Computer Science
North University of China

CS101 Fall 2017

Introduction to
Computer Science

Instructor: Qingsong Guo
School of Computer Science & Technology

http://abelgo.cn/cs101.html

Think in Data: Data Manipulation

Machine Language, Instructions, and Program Execution

Computer Architecture

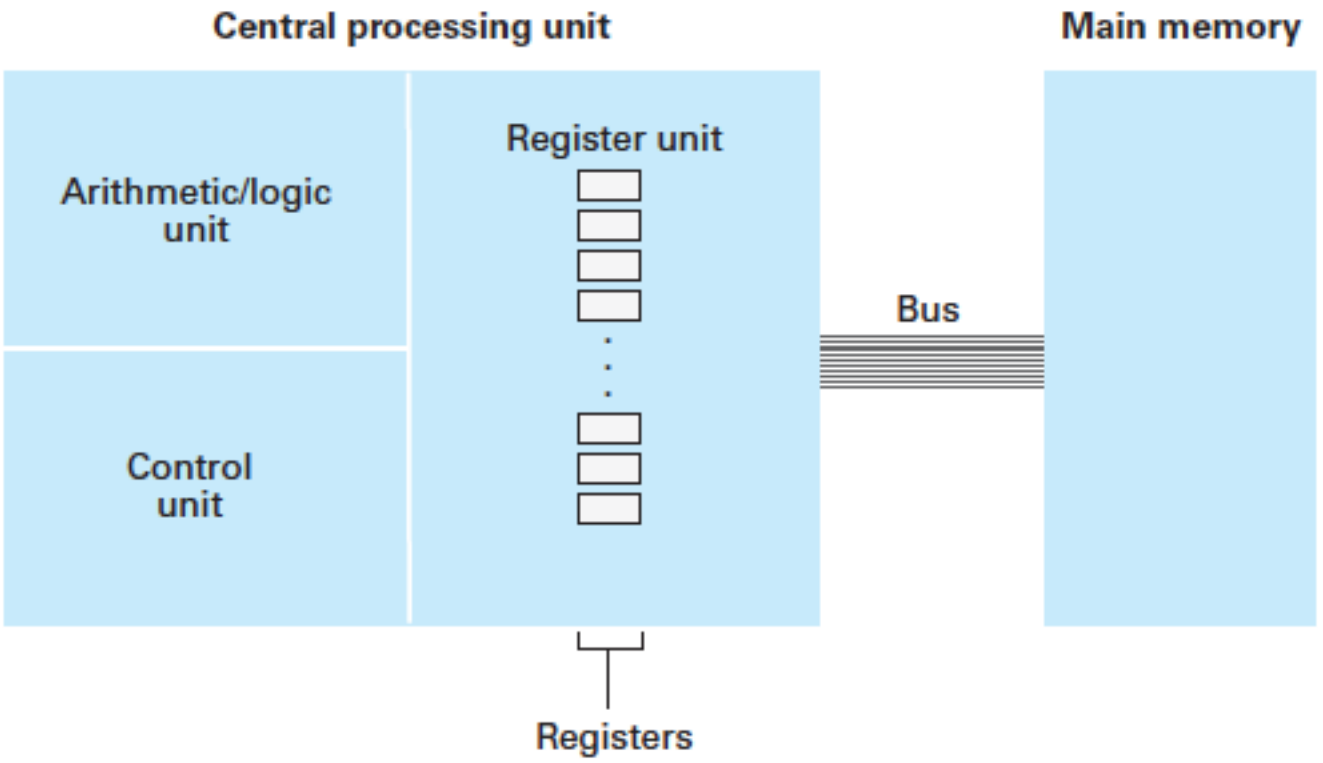
Central Processing Unit (CPU) or processor

- Arithmetic/Logic unit versus Control unit
- Registers
  - General purpose
  - Special purpose

Bus

Motherboard

CPU, Main Memory, and Bus



CPU and main memory connected via a bus

Stored Program Concept

In the early days of computer

- The steps that each device executed were built into the control unit as a part of the machine
- Lack of flexibility

Stored-program concept

- Credited to John von Neumann
- A program, just as data, can be encoded as bit patterns and stored in main memory. From there, the CPU can then extract the instructions and execute them. In turn, the program to be executed can be altered easily.

Terminology

To apply the stored-program concept, we need the following two concepts.

Machine instruction:

- An instruction (or command) encoded as a bit pattern recognizable by the CPU

Machine language:

- The set of all instructions recognized by a machine

The Instruction Repertoire (计算机指令系统)

Reduced Instruction Set Computing (RISC)

- Few, simple, efficient, and fast instructions
- Examples: PowerPC from Apple/IBM/Motorola and SPARK from Sun Microsystems

Complex Instruction Set Computing (CISC)

- Many, convenient, and powerful instructions
- Example: Pentium from Intel

Machine Instruction Types

Data Transfer
- copy data from one location to another
- transfer/move (copy/clone)
- LOAD/STORE, I/O instructions

Arithmetic/Logic
- use existing bit patterns to compute a new bit patterns
- AND, OR, and XOR
- SHIFT/ROTATE

Control
- direct the execution of the program rather than manipulating data
- JUMP:
  - unconditional jumps: “skip to Step 5”
  - conditional jumps: “skip to Step 5 if the value obtained is 0.”

Algorithm 1: Adding Values Stored in Memory

Step 1. Get one of the values to be added from memory and place it in a register.

Step 2. Get the other value to be added from memory and place it in another register.

Step 3. Activate the addition circuitry with the registers used in Step 1 and 2 as inputs and another register designated to hold the result.

Step 4. Store the result in memory.

