COMPS462
Embedded Systems

Lecture 1: Introduction, Architecture, Embedded Systems, ARM Programming, Introduction to I/O

These slides reproduced from University of Texas, Embedded System Course with some minor changes.

Agenda

- Course Description
  - Book, Labs, Equipment
  - Grading Criteria
  - Expectations/Responsibilities
  - Prerequisites
- Embedded Systems
  - Microcontrollers
  - Input/Output
- ARM Architecture
  - Programming
  - Integrated Development Environment (IDE)

Useful Info

- Office hours: Fridays 10am-12pm

Assignments	20%
Labs	25%
Midterm exam	25%
Final exam	30%

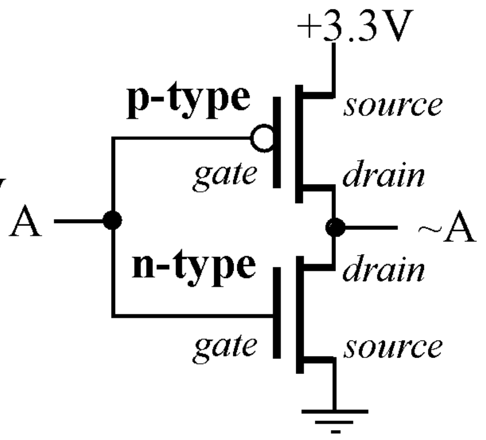
- Read the book and lab manual!
- Check Canvas often

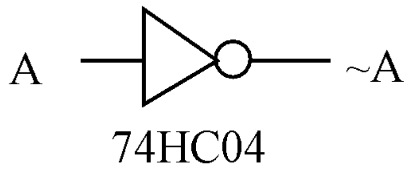
Action Items

- Come introduce yourselves
- Order board, LCD
- Install Keil 5 software IDR
- Read Chapter 1 of book

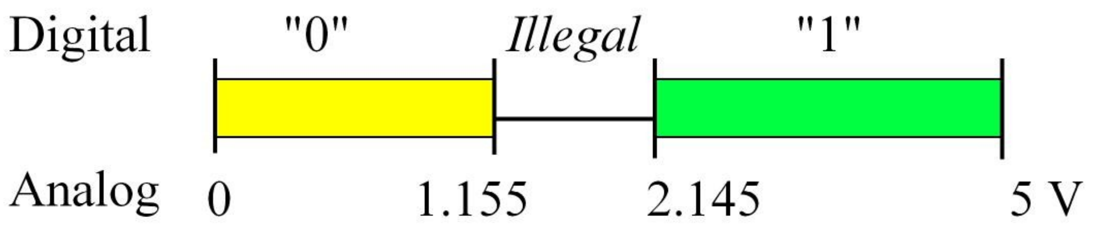
Review: Digital Logic

A	p-type	n-type	~A
0 V	active	off	+3.3V
+3.3V	off	active	0V





A	~A
0	1
1	0

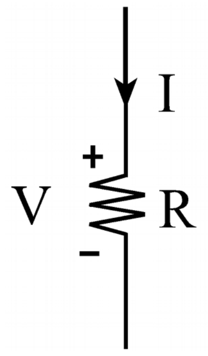


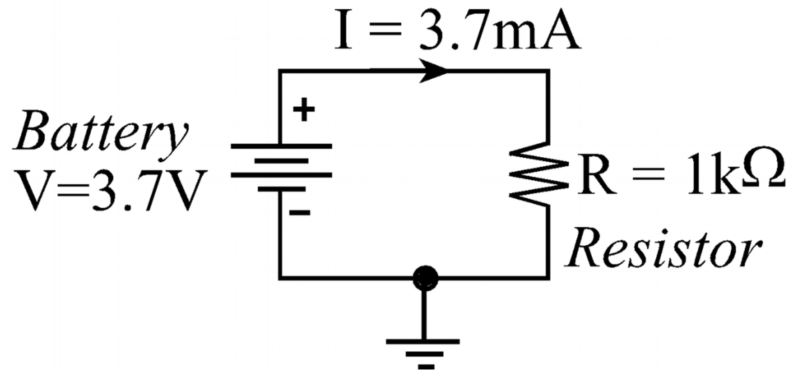
- ❑ AND, OR, NOT
- ❑ Flip flops
- ❑ Registers

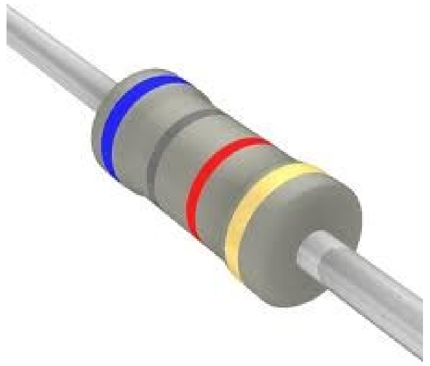
Positive logic:	Negative logic :
True is higher voltage	True is lower voltage
False is lower voltage	False is higher voltage

Review: Ohm's Law

*V = I * R*	*Voltage = Current * Resistance*
I = V / R	*Current = Voltage / Resistance*
R = V / I	*Resistance = Voltage / Current*



