汇编语言与逆向技术

第1章 汇编语言基本概念

王志
zwang@nankai.edu.cn

南开大学 网络空间安全学院

2024-2025学年

投票 最多可选1项

设置

大家是否听说过汇编语言?

A 没有听说过汇编语言

B 听说过汇编语言，但是没有使用过

C 使用过汇编语言

D 打过逆向或者PWN的CTF比赛

提交

南開大學 Nankai University

大学四大“神”课

- **汇编语言**不会编
- 微机原理闹危机
- 随机过程随机过
- 实变函数学十遍

汇编语言与逆向技术

- 学分：2.5
- 教学：
  - 2024-2025学年第一学期（2-18周）
  - 星期五 10：00-11：40 ，津南**公教楼B**区423
- 实验：
  - 2024-2025学年第一学期（**4-18周**）
  - 星期五 14：00-15：40，津南**实验楼**A区205、210

汇编语言与逆向技术

- 授课教师：王志、邓琮弋
  - 王志，zwang@nankai.edu.cn
  - 邓琮弋，dengcongyi0701@163.com

课程教材和拓展阅读资料

•Intel汇编语言程序设计（第五版），Assembly Language for Intel-Based Computers （Fifth Edition），【美】Kip R. Irvine著，温玉杰、梅广宇、罗云彬等译，电子工业出版社；

•加密与解密（第四版），段钢 编著，电子工业出版社；

课程教材和拓展阅读资料

- Practical Reverse Engineering, Bruce Dang, Alexandre Gazet and Elias Bachaalany, Wiley;
- 逆向工程核心原理，【韩】李承远 著，武传海 译，人民邮电出版社；
- Practical Malware Analysis, Michael Sikorski and Andrew Honig, No Starch Press;
- IDA Pro 权威指南（第二版），【美】Chris Eagle 著，石华耀、段桂菊译，人民邮电出版社

考试成绩

- 平时成绩 25%
  - 考勤、课堂交互、课后讨论（雨课堂）
- 实验成绩 25%
  - 课后习题、实验报告（雨课堂）
- 期末考试 50%
  - 闭卷考试

课程目标

- **目标 1：** 了解处理器基本架构，能使用汇编语言在IA32和ARM处理器上开发程序；
- **目标 2：** 了解Windows 二进制可执行程序文件结构，定位和提取程序的入口地址、导入表、导出表、节表等基本信息，推演出程序使用的变量、数组、结构体、控制逻辑、函数等高级语言信息；
- **目标 3：** 了解反汇编、静态分析、动态分析等逆向技术原理，应用典型工具实现深入的软件逆向分析；
- **目标 4：** 了解软件保护技术的基本原理

课程目标

- 学习课程 ≠ 掌握技术 ≠ 成为专家
- 引导激发 -> 自我探索 + 练习 + 坚持
- 全国大学生信息安全创新实践能力赛
- 天津市大学生信息安全竞赛（京津冀大学生信息安全网络攻防大赛）
- 校内国创、市创、百项
- 进入实验室，参与科研项目

课程内容

- 汇编语言
  - Intel处理器X86汇编、华为鲲鹏处理器ARM汇编
- 逆向分析
  - 静态逆向分析
  - 动态逆向分析
- Windows
  - 可执行文件结构
- 软件保护

课程微信群

群聊：2024《汇编语言与逆向技术》

第1章 汇编语言基本概念

问题1：收获

- **通过本章节学习我们能学到什么？**
- 答: 本课程将展示汇编语言的基础知识，包括语法、指令集、内存管理和程序结构。还将学习如何使用汇编器和链接器，以及如何调试和优化代码。
- 本课程将介绍一些实际应用案例，帮助理解汇编语言的应用场景和重要性。

问题2：环境配置

- **学习汇编语言需要什么硬件和软件环境？**
- 答: Intel Core或AMD 处理器的Windows系统

记事本（Notepad.exe)

MASM32 SDK https://www.masm32.com/


```
.386
.model flat, stdcall
option casemap :none
include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
include \masm32\include\masm32.inc
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\masm32.lib

.data
    str_hello BYTE "Hello World!", 0

.code
start:
    invoke StdOut, addr str_hello
    invoke ExitProcess, 0
END start
```

问题3：工具

- **什么是汇编器和链接器？**
- 答: 汇编器将汇编语言代码转换为机器语言代码。

链接器则是将一个或多个由汇编器生成的对象文件合并成单个可执行文件的工具。

链接器处理符号解析、重定位和内存分配。

主观题 10分

设置

为什么不直接把汇编语言编译成可执行的exe程序，取消中间的链接过程?