Step 5. Stop

Algorithm 2: Dividing Values Stored in Memory

Step 1. LOAD a register with a value from memory.

Step 2. LOAD another register with another value from memory.

Step 3. If this second value is zero, JUMP to Step 6

Step 4. Divide the contents of the first register by the second register and leave the result in a third register

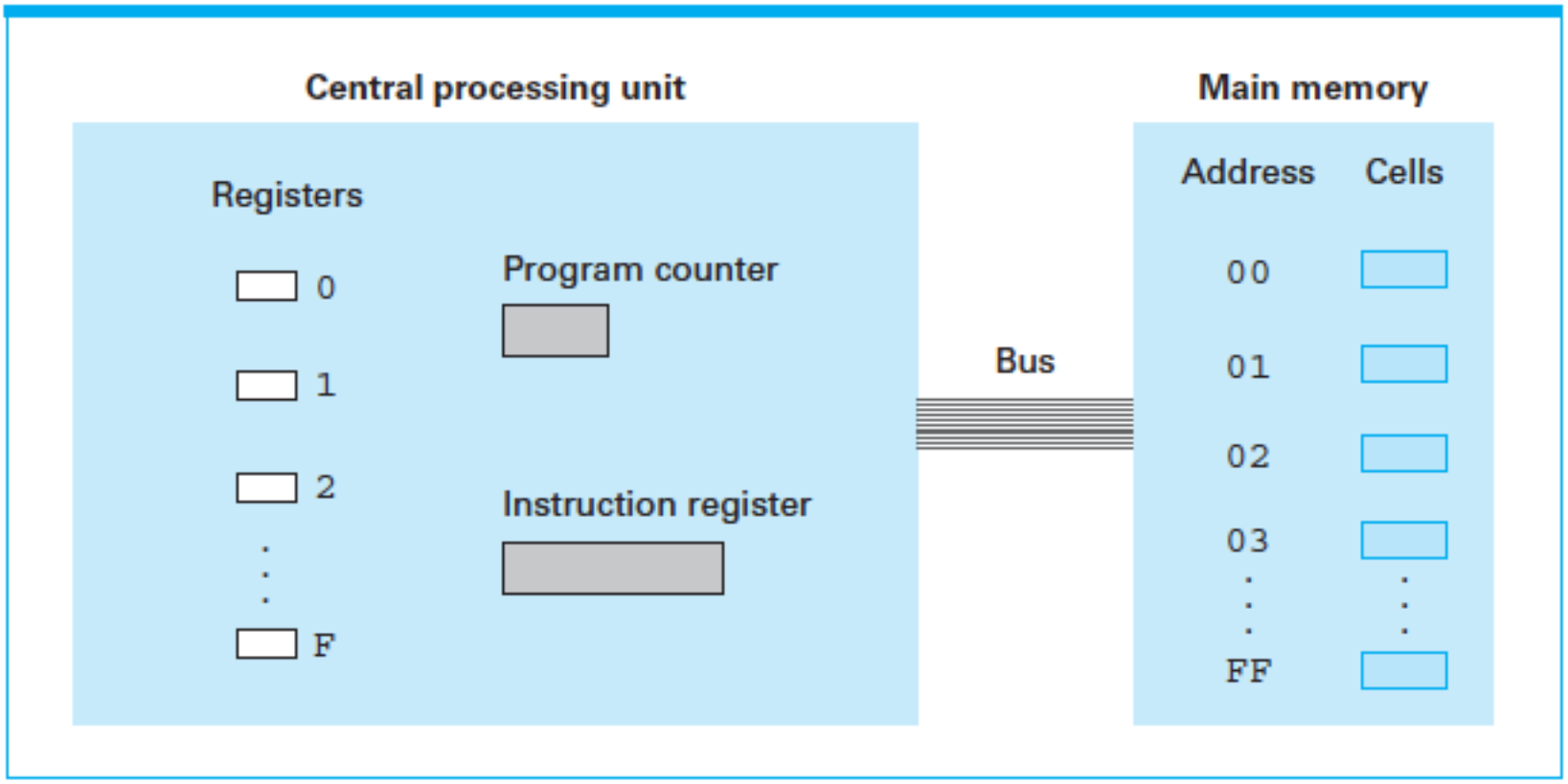
Step 5. STORE the contents of the third register in memory.

Step 6. Stop

An Illustrative Machine – Appendix C

Operation code (Op-code,操作符)

- 16 genera-purpose registers: 0-15 in binary, 0-F in Hex
- 256 main memory cells: 0–255 in binary, 00-FF in Hex



Machine Instructions

Operation code (Op-code, 操作符)

- Specifies which elementary operation to execute
- STORE, SHIFT, XOR, JUMP, etc.