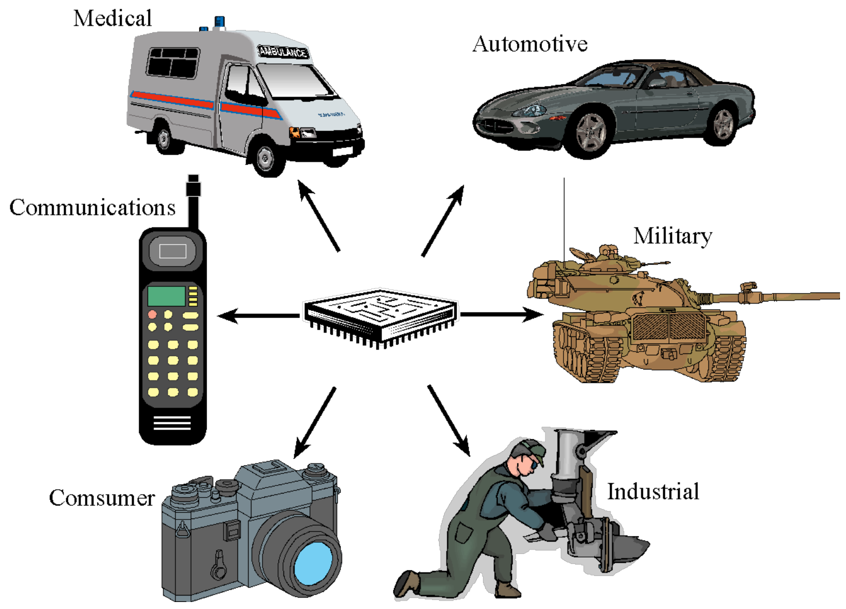


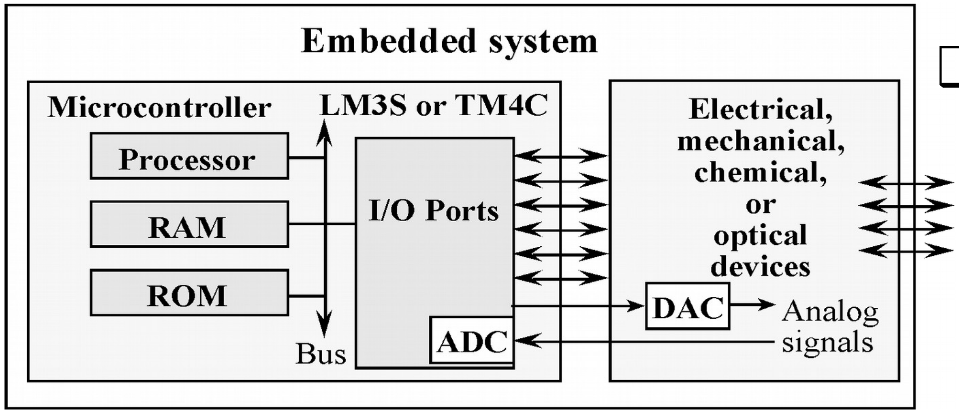


•*P = V * I*	*Power = Voltage * Current*
•*P = V^2 / R*	*Power = $Voltage^2$ / Resistance*
•*P = I^2 * R*	*Power = $Current^2$ * Resistance*

1 amp is 6.241×10^{18} electrons per second = 1 coulomb/sec

Embedded System



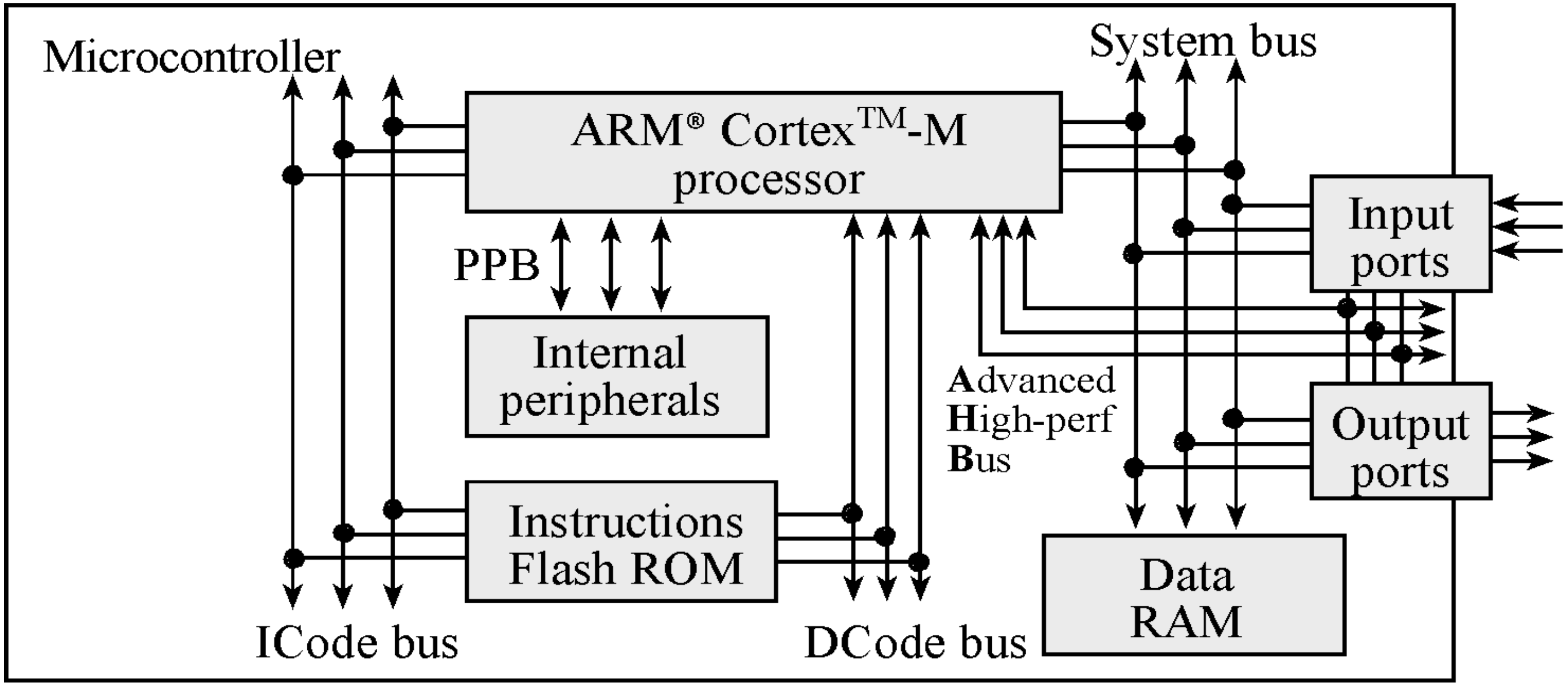


- ❑ Embedded Systems are everywhere
  - ❖ Ubiquitous, invisible
  - ❖ Hidden (computer inside)
  - ❖ Dedicated purpose
- ❑ Microprocessor
  - ❖ Intel: 4004, ..8080,.. x86
  - ❖ Freescale: 6800, .. 9S12,.. PowerPC
  - ❖ ARM, DEC, SPARC, MIPS, PowerPC, Natl. Semi.,...
- ❑ Microcontroller
  - ❖ Processor+Memory+ I/O Ports (Interfaces)

