مدينة زويل للعلوم والتكنولوجيا
Zewail City of Science and Technology
COMMUNICATION AND INFORMATION ENGINEERING

CIE 314
Embedded Systems Fundamentals

Lecture #3
Basic microcomputer structure

Instructor:
Dr. Ahmad El-Banna

SPRING 2017

Agenda

- Base Microcomputer Structure
- Microcontrollers Versus Microprocessors
- Central Processing Unit
- System Buses
- Design Tips

3.1 BASE MICROCOMPUTER STRUCTURE

- Base Structure
- Architecture

BASE MICROCOMPUTER STRUCTURE

- Central Processing Unit
- System Memory
  - Program Memory
  - Data Memory
- Input-Output Subsystem
- System Buses
  - Address bus
  - Data bus
  - Control bus
- Power & Support Logic

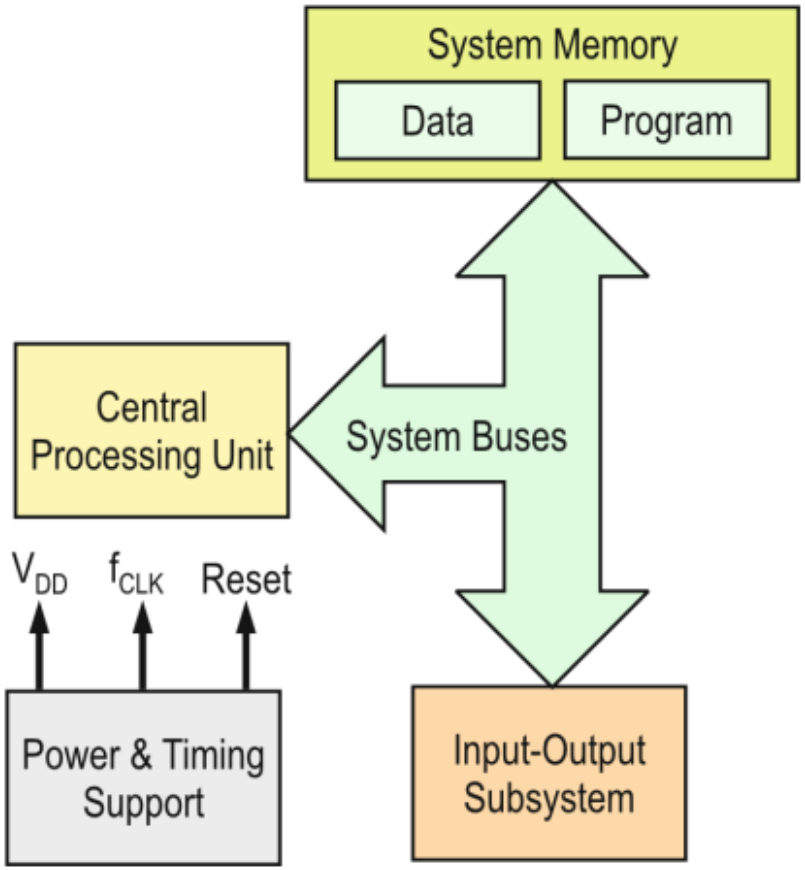


Fig. 3.1 General architecture of a microcomputer system

MINIMAL HARDWARE

- Central Processing Unit (CPU)
  - Fetches, decodes, and executes instructions from memory
- System Memory
  - Program memory
  - Data memory
- Input/Output Interface
  - Connect to the external world
- Buses: Sets of lines grouped according to their function
  - Address bus: Indicate address to be accessed
  - Data bus: carry data or instruction being transferred
  - Control bus: regulate the activity in address & data buses

3.2 MICROCONTROLLERS VERSUS MICROPROCESSORS

- Microprocessor Units
- Microcontroller Units
- RISC Versus CISC Architectures
- Programmer and Hardware Model

MICROPROCESSOR UNITS

- Abbreviated MPUs
- Contain a General Purpose CPU
  - ALU, CU, Registers, & BIU
- Require External Components to form a basic system
  - Buses
  - Memory
  - I/O interfaces & devices
- Additional Characteristics
  - Architecture optimized for accelerating data processing
  - Include elements to accelerate instruction execution

Microprocessors..

- Examples of systems designed around an MPUs
  - Personal computers
  - Mainframes.
- Manufacturers of MPU's include
  - INTEL, Freescale, Zilog, Fujitsu, Siemens, and many others.
  - Intel's initial **4004** in **1971** was built using **10 µm** technology, ran at **400 kHz** and contained **2,250** transistors.
  - Intel's **Xeon E7** MPU, released in **2011**, was built using **32 nm** technology, runs at **2 GHz** and contains **2.6×10^9** transistors.
- MPUs are the most powerful type of processing components available to implement a microcomputer.
- However, most small embedded systems, do not need the large computational and processing power that characterize microprocessors.

