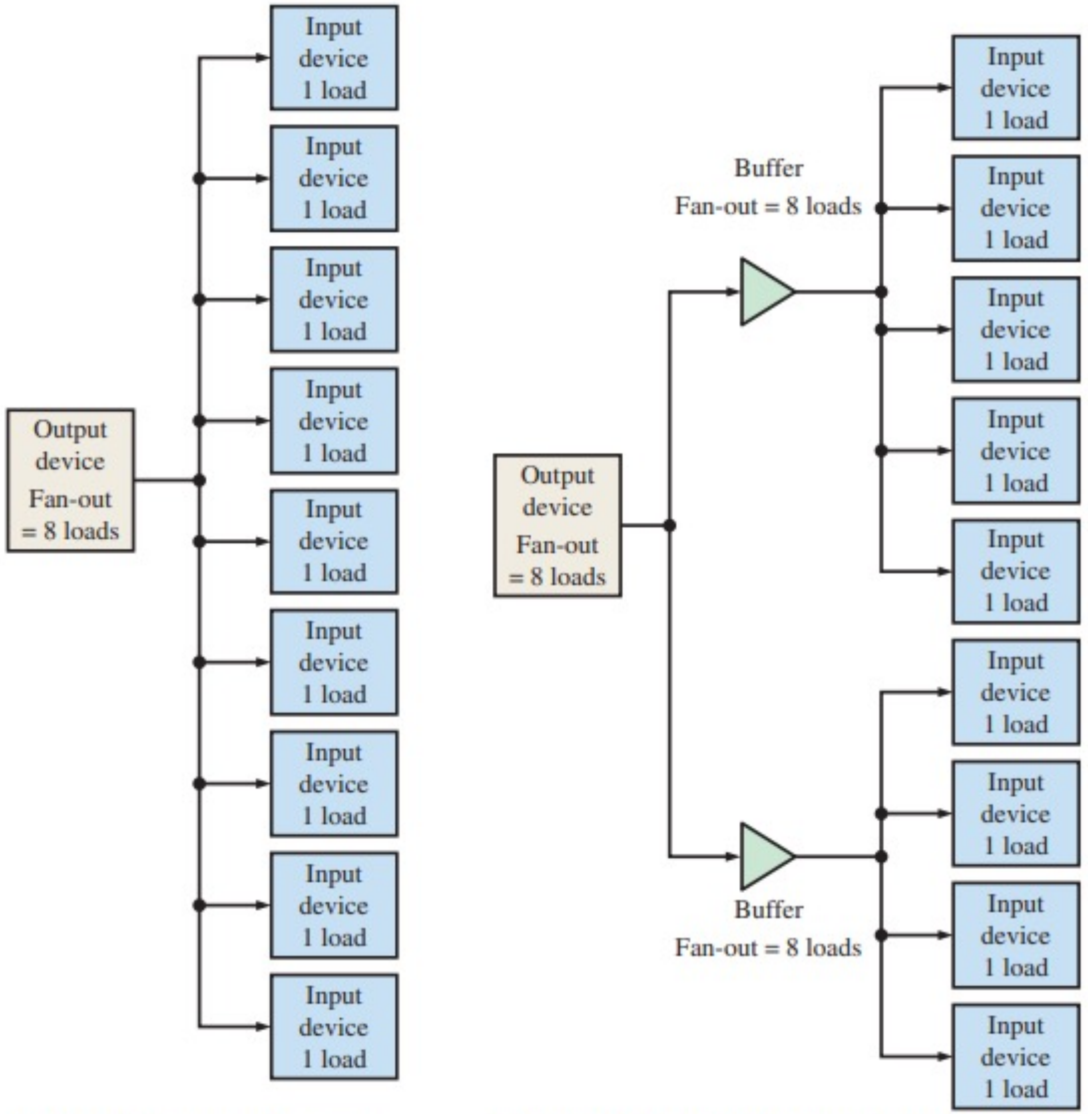


FIGURE 14–7 A wait-state generator programmed for one wait state.

Signal Loading and Buffering



(a) Nine input device loads exceed 8-load fan-out of output device

(b) Buffering of output device prevents signal loading

FIGURE 14–5 Buffers are used to prevent overloading of driving device.

Device Selection

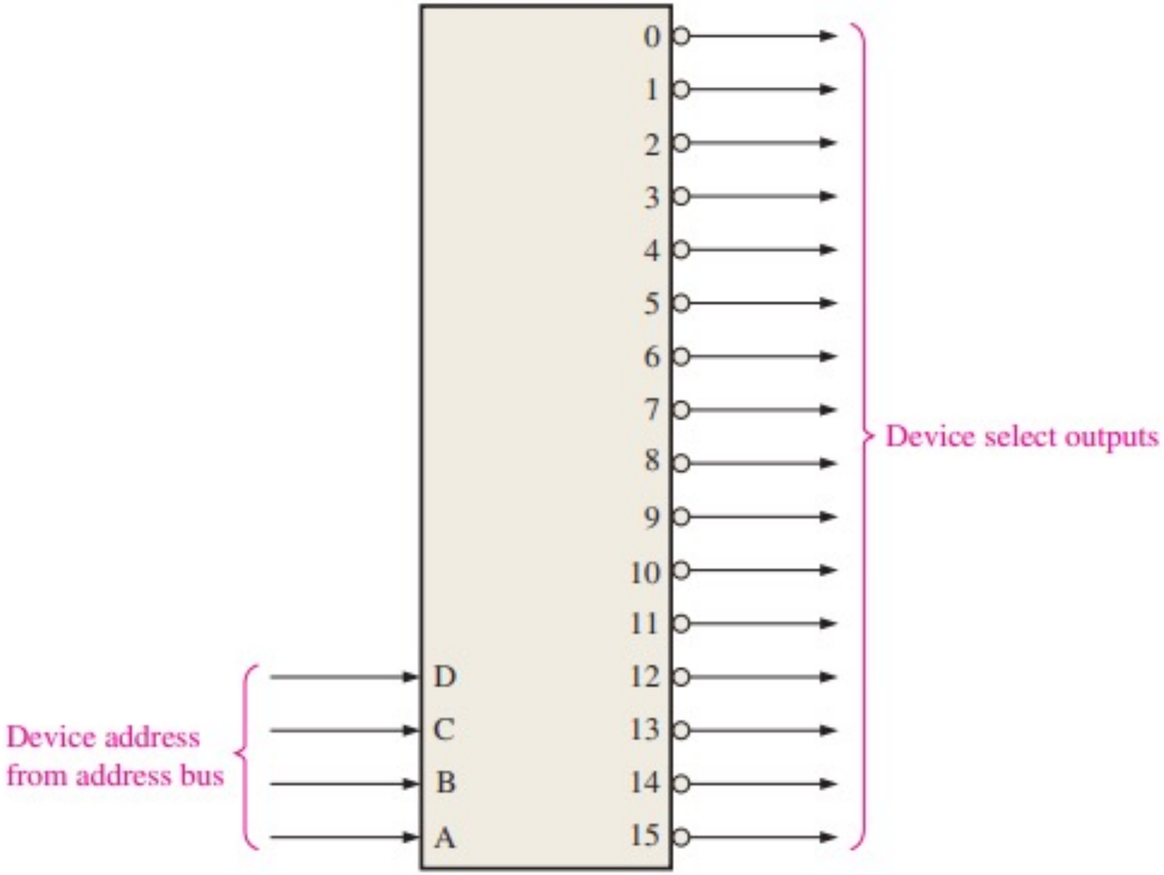


FIGURE 14–6 Address decoding for the purpose of device selection.

EXAMPLE 14–1

For the wait-state generator in Figure 14–8, how many waits states will be generated when the device is selected?

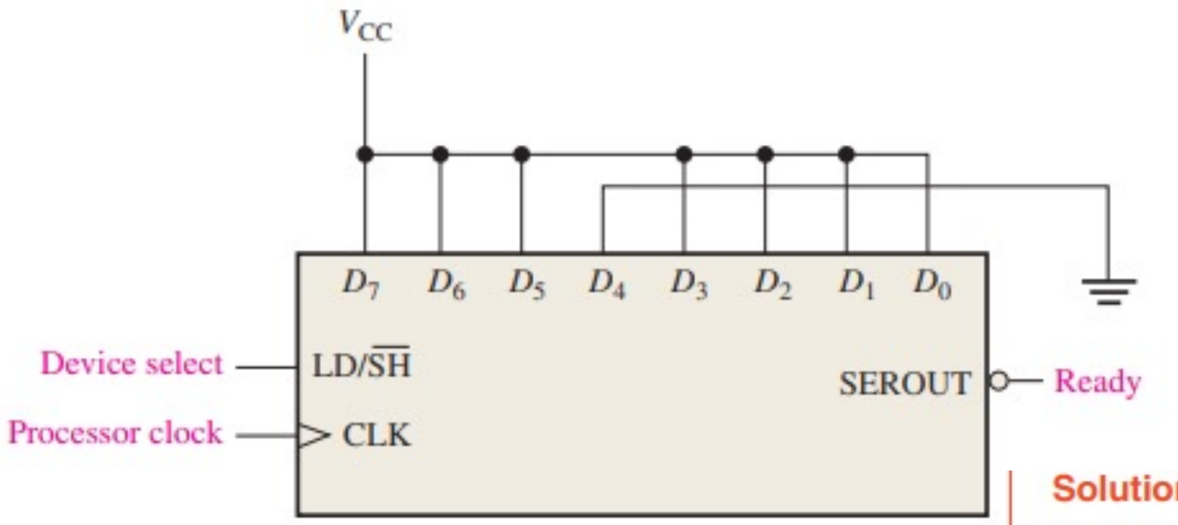


FIGURE 14–8

Solution

The initial pattern loaded into the shift register is 11101111_2. This shifted pattern for each clock and the corresponding number of wait states are

Clock 1 (0 wait states): 11110111_2

Clock 2 (1 wait state): 11111011_2

Clock 3 (2 wait states): 11111101_2

Clock 4 (3 wait states): 11111110_2

On the fourth clock after Device select goes LOW, the most significant bit of the SEROUT line for the shift register goes LOW. This causes the Ready output to go LOW, terminating the bus cycle. Therefore, the wait-state generator inserts **three** wait states.

3. The Processor: Basic Operation

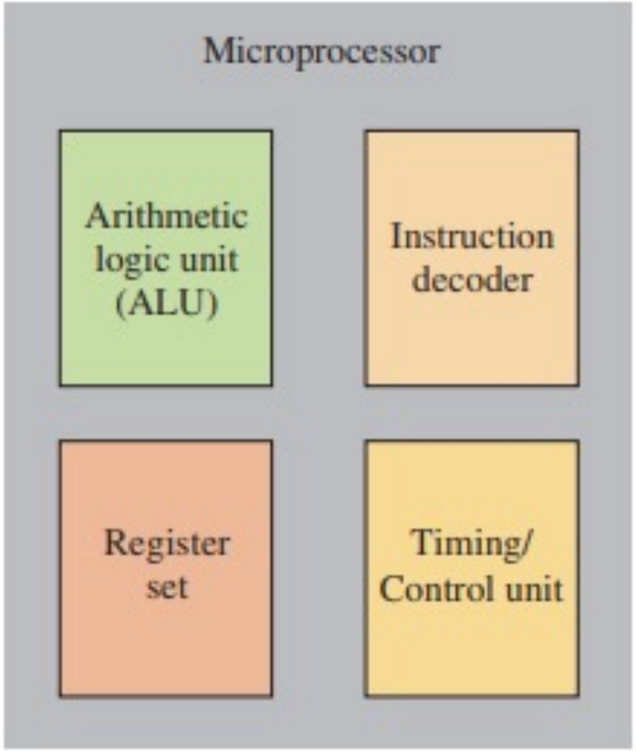


FIGURE 14–9 Elements of a microprocessor (CPU).

