CS301
Embedded System and Microcomputer Principle

Lecture 1: Introduction

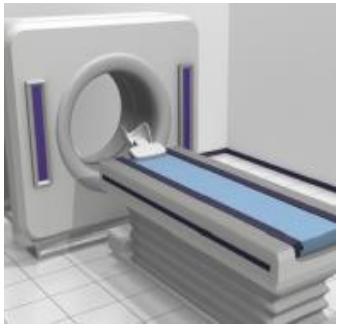
2023 Fall

Outline

- **Embedded systems**
- Microcontroller
- Development Environment
- Course information

Embedded Systems

- A computer, pretending not to be a computer
- Often there to replace previously electromechanical components
  - Physical Things Augmented with Computing/Communication
  - Computing/Communication “Embedded” into Physical Things



What is Embedded System?

- "An embedded system is an application that contains at least one programmable computer ... and which is used by individuals who are unaware that the system is computer-based."

  -- Michael J. Pont, *Embedded C*

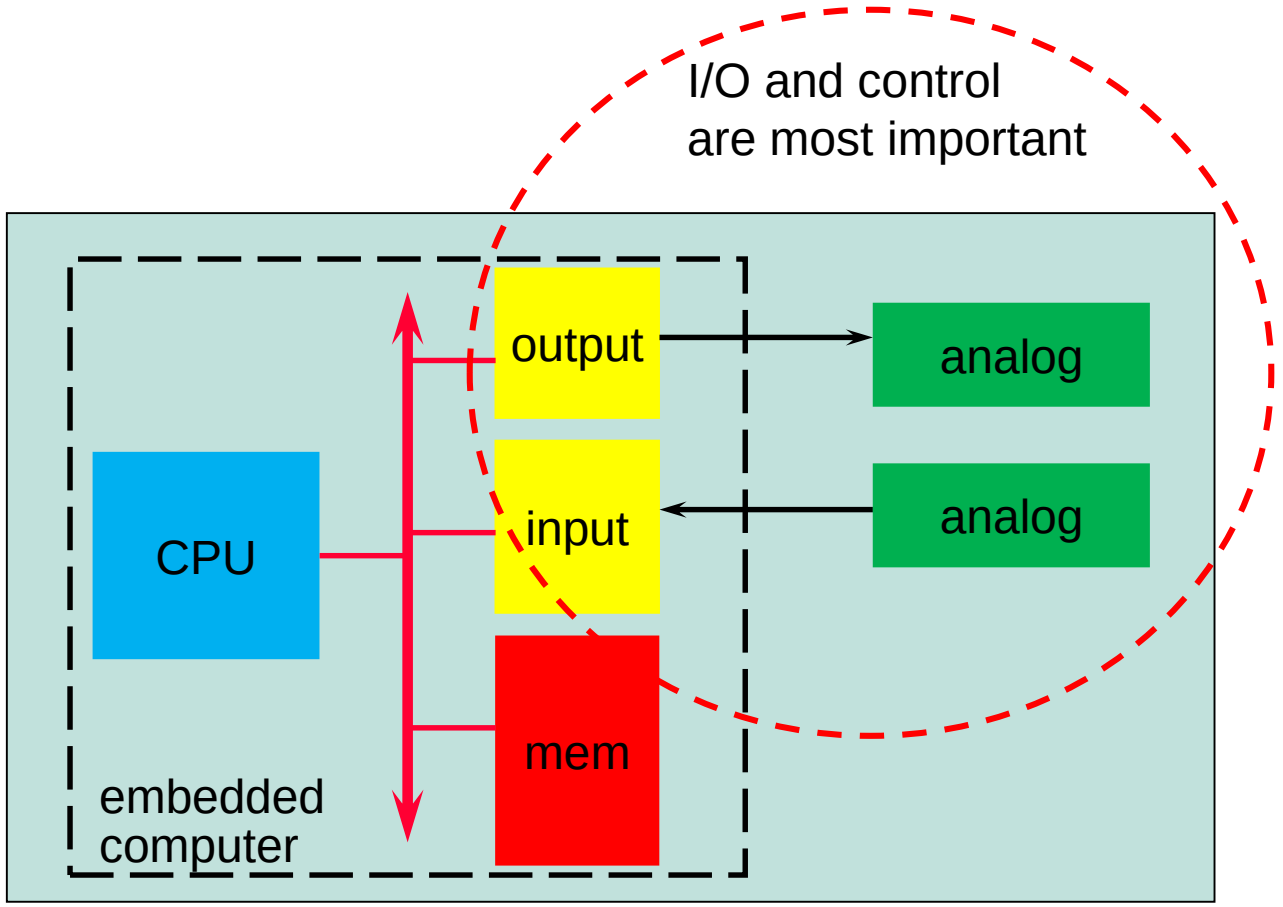
- Programmable computers require programs
  - embedded software

What is Embedded System?

- "Embedded systems are information processing systems that are embedded into an enclosing product"
  -- Peter Marwedel, *Embedded System Design*
  - Main reason for buying is **not** information processing
- Any device that includes a programmable processor but is not itself a general-purpose computer
- Take advantage of application characteristics to optimize the design:
  - Do not need all general-purpose bells and whistles

Embedded systems inside

- Same basics inside as general purpose computer



Embedded = Smart

- Computers embedded into objects
  - Augment objects with programmatic control, communication, sensing, etc
- Let the world know you:
  - Make physical objects/phenomena accessible to digital world
- Let you know the world:
  - Give intelligence/life to physical objects so that they can sense/react
- Smart home, Intelligent Transportation System, smart city

Some Concepts to Clarify

- Embedded systems refer to not only small devices or gadgets






- But also large, complex systems requiring strict reliability and real-time responses





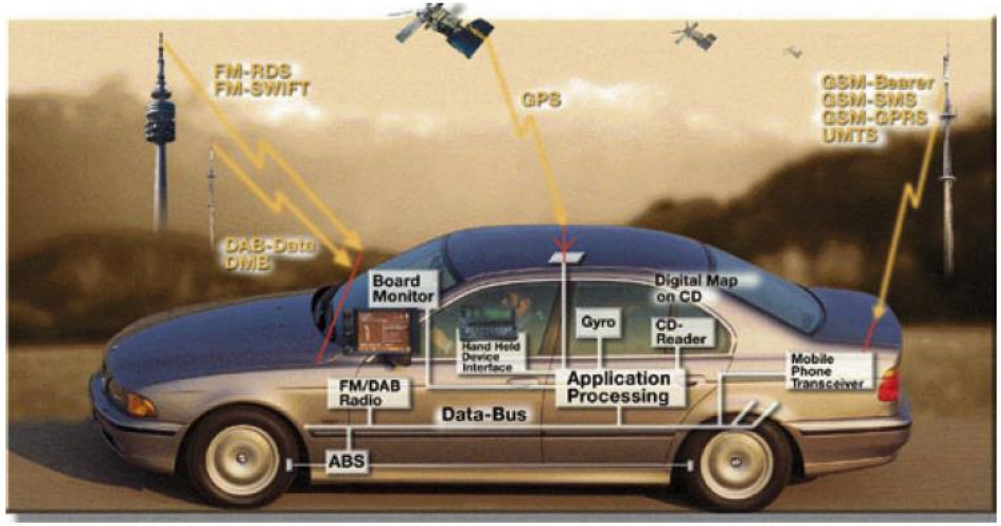
Some Concepts to Clarify

- A product, e.g., video decoder, may be implemented using pure hardware or microprocessor + embedded software
  - A chip may be implemented using pure logic gates or a microprocessor + peripheral logic