MICROCONTROLLER UNITS

- Abbreviated MCUs
- Contain a Microprocessor Core
  - Usually less complex than that of an MPU
- Include memory and peripherals in a single chip
  - Denominated computer-on-a-chip
  - Most MCUs do not provide external buses
- On-chip Memory
  - Includes both program and data memory
- Typical Peripherals
  - Timers
  - I/O ports
  - Data converters

TYPICAL MICROCOMPUTER

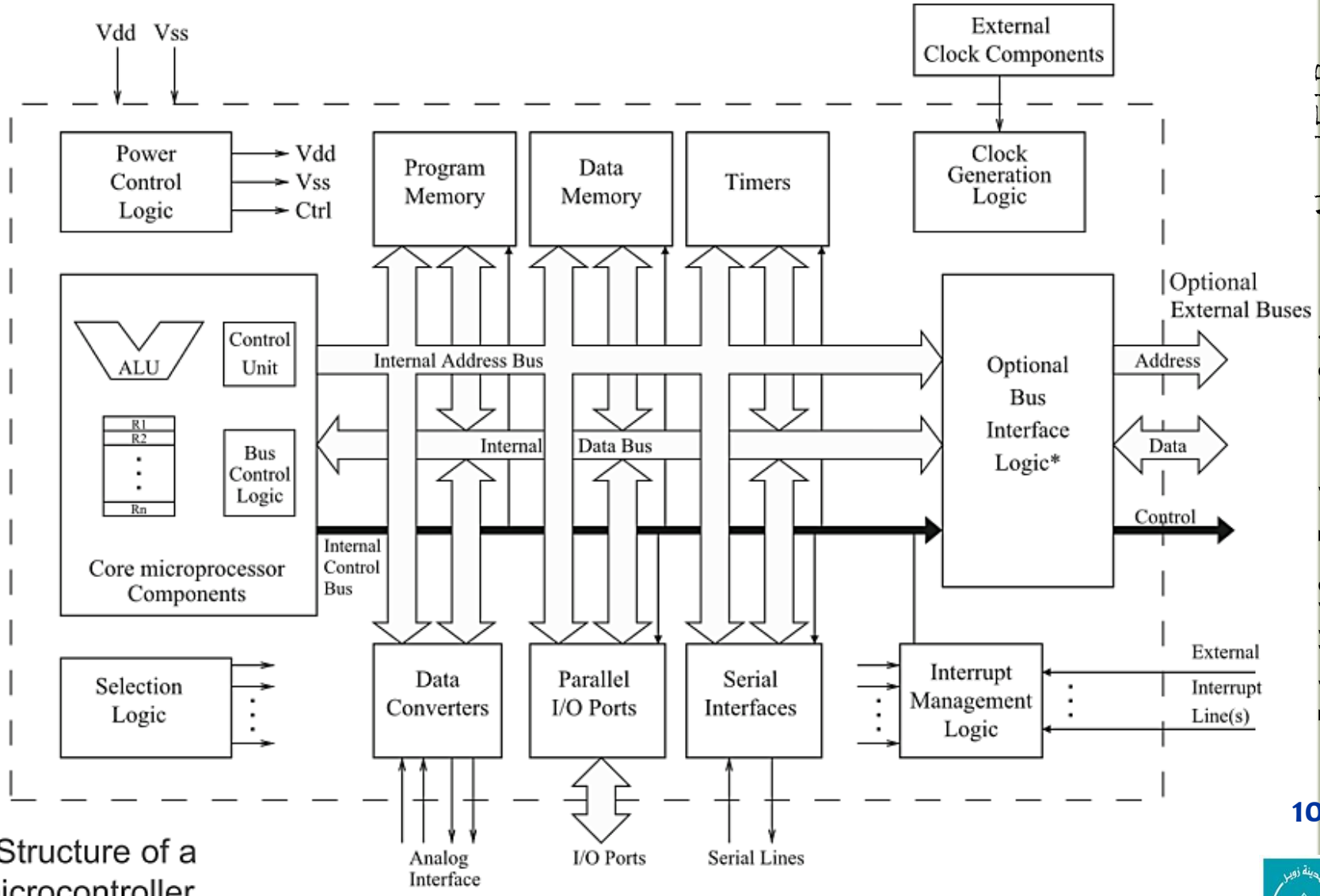


Fig. 3.2 Structure of a typical microcontroller

SOME MCU SERIES

Table 3.1 A sample of MCU families/series

Company	4-bits	8-bits	16-bits	32-bits
EM Microelectronic	EM6807	EM6819		
Samsung	S3P7xx	S3F9xxx	S3FCxx	S3FN23BXZZ
Freescale semiconductor		68HC11	68HC12	
Toshiba		TLCS-870	TLCS-900/L1	TLCS-900/H1
Texas instruments			MSP430	TMS320C28X
			TMS320C24X	Stellaris line
Microchip		PIC1X	PIC2x	PIC32

RISC VS CISC

- CISC (Complex Instruction Set Computer)
  - Variable length instructions
  - Large instruction set
  - Focuses in accomplishing as much as possible with each instruction
    - Helps programmer's tasks
    - Augments hardware complexity
- RISC (Reduced Instruction Set Computer)
  - Fixed length instructions
  - Short (reduced) instruction set
  - Focuses on simple instructions
    - Makes programming harder
    - Simplifies the hardware structure

PROGRAMMER'S AND HW MODEL (1/2)

- General-purpose, High-level Programming
  - Do not require knowledge of the underlying hardware
  - Centered on compiler-level abstraction
- Embedded Systems Programming
  - Need to consider both, hardware and software models
    - Requires awareness of underlying hardware infrastructure
    - Needs to use software programmer's model
  - Might require some assembly-level programming
    - Mostly programmed in C-language
    - Allows mixed assembly- C-language programming
    - Other languages might be available

PROGRAMMER'S AND HW MODEL (2/2)

- Hardware Model
  - Knowledge of hardware characteristics is indispensable
    - Structure, protocols, timing, power, limitations
  - Supports the programmer's model
- Programmer's Model Focus
  - Instruction set
  - Syntax
  - Addressing modes
  - Memory map
  - Transfers and execution time