Microcontroller

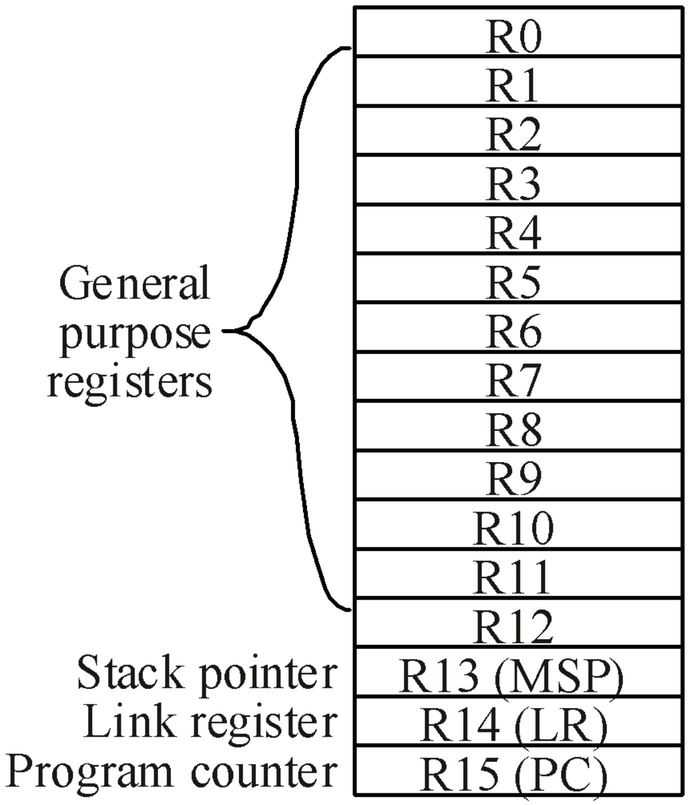
- ❑ Processor – Instruction Set + memory + accelerators
  - ❑ Ecosystem
- ❑ Memory
  - ❑ Non-Volatile
    - o ROM
    - o EPROM, EEPROM, Flash
  - ❑ Volatile
    - o RAM (DRAM, SRAM)
- ❑ Interfaces
  - ❑ H/W: Ports
  - ❑ S/W: Device Driver
  - ❑ Parallel, Serial, Analog, Time
- ❑ I/O
  - ❑ Memory-mapped vs. I/O-instructions (I/O-mapped)

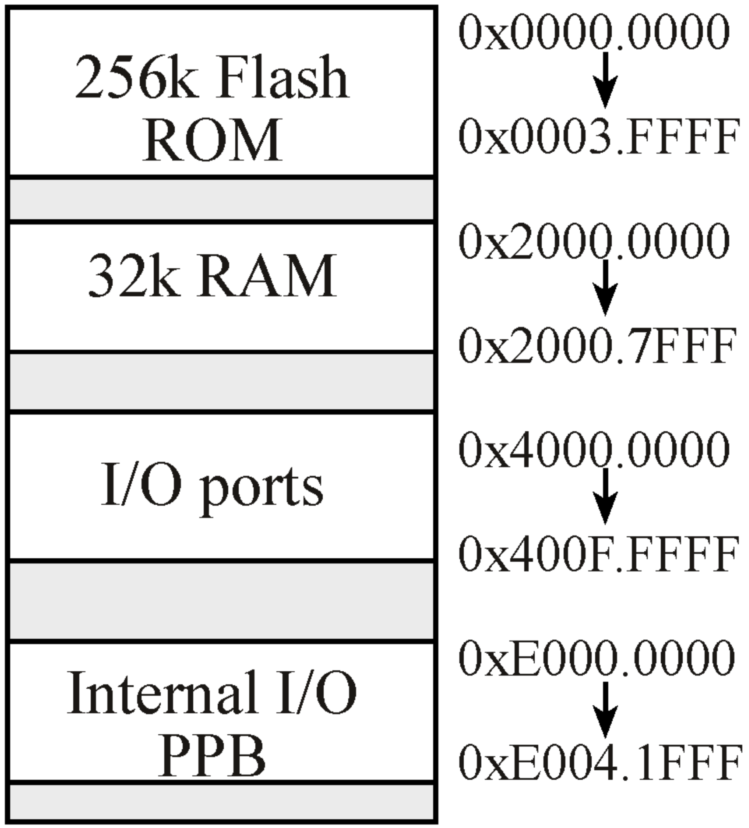
ARM Cortex M4-based System



- ARM Cortex-M4 processor
- *Harvard* architecture
  - Different busses for instructions and data