  → There must be some processors + software
- An embedded system may or may not have OS
  - Simple systems may be implemented by a single program that runs continuously
  - Systems that need to control and respond to many activities may require an OS for management

Benefits of Embedded Systems

- Reduced cost
- Increased functionality
- Improved performance
- Increased overall dependability

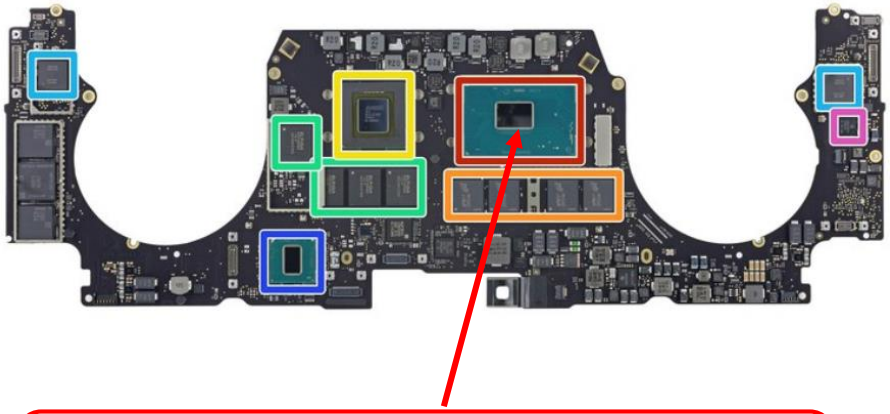


Outline

- Embedded systems
- **Microcontroller**
- Development Environment
- Course information

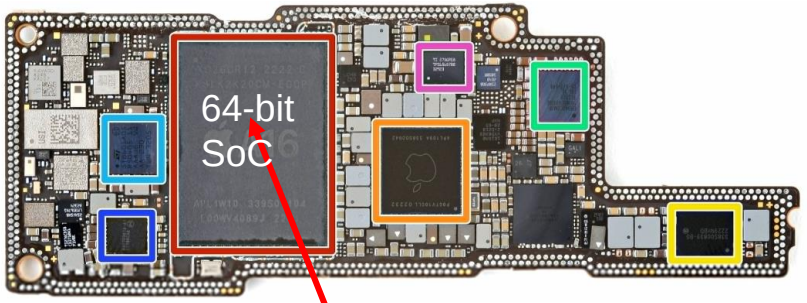
Teardown of MacBook vs iphone 14

Macbook



- Intel Core i7-6700HQ 2.6 GHz (up to 3.5 GHz) quad-core processor
- Micron MT52L1G32D4PG-093 4 GB LPDDR3 (four chips for 16 GB total)
- AMD Radeon Pro 450 GPU
- Elpida (Micron) EDW4032BABG-70-F 512 MB GDDR5 RAM (four chips for 2 GB total)

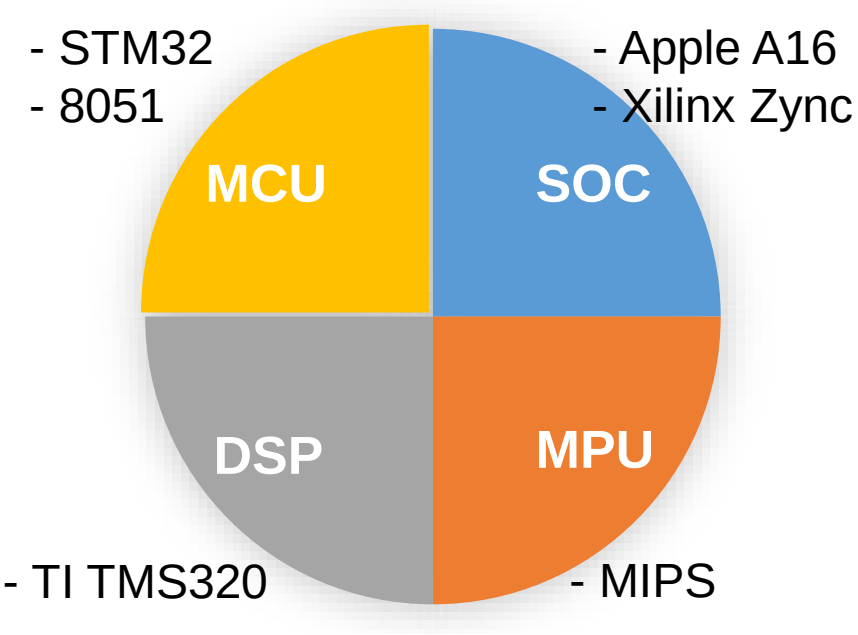
iphone 14



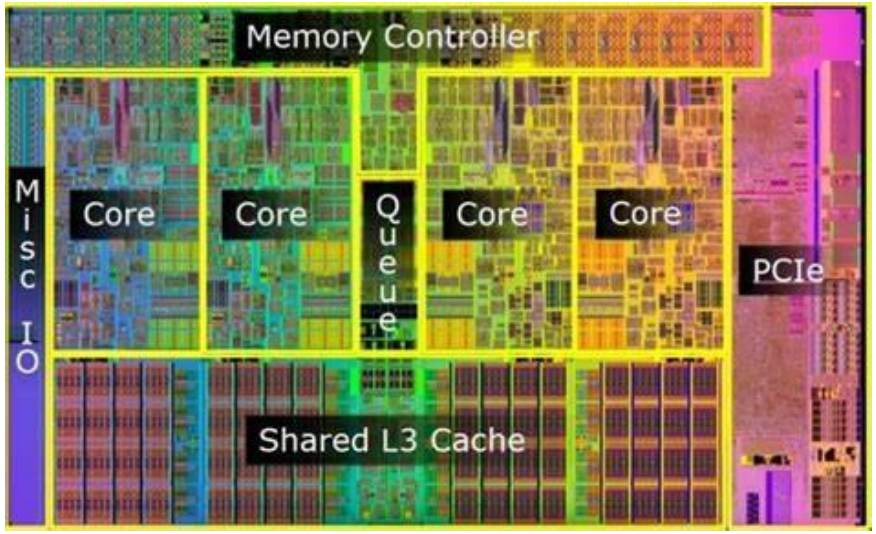
- Apple APL1W10/339S01104 A16 64-bit hexa-core applications processor w/ penta-core GPU layered underneath most likely Samsung K3LK2K20CM-EGCP 6 GB LPDDR5 SDRAM memory
- Apple APL109A/338S00942 power management
- Apple/Dialog Semiconductor 338S00839-B0 power management

Embedded Processor

- Microprocessors (微处理器)
- DSPs (数字信号处理器)
- Microcontrollers (微控制器/单片机)
- SOC (片上系统)