作答

问题4：区别

•汇编语言与高级编程语言有什么区别？

•答: 汇编语言是低级编程语言，直接对应计算机的机器指令，高级编程语言提供了更多的语义解释和功能封装。

汇编语言使用助记符代表具体的机器指令，使其在语法上比0和1的二进制代码更易于理解。高级语言隐藏了硬件的复杂性，牺牲了性能和对底层硬件的控制。

学习汇编语言能让程序员更直接地控制硬件资源，实现高效、精确的编程，尤其在关注性能或指定硬件的应用中尤为重要。

主观题 10分

设置

大家都学习或者听说过哪些计算机编程的高级语言？

作答

问题5：联系

•高级编程语言与汇编语言有什么关系？

•答：高级语言代码在执行前需要转换成更接近硬件的汇编语言或机器代码。

假设以下C++代码段计算两个整数的和：

```
int a = 5;
int b = 10;
int sum = a + b;
```

在执行前将被编译器转换成汇编语言。

```
mov eax, 5     ; 将5赋值给EAX寄存器
mov ebx, 10    ; 将10赋值给EBX寄存器
add eax, ebx   ; 将EAX和EBX寄存器的值相加，结果存储在EAX中
mov sum, eax   ; 将EAX寄存器的值赋给sum变量
```

寄存器是CPU用于临时存储计算数据的小型存储区域， eax 和 ebx 是两个常用的寄存器。

高级编程语言与汇编语言的转换，使程序员可以编写更抽象、易于理解和维护的代码，同时由编译器负责处理底层的硬件交互。

问题6：限制

•汇编语言是可移植的吗？

•答: 汇编语言是依赖于特定硬件架构的。

为一个特定类型的处理器编写的汇编代码通常无法在不同类型的处理器上直接运行

硬件依赖性限制了汇编语言代码的可移植性。

问题7：应用

- **汇编语言在现代编程中的应用是什么？**
- 答: 尽管高级语言在许多应用中更常见，但汇编语言在性能攸关任务、系统底层编程、硬件操作中仍然有其独特的应用。例如，操作系统的核心部分、高性能游戏和图形处理程序，以及嵌入式系统通常会用到汇编语言。

问题8：交互

- **汇编语言如何与硬件交互？**
- 答: 汇编语言通过使用特定于处理器的指令直接与硬件交互

指令可以控制处理器的寄存器、执行算术和逻辑运算、管理内存访问等。

指令使得汇编语言在精细化硬件控制场景中十分有效

问题9：提升

- **汇编语言如何帮助理解高级编程语言？**
- 答: 汇编语言更好地理解高级语言的底层机制，例如内存管理、指针等概念，以及编译器是如何将高级代码转换为机器代码的，从而帮助开发者编写更高效、更优化的高级语言代码。

汇编语言

单选题 10分

设置

汇编、C++、Java、Python语言出现时间排序

A C++ <Assembly< Java < Python

B Java < C++ <Assembly < Python

C Python <Assembly < C++ < Java

D Assembly <C++ < Python < Java

提交

什么是汇编语言（Assembly Language）

- 汇编语言是所有程序设计语言中最古老的语言（Born in 1949）
  - 与机器语言最为接近
  - 可以直接访问硬件
  - 需要了解计算机体系结构和操作系统

BASIC, designed in 1964, wa
product ever made by Micros
early years due to how powe