ARM ISA: Registers, Memory-map

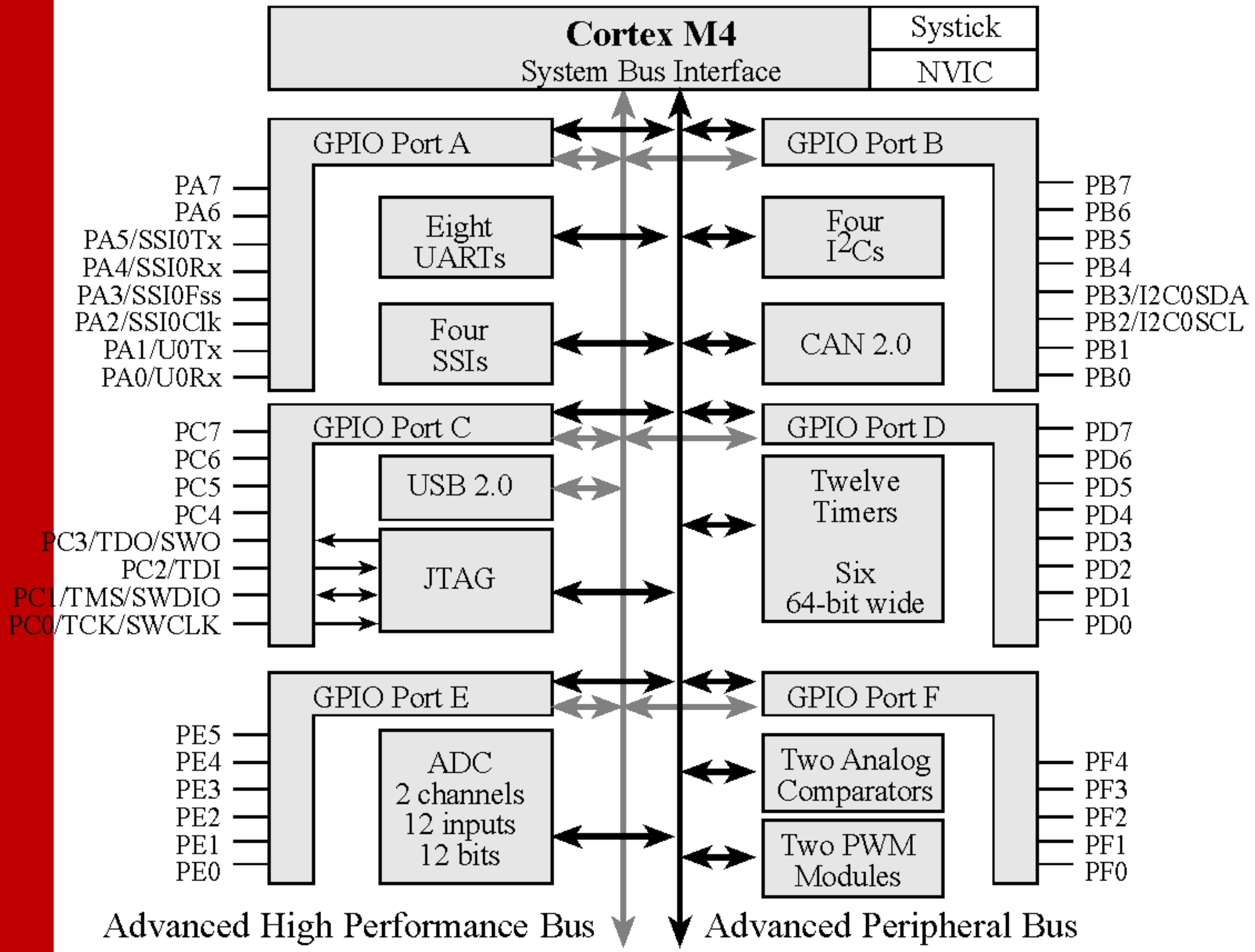




Condition Code Bits		***Indicates***
N	negative	Result is negative
Z	zero	Result is zero
V	overflow	Signed overflow
C	carry	Unsigned overflow