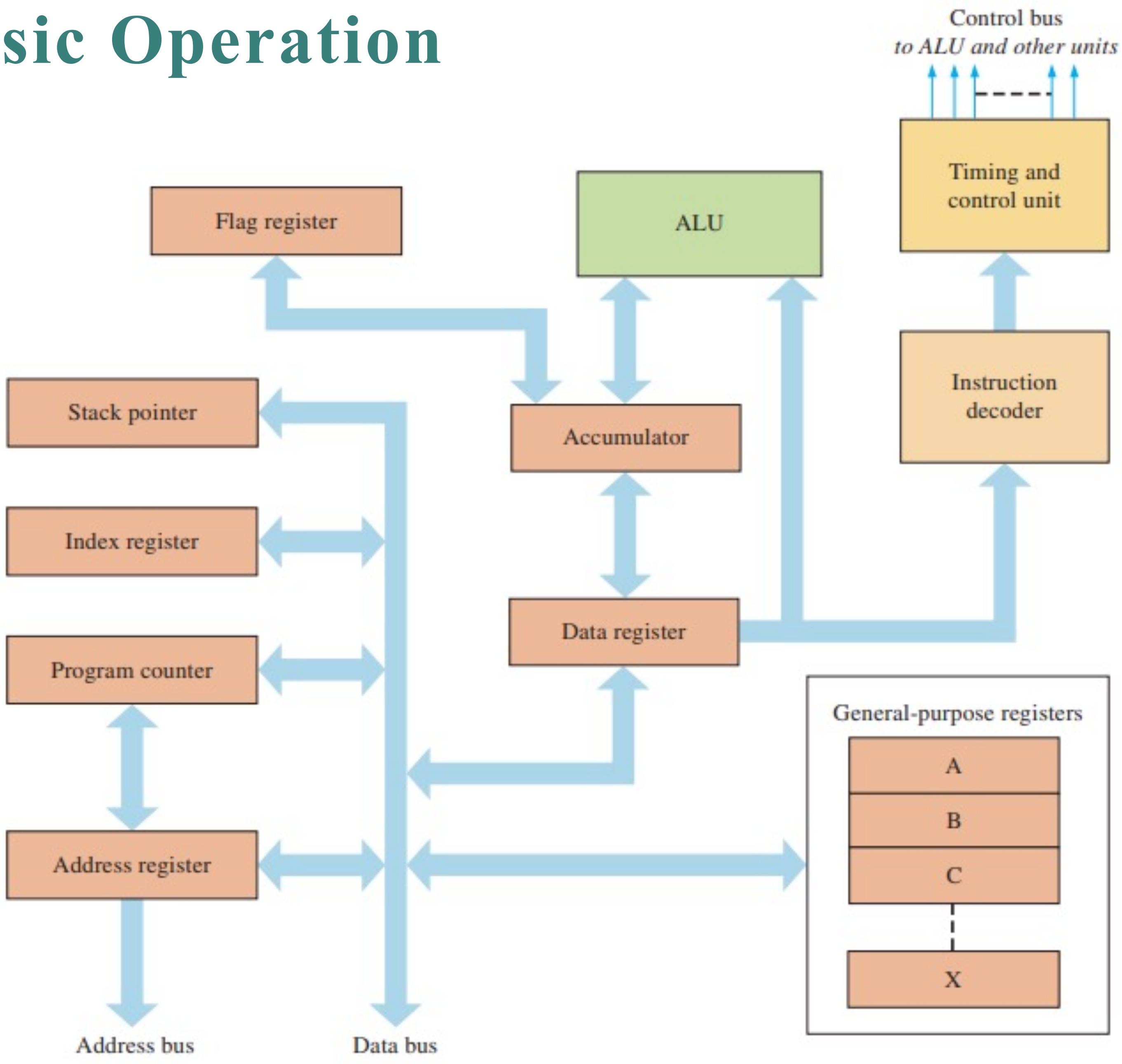


FIGURE 14–10 Basic model of a simplified processor.

The Fetch/Execute Cycle

- During the fetch phase, an instruction is read from the memory and decoded by the instruction decoder.
- During the execute phase, the processor carries out the sequence of operations called for by the instruction. As soon as one instruction has been executed, the processor returns to the fetch phase to get the next instruction from the memory.

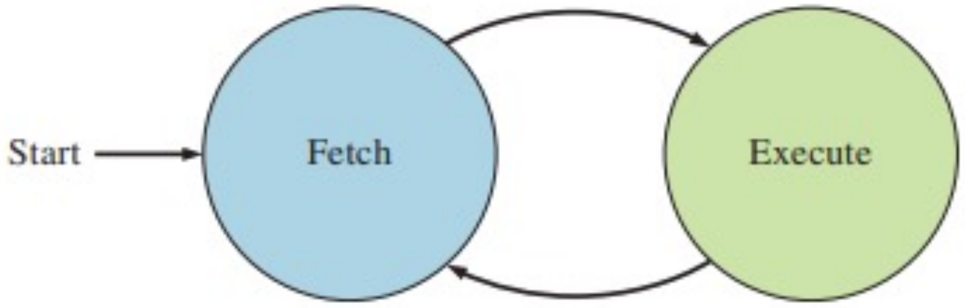


FIGURE 14–11 The fetch/execute cycle of a processor.

Pipelining

A technique where the microprocessor begins executing the next instruction in a program before the previous instruction has been completed is called pipelining.

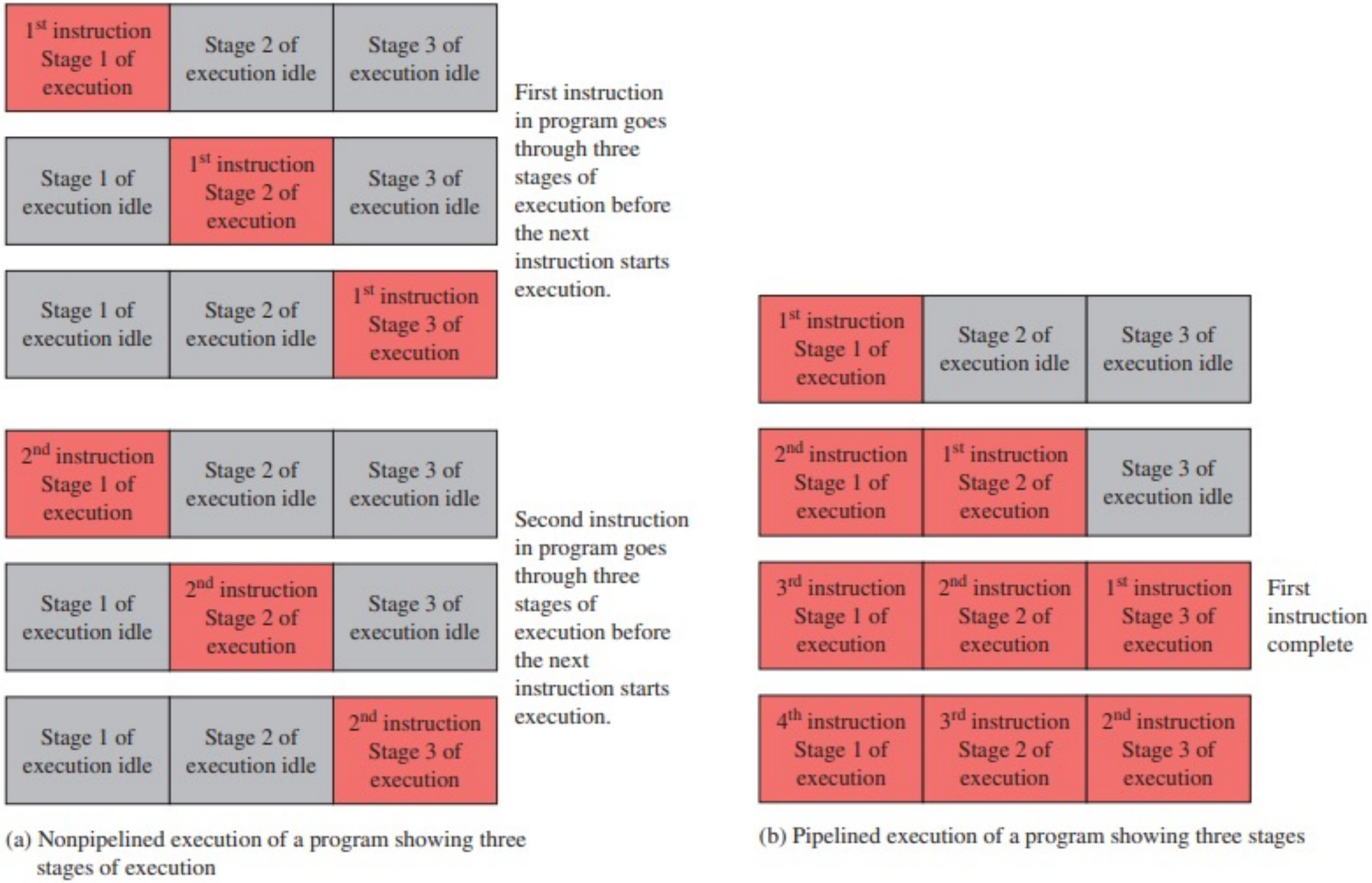


FIGURE 14–12 Illustration of pipelining.

Processor Elements

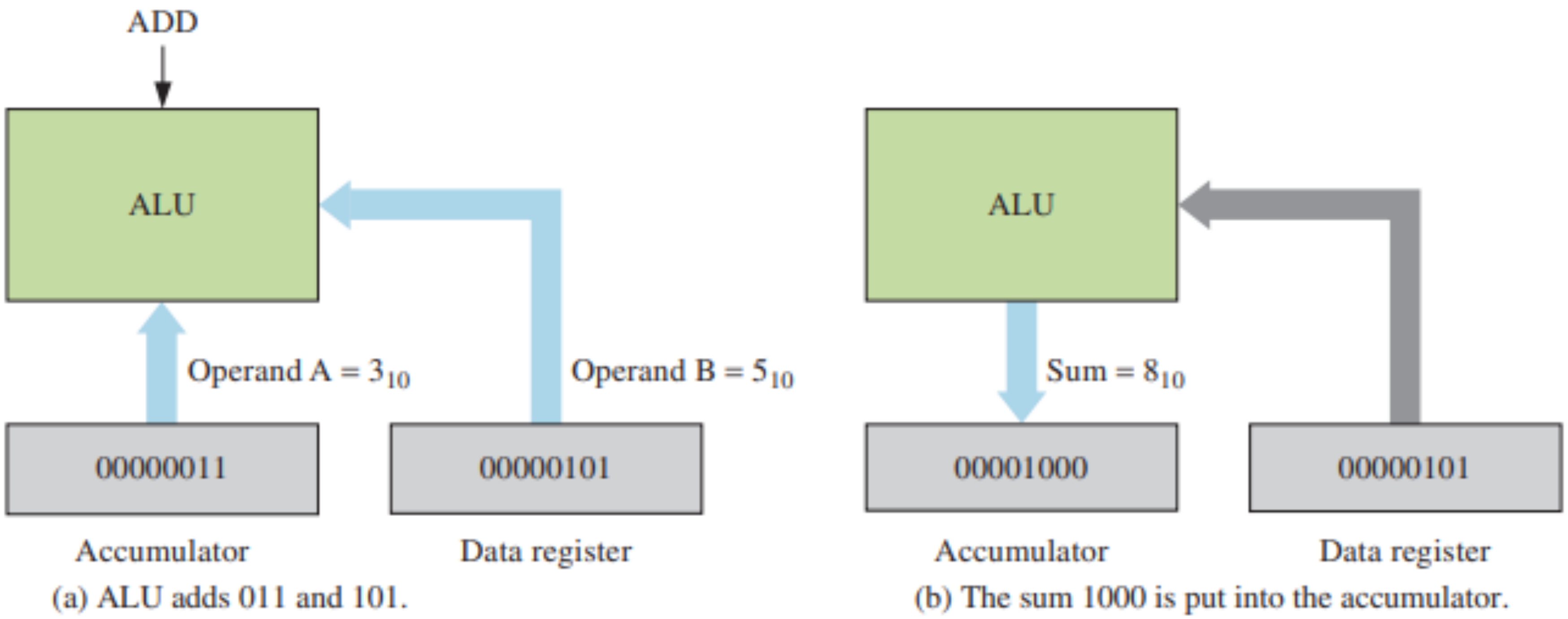


FIGURE 14–13 Example of the ALU adding two operands.

