Introduction to Microcontrollers

中山 大 学

数据科学与计算机学院

郭雪梅

Tel:39943108

Review

❿ Definitions (define in 16 words or less, choose the word, or multiple choice)

- volatile, nonvolatile, RAM, ROM, port
- basis, nibble, precision, kibibyte, mebibyte
- signed/unsigned, 8-bit, 16-bit, 32-bit
- bus, ~~address bus~~, ~~data bus~~
- memory-mapped, I/O mapped
- Harvard architecture, von Neumann
- ALU, D flip-flop, registers
- device driver, CISC, RISC
- friendly, mask, toggle, heartbeat, breakpoint
- Negative logic, positive logic, open collector
- Voltage, current, power, Ohm's Law

Review

- Number conversions - *convert one format to another*
  - alternatives, binary bits
  - signed decimal e.g., -56
  - unsigned decimal e.g., 200
  - binary e.g., 11001000_2
  - hexadecimal e.g., 0xC8
- Addressing modes *(book Sec 3.3.2)*
  - Immediate e.g., MOV R0,#0,
  - Indexed e.g., LDR R0,[R1] LDR R0,=123
  - PC-relative e.g., BL subroutine
  - Register list, e.g., PUSH {R1, R4-R6}

Review

根据两个数字的运算，
确定NZVC

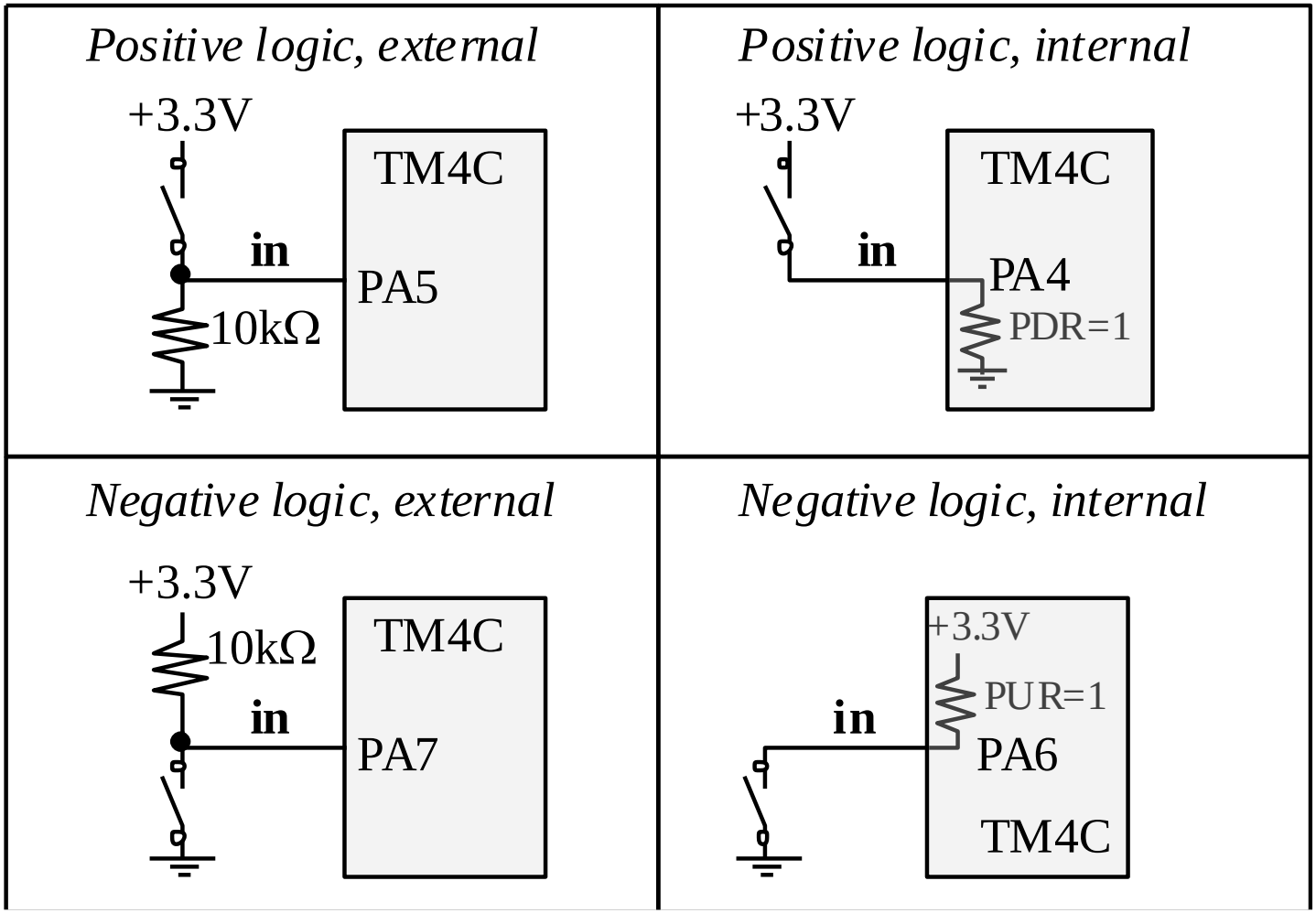
- Cortex-M4 operation & instructions
  - Definition of N,Z,V,C
    - What do they mean? How do we use them?
  - Thumb-2 instructions on reference sheet
  - Components in address space
  - Subroutine linkage
  - Stack operations
- Switch and LED interfaces