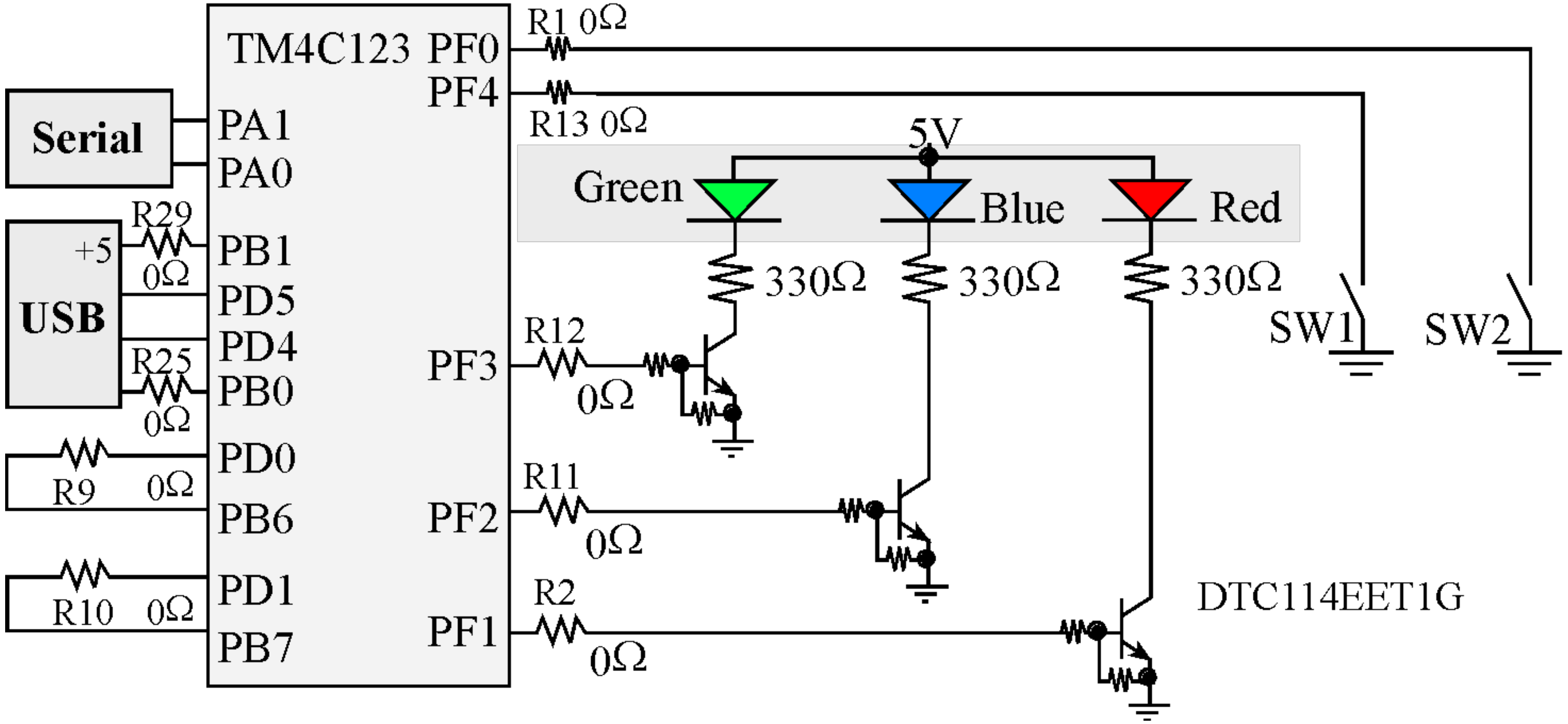
TI TM4C123 Microcontroller

Texas Instruments TM4C123



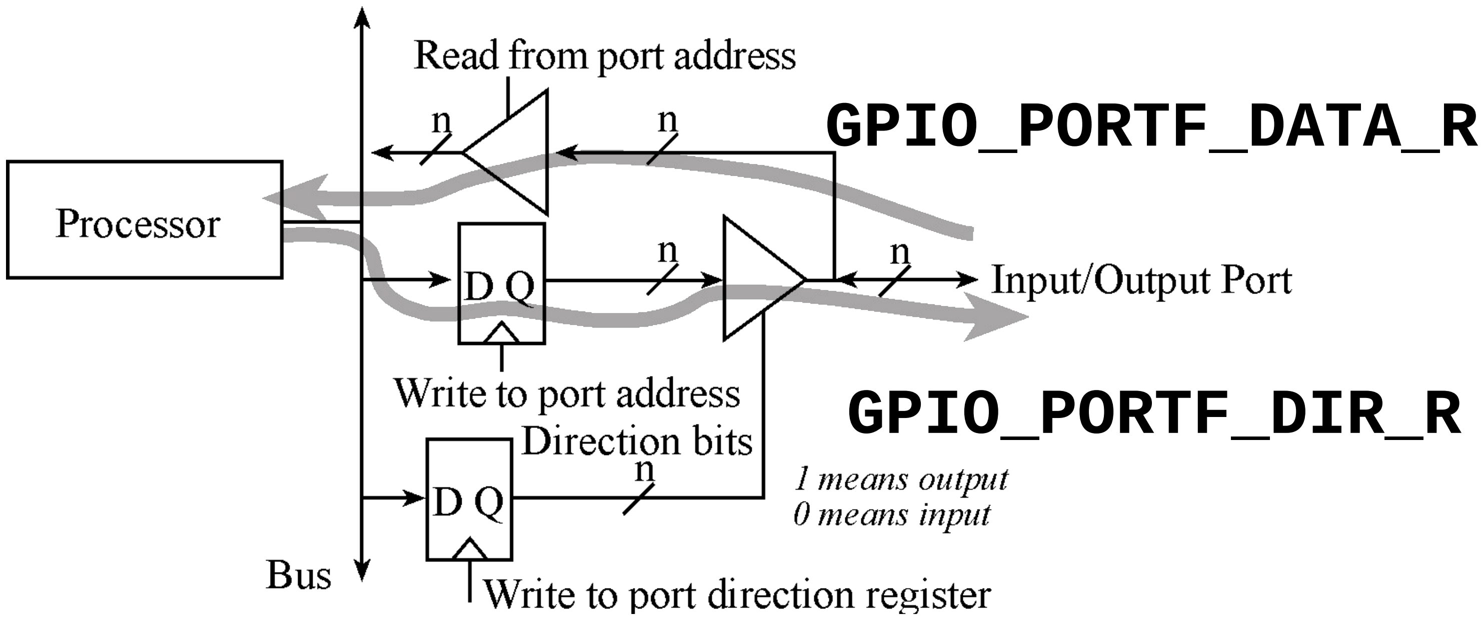
ARM Cortex-M4
+ 256K EEPROM
+ 32K RAM
+ JTAG
+ Ports
+ SysTick
+ ADC
+ UART

LaunchPad Switches and LEDs



- ❑ **The switches on the LaunchPad**
  - ❖ **Negative logic**
  - ❖ **Require internal pull-up (set bits in PUR)**
- ❑ **The PF3-1 LEDs are positive logic**

I/O Ports and Control Registers



The input/output direction of a bidirectional port is specified by its direction register.

`GPIO_PORTF_DIR_R` , specify if corresponding pin is input or output:

- 0 means input
- 1 means output

I/O Ports and Control Registers

Address	7	6	5	4	3	2	1	0	Name
400F.E608	-	-	**GPIOF**	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_RCGCGPIO_R
4002.53FC	-	-	-	DATA	DATA	DATA	DATA	DATA	GPIO_PORTF_DATA_R
4002.5400	-	-	-	DIR	DIR	DIR	DIR	DIR	GPIO_PORTF_DIR_R
4002.551C	-	-	-	DEN	DEN	DEN	DEN	DEN	GPIO_PORTF_DEN_R

- **Initialization (executed once at beginning)**
  1. Turn on clock in **`SYSCTL_RCGCGPIO_R`**
  2. Wait two bus cycles (two NOP instructions)
  3. Write *DIR* bit, 1 for output or 0 for input
  4. Set *DEN* bits to 1 to enable data pins
- **Input/output from pin**