In addition to that, the 1970s

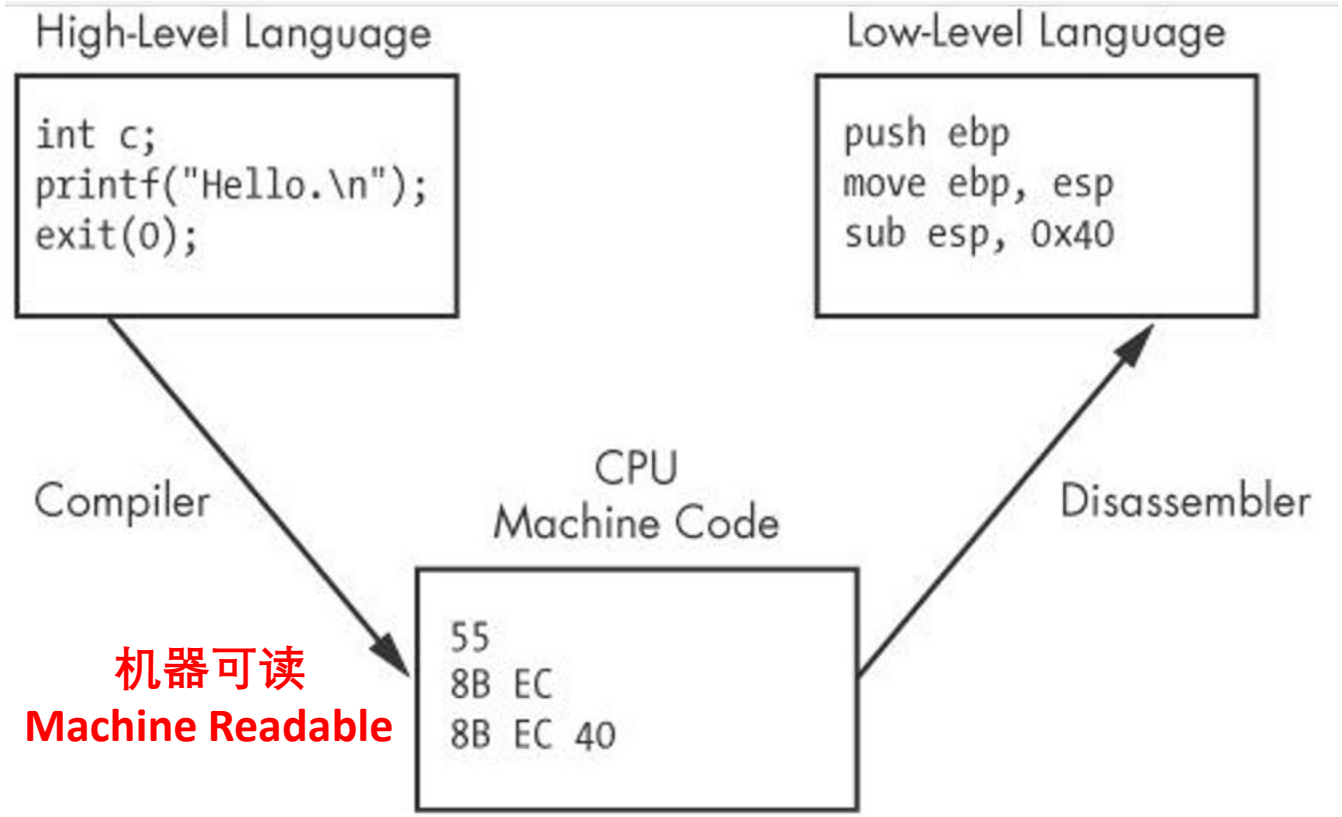
- **Smalltalk (1972)**, which e
  introduced things that are
- **C (1972)** was the very first
  Unix to be used on a broa
  coding languages today.
- **SQL (1972)** revolutionized
  queries.
- **MATLAB (1978)** remains
  primarily used in research