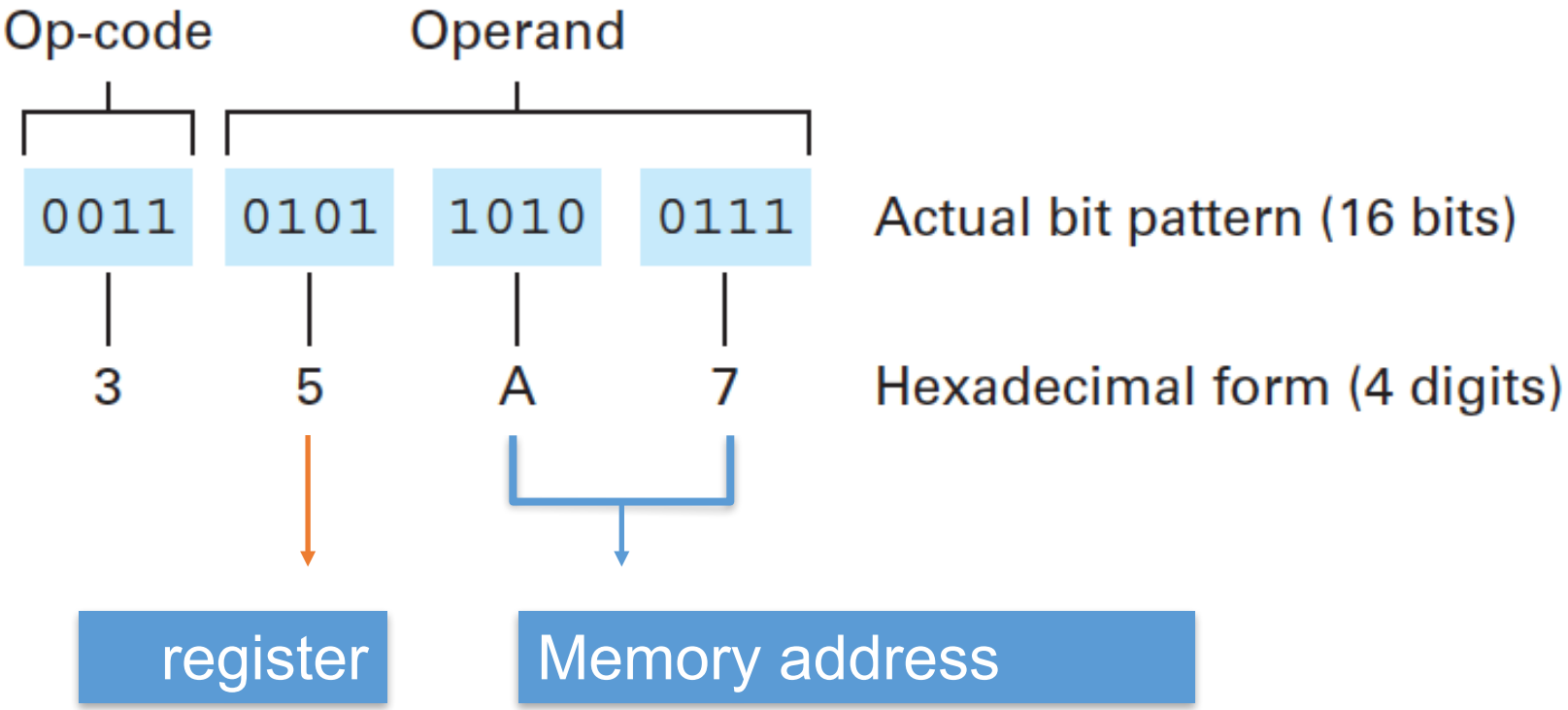
Operand (操作数, data)

- Gives more detailed information about the operation
- STORE: the operand indicates which register contains the data to be stored and which memory cell is to receive the data