  Input: Read from **`GPIO_PORTF_DATA_R`**

  Output: Write **`GPIO_PORTF_DATA_R`**

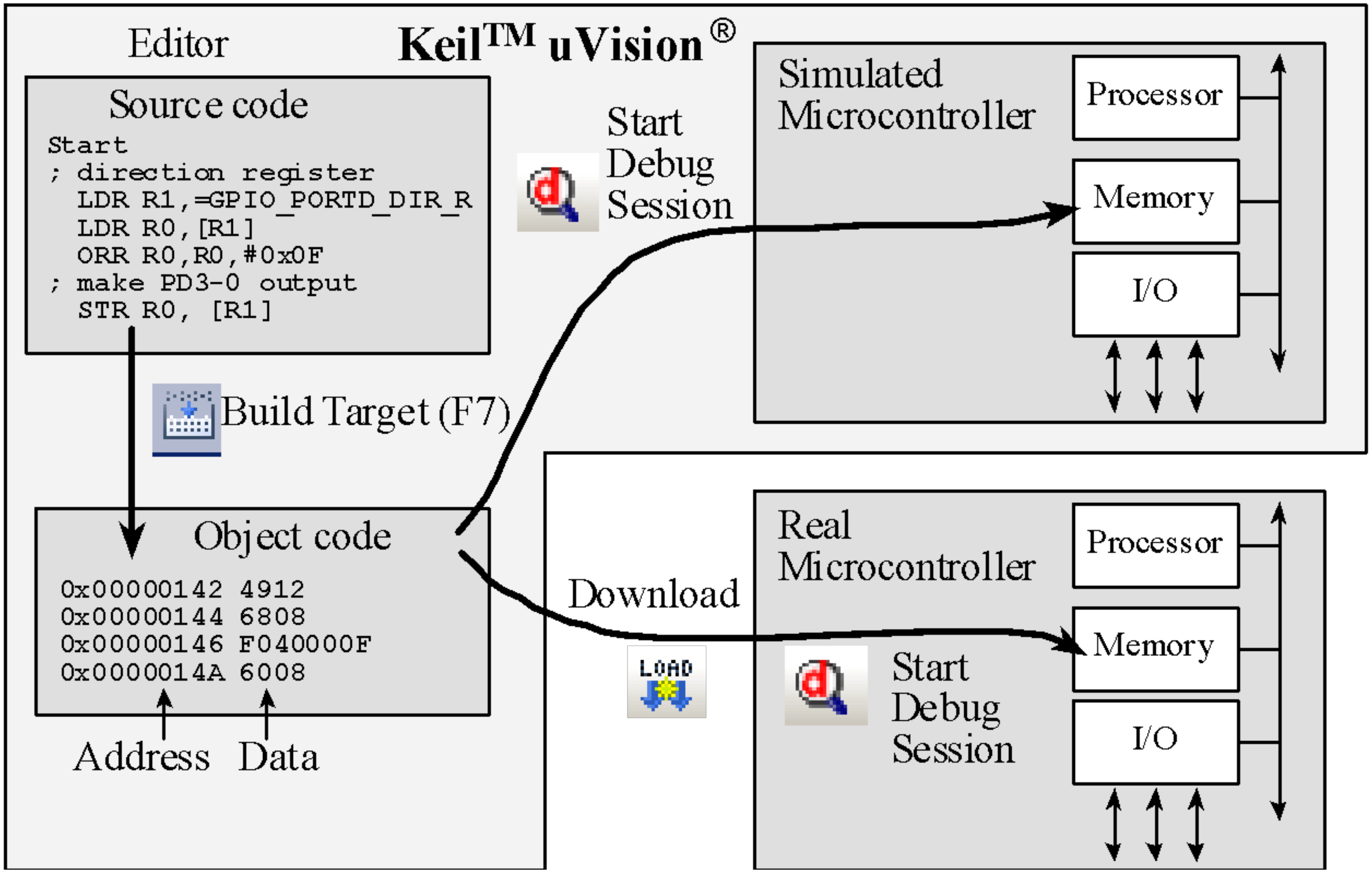
ARM is a Load-Store machine

Code to set (to 1) bit 5 of memory address 0x400FE608

```
SYSCTL_RCGCGPIO_R  EQU   0x400FE608
                                  ; EQU pseudo-op allows use of
                                  ; symbolic name to represent a constant


LDR R1, =SYSCTL_RCGCGPIO_R    ; R1 holds 0x400FE608
LDR R0, [R1]                  ; R0 holds contents of
                              ; location 0x400FE608
ORR R0, R0, #0x20             ; bit5 of R0 is set to 1
STR R0, [R1]                  ; write R0 contents back to
                              ; location 0x400FE608
```

SW Development Environment

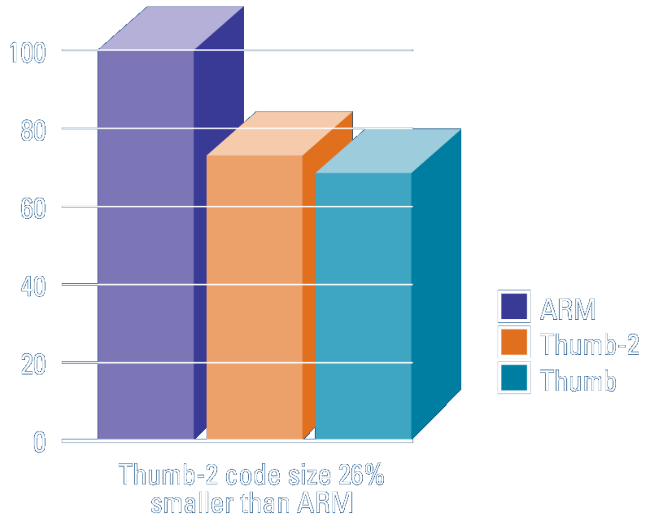


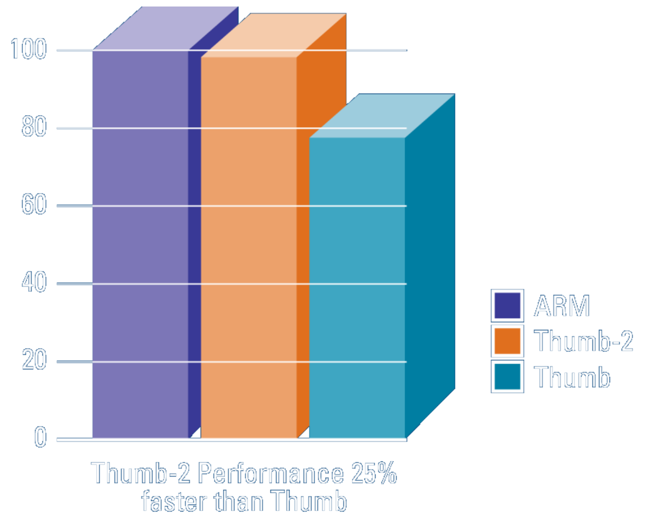
```
Start
; direction register
  LDR R1,=GPIO_PORTD_DIR_R
  LDR R0,[R1]
  ORR R0,R0,#0x0F
; make PD3-0 output
  STR R0, [R1]
```

```
0x00000142 4912
0x00000144 6808
0x00000146 F040000F
0x0000014A 6008
```

ARM ISA: Thumb2 Instruction Set

- ❑ Variable-length instructions
  - ❖ ARM instructions are a fixed length of 32 bits
  - ❖ Thumb instructions are a fixed length of 16 bits
  - ❖ Thumb-2 instructions can be either 16-bit or 32-bit
- ❑ Thumb-2 gives approximately 26% improvement in code density over ARM
- ❑ Thumb-2 gives approximately 25% improvement in performance over Thumb