编译与反汇编

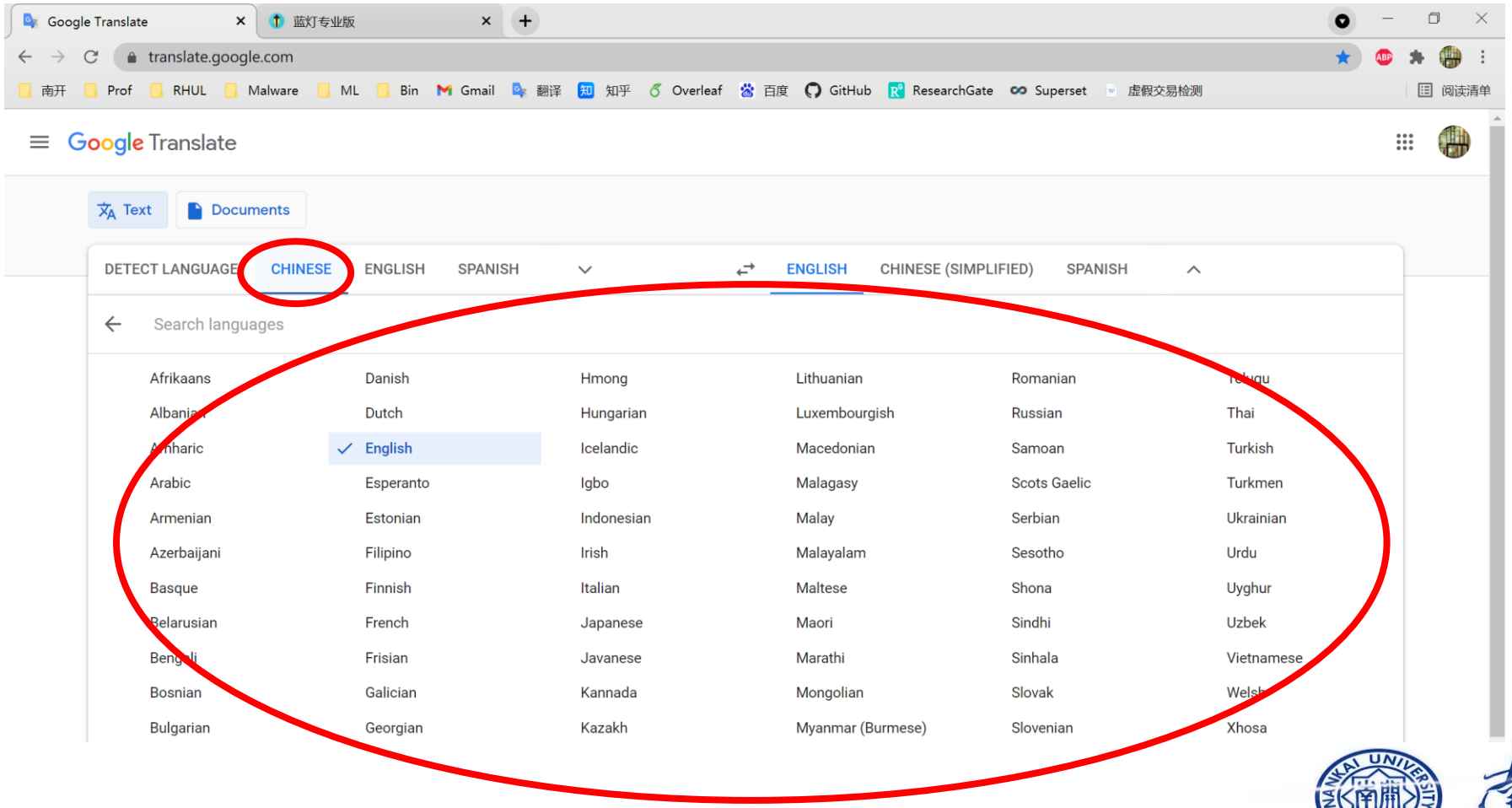
人可读
Human Readable

翻译
Compile

机器可读
Machine Readable



Google翻译


Google Translate
蓝灯专业版
translate.google.com
南开 Prof RHUL Malware ML Bin Gmail 翻译 知乎 Overleaf 百度 GitHub ResearchGate Superset 虚假交易检测 阅读清单
Google Translate
Text
Documents
DETECT LANGUAGE
CHINESE
ENGLISH
SPANISH
ENGLISH
CHINESE (SIMPLIFIED)
SPANISH
Search languages
Afrikaans
Albanian
Amharic
Arabic
Armenian
Azerbaijani
Basque
Belarusian
Bengali
Bosnian
Bulgarian
Danish
Dutch
English
Esperanto
Estonian
Filipino
Finnish
French
Frisian
Galician
Georgian
Hmong
Hungarian
Icelandic
Igbo
Indonesian
Irish
Italian
Japanese
Javanese
Kannada
Kazakh
Lithuanian
Luxembourgish
Macedonian
Malagasy
Malay
Malayalam
Maltese
Maori
Marathi
Mongolian
Myanmar (Burmese)
Romanian
Russian
Samoan
Scots Gaelic
Serbian
Sesotho
Shona
Sindhi
Sinhala
Slovak
Slovenian
Thai
Turkish
Turkmen
Ukrainian
Urdu
Uyghur
Uzbek
Vietnamese
Welsh
Xhosa

什么是汇编语言

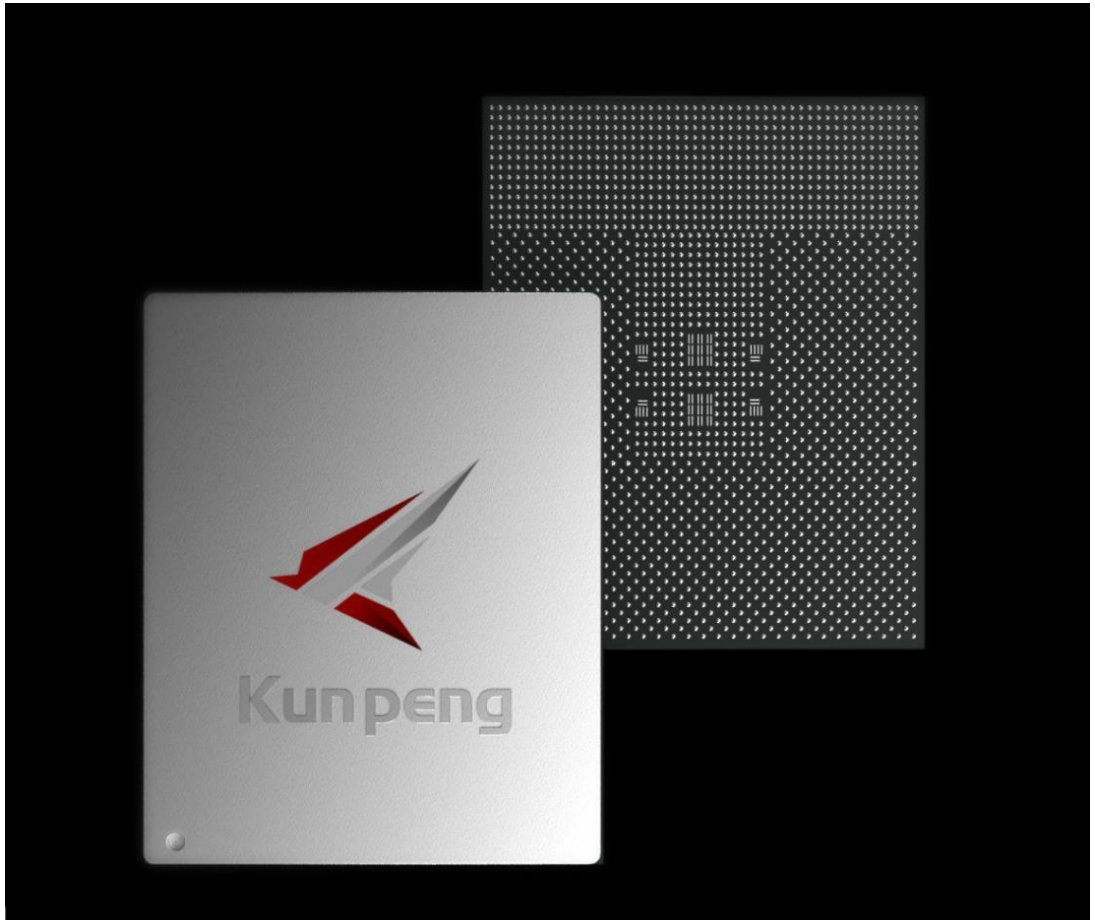
- 汇编语言也称为符号语言
  - 用助记符代替机器指令的操作码
    - 机器指令 55，对应的汇编指令是push ebp
  - 用地址符号或标号代替指令或操作数的地址
    - 例如，将一个数据从内存读到CPU的寄存器中