计算题：
根据已知条件计算，
使用欧姆定律进行计算；

Review

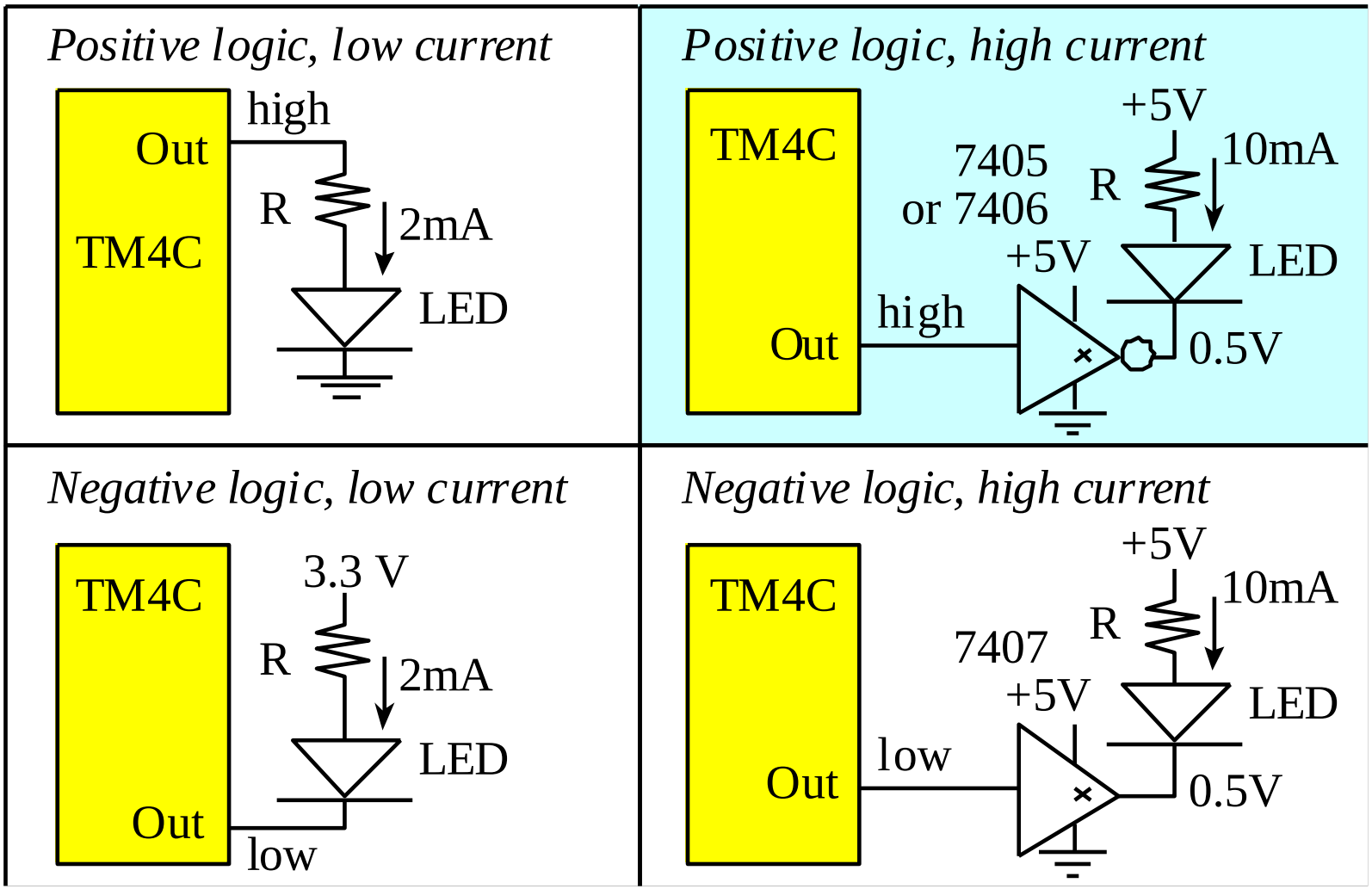
- Simple programs (assembly and C)
  - specify an I/O pin is an input
  - specify an I/O pin is an output
  - clear an I/O output pin to zero
  - set an I/O output pin to one
  - toggle an I/O output pin
  - check if an I/O input pin is high or low
  - add, sub, shift left, shift right, and, or, eor
  - subroutine linkage
-