3.3 CENTRAL PROCESSING UNIT

- Control Unit
- Arithmetic Logic Unit
- Bus Interface Logic
- Registers & Flags

CPU COMPONENTS

- Hardware
  - Control unit (CU)
  - Arithmetic Logic Unit (ALU)
  - Register set
  - Bus interface logic (BIL)
- Software
  - Instruction set
  - Addressing modes

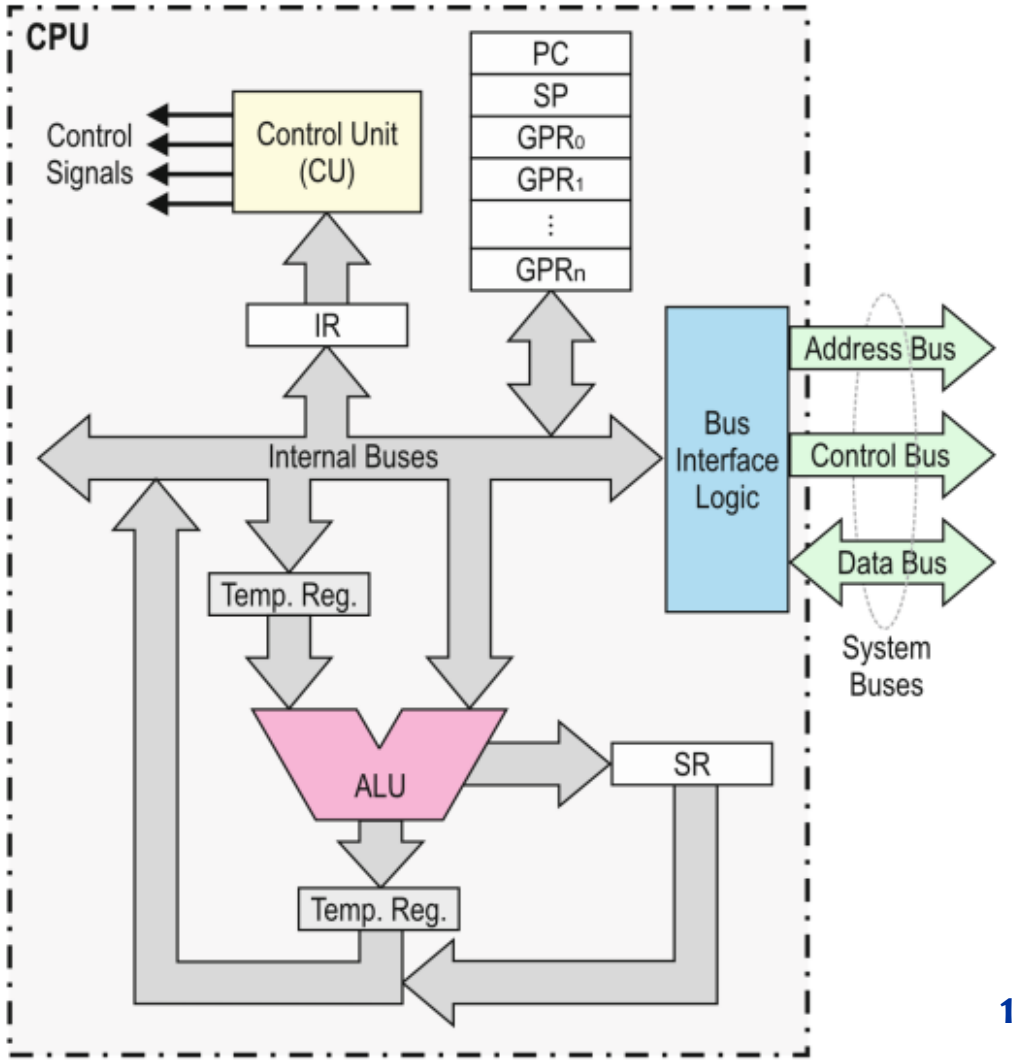


Fig. 3.3 Minimal architectural components in a simple CPU

CONTROL UNIT

- Operation
  - Governs the CPU working like a finite state machine
  - Cycles forever through three states:
    - Fetch
    - Decode, and
    - Execute
  - The fetch-decode-execute cycle is also known as the instruction cycle or CPU cycle