汇编语言

- CPU
  - 寄存器
  - 传送指令
  - 算数指令
  - 位运算指令
  - 串操作指令
  - 跳转指令
- 内存
  - 寻址方式

基于ARMv8架构的鲲鹏处理器

- ARM寻址方式
- ARM指令集
- ARM伪指令
- ARM汇编语言程序结构
- ARM编译与调试工具



主观题 10分

设置

X86和ARM架构的区别?

正常使用主观题需2.0以上版本雨课堂

作答

南開大學 Nankai University

为什么学习汇编语言

？

现状

- 程序语言现状
  - （机器语言）代码冗长，晦涩难懂
  - （高级语言）执行效率低，难以逆向分析和修改

需求

- 高速度、高效率
  - 直接对应机器语言，代码体积小，程序更优化，直接的硬件操作加深系统、硬件理解
  - 向上软件系统
  - 向下硬件系统
- 软件程序分析
  - 计算机病毒分析、协议分析、漏洞发掘
  - 软件盗版、破解、代码盗用
  - 数据挖掘与计算机取证

案例：XcodeGhost的病毒

- Xcode 编译器是开发Mac OS 和iOS 应用程序的编译器
- Xcode编译器官方下载服务器在国外，中国下载速度非常慢
- 第三方的Xcode编译器有病毒，编译的时候把恶意代码输入到程序里面
- 导致其编译出来的App都带有后门代码，会在最终客户端运行时将隐私信息提交给第三方
- 根据盘古团队2018年数据显示，已检测到超过800个不同版本的应用感染了XcodeGhost病毒

案例:XcodeGhost的病毒

部分受病毒影响的 APP 及版本

相关 APP	版本
滴滴打车	3.9.7
同花顺	9.26.03
中国联通网上营业厅	3.2
中信银行动卡空间	3.3.12
微信 IOS	6.2.5
网易公开课	4.2.8
愤怒的小鸟 2	2.1.1
炒股公开课	3.10.02-3.10.01
股票雷达	5.6.1
南京银行	3.6-3.0.4
南方航空	2.6.5.0730-2.6.5

1 病毒如何植入?

2 如何分析病毒?

为什么要学习汇编语言

- 软件优化（optimization）
- 软件自动修复（automated repair）
  - 软件补丁（patch），不用重新编译
- 软件插装（instrumentation）
  - 虚拟化技术
- 软件知识产权保护（software hardening）
  - 水印、指纹、混淆、版权保护
- 软件调试（debugging）

为什么学习汇编语言

- 汇编语言和逆向工程是**系统开发、系统安全**的基础课程
  - 系统内核开发、硬件驱动开发
  - 软件安全分析
  - 计算机病毒分析
  - 网络渗透与入侵检测
  - 漏洞分析

汇编语言缺点

- 学习难度：需要了解硬件和系统的知识
- 可移植性：CPU架构和指令集的差异，移植性较弱
- 复杂性：程序较长，结构较松散

程序员较少选择汇编直接编程

主观题 10分

设置

有哪些场景需要使用汇编语言?