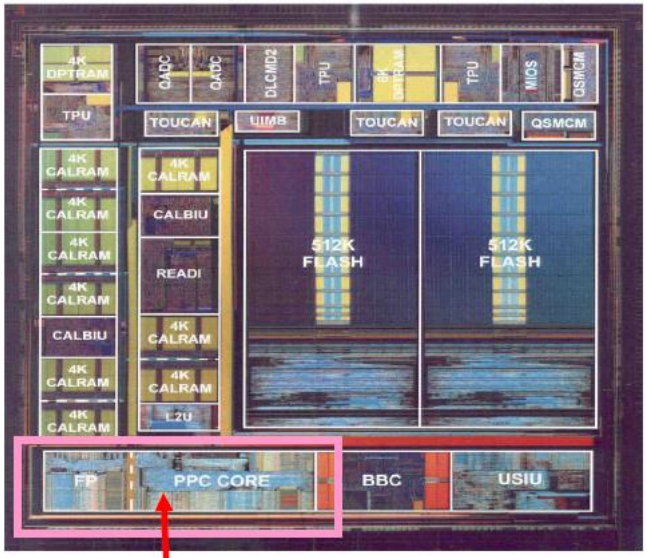


CPU vs SoC vs Microcontroller

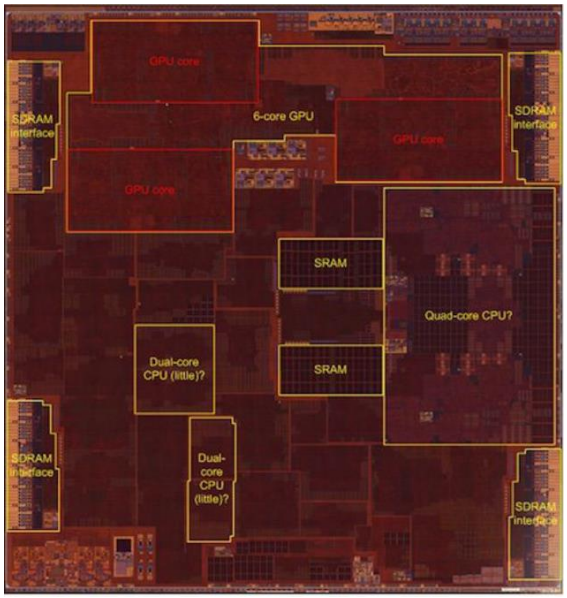


Intel's Core i7
CPU Die map

Freescale PPC5554 Die map
(CPU is only a part of MCU)



PowerPC Core + FP



Apple A10 SoC Teardown
rreveals (Multiple
processors within one chip)

Microcontroller

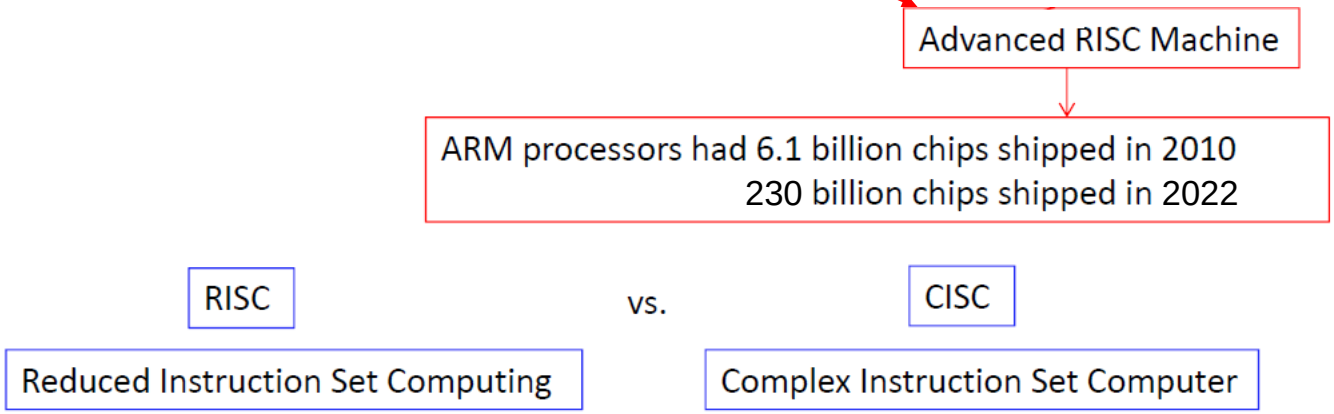
- A microcontroller(MCU) is a complete computer system optimized for hardware control that encapsulates the entire processor, memory and all of the I/O peripherals on a single piece of silicon. An MCU usually include:
  - An 8/16/32-bit microprocessor.
  - A little measure of RAM.
  - Programmable ROM and flash memory.
  - Parallel and serial I/O.
  - Timers and signal generators
  - ADC and DAC

Why MCU

- Smaller board area
- Low cost
- Low power consumption
- High reliability
- Dedicated for special purpose application
- Microcontrollers save time and money!

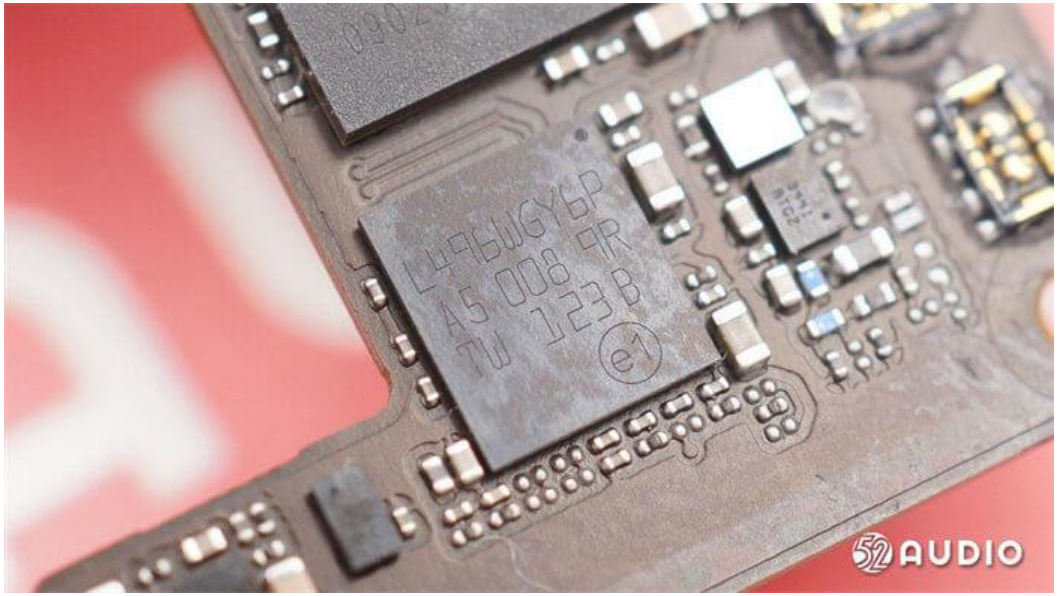
ARM based Microcontroller

- Acorn Computers Ltd. was a British computer company established in Cambridge, England, in 1978.
- In 1983 the Acorn started its Acorn RISC Machine (ARM) project and the resulting Reduced Instruction Set Computing (RISC) processor would eventually become known as the 32-bit ARM1
- Some companies (Intel, Marvell, Qualcomm, Apple, etc) paid for 'architectural license' which allows to design own cores.