The Processor and the Memory

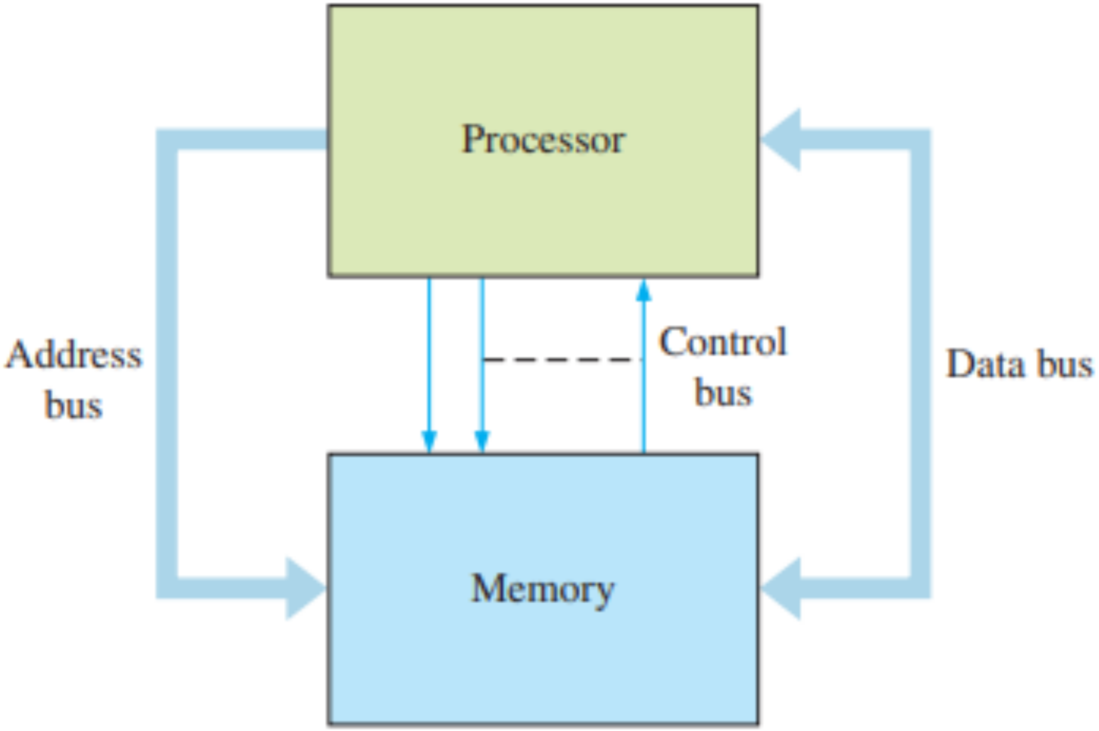
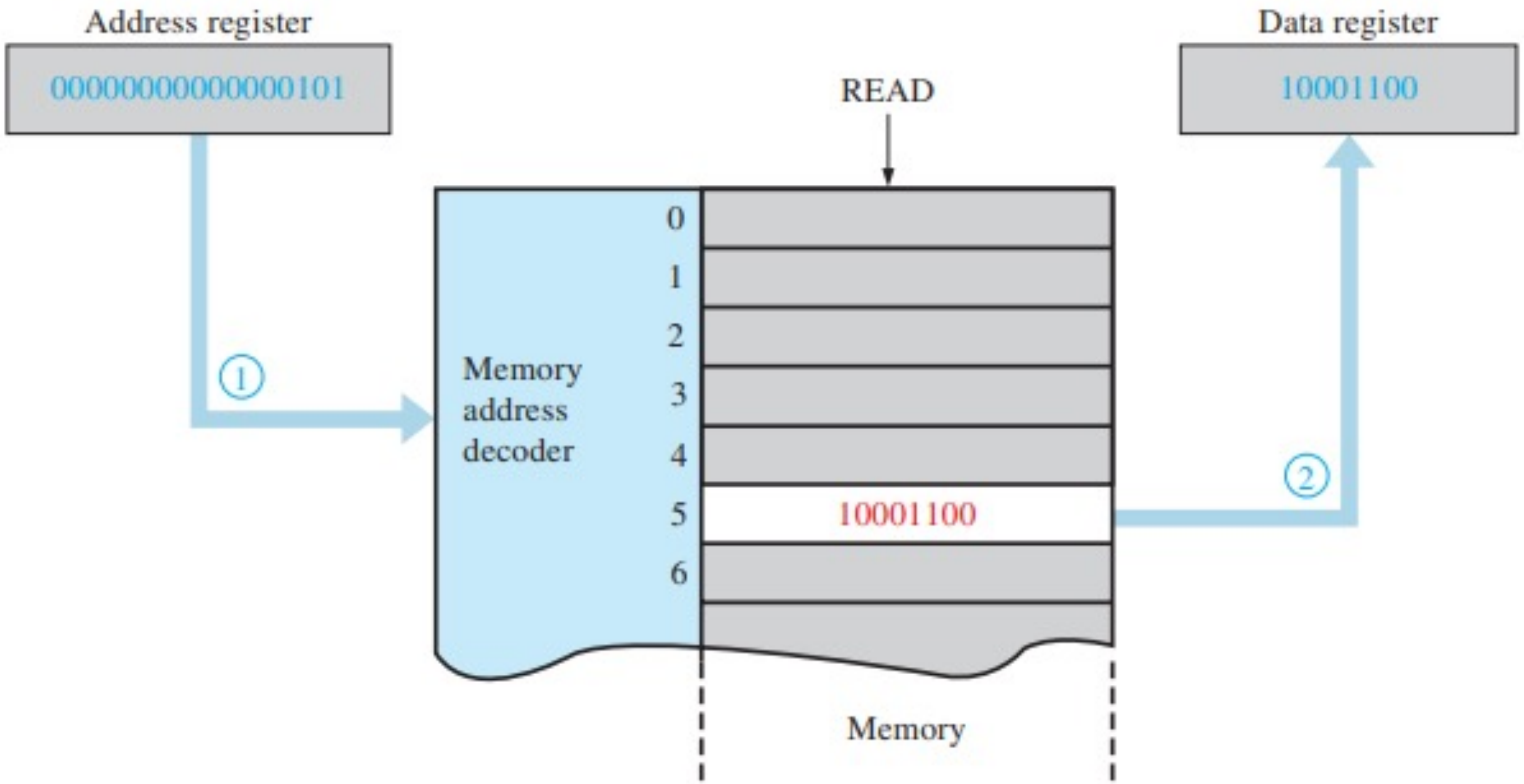


FIGURE 14–14 A processor and memory.

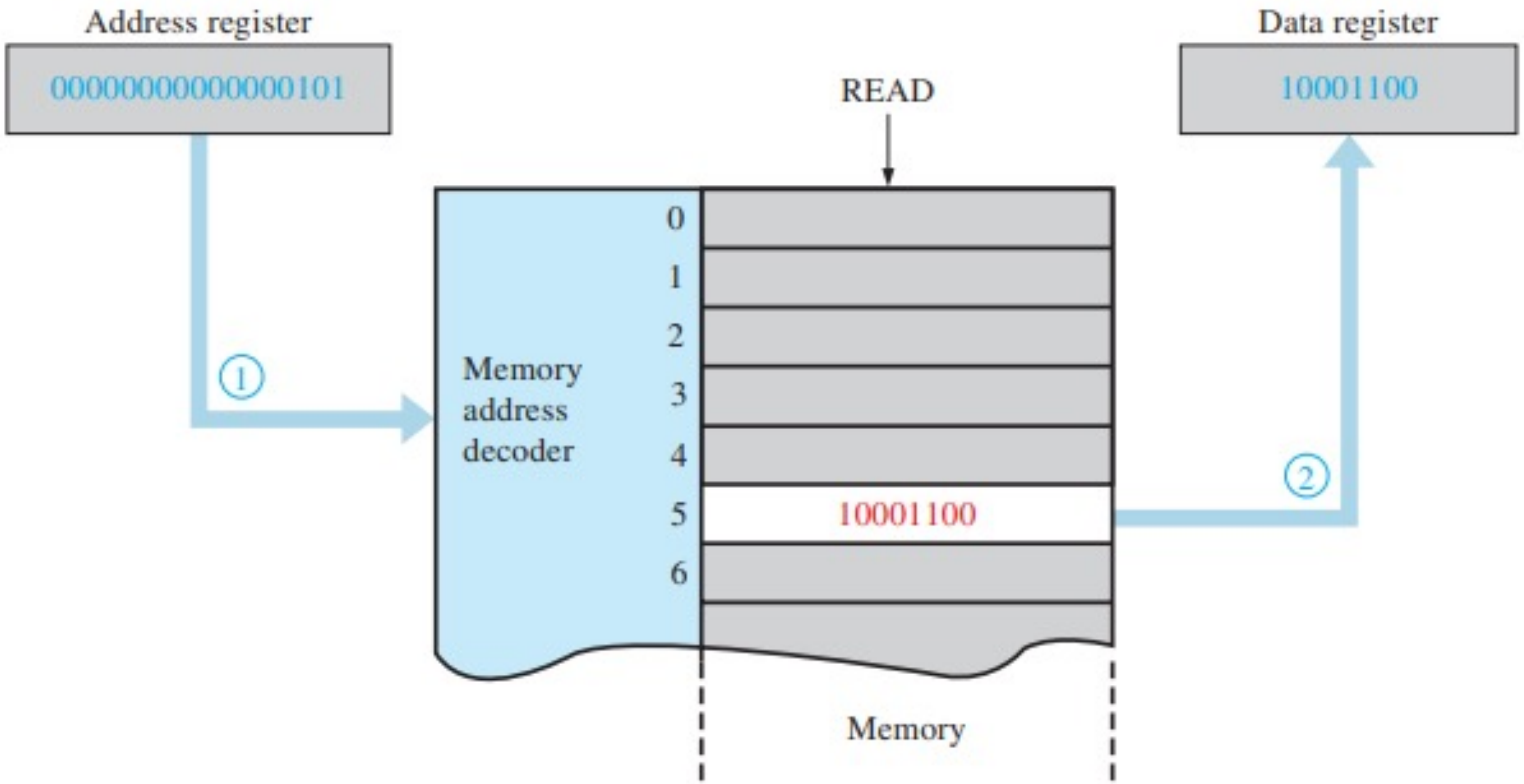
The Read Operation



① Address 5_{10} is placed on address bus and followed by the read signal.
② Contents of address 5_{10} in memory is placed on data bus and stored in data register.

FIGURE 14–15 Illustration of the read operation.