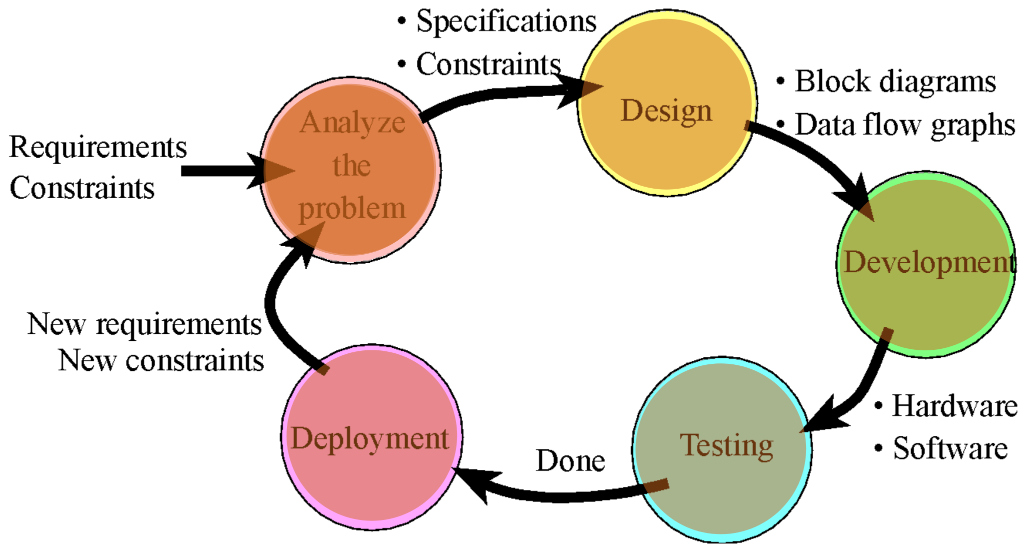


ARM Cortex M4-based System

- “RISC” machine
  - *Pipelining* provides effectively single cycle operation for many instructions
  - Thumb-2 configuration employs both 16 and 32 bit instructions

CISC	RISC
Many instructions	Few instructions
Instructions execute in varying times	Instructions execute in varying times (in the past, just 1 or 2 cycles)
Many instructions access memory	Few memory instructions Load from memory to register Store from register to memory
Many instructions both read and write memory	A few special “atomic memory operations” can read and write memory in single inst.
Fewer and some specialized registers	Many identical general-purpose registers
Numerous address modes	Limited number of addressing modes: register, immediate, and indexed.

Product Life Cycle



Analysis (What?)

- ❖Requirements -> Specifications

Design (How?)

- ❖High-Level: Block Diagrams
- ❖Engineering: Algorithms, Data Structures, Interfacing

Implementation(Real)

- ❖Hardware, Software

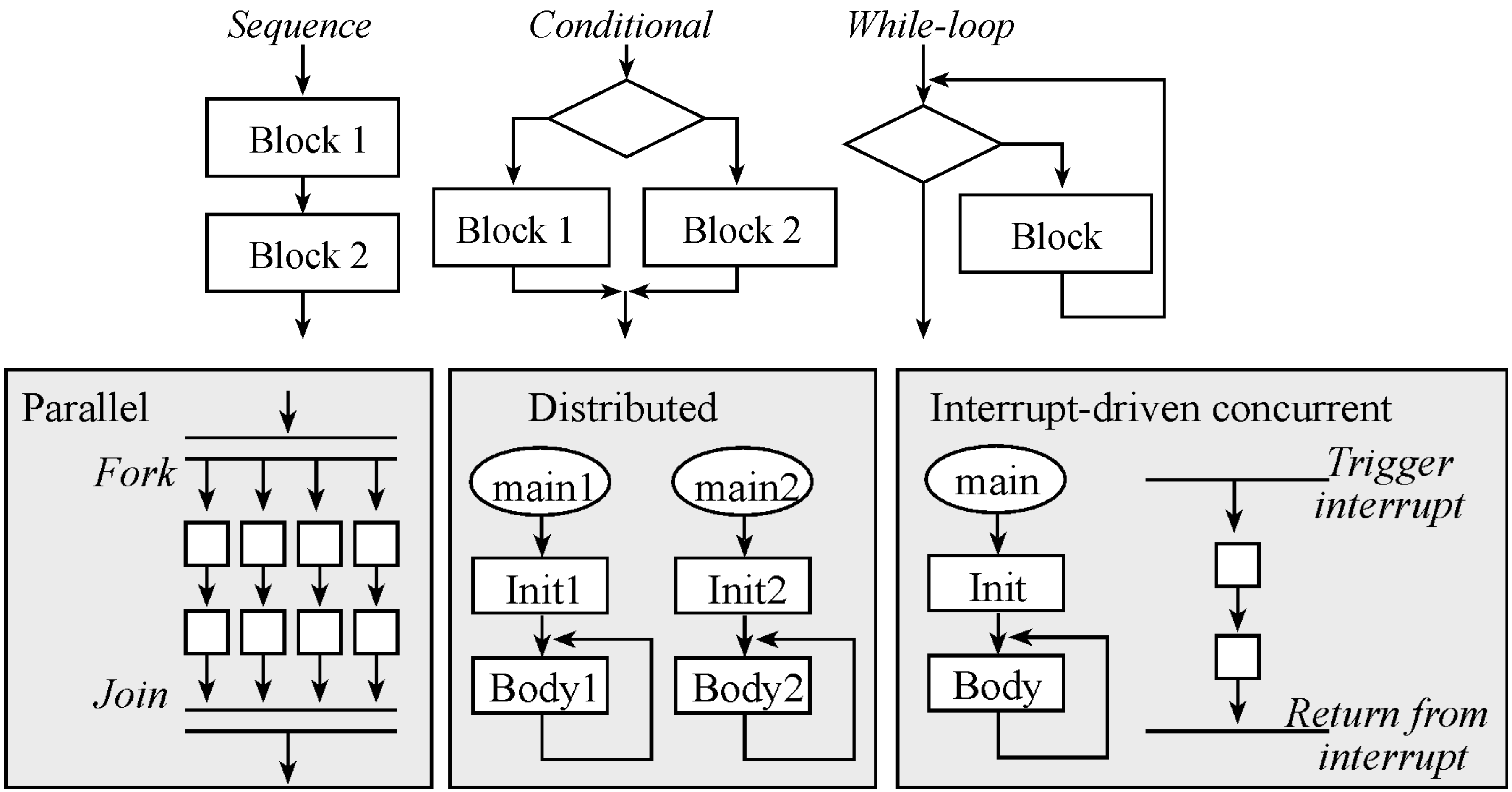
Testing (Works?)