作答

计算机编程语言

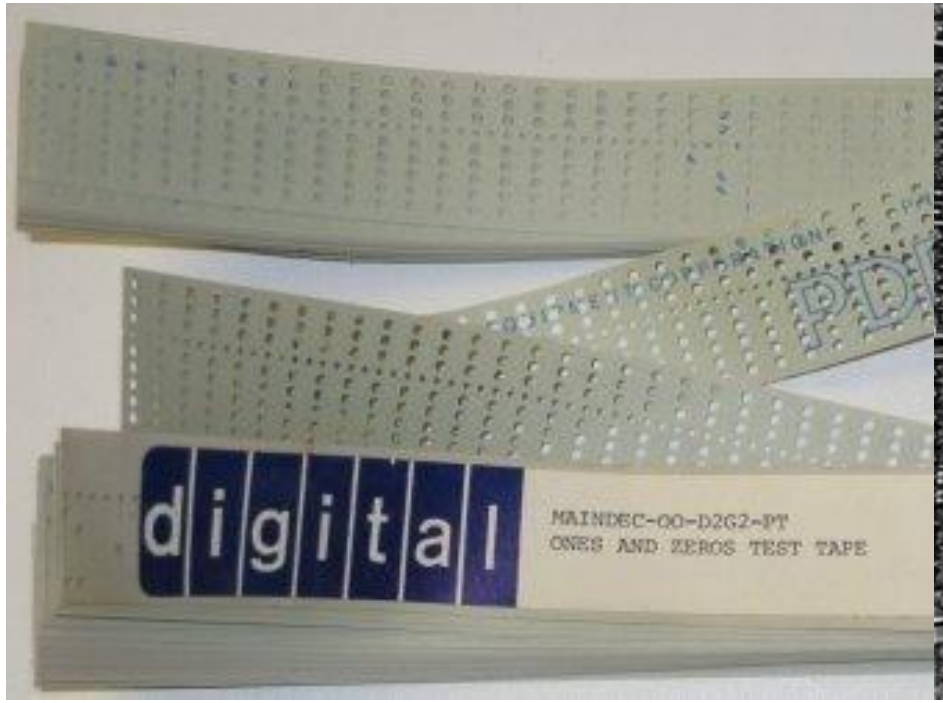
计算机编程语言

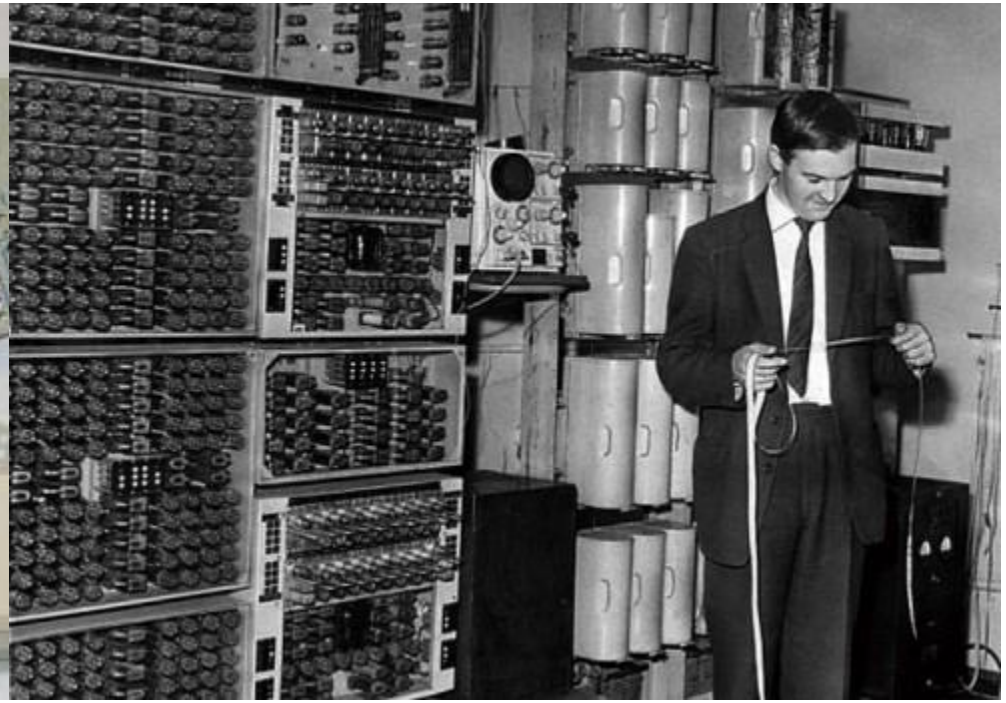
- 机器语言
- 汇编语言
- 高级语言

计算机编程语言

- 1946年第一台计算机问世
- 人机交互成为研究方向
- 更高效、更简便
  - 二进制机器语言
  - 助记符汇编语言（1949年）
  - 可以跨平台编译和执行的高级语言