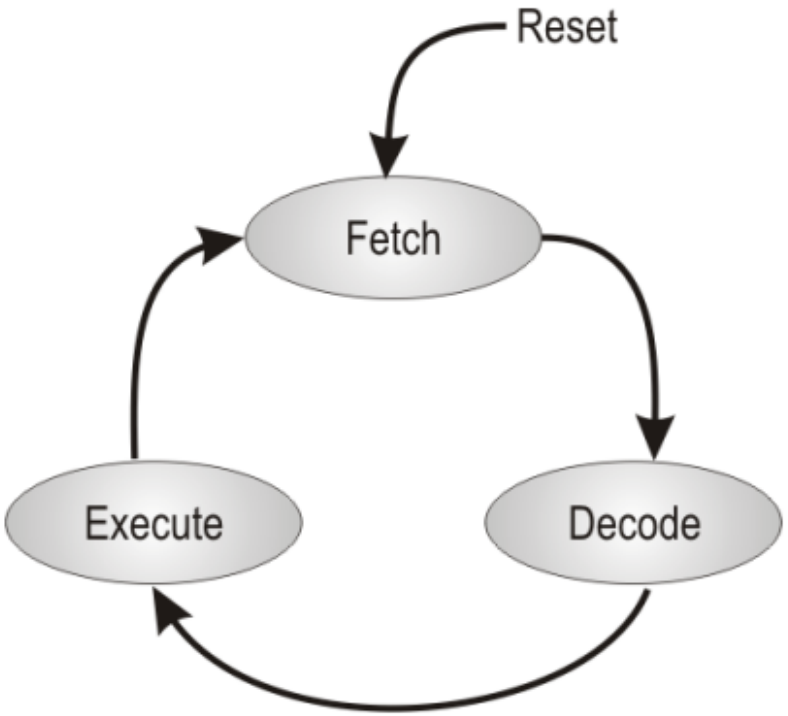


Fig. 3.4 States in control unit operation: fetch, decode, and execute

INSTRUCTION CYCLE

- Fetch
  - The program counter (PC) provides the address of the instruction to be fetched from memory
  - The instruction pointed by the PC is brought from memory into the CPU's instruction register (IR)
- Decode
  - The instruction meaning is deciphered
  - The decoded information is used to send signals to the appropriate CPU components to execute the instruction
- Execute
  - The CU commands the corresponding functional units to perform the actions specified by the instruction
  - At the end of the execution phase, the PC has been incremented to point to the address of the next instruction in memory

ARITHMETIC LOGIC UNIT

- Performs Supported Logic and Arithmetic Operations
  - Logic: AND, OR, NOT, XOR, SHIFT, ROTATE
  - Arithmetic: ADD, SUB, CMP
- Datapath Width
  - Established by the ALU operand width
  - Also sets the width of the data bus and data registers
- Example: A 16-bit CPU
  - 16-bit ALU operands
  - 16-bit Data bus
  - 16-bit Registers

BUS INTERFACE LOGIC (BIL)

- Coordinates the interaction between the internal buses and the system buses, if externally accessible
- Defines how the external address, data, and control buses operate
- Present even in MCUs with no external buses to coordinate internal bus activity
- Transparent to the programmer

REGISTERS

- Provide temporary storage for:
  - Data & operands
  - Memory addresses
  - Control words
- Fastest form of storage
- Smallest Capacity
- Volatile Contents
  - Contents lost when CPU is de-energized
- Register Types
  - General Purpose
  - Special Purpose

GENERAL PURPOSE REGISTERS

- Are not tied to specific functions
  - Are available for programmer's general usage
- Can hold data, variables, or addresses
  - Usage depend on addressing mode and programmer's designation
- Number of registers depend on CPU architecture
  - Accumulator architectures have only a few
    - Some as little as two GP registers
  - RISC CPUs use a register file with dozens of registers

SPECIAL PURPOSE REGISTERS

- Instruction Register (IR)

  Holds the instruction being currently decoded and executed

- Program Counter (PC)
  - Holds the address of the next instruction to be fetched from memory
- Stack Pointer (SP)
  - Holds the address of the current top-of-stack (TOS)
- Status Register (SR)
  - Holds the current CPU status
  - Status is indicated by a set of *flags*
  - A Flag: an individual bit indicating some condition

COMMON STATUS FLAGS

- Zero Flag (ZF)
  - Set when the result of an ALU operation is zero, and cleared otherwise
- *Carry Flag (CF)*
  - Set when an ALU arithmetic operation produces a carry
- *Negative or sign flag (NF)*
  - Set if the result of an ALU operation is negative and cleared otherwise
- *Overflow Flag (VF)*
  - This flag signals overflow in ALU addition or subtraction operations with signed numbers
- *Interrupt Flag (IF)*
  - Also called the *Global Interrupt Enable* (GIE)
  - Is not associated to the ALU status
  - Indicates whether or not the CPU would accept interrupts

MSP430 STATUS REGISTER

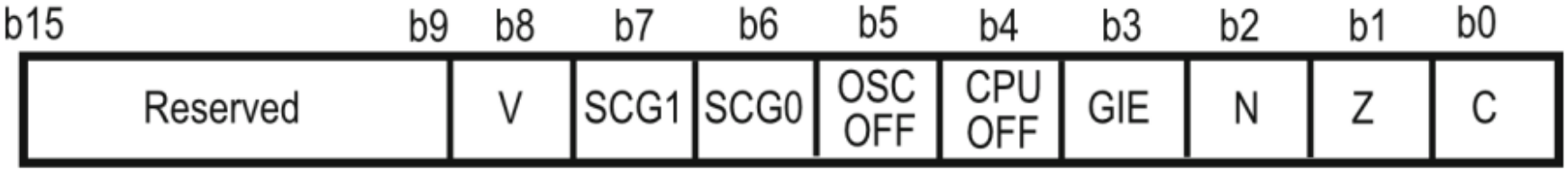


Fig. 3.5 MSP430 status register

UNDERSTANDING FLAG OPERATIONS

Example 3.1 *The following operations are additions performed by the ALU using 8-bit data. For each one, determine the Carry, Zero, Negative, and Overflow flags.*

01001010 +	10110100 +	10011010 +	11001010 +
01111001 =	01001100 =	10111001 =	00011011 =
0 11000011	1 00000000	1 01010011	0 11100101
↑ ↑	↑ ↑	↑ ↑	↑ ↑
C N	C N	C N	C N

Solution: *The operands have eight bits, so this length is our reference for the flags when we look at the result. The most significant bit in this group is flag* N. *The bit to the left is* C. *In hex form, these additions are, respectively,* 4Ah + 79h = C3h*;* B4h + 4Ch = 100h*;* 9Ah + B9h = 153h*; and* CAh + 1Bh = E5h. *The zero flag is set if the result is 0, discarding the carry, and the overflow flag is set if the addition of numbers of the same sign (that is, with equal most significant bit) yield a result of different sign (signaled by N). With this information we have then:*

Operation 4Ah + 79h = C3h : C = 0, N = 1, Z = 0 *and* V = 1.
Operation B4h + 4Ch = 100h : C = 1, N = 0, Z = 1 *and* V = 0.
Operation 9Ah + B9h = 153h : C = 1, N = 0, Z = 0 *and* V = 1.
Operation CAh + 1Bh = E5h : C = 0, N = 1, Z = 0 *and* V = 0.