- ❖Validation:Correctness
- ❖Performance: Efficiency

Maintenance (Improve)

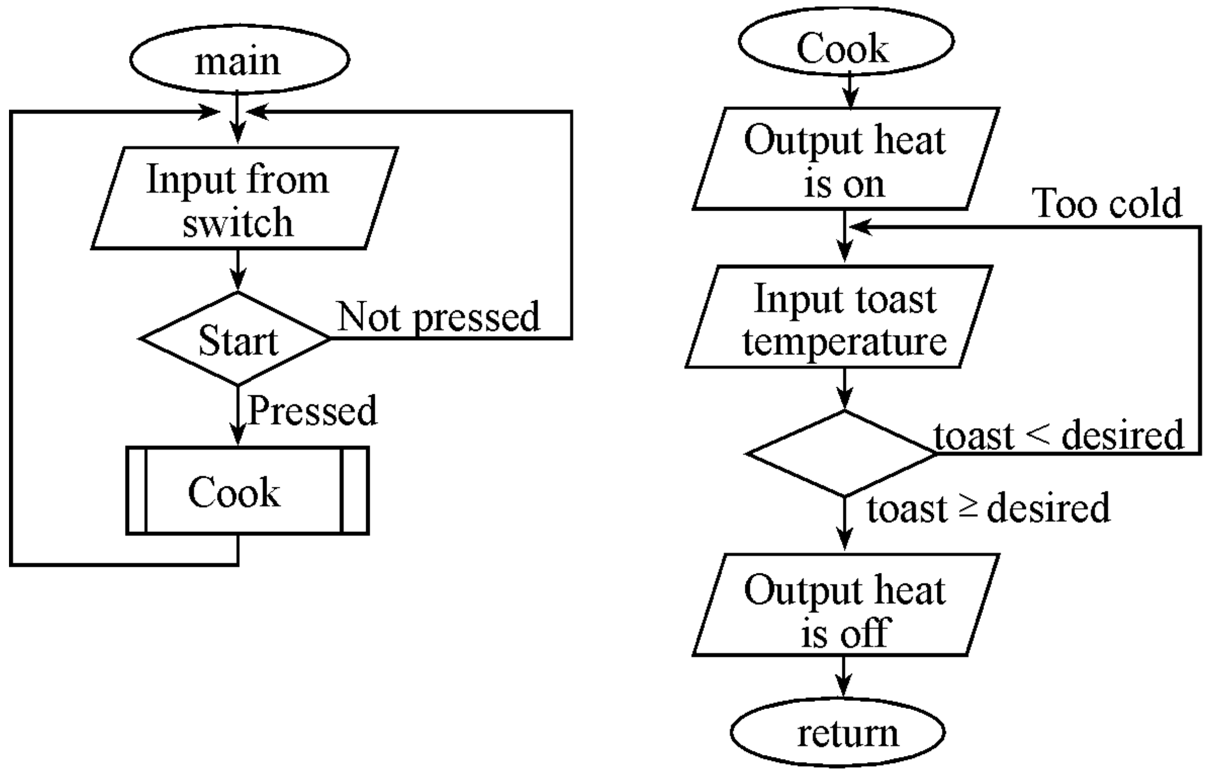
Structured Programming

Common Constructs (as Flowcharts, also pseudocode)



Flowchart

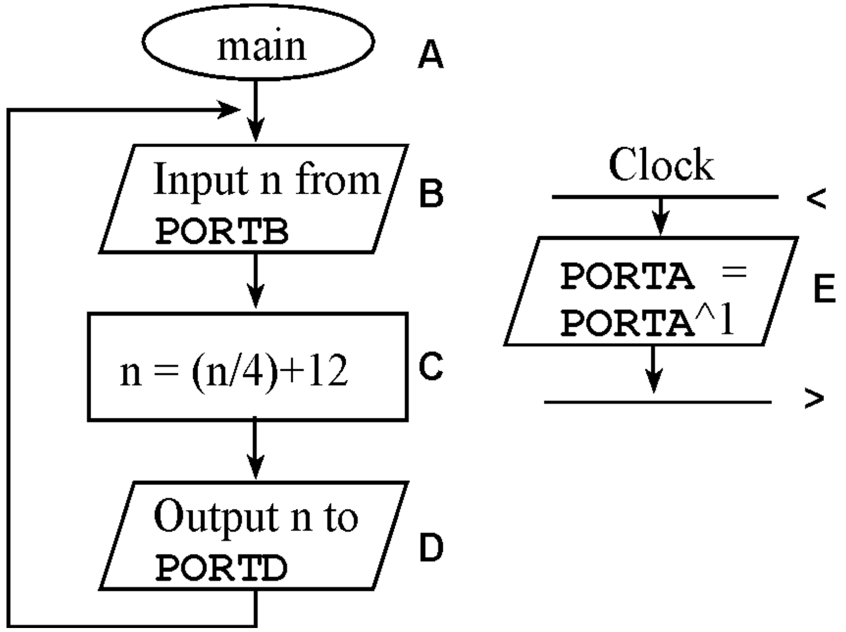
Toaster oven:

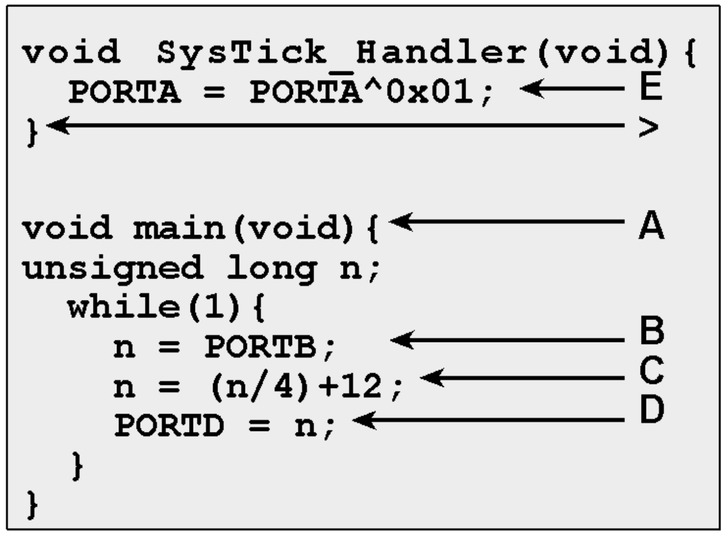


Coding in assembly and/or high-level language (C)

Flowchart

- **Example 1.3.** Design a flowchart for a system that performs two independent tasks. The first task is to output a 20 kHz square wave on **PORTA** in real time (period is 50 ms). The second task is to read a value from **PORTB**, divide the value by 4, add 12, and output the result on **PORTD**. This second task is repeated over and over.





```
void  SysTick_Handler(void){
  PORTA = PORTA^0x01;  <------- E
}<-------------------------------- >

void main(void){ <--------------- A
unsigned long n;
  while(1){
    n = PORTB;  <---------------- B
    n = (n/4)+12;  <------------- C
    PORTD = n;  <---------------- D
  }
}
```