The Write Operation

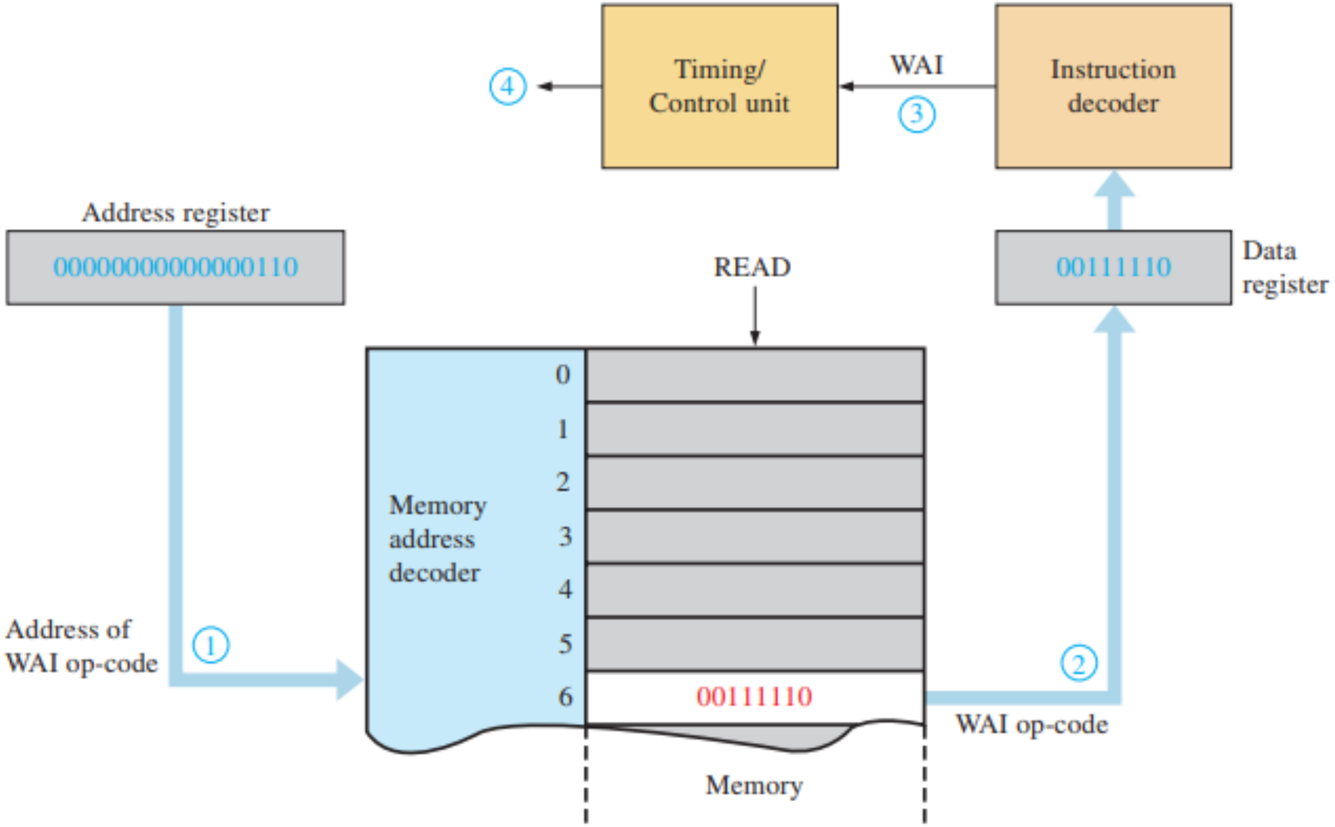


1. Address 5_{10} is placed on address bus and followed by the read signal.
2. Contents of address 5_{10} in memory is placed on data bus and stored in data register.

FIGURE 14–15 Illustration of the read operation.

4. The Processor: Addressing Modes

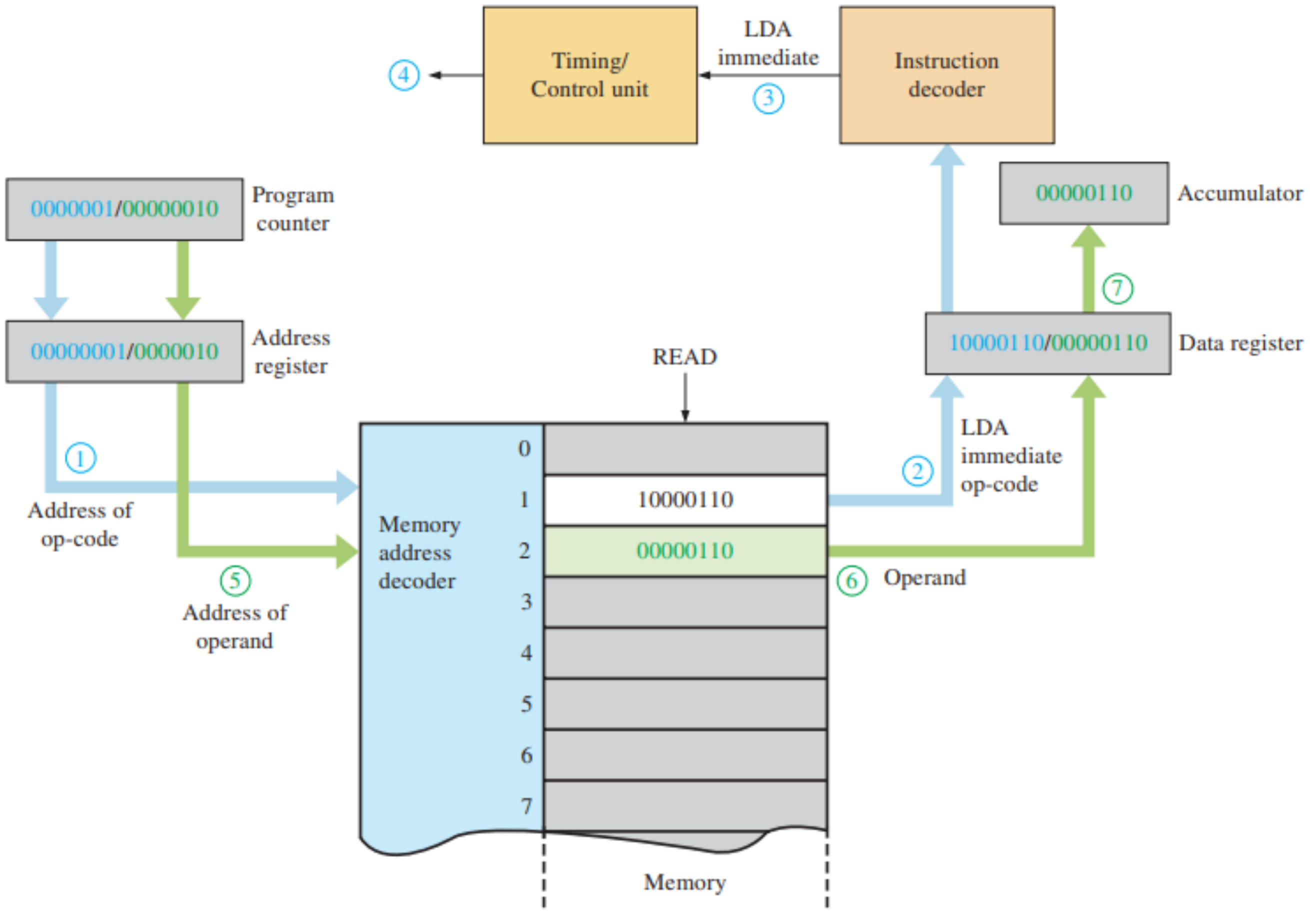
Inherent Addressing



1. Address code (6_{10}) is placed on address bus.
2. Data are placed on data bus and stored in data register by the read signal.
3. Instruction is decoded.
4. Timing/Control unit stops processor operation.

FIGURE 14–17 Fetch/execute cycle for the wait (WAI) instruction. This illustrates inherent addressing.

Immediate Addressing



First fetch/Execute cycle

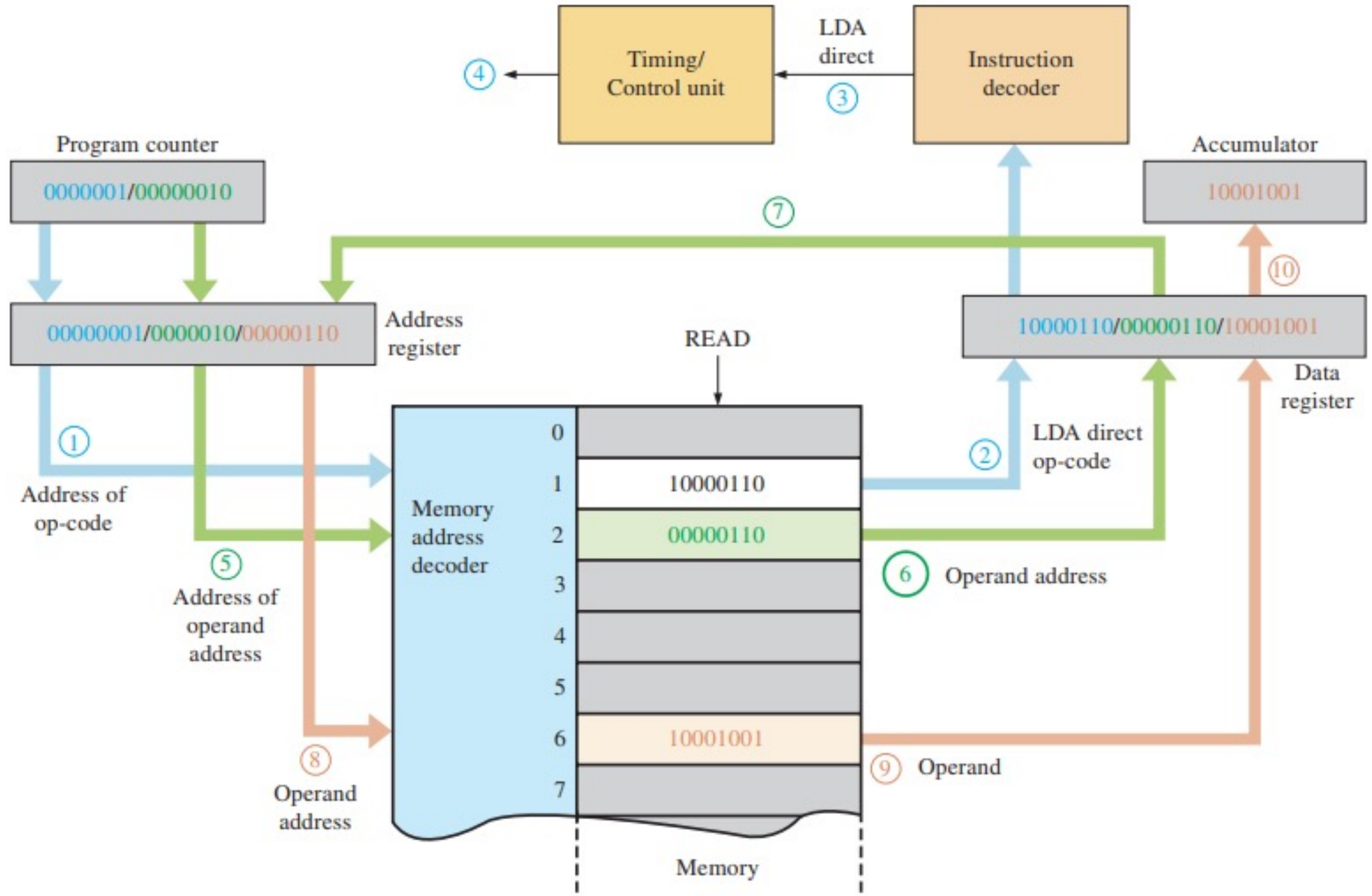
1. Address of LDA immediate op-code (1_{10}) is placed on address bus.
2. LDA immediate op-code is placed on data bus and stored in data register by the read signal.
3. LDA instruction is decoded.
4. Timing/Control unit initiates a read operation to fetch the operand.