3.4 SYSTEM BUSES

- Data Bus
- Address Bus
- Control Bus

SYSTEM BUSES

- Data bus
  - Set of lines carrying data and instructions
  - Data bus lines are bidirectional to allow for reads & writes
  - READ: A transfer into a CPU register from memory or I/O
  - WRITE: A transfer or from a CPU register to memory or I/O
- Address bus
  - Set of lines transporting the address information which uniquely identifies a data cell in memory or peripheral device
- Control bus
  - Set of lines carrying the signals that regulate the system activity

DESIGN TIPS

Fig. 6.1 Components in a basic CPU interface: power source, clock generator, power-on-reset, and boot sequence

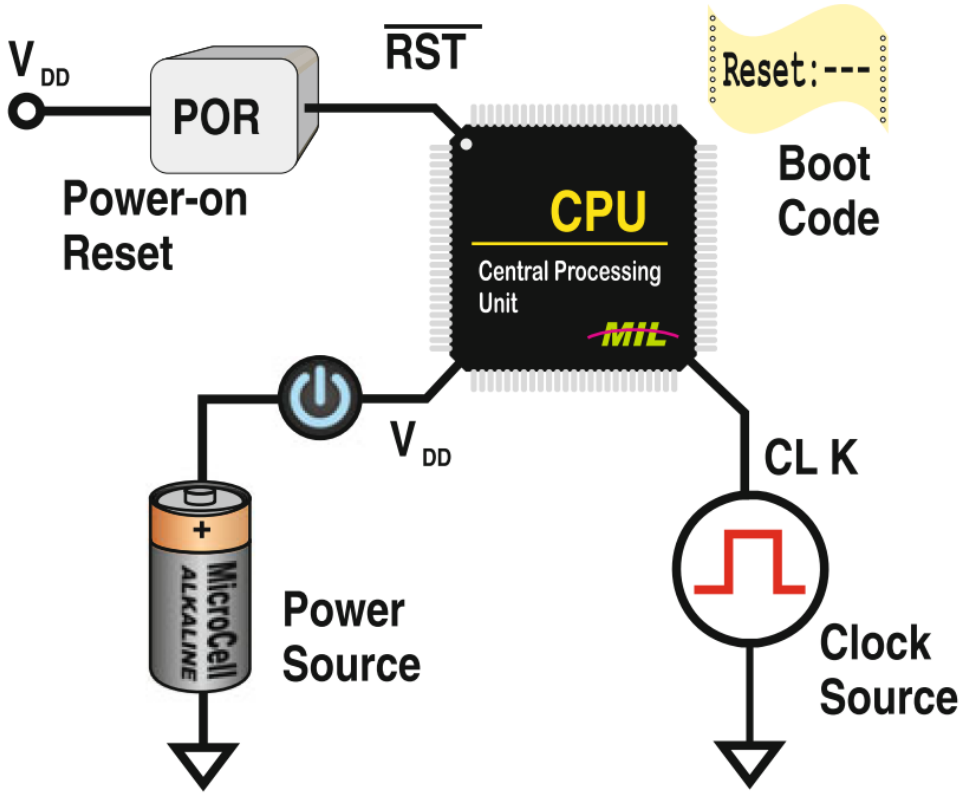
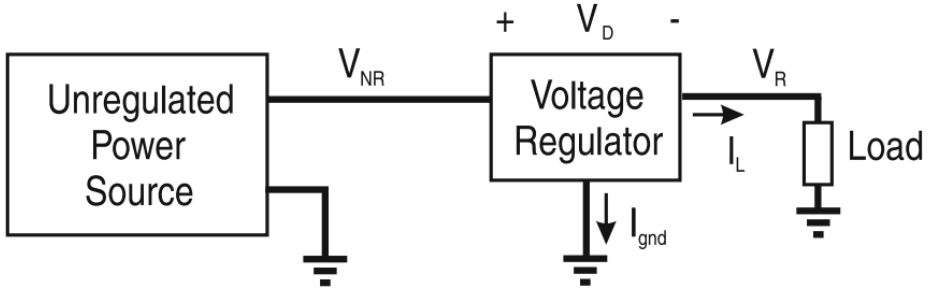


Fig. 6.4 Non-regulated source feeding a load through a 3-terminal linear voltage regulator