机器语言





汇编语言

- **助记符**代替机器语言中的二进制**操作码**
- **地址符号**或**标号**，代替指令或操作数的二进制**地址**

```
.text:00401000
.text:00401000 68 00 30 40 00      start   proc near
.text:00401005 FF 15 08 20 40 00           push    offset Format   ; "Please enter a challenge: "
.text:0040100B 83 C4 04                    call    ds:printf
.text:0040100E 68 B1 30 40 00              add     esp, 4
.text:00401013 68 1B 30 40 00              push    offset Str
.text:00401018 FF 15 00 20 40 00           push    offset aS       ; "%s"
.text:0040101E 83 C4 08                    call    ds:scanf
.text:00401021 68 B1 30 40 00              add     esp, 8
.text:00401026 FF 15 04 20 40 00           push    offset Str      ; Str
.text:0040102C 83 C4 04                    call    ds:strlen
.text:0040102F 83 F8 06                    add     esp, 4
.text:00401032 0F 82 D5 00 00 00           cmp     eax, 6
.text:00401038 68 4E 30 40 00              jb      loc_40110D
.text:0040103D FF 15 08 20 40 00           push    offset aPleaseEnterThe ; "Please enter the solution: "
.text:00401043 83 C4 04                    call    ds:printf
.text:00401046 68 AD 30 40 00              add     esp, 4
.text:0040104B 68 A9 30 40 00              push    offset dword_4030AD
.text:00401050 68 A5 30 40 00              push    offset dword_4030A9
.text:00401055 68 A1 30 40 00              push    offset dword_4030A5
.text:0040105A 68 6A 30 40 00              push    offset dword_4030A1
.text:0040105F FF 15 00 20 40 00           push    offset aUUUU    ; "%u-%u-%u-%u"
.text:00401065 83 C4 14                    call    ds:scanf
.text:00401068 83 F8 04                    add     esp, 14h
.text:0040106B 0F 82 AC 00 00 00           cmp     eax, 4
                                           jb      loc_40111D
```

高级语言

- 高级语言更接近数学语言或者自然语言
  - 偏数学语言：Pascal、Ocaml、R
    - 函数式编程、多范式编程、基于规则的编程
  - 偏自然语言：C++、Java、Python
    - 面向对象（类、继承、多态）、面向过程

高级语言

- 高级语言与汇编语言和机器语言之间是一对多的关系。
  - 一条高级语言指令，编译之后，对应着多条机器码

```
int printf(const char *format, ...)
```

```
int scanf(const char *format, ...)
```

```
push    offset Format   ; "Please enter a chall
call    ds:printf
add     esp, 4
push    offset Str
push    offset aS       ; "%s"
call    ds:scanf
add     esp, 8
```

可移植

•如果一种语言的程序源代码可以在多种计算机系统上编译并运行，那么就说这种语言就是可移植的。

```
; Format      : Portable executable for 80386 (PE)
; Imagebase   : 400000
; Timestamp   : 4EA893FB (Wed Oct 26 23:12:59 2011)
; Section 1. (virtual address 00001000)
; Virtual size                    : 00000145 (    325.)
; Section size in file            : 00000200 (    512.)
; Offset to raw data for section: 00000400
; Flags 60000020: Text Executable Readable
; Alignment     : default

                .686p
                .mmx
                .model flat
```

多选题 10分

设置

下面哪种编程语言是可移植的?

A 机器语言

B 汇编语言

C Python

D Java

E C++

提交

投票 最多可选1项

设置

C++中是否可以嵌入汇编语言?

A 可以嵌入汇编语言

B 不可以嵌入汇编语言

提交

南開大學 Nankai University

可移植

- C++中可以使用汇编语言
  - 使用高级结构和访问底层细节之间提供了一种**折中方案**
  - 直接访问CPU，会使C++程序**丧失**可移植性

C++

```
#include <stdio.h>

int main() {
    /* Add 10 and 20 and store result into register %eax */
    __asm__ ( "movl $10, %eax;"
              "movl $20, %ebx;"
              "addl %ebx, %eax;"
    );

    /* Subtract 20 from 10 and store result into register %eax */
    __asm__ ( "movl $10, %eax;"
                    "movl $20, %ebx;"
                    "subl %ebx, %eax;"
    );

    /* Multiply 10 and 20 and store result into register %eax */
    __asm__ ( "movl $10, %eax;"
                    "movl $20, %ebx;"
                    "imull %ebx, %eax;"
    );

    return 0 ;
}
```

汇编语言 VS 高级语言

特性	汇编语言	高级语言
接近硬件	是（直接操作硬件指令）	否（抽象层次较高）
可读性	较低（使用助记符）	较高（使用英语单词和语法结构）
学习难度	高（需要硬件知识）	较低（更直观易懂）
性能	高（直接硬件控制）	取决于语言和编译器优化
可移植性	低（依赖具体硬件架构）	高（通常不依赖硬件）
应用领域	系统编程、嵌入式系统等	一般应用程序开发、企业级应用等
维护和扩展性	困难（代码复杂，可读性差）	较容易（结构化、模块化编程）

主观题 10分

设置

有哪些场景需要用到汇编语言?