Switch Interface



Know voltage, current, power

LED interfaces



Know voltage, current, power, Ohm's Law

- **Number conversions - *convert one format to another***
  - **decimal digits**
  - **signed decimal e.g., -56**
  - **unsigned decimal e.g., 200**
  - **binary e.g., %11001000**
  - **hexadecimal e.g., 0xC8**

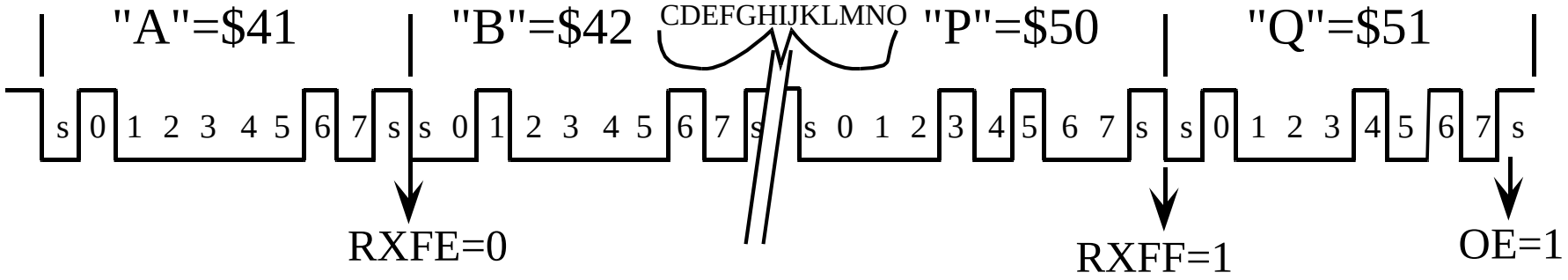
Final Exam Review

- Instruction detail and Cortex-M operation
  - 8-bit addition, subtraction yielding result, N, Z, V, C
    - No N Z V C this semester
- Simple programs
  - specify an I/O pin is an input
  - specify an I/O pin is an output
  - clear an I/O output pin to zero
  - set an I/O output pin to one
  - toggle an I/O output pin
  - check if an I/O input pin is high or low
  - add, sub, shift left, shift right, and, or, eor
  - subroutine linkage

Final Exam Review

- Switch & LED interfacing
- GPIO Ports
  - *friendly* programming practices
  - LED and switch interfacing
  - bit-specific addressing (no bit-specific addressing this year)
- SysTick Timer
  - initialization
  - operational parameters
    - period
  - busy-wait delay or periodic interrupt
- UART
  - Operation, programming, start bit, stop bit, rates
- Real time and communication systems
  - throughput ≡ bandwidth