Apple AirPods

- On Charge Station: STM32L496: Arm Cortex-M4 32-bit MCU+FPU

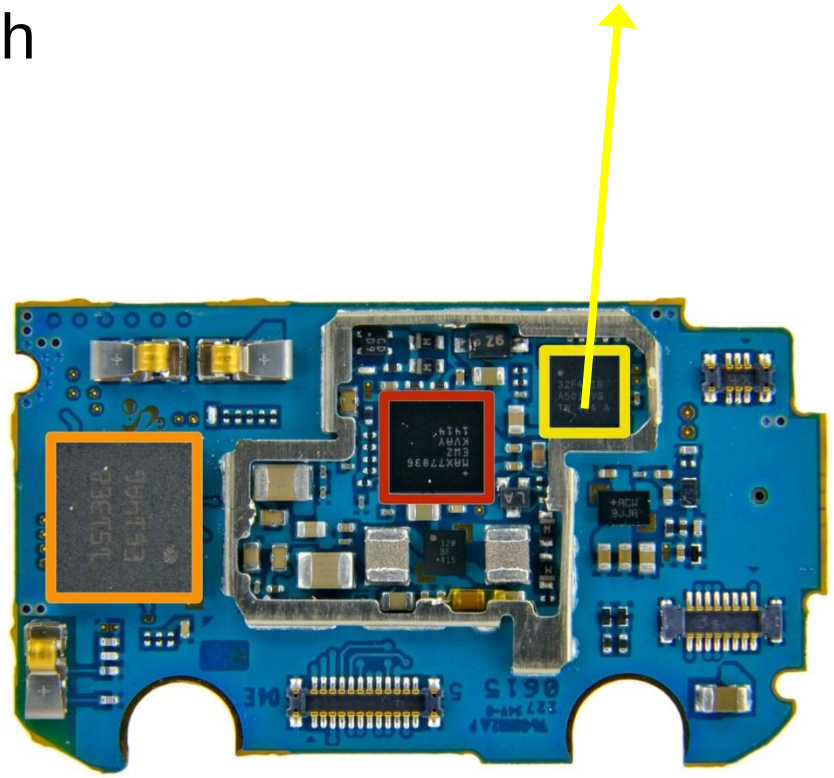


Apple TV

- ST Microelectronics STM32L 151QD ultra-low-power ARM Cortex-M3 MCU

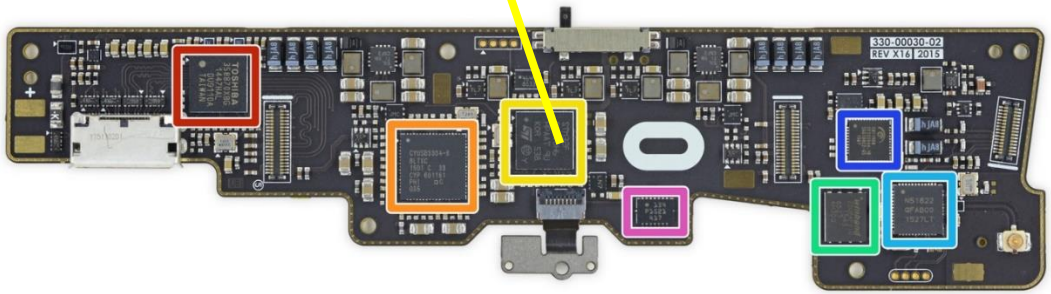
Samsung Galaxy Gear

- STMicroelectronics STM32F401B ARM-Cortex M4 MCU with 128KB Flash



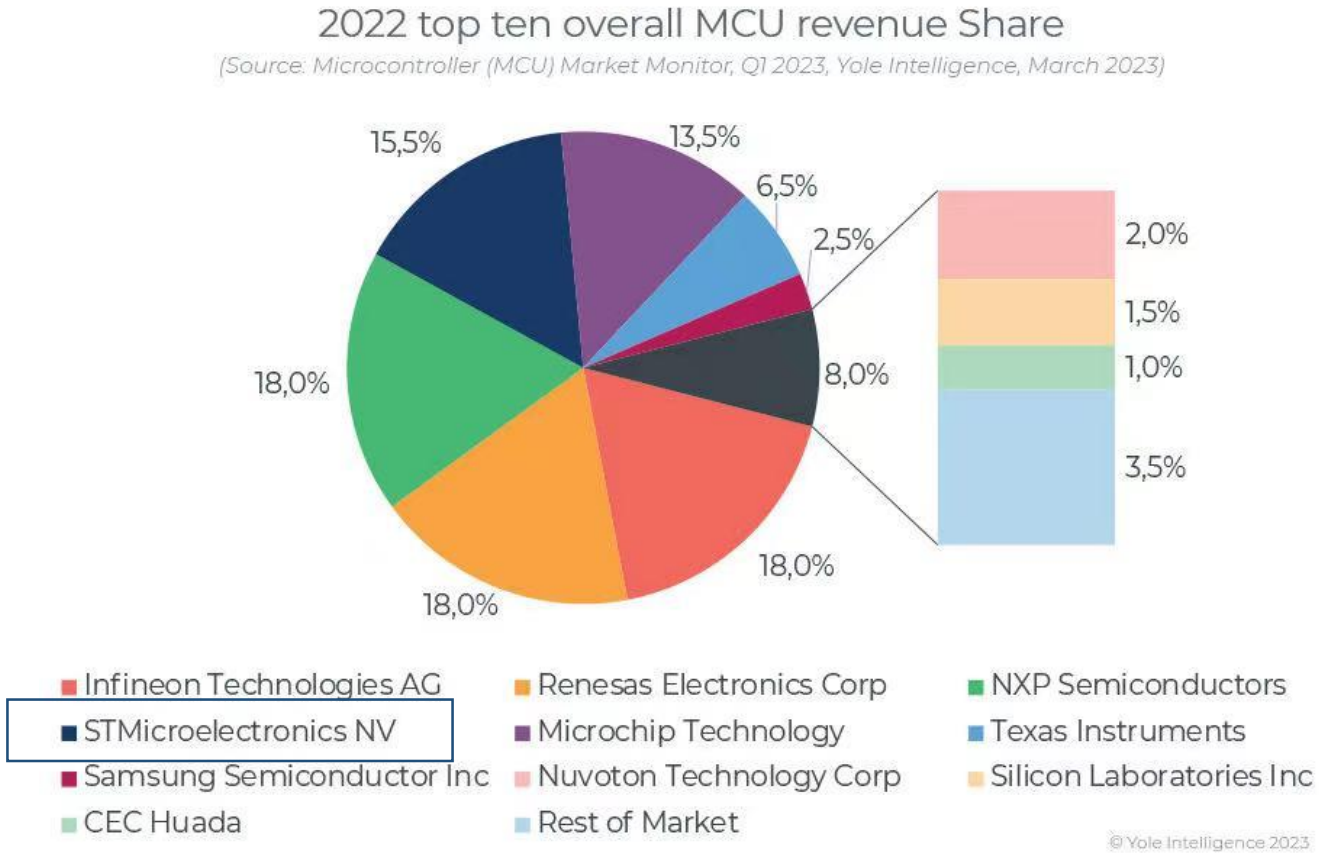
Oculus VR

- Facebook's $2 Billion Acquisition Of Oculus in 2014
- ST Microelectronics STM32F072VB ARM Cortex-M0 32-bit RISC Core Microcontroller