Second fetch/Execute cycle

5. Address of operand (2_{10}) is placed on address bus.
6. Operand is placed on data bus and stored in data register by the read signal.
7. Operand is loaded into accumulator.

FIGURE 14–18 Illustration of immediate addressing. The process steps are numbered in sequence, and the cycle operations are color-coded.

Direct Addressing



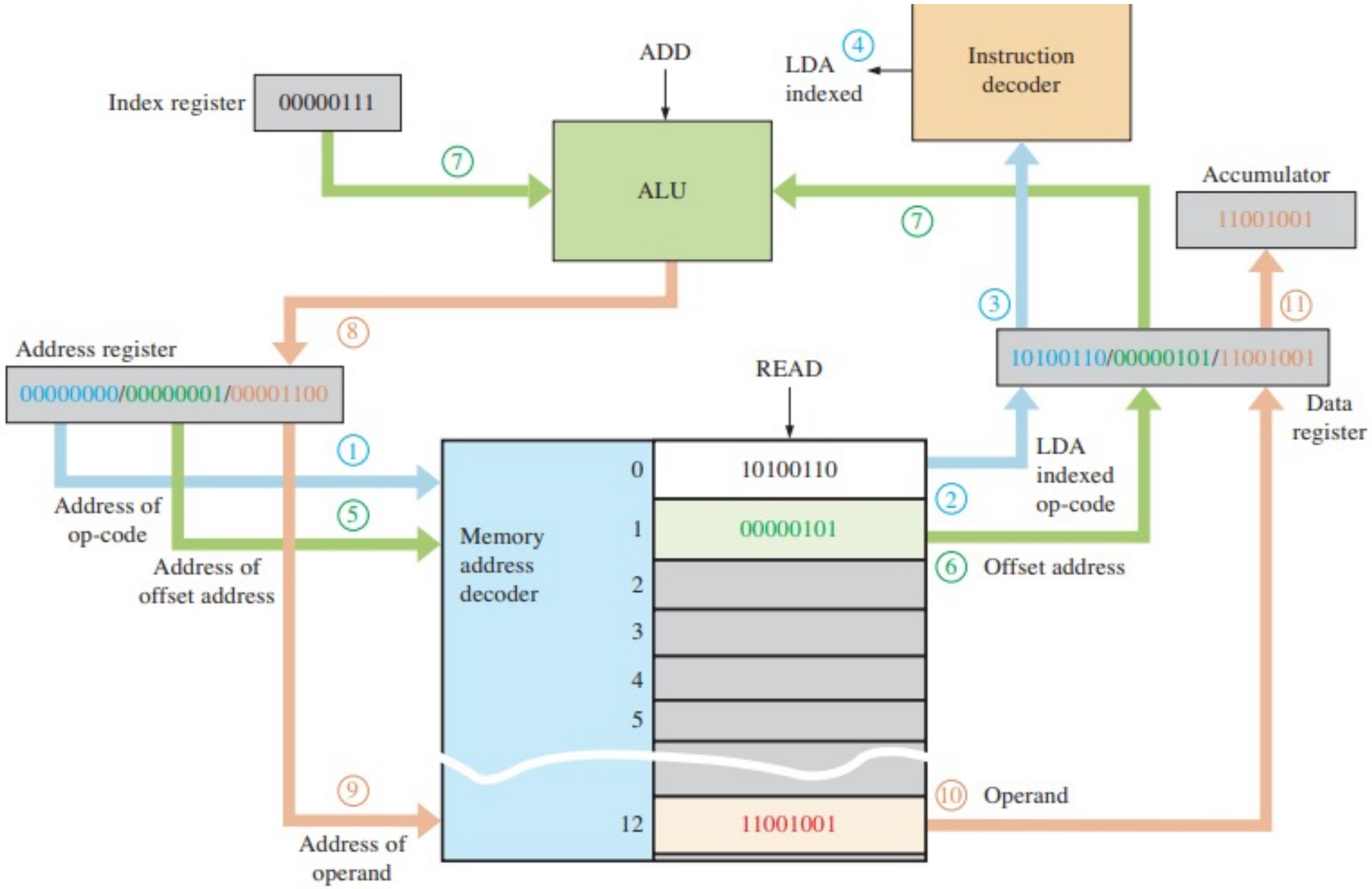
First fetch/Execute cycle
1. Address of LDA direct op-code (1_{10}) is placed on address bus.
2. LDA direct op-code is placed on data bus and stored in data register by the read signal.
3. LDA instruction is decoded.
4. Timing/Control unit initiates a read operation to fetch the address of the operand.

Second fetch/Execute cycle
5. Address of operand address (2_{10}) is placed on address bus.
6. Operand address is placed on data bus and stored in data register by the read signal.
7. Operand address (6_{10}) is loaded into address register.

Third fetch/Execute cycle
8. Operand address (6_{10}) is placed on address register.
9. Operand is placed on data bus and loaded into data register.
10. Operand is loaded into accumulator.

FIGURE 14–19 Illustration of direct addressing.

Indexed Addressing



First fetch/Execute cycle
- 1 Address of LDA indexed op-code (0_{10}) is placed on address bus.
- 2 LDA indexed op-code is placed on data bus and stored in data register by the read signal.
- 3 LDA indexed instruction is decoded.
- 4 Timing/Control unit initiates a read operation to fetch the address of the operand.

Second fetch/Execute cycle
- 5 Offset address (5_{10}) is selected at address 1_{10}.
- 6 Offset address is placed on data bus and stored in data register by the read signal.
- 7 Offset address is added to contents of index register (7_{10}) by the ALU to produce address of operand (12_{10}).

Third fetch/Execute cycle
- 8 Address of operand (12_{10}) is loaded into address register.
- 9 Address of operand is placed on address bus.
- 10 Operand is placed on data bus and stored in data register by the read signal.
- 11 Operand is transferred into accumulator.

FIGURE 14–20 Illustration of indexed addressing.

5. The Processor: Special Operations

Interrupts and Exceptions

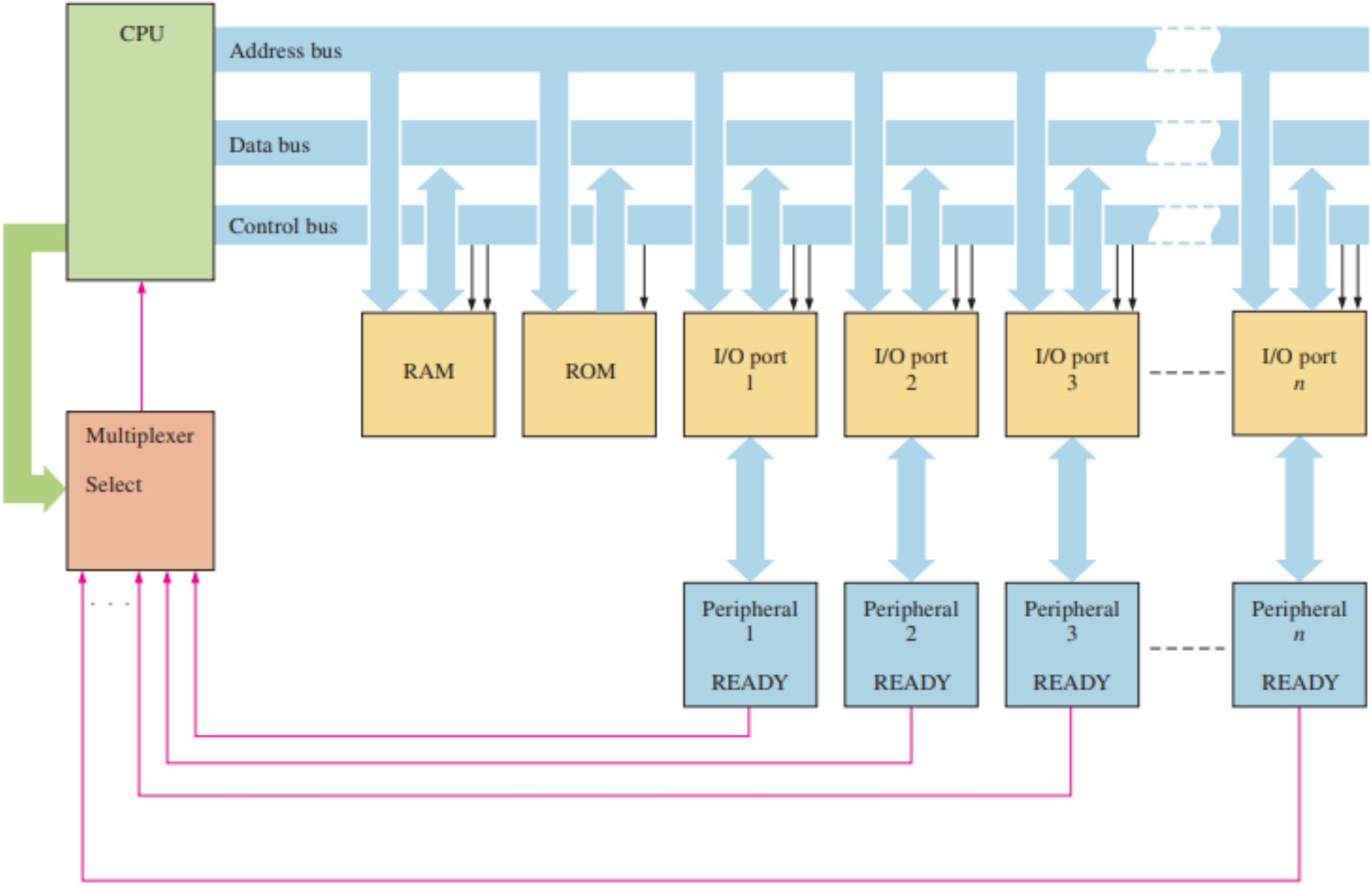
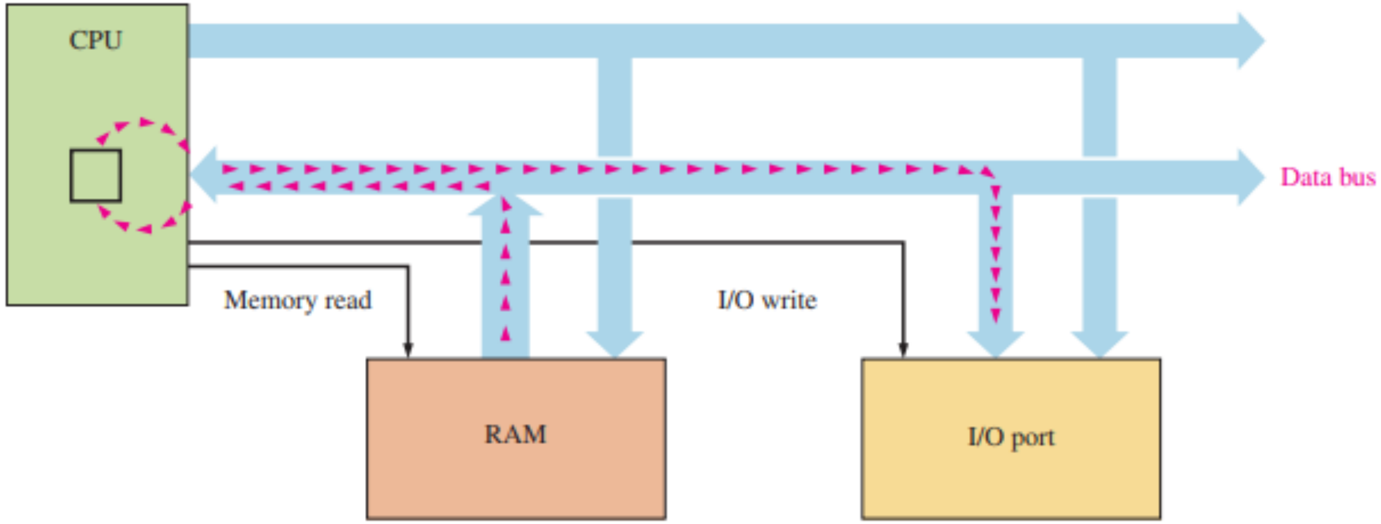
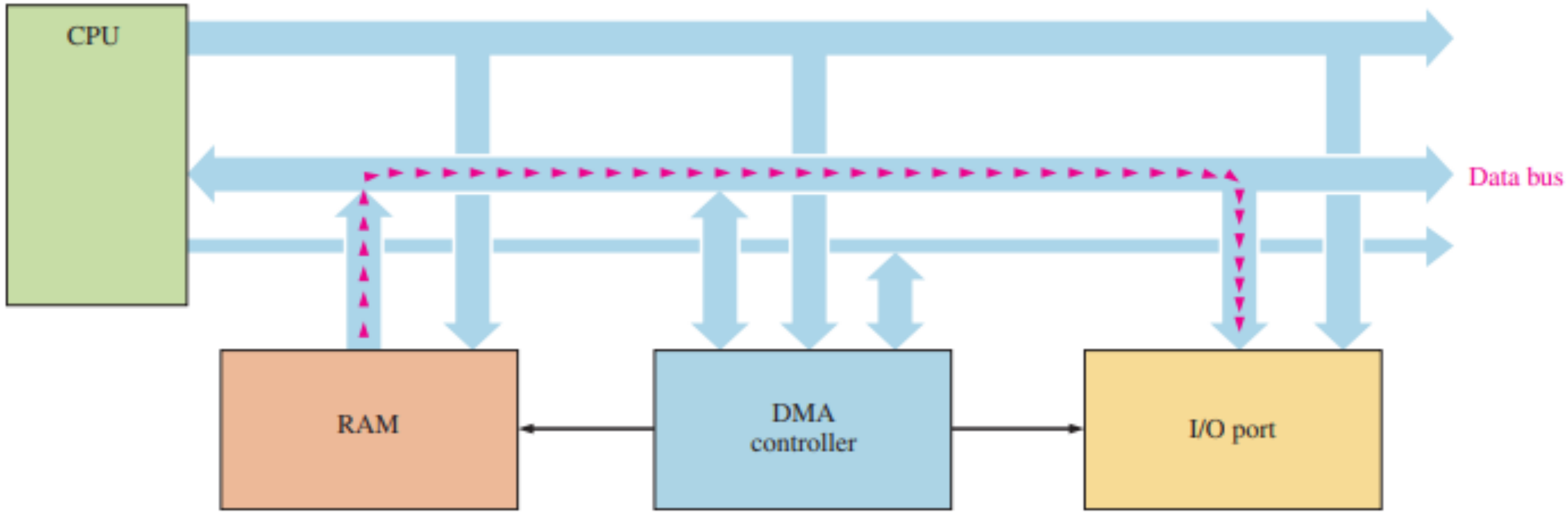