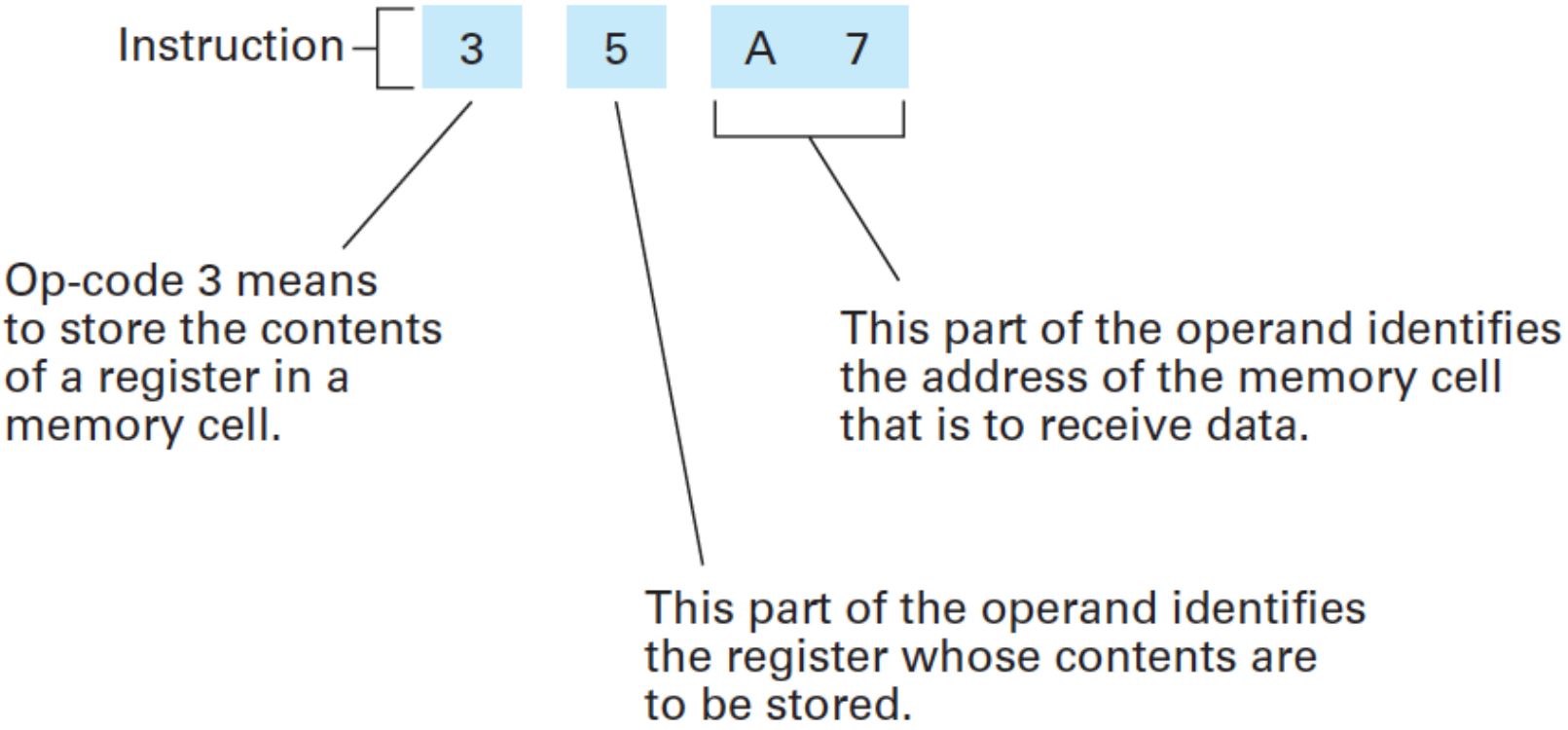
The illustrative machine language

- The illustrative machine consists of 12 instructions
- Interpretation of operand varies depending on op-code

The Composition Of An Instruction



Decoding the Instruction 35A7



An Program Implements Algorithm 1 By Using The Illustrative Machine

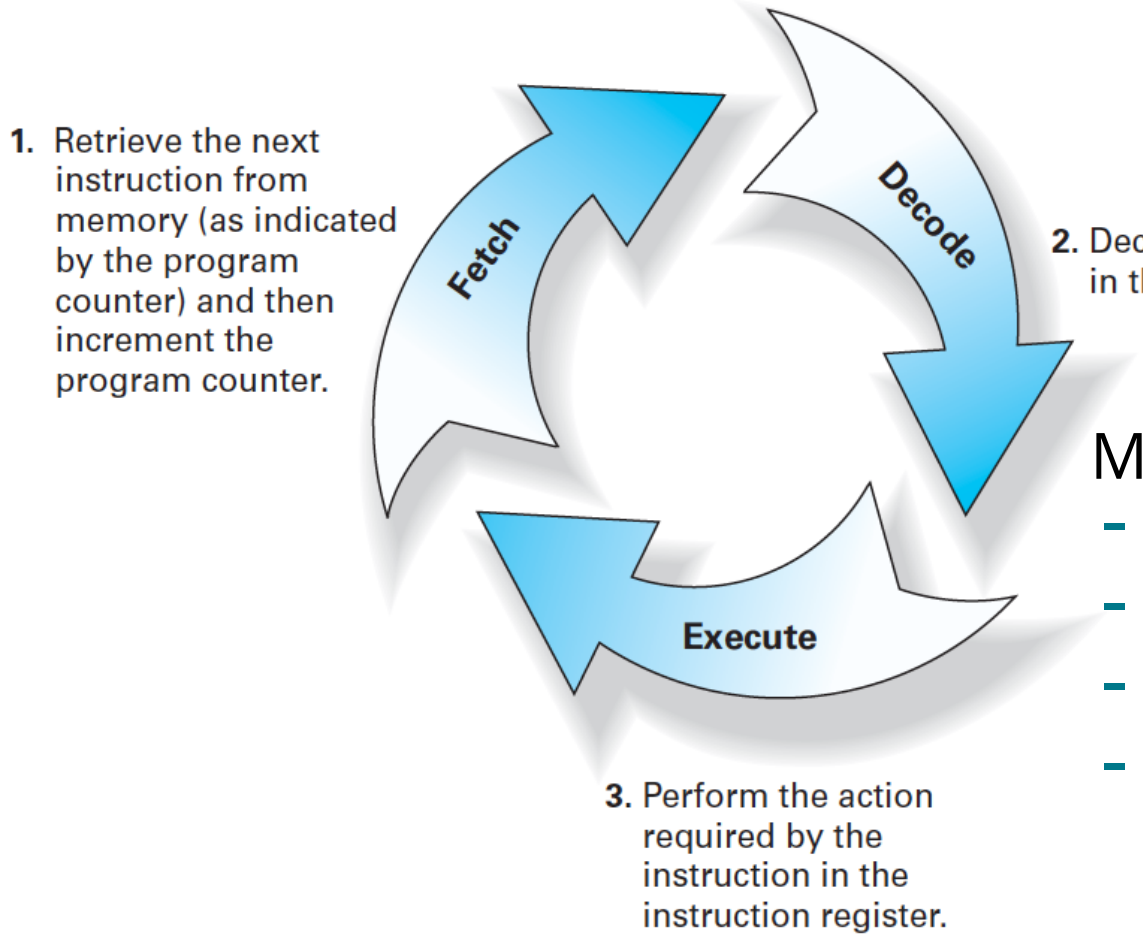
Encoded instructions	Translation
156C	Load register 5 with the bit pattern found in the memory cell at address 6C
166D	Load register 6 with the bit pattern found in the memory cell at address 6D
5056	Add the contents of register 5 and 6 as though they were two's complement representation and leave the result in register 0.
306E	Store the contents of register 0 in the memory cell at address 6E.
C000	Halt.

A program consists of a collection of instructions encoded for a specific purpose.