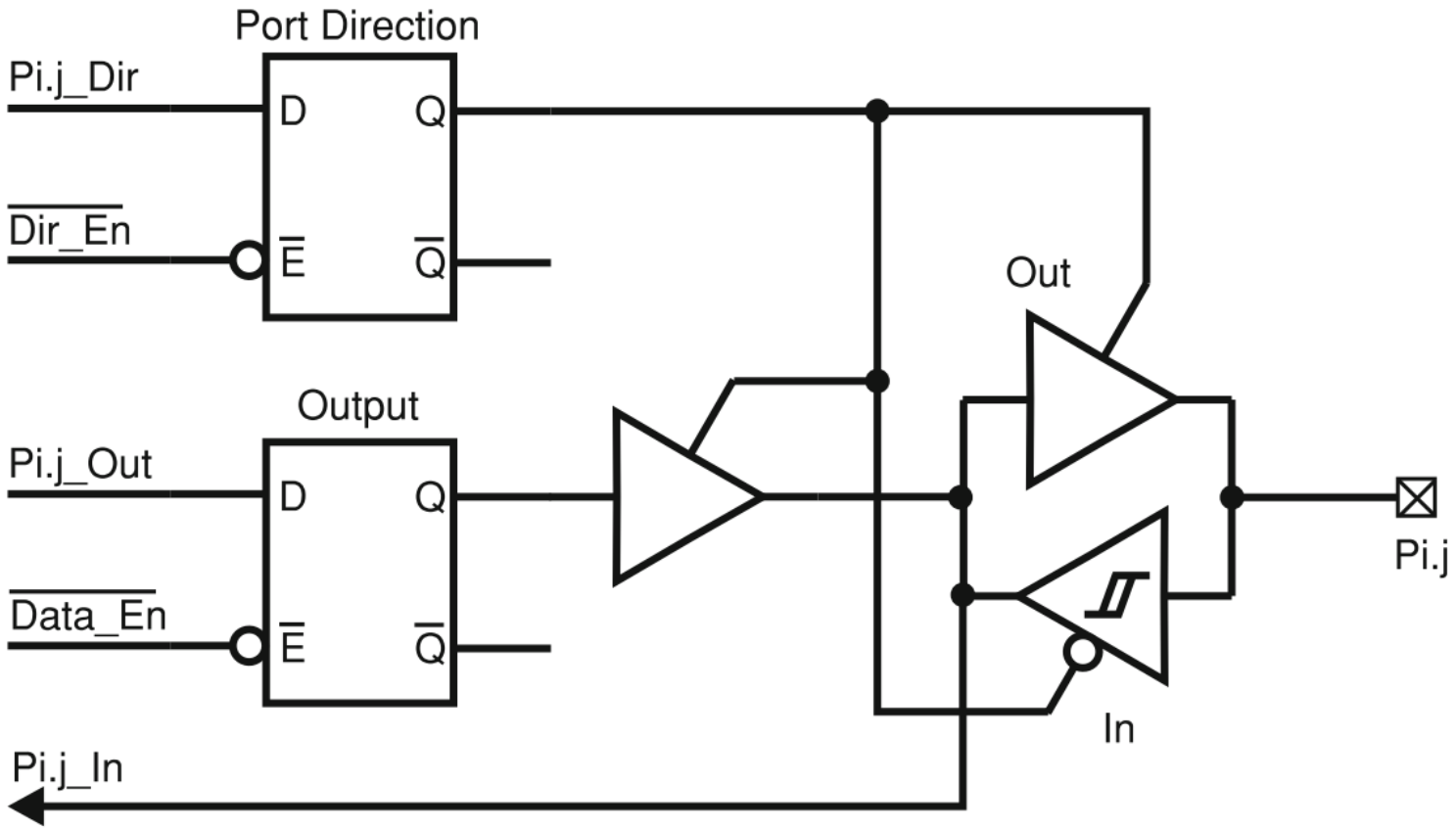


Fig. 8.2 Basic structure of an Input/Output pin driver

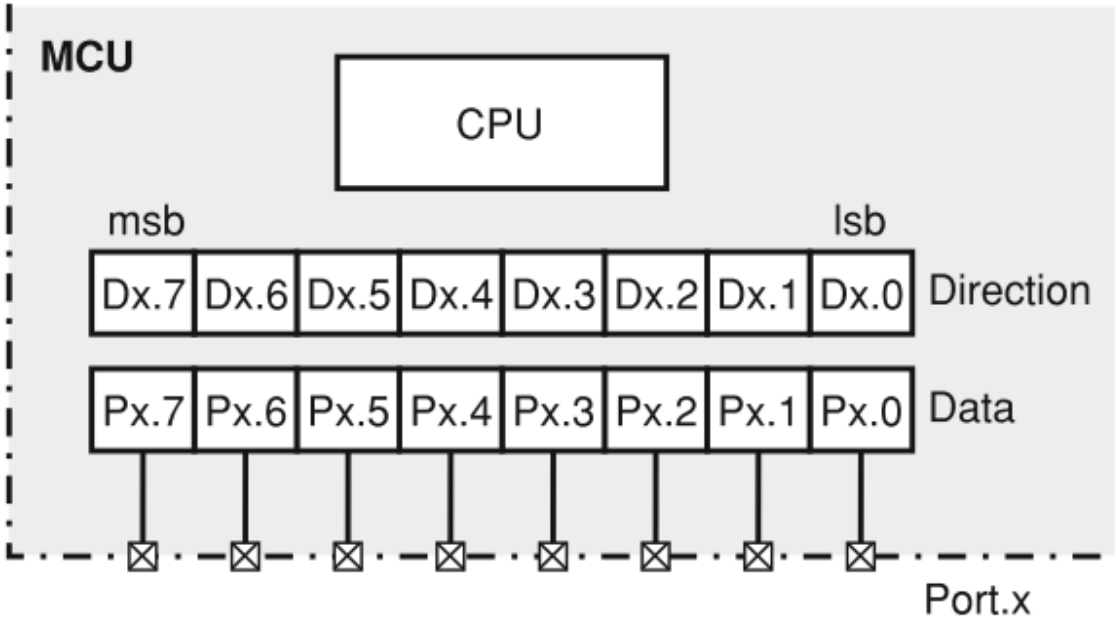


Fig. 8.4 Programmer's model for a simple GPIO port containing only data and direction registers