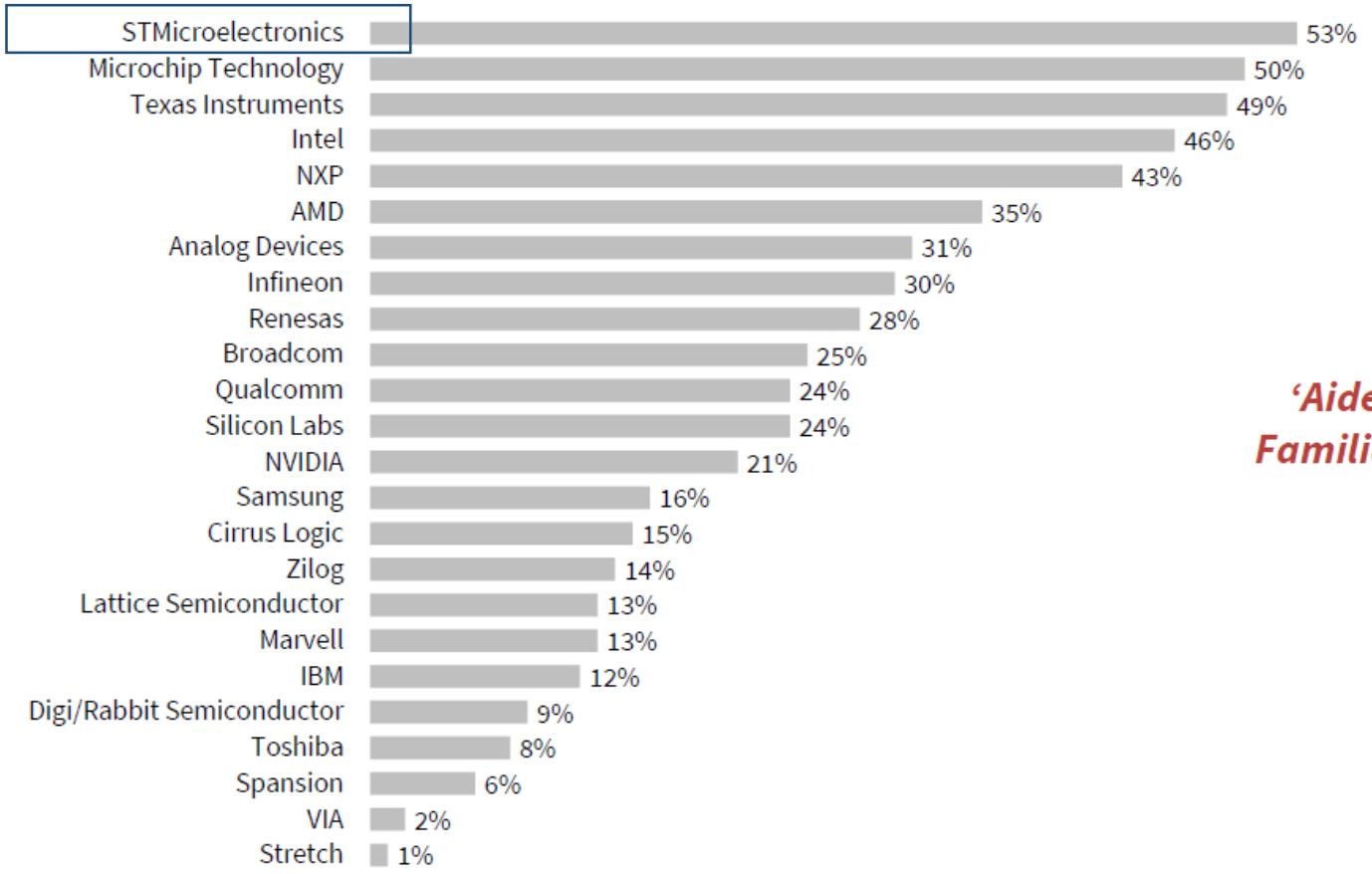
ARM MCUs

- ARM does not produce chips but designs & sells the license of its of architecture to others.
  - More than 200 companies have bought the Arm architecture and provide their Arm chips.



Familiarity with MCU vendors

- STMicro, Microchip, TI, Intel, and NXP are the most well-known processor vendors



--- 2023 Embedded Survey by Aspencore

Outline

- Embedded systems
- ARM Microcontroller
- **Development Environment**
- Course information

Levels of Program Code

C Program

```
int main(void){
  int i;
  int total = 0;
  for (i = 0; i < 10; i++) {
    total += i;
  }
  while(1); // Dead loop
}
```

Compile

Assembly Program

```
        MOVS r1, #0
        MOVS r0, #0
        B    check
loop    ADD  r1, r1, r0
        ADDS r0, r0, #1
check   CMP  r0, #10
        BLT  loop
self    B    self
```

Assemble

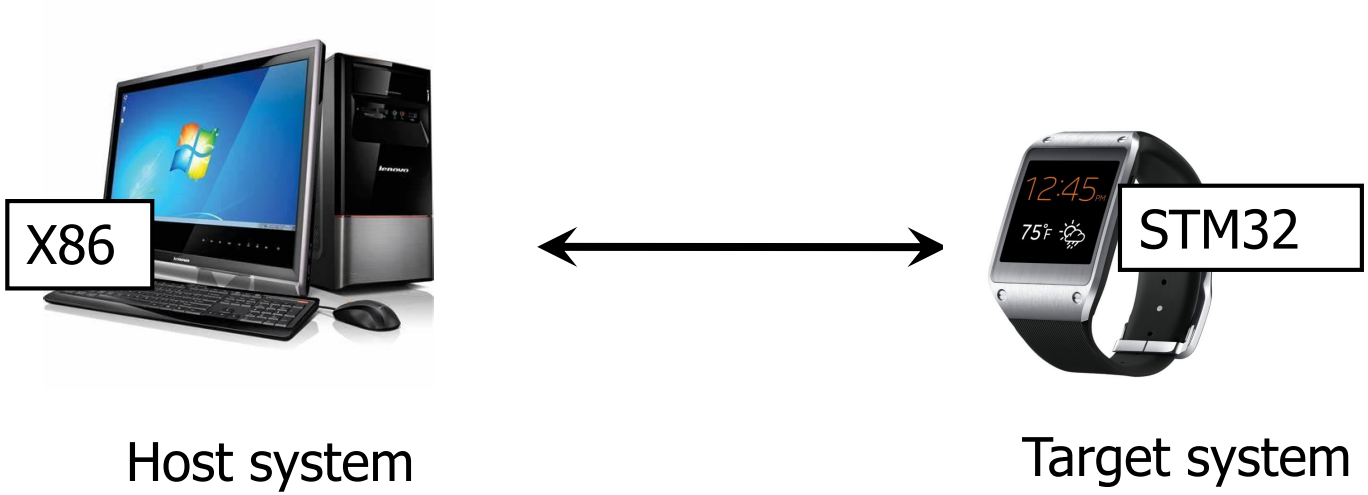
Machine Program

```
0010000100000000
0010000000000000
1110000000000001
0100010000000001
0001110001000000
0010100000001010
1101110011111011
1011111100000000
1110011111111110
```

- **High-level language**
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- **Assembly language**
  - Textual representation of instructions
  - Human-readable format instructions
- **Hardware representation**
  - Binary digits (bits)
  - Encoded instructions and data
  - Computer-readable format instructions

Development Environment

- Host: a computer running programming tools for development of the programs
- Target: the HW on which code will run
- After program is written, compiled, assembled and linked, it is transferred to the target