FIGURE 14–22 Basic concept of CPU polling peripheral devices.

Direct Memory Access (DMA)



(a) Data transfer handled by the CPU



(b) Data transfer handled by the DMA controller

FIGURE 14–24 Illustration of DMA vs CPU data transfer.

Direct Memory Access (DMA)

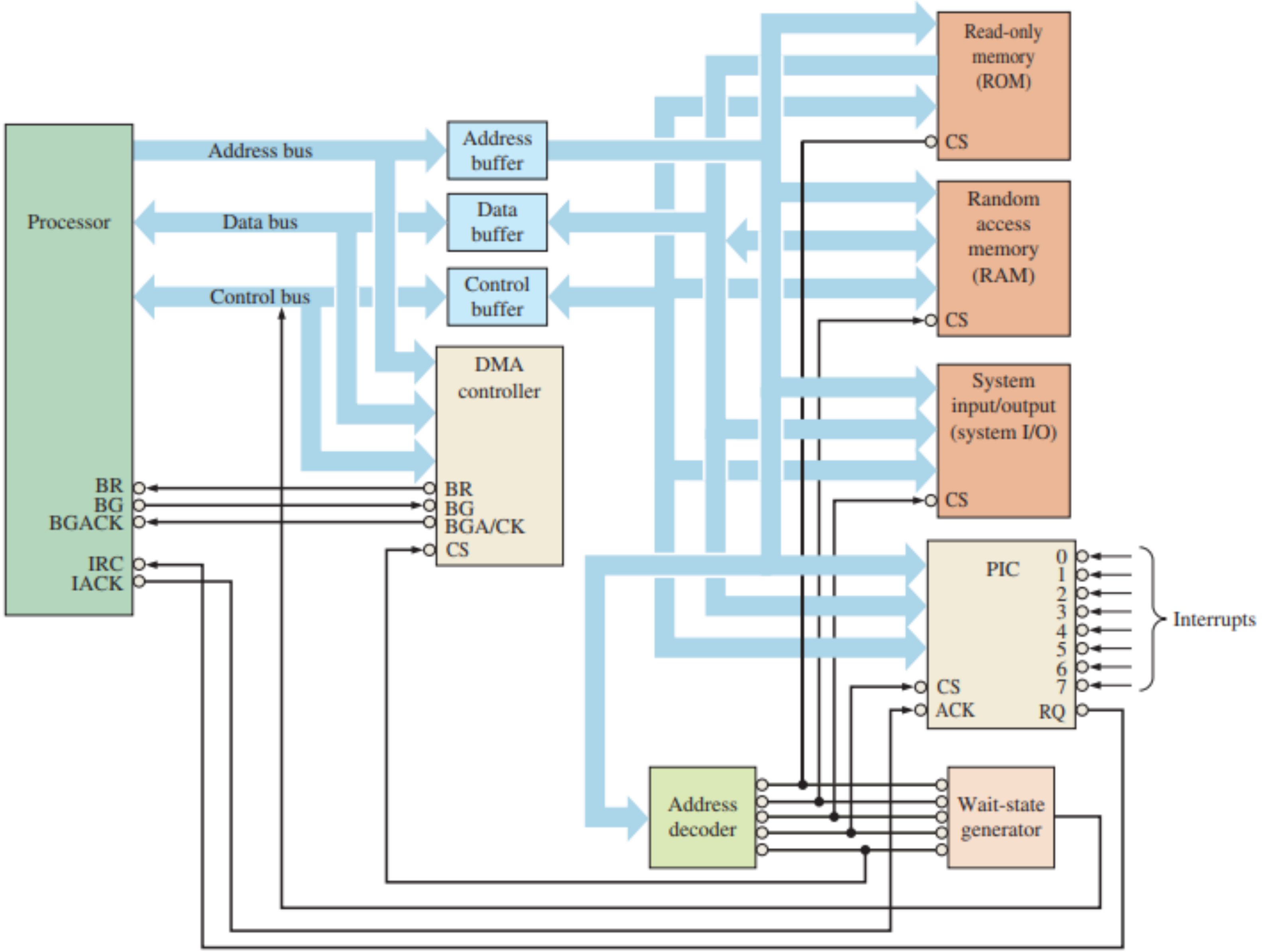


FIGURE 14–25 Block diagram of a typical computer.

6. Operating Systems and Hardware

The operating system (OS) of a computer is a special program that establishes the environment in which application programs operate. The operating system provides the functional interface between application programs in the system, called processes, and the computer hardware.

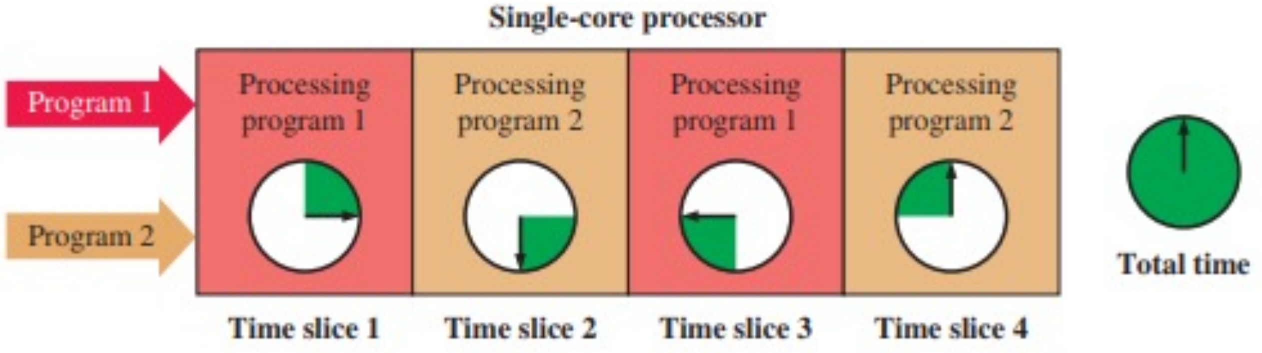


FIGURE 14–26 Simplified model of processor multitasking.

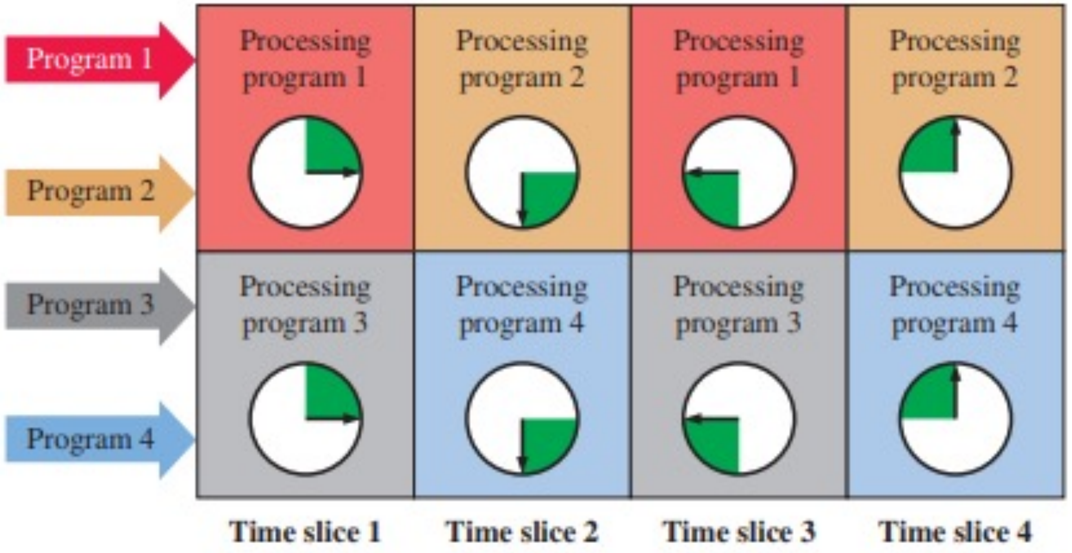


FIGURE 14–27 Multitasked multiprocessing in a multicore processor.