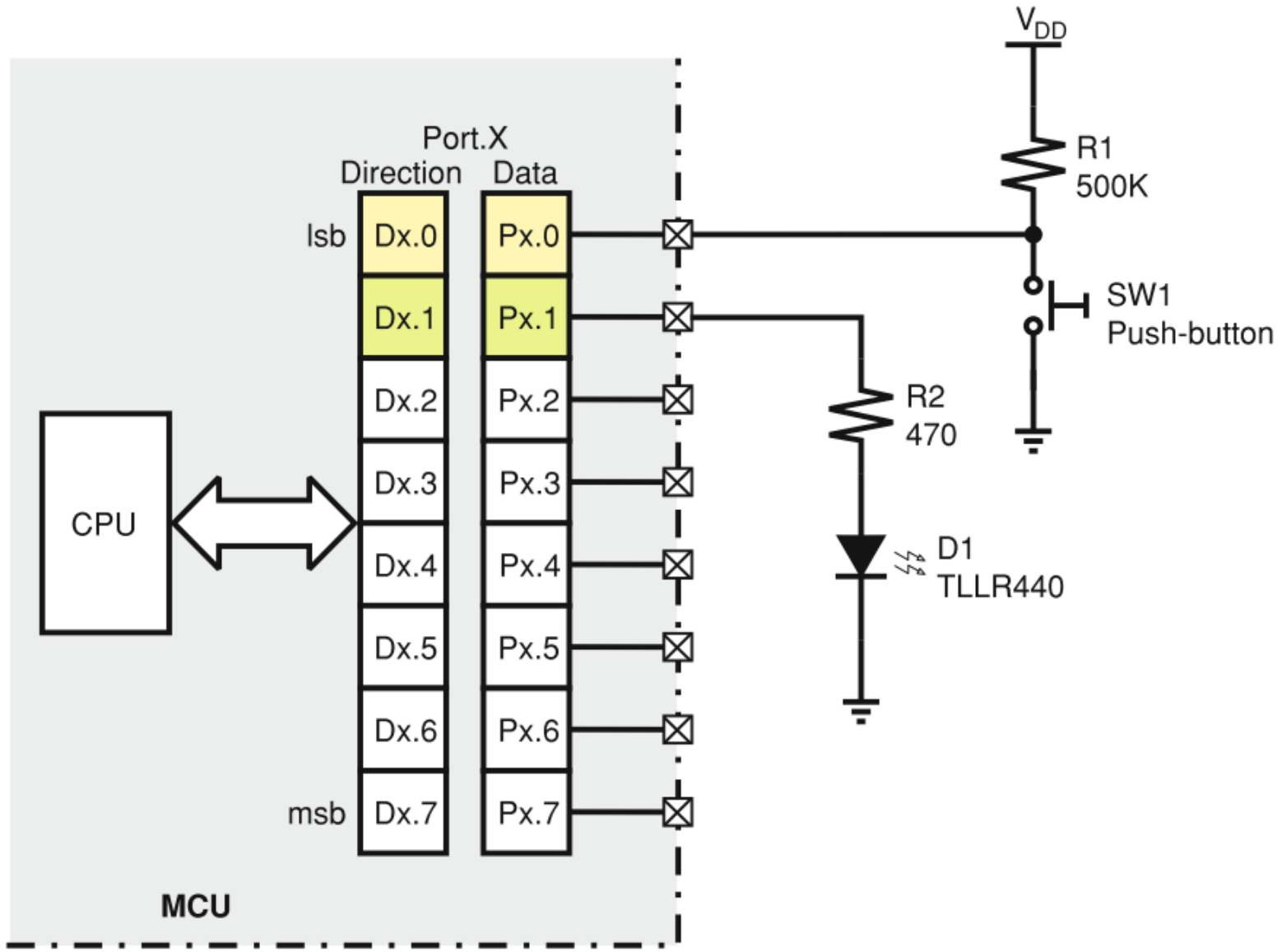
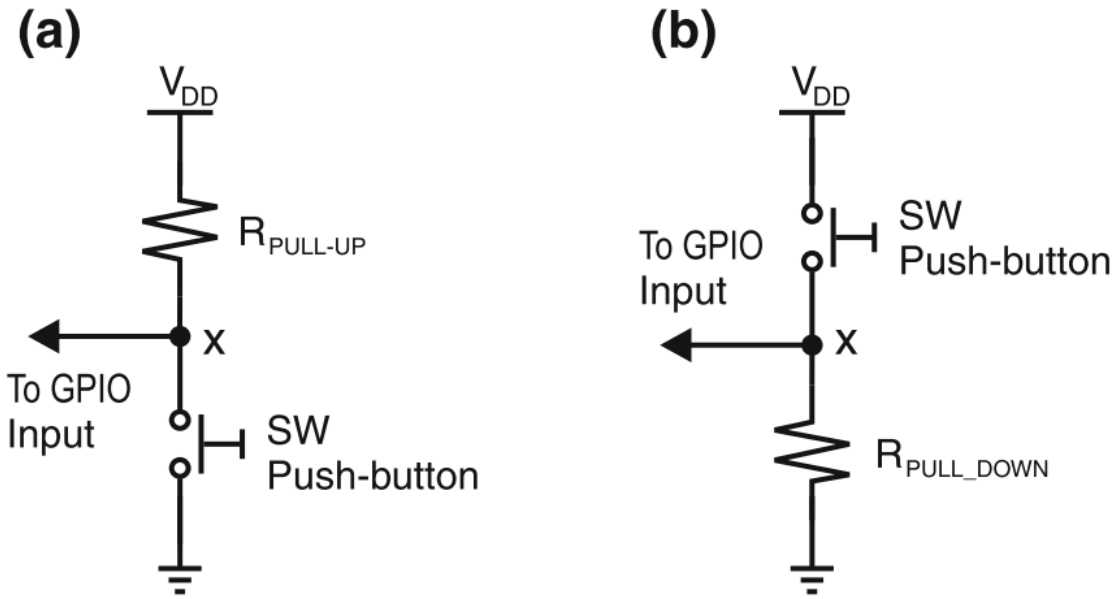