Development Environment

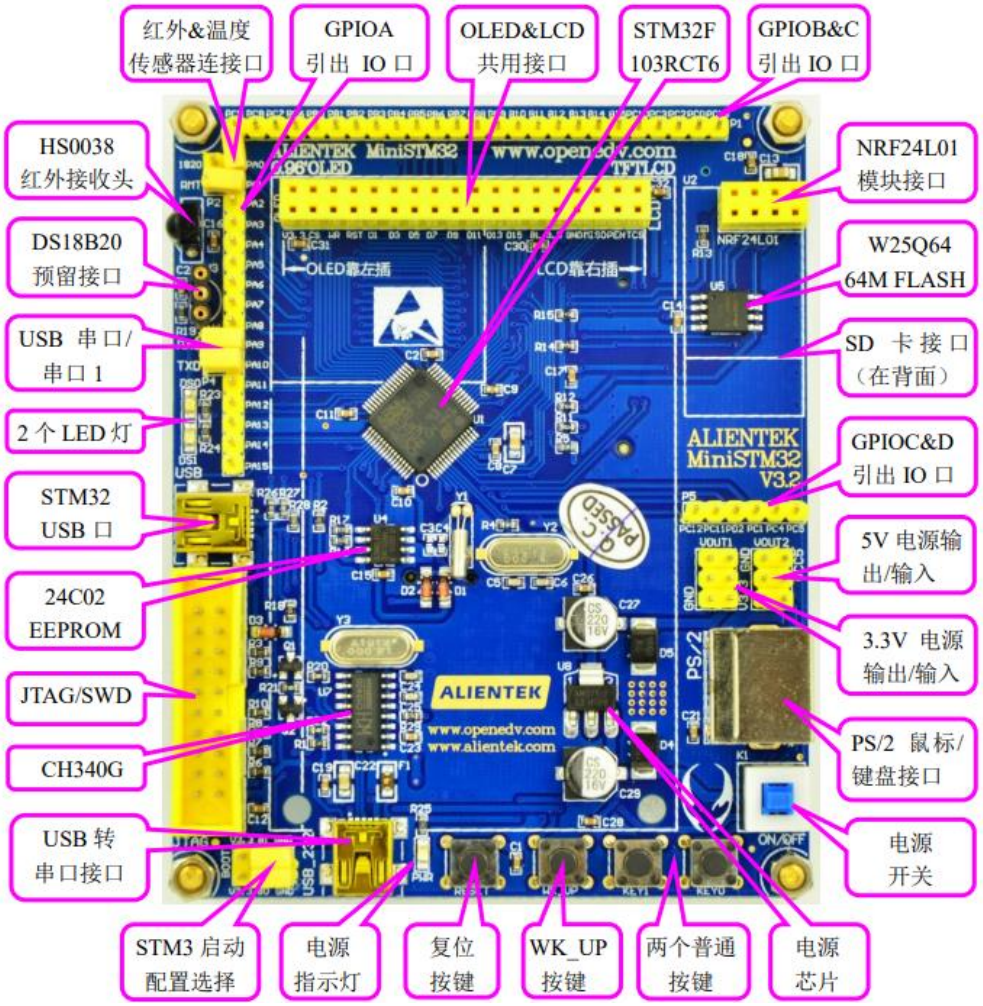
- Development board:
  - Before real hardware is built, software can be developed and tested using development boards
  - Development boards usually have the same CPU as the end product and provide many IO peripherals for the developed software to use as if it were running on the real end product
- Tools for program development
  - *Integrated Development Environment* (IDE): cross compiler, linker, loader, …
  - OS and related libraries and packages

STM32 CubeIDE

KEIL Tools by ARM
µVision®5
Integrated Development Environment
KEIL is a trademark and µVision is a registered trademark of ARM Ltd. All rights reserved.
This product is protected by US and international laws.

What We Need in Lab

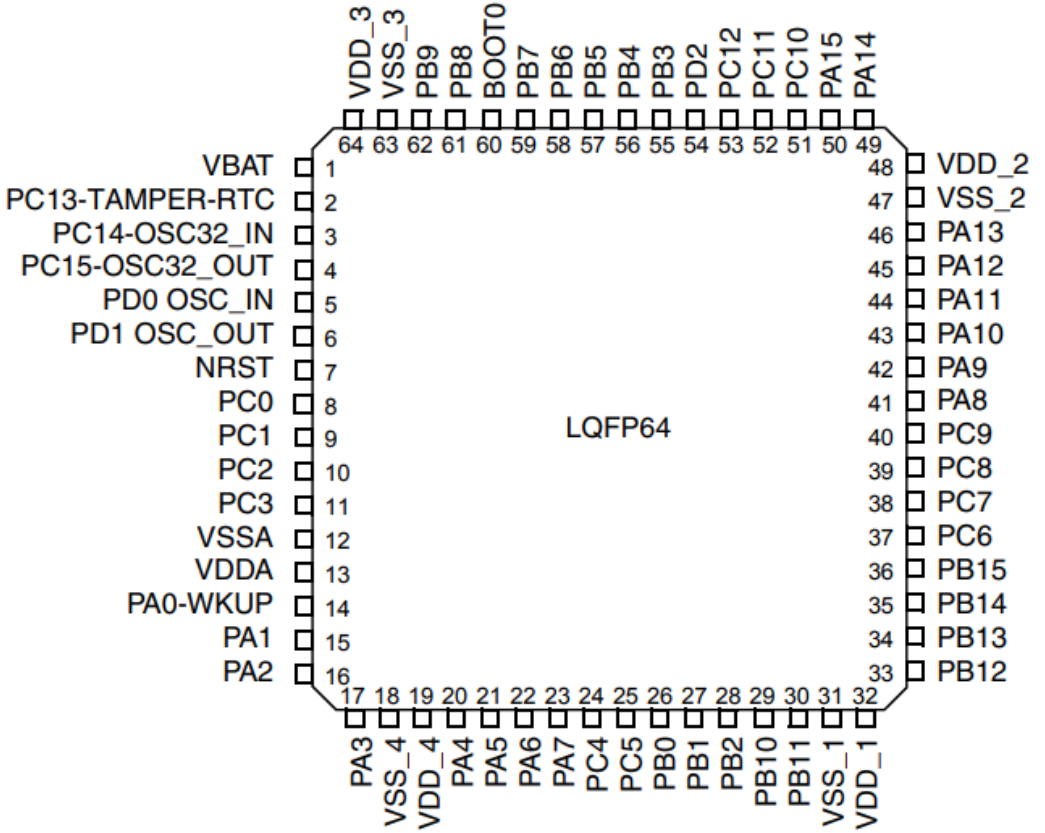
- ST Microelectronics STM32F103 (正点原子miniSTM32)



STM32 CubeIDE

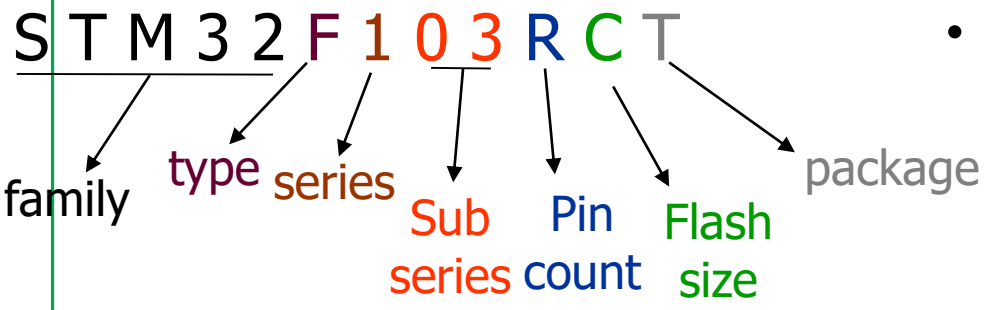
ARM MCU outside view

- STM32F103RCT6 outside view (pin-out):
- 64 pin version



STM32 Naming Conventions

- Family
  - Names of the new Arm products of ST begin with STM32.
- Type
  - L: Low Power
  - F: Mainstream (Foundation)
  - H: High performance
  - W: Wireless



- Series
  - 0: Cortex-M0
  - 1,2: Cortex-M3
  - 3,4: Cortex-M4
  - 7: Cortex-M7
- Sub series
  - Chips with higher sub-series numbers have richer configurations
- Package
  - H: BGA (Ball Grid Array)
  - T:LQFP (Low-profile Quad Flat Pack)
  - U:QFN (Quad Flat No-leads)
  - Y:WLCSP
- Pin count