7. Programming

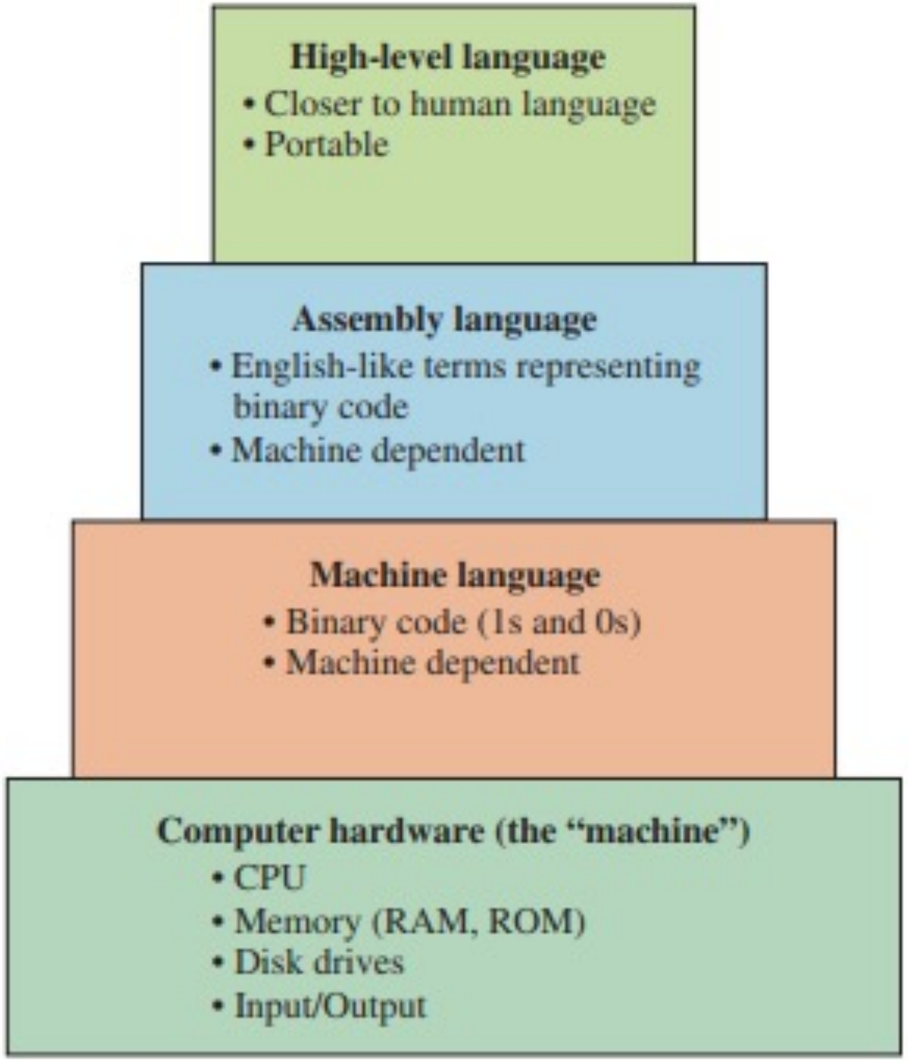


FIGURE 14–28 Hierarchy of programming languages relative to computer hardware.

7. Programming

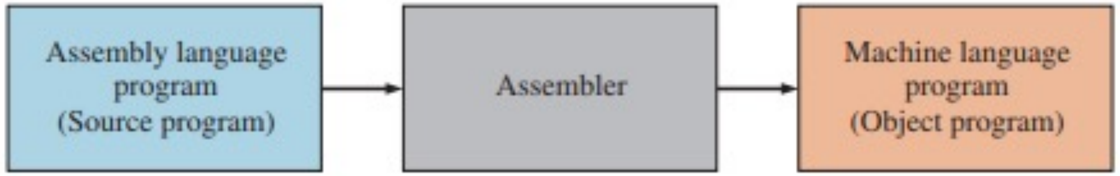


FIGURE 14–29 Assembly to machine conversion using an assembler.

FIGURE 14–30 High-level to machine conversion with a compiler.

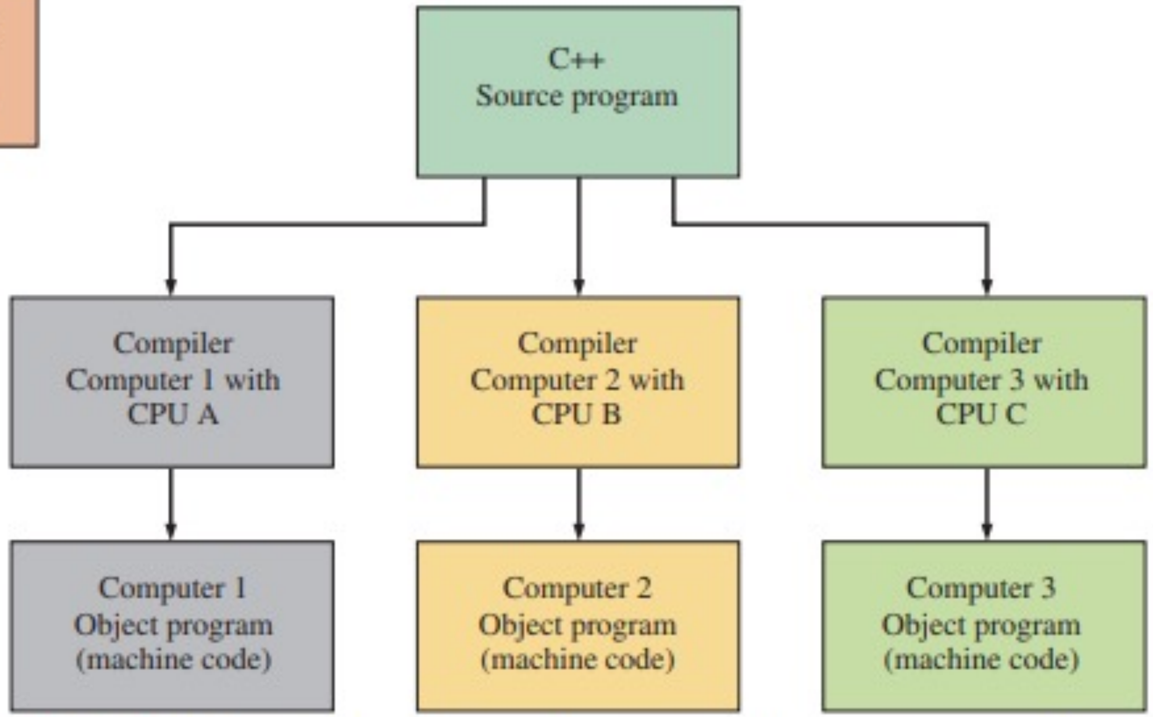


FIGURE 14–31 Machine independence of a program written in a high-level language.

8. Microcontrollers and Embedded Systems

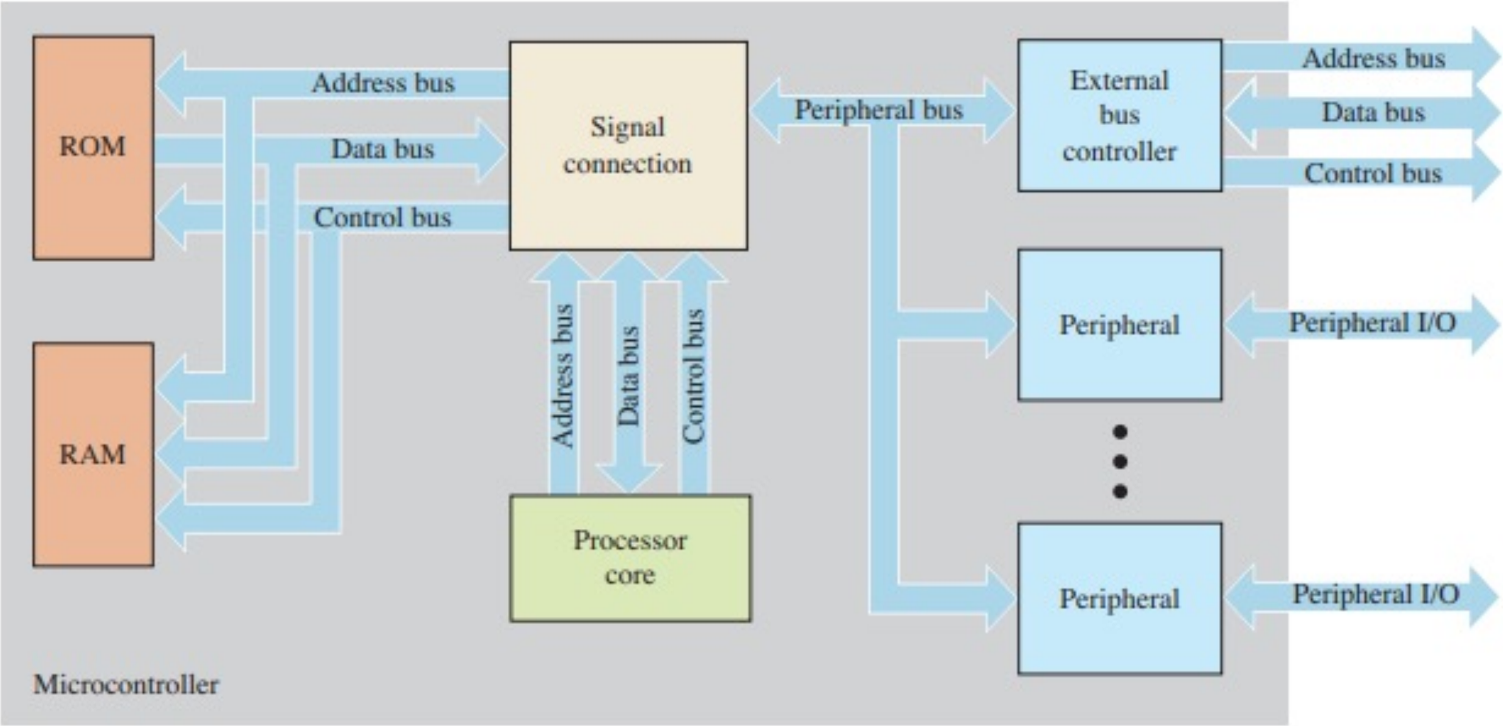


FIGURE 14–34 Simplified microcontroller block diagram.

8. Microcontrollers and Embedded Systems

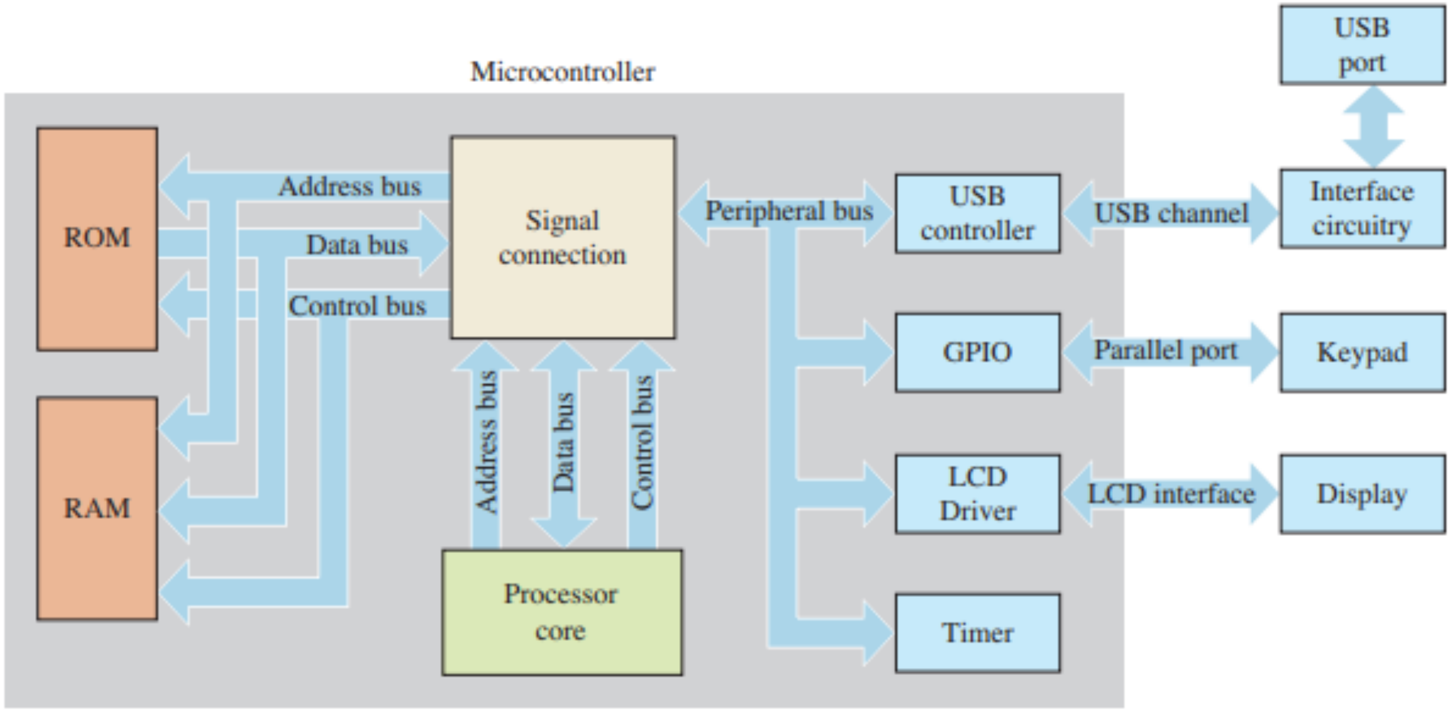


FIGURE 14–35 Microcontroller block diagram for programmable calculator.