Program Execution

Controlled by two special-purpose registers

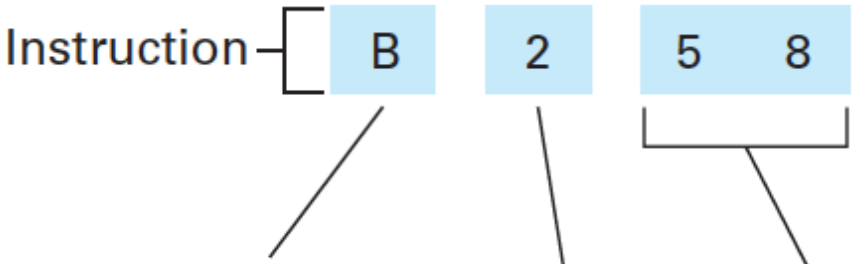
- **Instruction register (IR)**: current instruction
- **Program counter (PC)**: address of next instruction



Machine Cycle

- It is a three-step process
- fetch,
- decode,
- execute

Decoding the Instruction B258

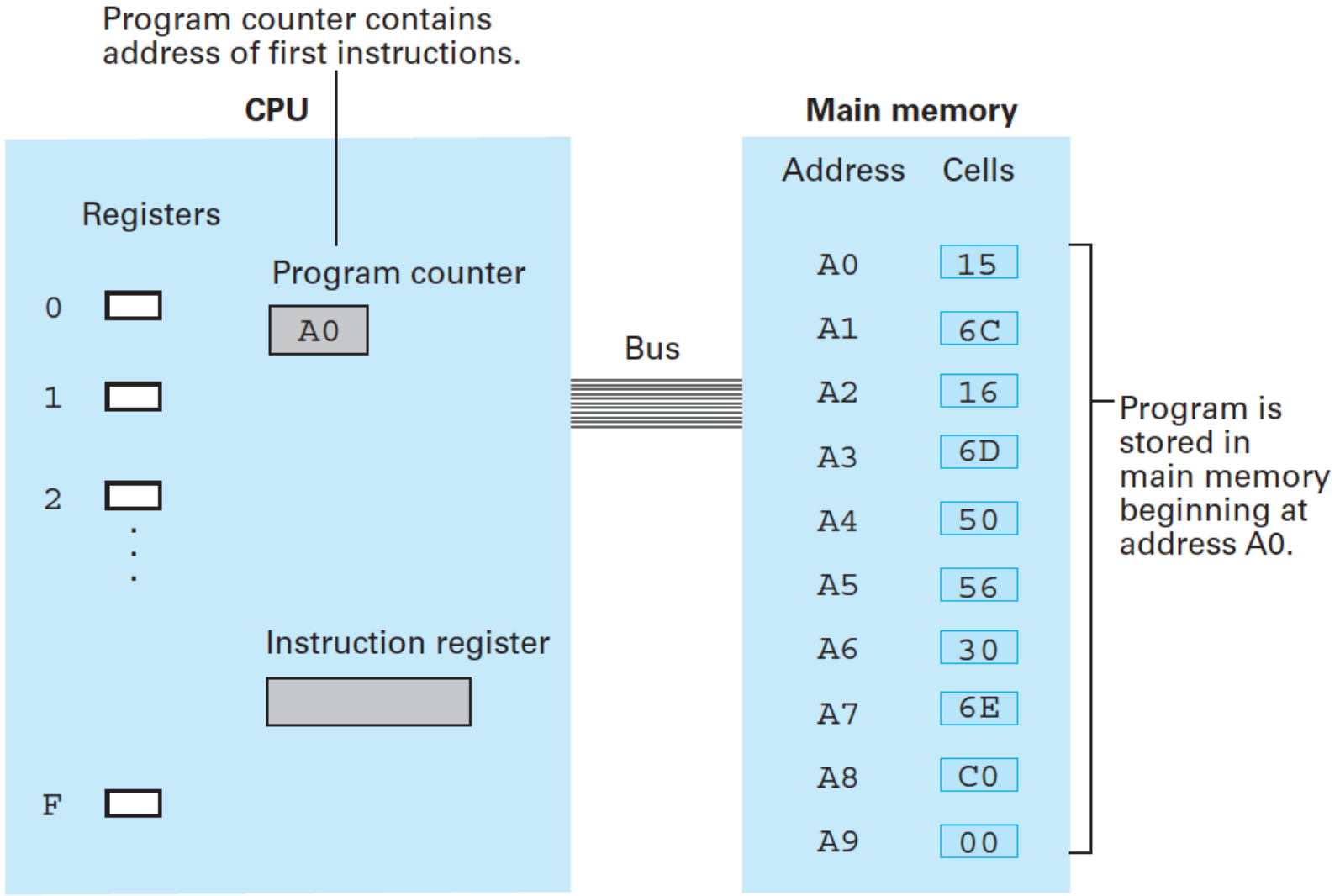


Op-code B means to change the value of the program counter if the contents of the indicated register is the same as that in register 0.