Fig. 8.5 Hardware setup for Example 8.1

Fig. 8.19 Common interfaces for momentary push-buttons. **a** MPB with pull-up resistor, **b** MPB with pull-down resistor



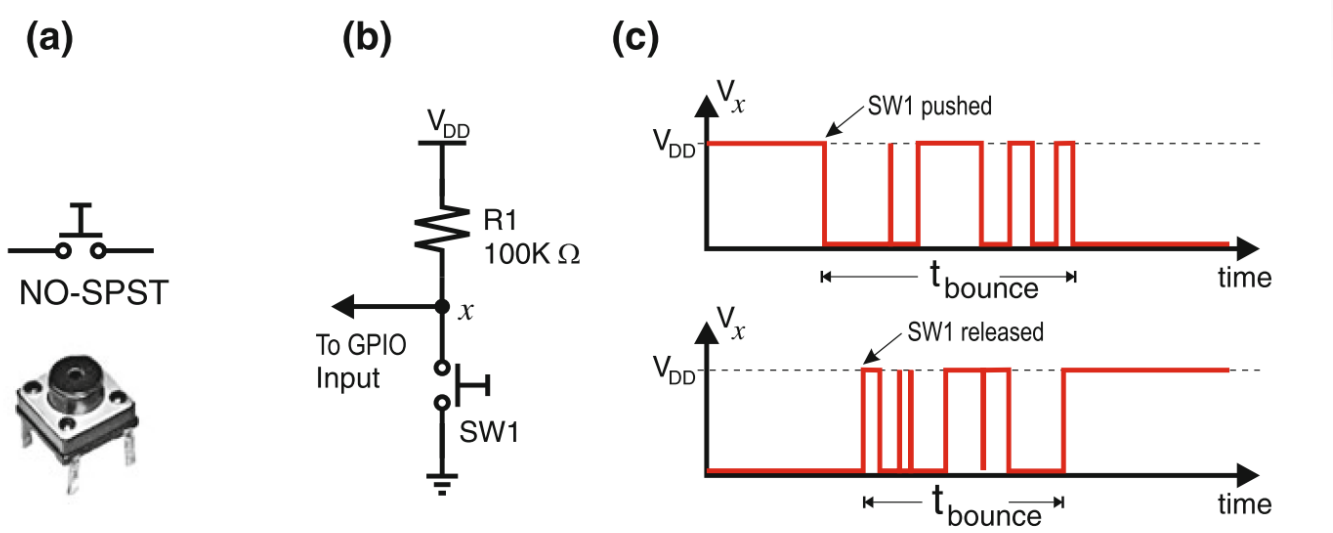
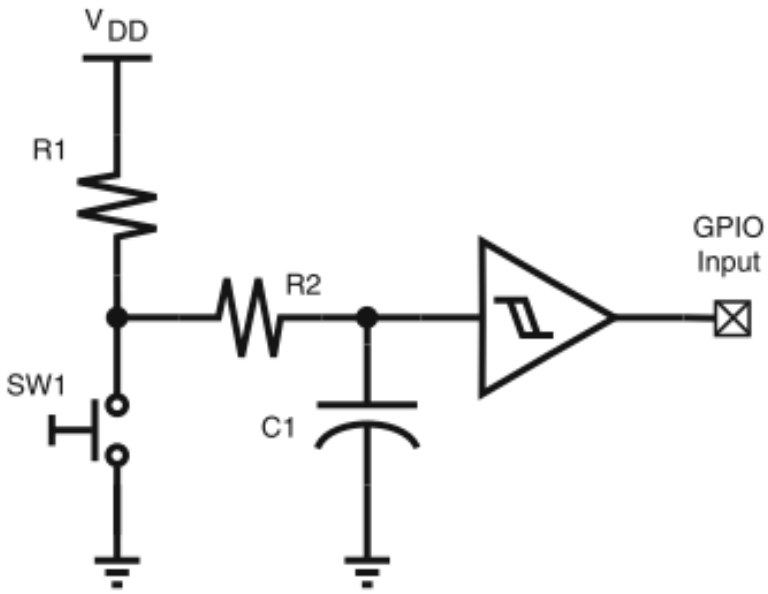


Fig. 8.20 Bouncy behavior of mechanical NO-MPB. **a** Actual Switch, **b** Interface Circuit, **c** Output waveform. The *top* and *bottom* waveforms correspond to the switch closing and opening, respectively

Fig. 8.22 RC network-based switch debouncing circuit



- For more details, refer to:
  - Chapter 3,6,8 at **Introduction to Embedded Systems,** Springer 2014 by Manuel Jiménez et al.

- The lecture is available online at:
  - *http://bu.edu.eg/staff/ahmad.elbanna-courses*

- For inquires, send to:
  - ahmad.elbanna@feng.bu.edu.eg