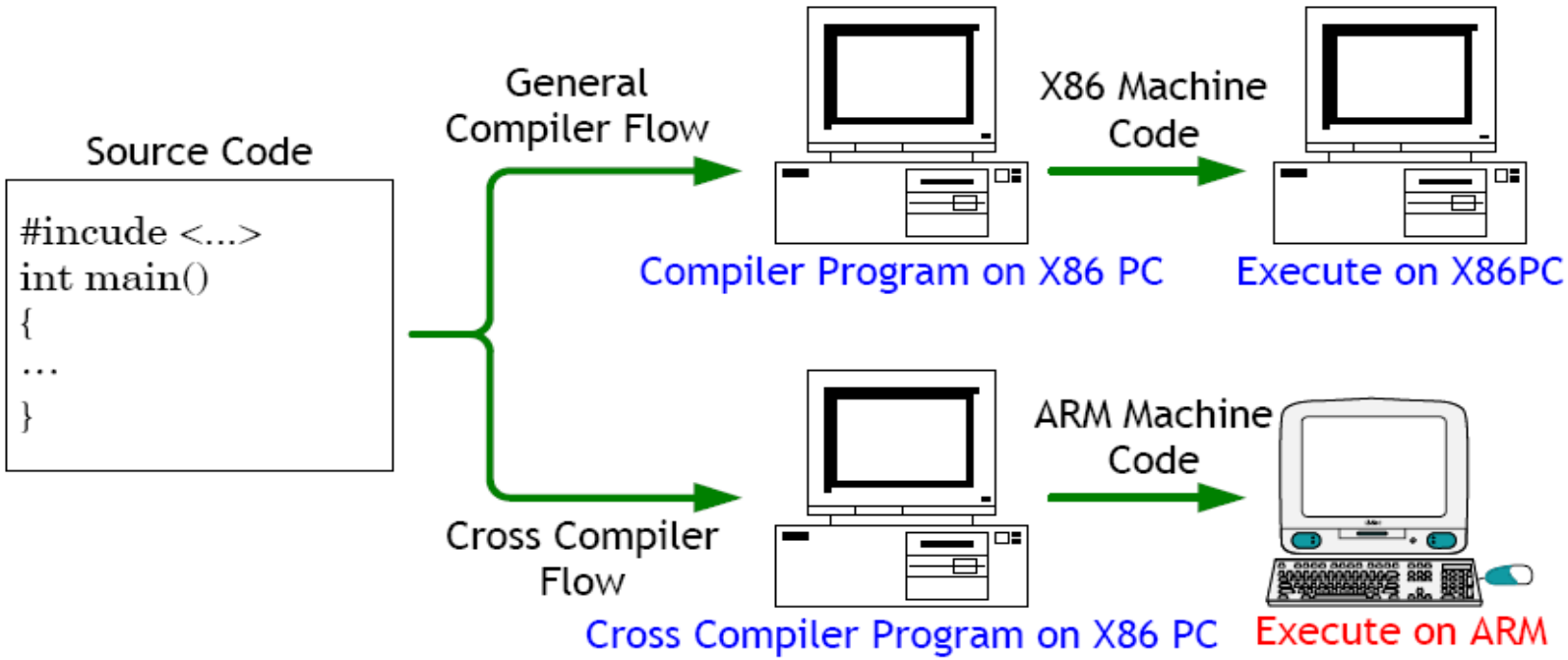
	F	G	K	T	S	C	R	V	Z
# of pins	20	28	32	36	44	48	64	100	144

- Flash size

	4	6	8	B	C	D	E	F	G	H	I
Group	**Low density**		**Medium density**		**High density**						
Flash	**16K**	**32K**	**64K**	**128K**	**256K**	**384K**	**512K**	**768K**	**1M**	**1.5M**	**2M**

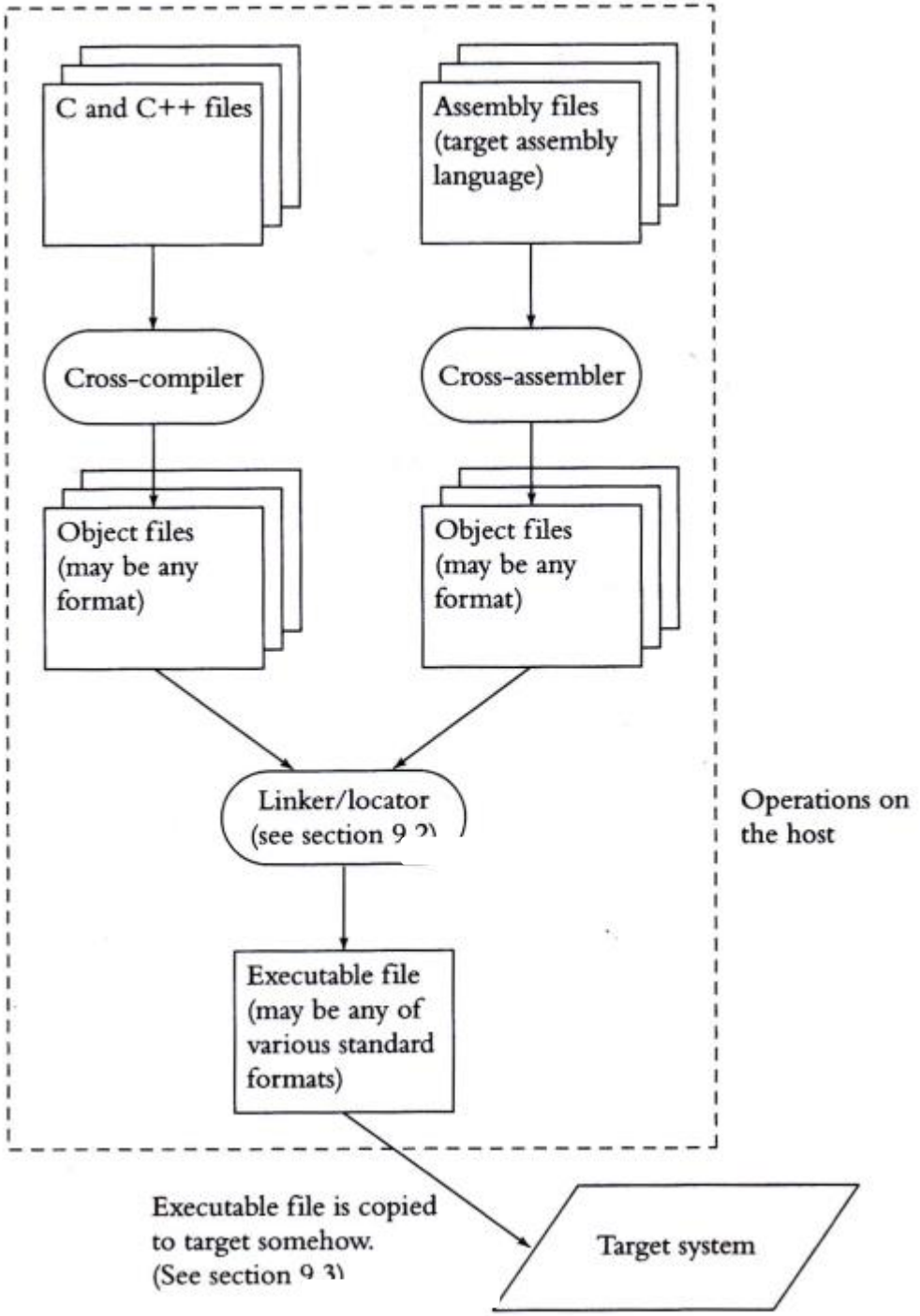
Cross Compiler

- Runs on host but generates code for target
  - Target usually have different architecture from host. Hence compiler on host has to produce binary instructions that will be understood by target