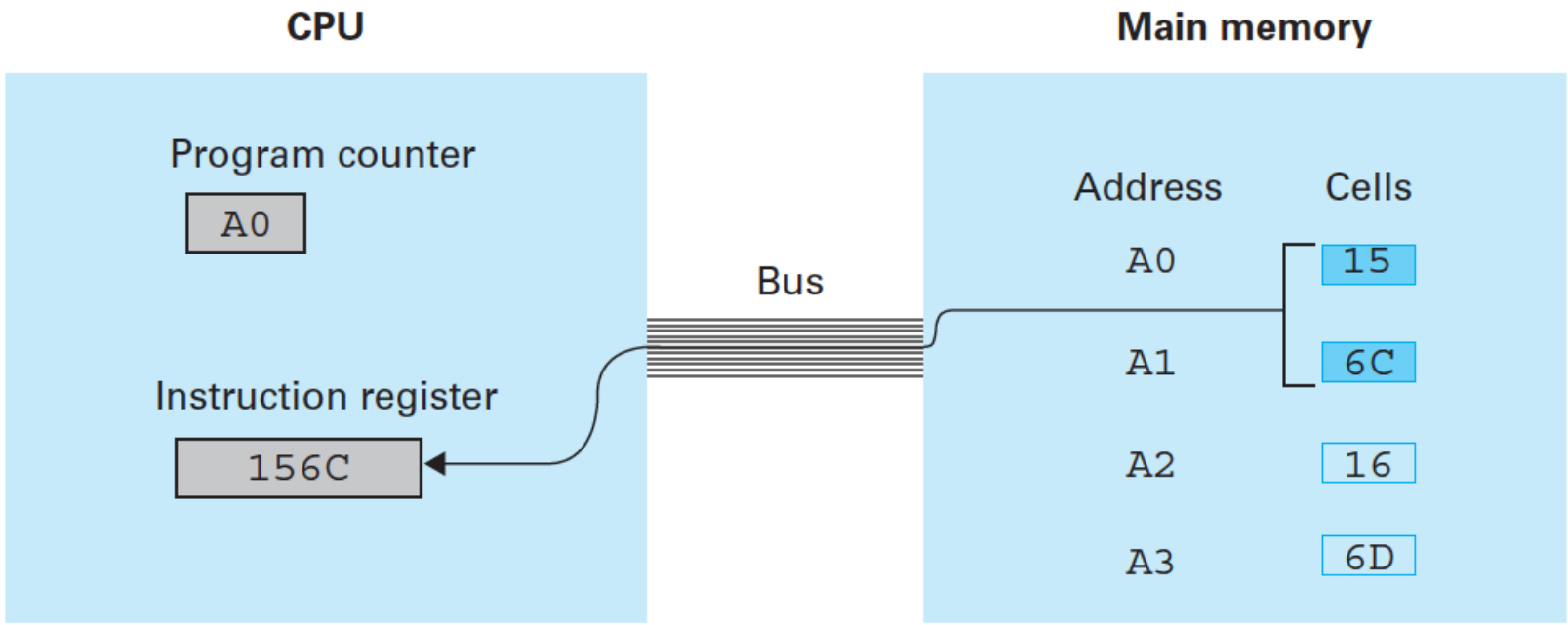
This part of the operand identifies the register to be compared to register 0.

This part of the operand is the address to be placed in the program counter.

The Program is Stored in Main Memory Ready for Execution

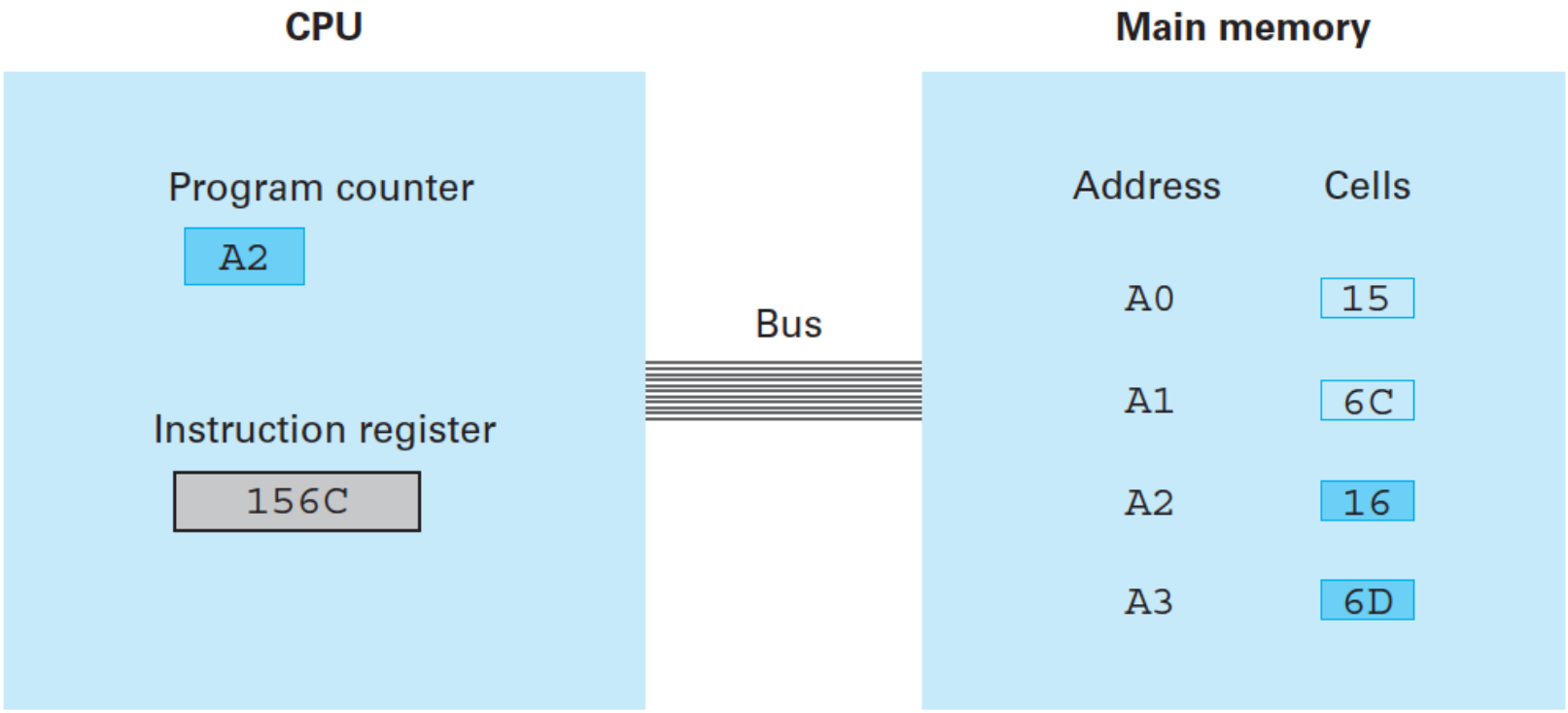


Performing the Fetch Step



a. At the beginning of the fetch step the instruction starting at address A0 is retrieved from memory and placed in the instruction register.