8. Microcontrollers and Embedded Systems

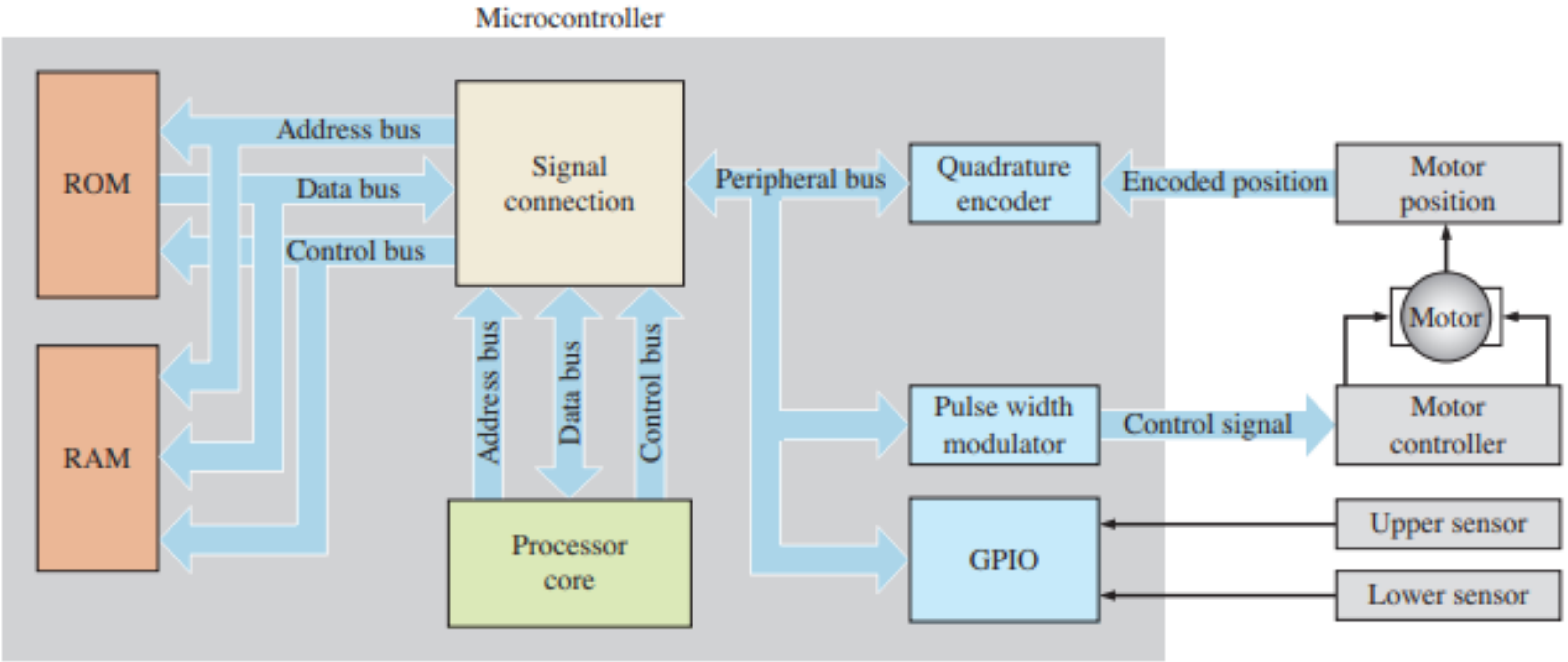
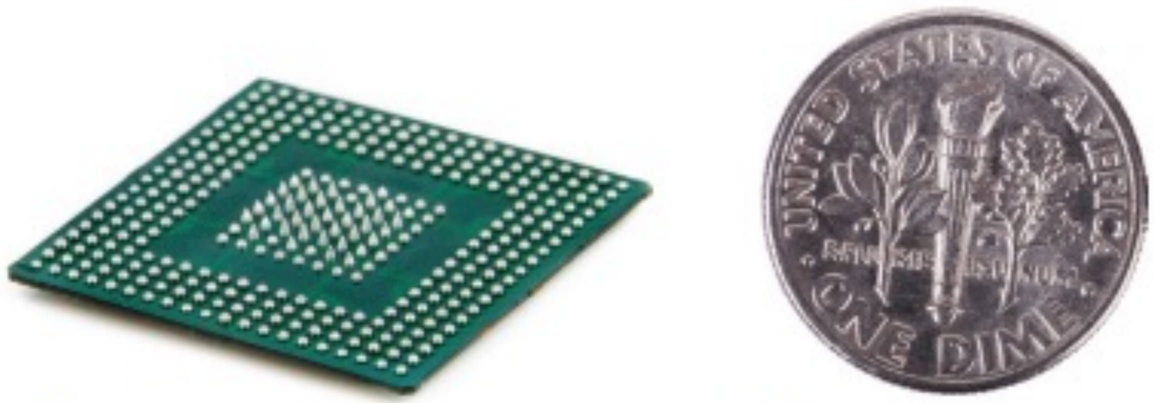


FIGURE 14–36 Basic block diagram for a robotics system.

9. System on Chip (SoC)

A system on chip (SoC) is an integrated circuit that combines all components of a computer or other electronic system on a single chip. The SoC offers reduced manufacturing costs and smaller system configurations; Package sizes can be smaller than a dime.



(a) (b)

FIGURE 14–37 A typical SoC ball-grid package. The bottom of the package with the BG contacts is shown. (a) Boris Sosnovyy/Shutterstock (b) Eldad Carin/Shutterstock.

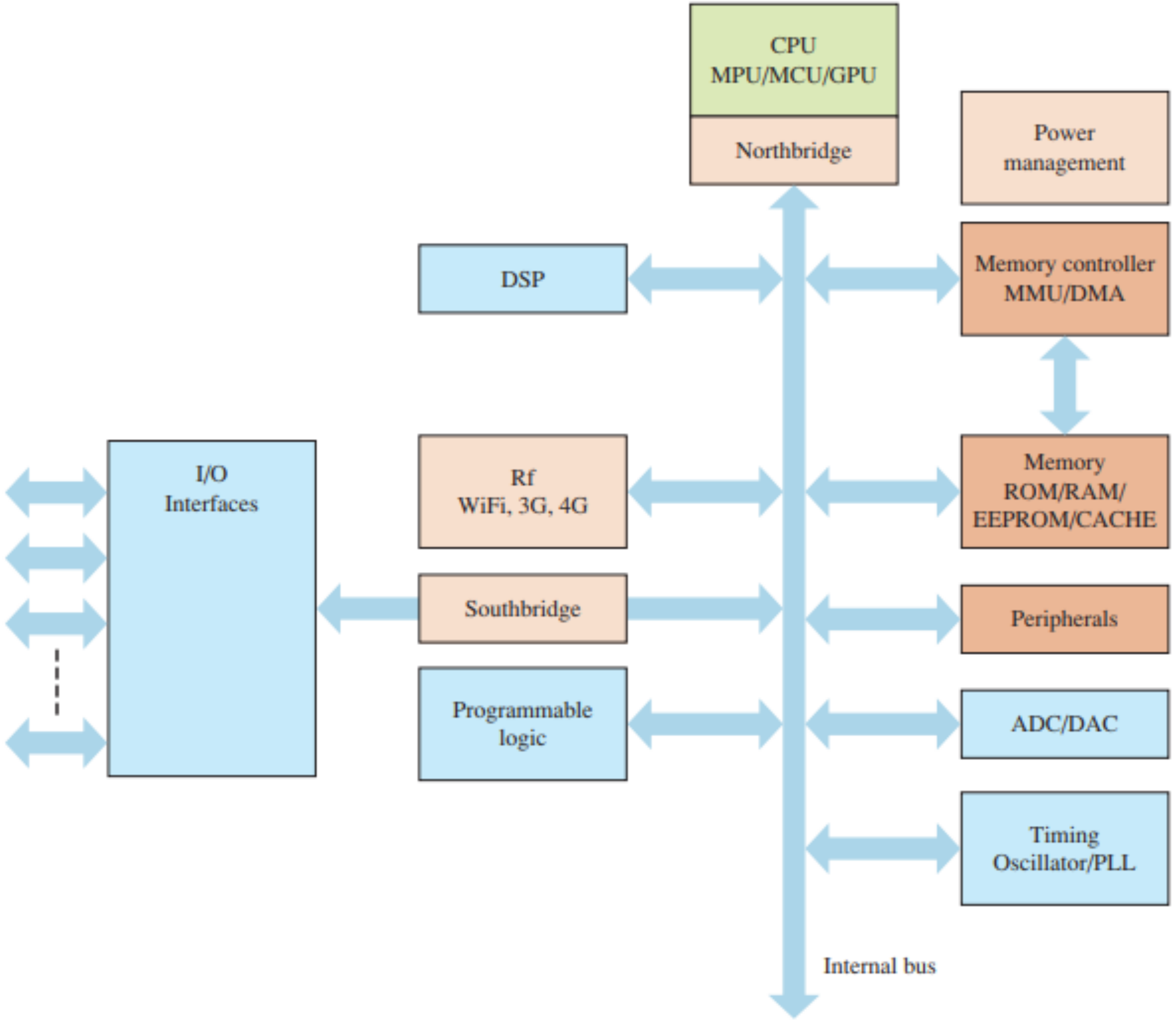


FIGURE 14–38 Generic block diagram of a typical SoC.

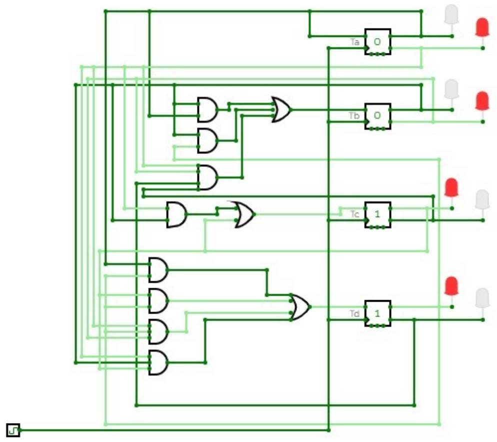
Vietnam National University HCMC

International University

School of Electrical Engineering

THE END

Lecture 14: Data Processing and Control



INSTRUCTOR: Dr. Vuong Quoc Bao