Development Process

- Cross Compiler
- Cross Assembler
- Linker
- Loader/Locator

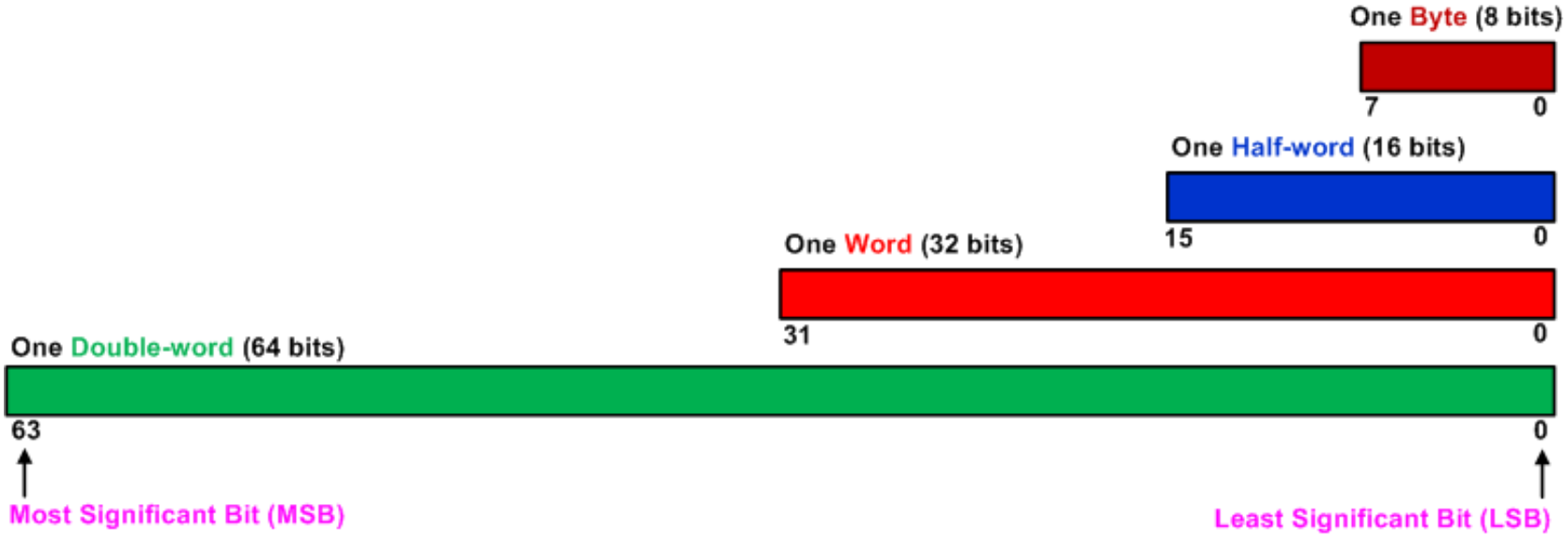


Outline

- Embedded systems
- ARM Microcontroller
- Development Environment
- **Course information**

Recall: Value notions

- For a 32 bit processor like the ARM
  - a word is equal to 32 bits or 4 bytes
  - a ‘half word’ is 16 bits or 2 bytes



Recall: Number System

Decimal	Binary	Octal	Hex
0	0000	00	0x0
1	0001	01	0x1
2	0010	02	0x2
3	0011	03	0x3
4	0100	04	0x4
5	0101	05	0x5
6	0110	06	0x6
7	0111	07	0x7
8	1000	010	0x8
9	1001	011	0x9
10	1010	012	0xA
11	1011	013	0xB
12	1100	014	0xC
13	1101	015	0xD
14	1110	016	0xE
15	1111	017	0xF

Recall: Number System

- Signed Integer Representation
- Three ways to represent signed binary integers:
  - Signed magnitude
    - Example: in a 5−bit system
    - $+7_{10} = 00111_2$
    - $-7_{10} = 10111_2$
  - One's complement
    - Example: in a 5−bit system
    - $+7_{10} = 00111_2$
    - $-7_{10} = 11000_2$ (bitwise NOT)
  - Two's complement (used in the modern computers)
    - Example: in a 5−bit system (
    - $+7_{10} = 00111_2$
    - $-7_{10} = 11001_2$ (One's complement plus one)

Recall: Characters

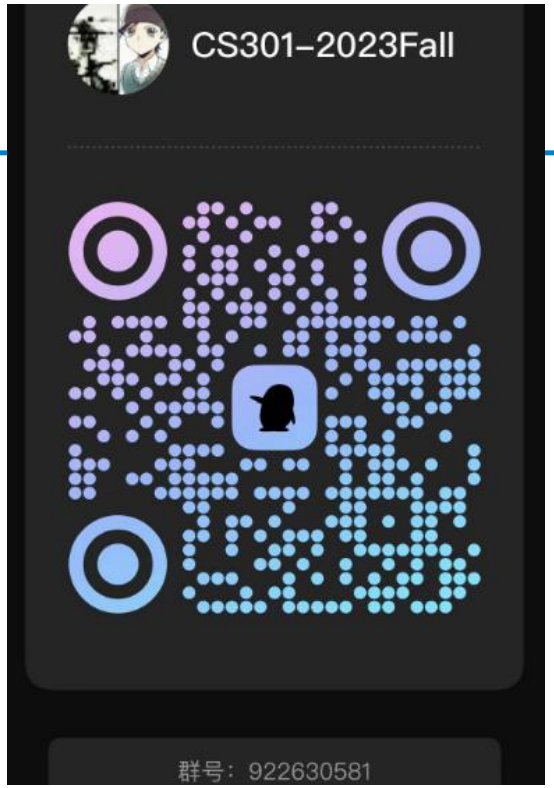
- American Standard Code for Information Interchange

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	NUL	32	20	SP	64	40	@	96	60	‘
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	’	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

About This Course

- Principles behind the design of the course:
  - Build the course around labs: Use labs to carry out the course contents. Labs are to develop a simple embedded system applications
  - Cover basic concepts in embedded system development: interrupt, clocking, I/O, real-time system, development tools
  - Term project development: innovation, development process, learning-by-doing

Course Information



- Course website: (Blackboard)
- Instructor:
  - Dr. Yuhui BAI（baiyh@sustech.edu.cn)
  - Office: 411 College of Engineering South
  - Office hour: Wednesday 14:00-16:00 (by appointment)
- QQ group
  - 922630581
- Lecture:
  - Tuesday 10:20-12:10, Room 102, Business Hall #3
- Lab:
  - Tuesday 14:00-15:50, Room 510, Lecture Hall #3 (WangQing)
  - Tuesday 16:20-18:10, Room 510, Lecture Hall #3

Prerequisites

- Be able to write C/Java programs
- Understand the steps in compiling and executing a program
- Digital Logic basics

Course contents

- By the end of the course you will:
- Learn at a low level how computers work
- Understand basic memory addressing, endianess, and data alignment
- Be able to understand basic ARM assembly
- Be able to program an STM32F103 board in C language
- Understand and be able to program interrupts
- Be able to program low-level GPIO and other hardware devices
- Learn how to interface various hardware components like displays, and communication interfaces (I2C, SPI, UART) with microcontrollers.

Schedule & Textbook

- Schedule
  - Refer to Blackboard->Syllabus
- Textbook
  - No Textbooks
  - Reference book:
    - Embedded Systems with ARM Cortex-M Microcontrollers, in Assembly Language and C (4th ed.), Yifeng Zhu
    - Embedded Systems : Architecture, Programming And Design(3rd ed.), Raj Kamal
    - The Definitive Guide to the ARM Cortex-M3 (2nd ed.), Joseph Yiu

Grading criteria

- Final Exam, 40%
  - Closed book
- Lab, 40%
  - Lab Attendance + Lab assignments 20%
  - Final Project 20%
- In-class Quiz, 20%
  - In-class assignments may be given during each lecture, and answers are submitted online or collected at the end of class.
  - Marking scheme for each in-class quiz:
    - Same mark as the actual mark if above 60;
    - 60, if 60 or below or you are absent with an accepted permission;
    - 0, if absence.

- Note: Do not do the assignments for other; second time, 0 point for the whole course; students. If caught, first time, 0 point for the In-class Assignments part