Performing the Fetch Step (cont.)



b. Then the program counter is incremented so that it points to the next instruction.

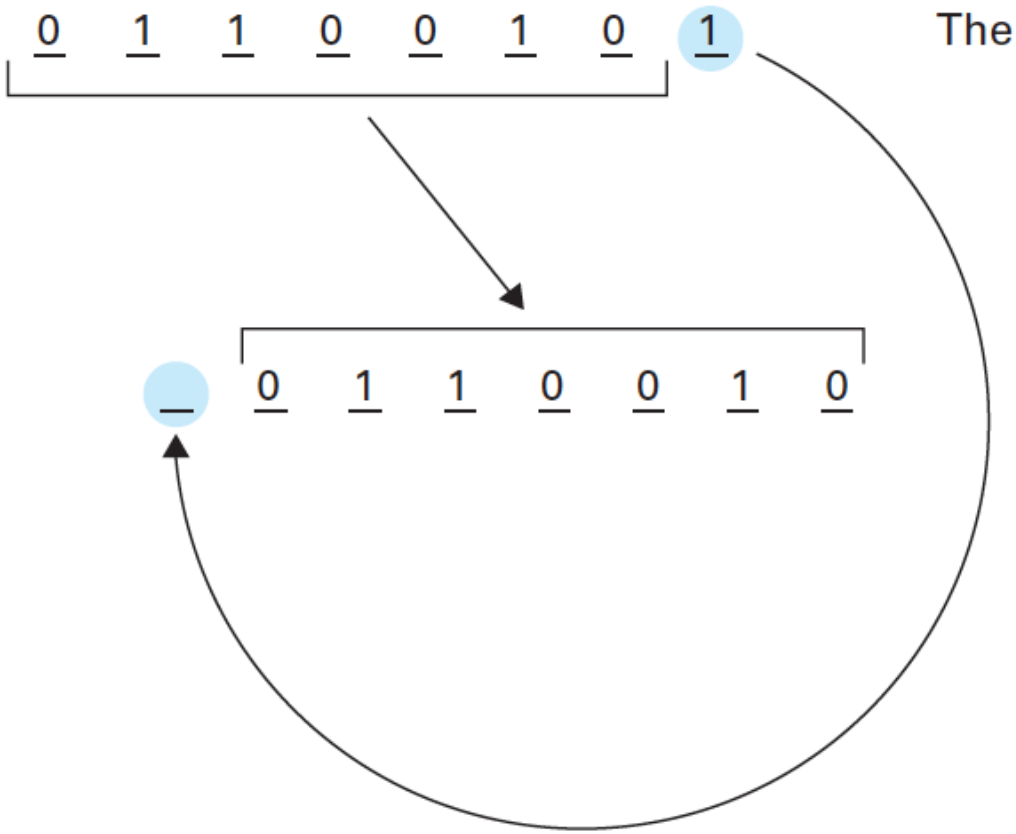
Arithmetic/Logic Operations

Logic operations
- AND, OR, XOR
- Masking

Rotate and Shift
- circular shift, logical shift, arithmetic shift

Arithmetic
- add, subtract, multiply, divide
- Precise action depends on how the values are encoded (two's complement versus floating-point).

Rotating 65 (hex) One Bit to the Right



The original bit pattern

The bits move one position to the right. The rightmost bit "falls off" the end and is placed in the hole at the other end.

1 0 1 1 0 0 1 0

The final bit pattern

Communicating with Other Devices

Peripheral devices

- Mass storage systems, printers, keyboards, mice, display screens, digital cameras, scanner, etc.

Controller (控制器)

- An intermediary apparatus that handles communication between the computer and a device
- Specialized controllers for each type of device
- General purpose controllers (USB and FireWire)