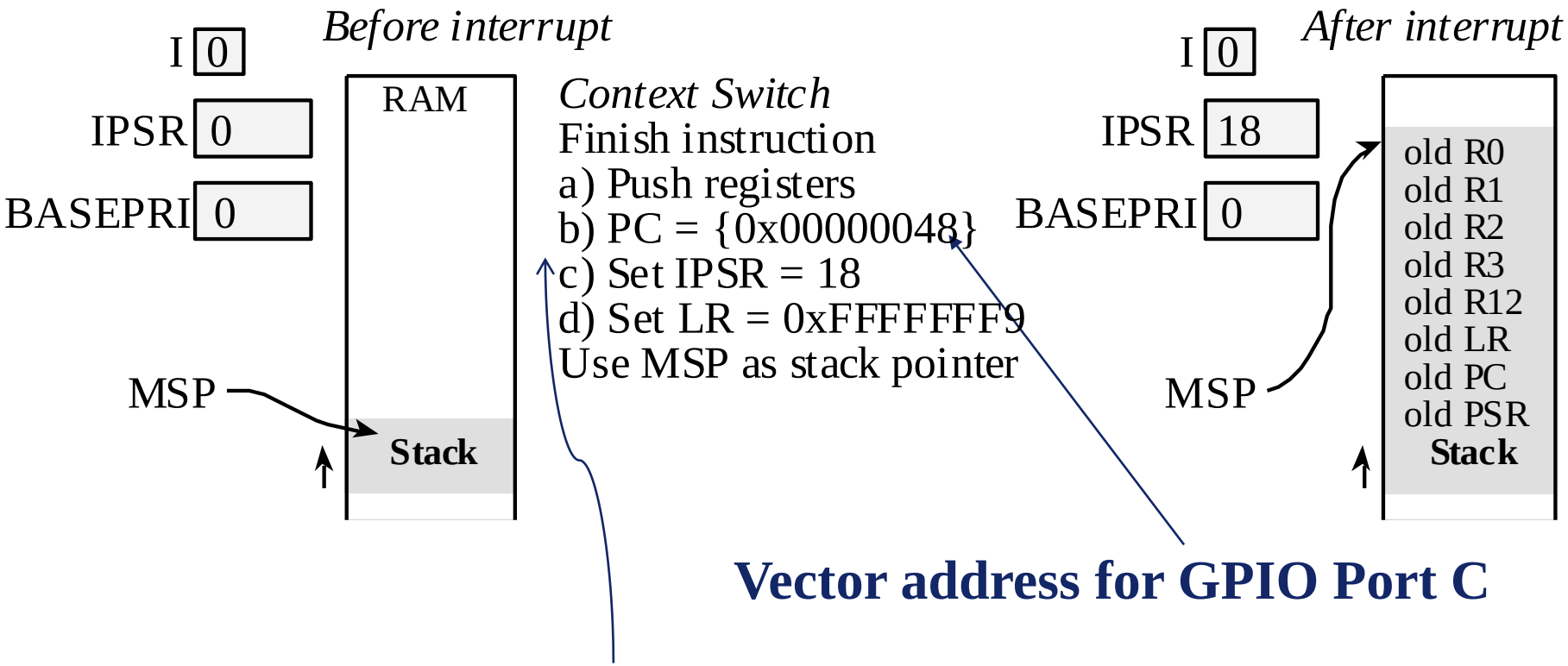
- **Baud rate** = **Baud16/16** = (Bus clock frequency)/(16*__divider__)
  - **Baud rate is total number of bits per unit time**
    - **Baudrate = 1 / bit-time**
  - **Bandwidth is data per unit time**
    - **Bandwidth = (data-bits / frame-bits) * baudrate**
- baud rate is 19200 bits/sec, then the **divider** should be 8,000,000/16/19200 or 26.04167,



Interrupts

- An **interrupt** is the automatic transfer of software execution in response to a hardware **event** that is **asynchronous** with the current software execution
- This hardware event is called **a trigger** and it breaks the execution flow of the **main thread** of the program
- The event causes the CPU to stop executing the current program and begin executing a special piece of code called an **interrupt handler** or **interrupt service routine (ISR)**
- Typically, the ISR does some work and then resumes the interrupted program

Interrupt Context Switch



To **return from an interrupt**, the ISR executes the typical function return **BX LR** . However, since the top 24 bits of **LR** are 0xFFFFFF, it knows to return from interrupt by popping the eight registers off the stack.

```
volatile uint32_t Counts;
#define PD0 (*((volatile uint32_t *)0x40007004))
void SysTick_Init(uint32_t period){
SYSCTL_RCGCGPIO_R |= 0x08; // activate port D
Counts = 0;
GPIO_PORTD_AMSEL_R &= ~0x01; // no analog
GPIO_PORTD_PCTL_R &= ~0x0000000F; // regular GPIO function
GPIO_PORTD_DIR_R |= 0x01; // make PD0 out
GPIO_PORTD_AFSEL_R &= ~0x01; // disable alt funct on PD0
GPIO_PORTD_DEN_R |= 0x01; // enable digital I/O on PD0
NVIC_ST_CTRL_R = 0; // disable SysTick during setup
NVIC_ST_RELOAD_R = period - 1;// reload value
NVIC_ST_CURRENT_R = 0; // any write to current clears it
NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0x40000000;
//priority 2
NVIC_ST_CTRL_R = 0x00000007; // enable with core clock and interrupts
EnableInterrupts();
}
void SysTick_Handler(void){
PD0 ^= 0x01; // toggle PD0
Counts = Counts + 1;
}
```

Program 9.7. Implementation of a periodic interrupt using SysTick (PeriodicSysTickInts_xxx.zip).