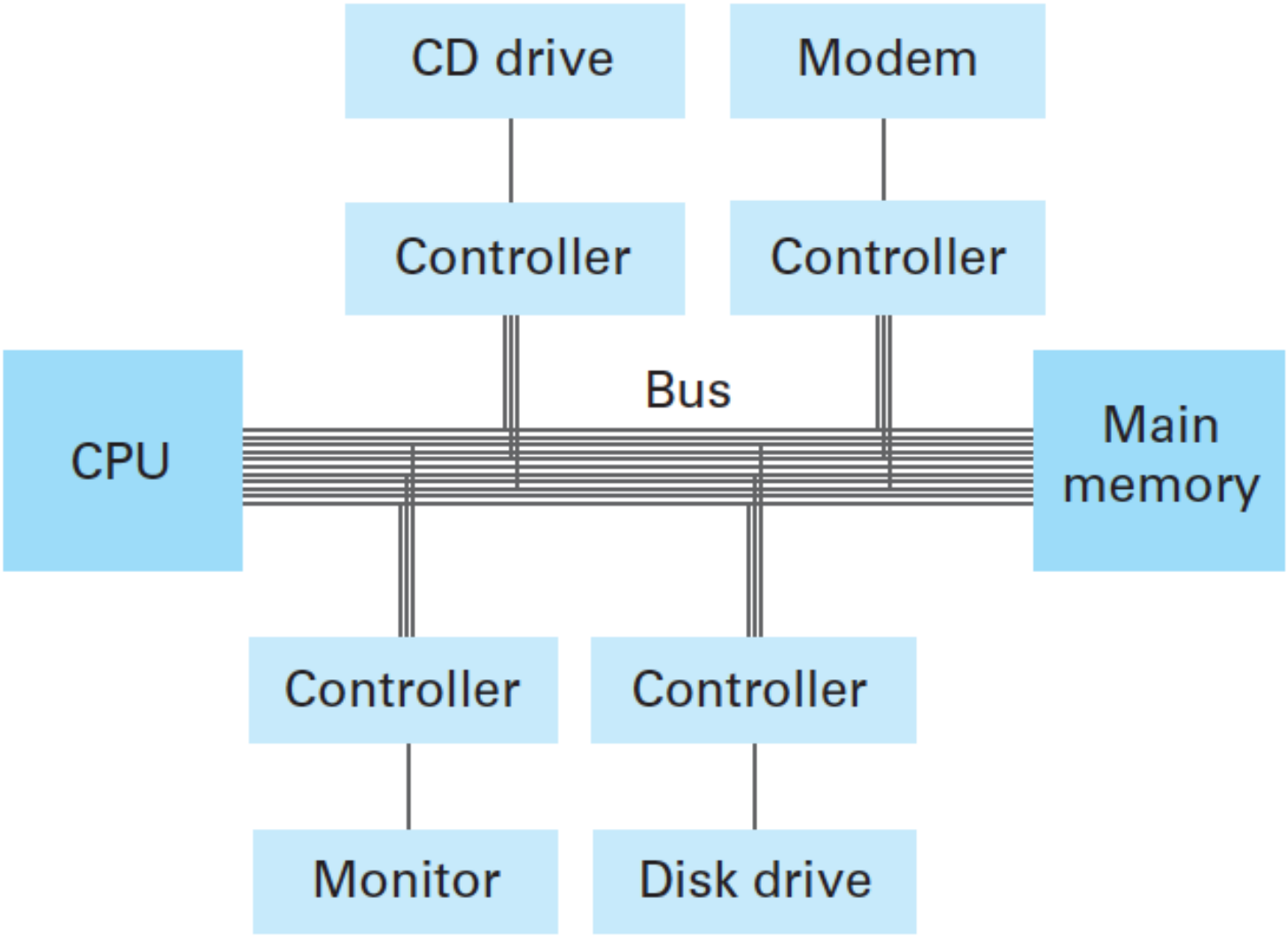
Port (端口)

- The point at which a device connects to a computer
- SATA, memory card slots, etc.

Memory-mapped I/O:

- CPU communicates with peripheral devices as though they were memory cells

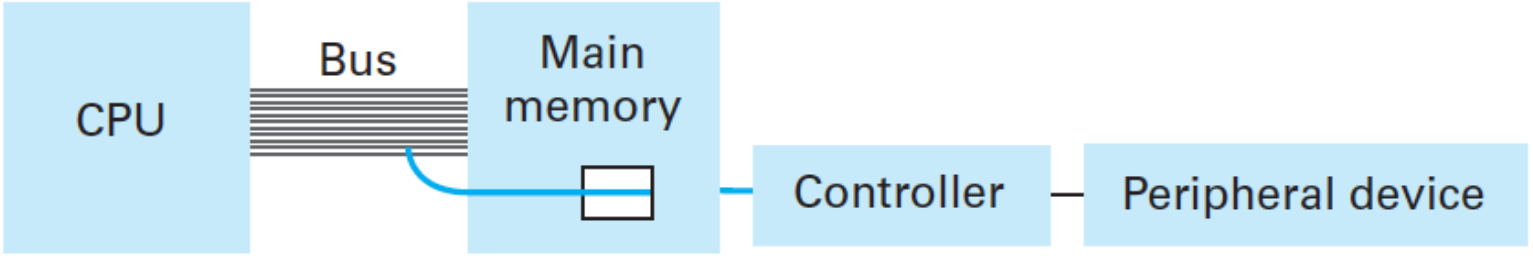
Controllers Attached to The Bus



Memory-mapped I/O (MMIO)

Memory-mapped I/O (MMIO)

- Use the same address space to address both memory and peripheral devices
- The memory and registers of the I/O devices are mapped to main memory addresses.
- Reuse the op-codes for communicating with memory



Port-mapped I/O (PMIO, isolated I/O)

- Main memory and peripheral devices use separate address spaces
- Using special I/O instructions to direct transfers to and from controllers

Communicating with Other Devices (cont.)

Direct memory access (DMA):

- Main memory access by a controller over the bus

von Neumann Bottleneck:

- CPU and controllers competing
- Insufficient bus speed impedes performance

Handshaking:

- The process of coordinating the transfer of data between components

Parallel Communication

- Several communication paths transfer bits simultaneously.

Serial Communication

- Bits are transferred one after the other over a single communication path.

Communication Rates

Measurement units

- bps: Bits per second
- Kbps: Kilo-bps (1,000 bps)
- Mbps: Mega-bps (1,000,000 bps)
- Gbps: Giga-bps (1,000,000,000 bps)

Notation difference

- **B** for byte and **b** for bit

Bandwidth

- Maximum available rate
- It is also used to describe capacity

Other Architectures

Several technologies have been applied to increase throughput

Pipelining

- Overlap steps of the machine cycle

Parallel Processing

- Use multiple processors simultaneously
- SISD: No parallel processing
- MIMD: Different programs, different data
- SIMD: Same program, different data

Information

Course site

- http://abelgo.cn/cs101.html

Office hours

- You should appoint it in advance
- Email: qingsongg@gmail

Course management system

- All course stuff could be found on Piazza