正常使用主观题需2.0以上版本雨课堂

作答

嵌入式系统

- 嵌入式系统的内存空间小
  - 电话、汽车、空调、打印机、摄像头、显卡、声卡、调制解调器等等
  - 汇编语言可以节省内存空间

实时系统

- 仿真、监控等实时系统要求精确计量时间和实时响应
  - 高级语言不能完全控制编译器生成的机器码
  - 汇编语言可以完全控制机器码，执行速度快

游戏机

•游戏机有专用的系统，要求程序在**大小**和**运行速度**两方面都要做高度优化。

- •充分利用目标系统的专用硬件特性
- •手动进行游戏速度优化

驱动程序

- 硬件设备需要驱动程序
  - 驱动程序把操作系统上通用的命令转换为对特殊硬件的具体细节操作的程序。
  - 打印机需要编写Windows、macOS、Linux等平台的打印驱动。

加密算法

- 高级语言的种种限制会阻碍位操作、数据加密等底层操作的有效实现
- 汇编语言会加快数据加密速度

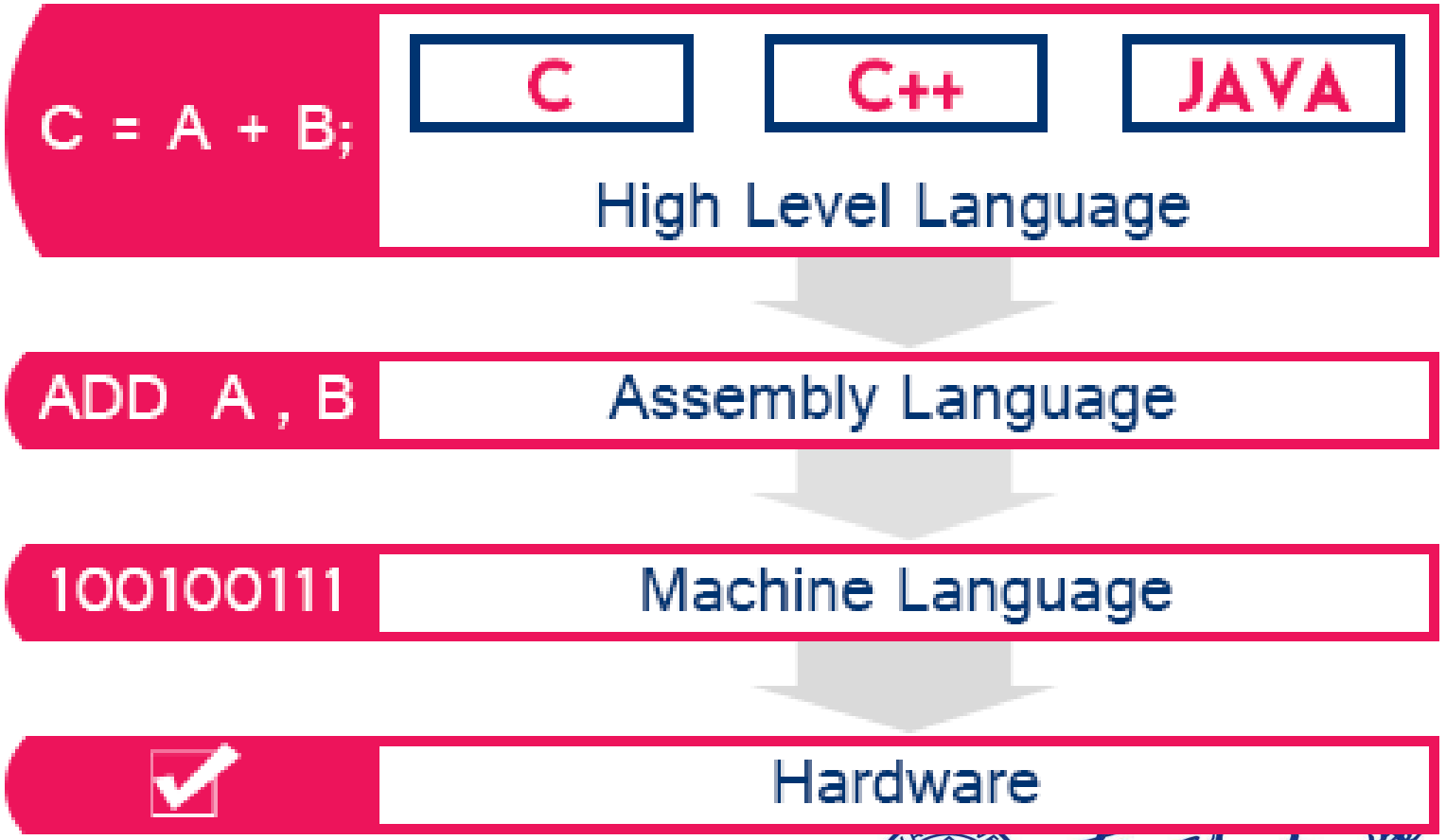
逆向分析

- 软件调试
- 软件漏洞挖掘
- 计算机病毒分析
- 软件知识产权保护

虚拟机的概念

虚拟机的概念

- 虚拟机
  - 对于每个语言层次，将其想象成一台假想的计算机（虚拟机）。
- 高层虚拟机的程序，通过解释或者翻译的方式，在底层虚拟机上执行

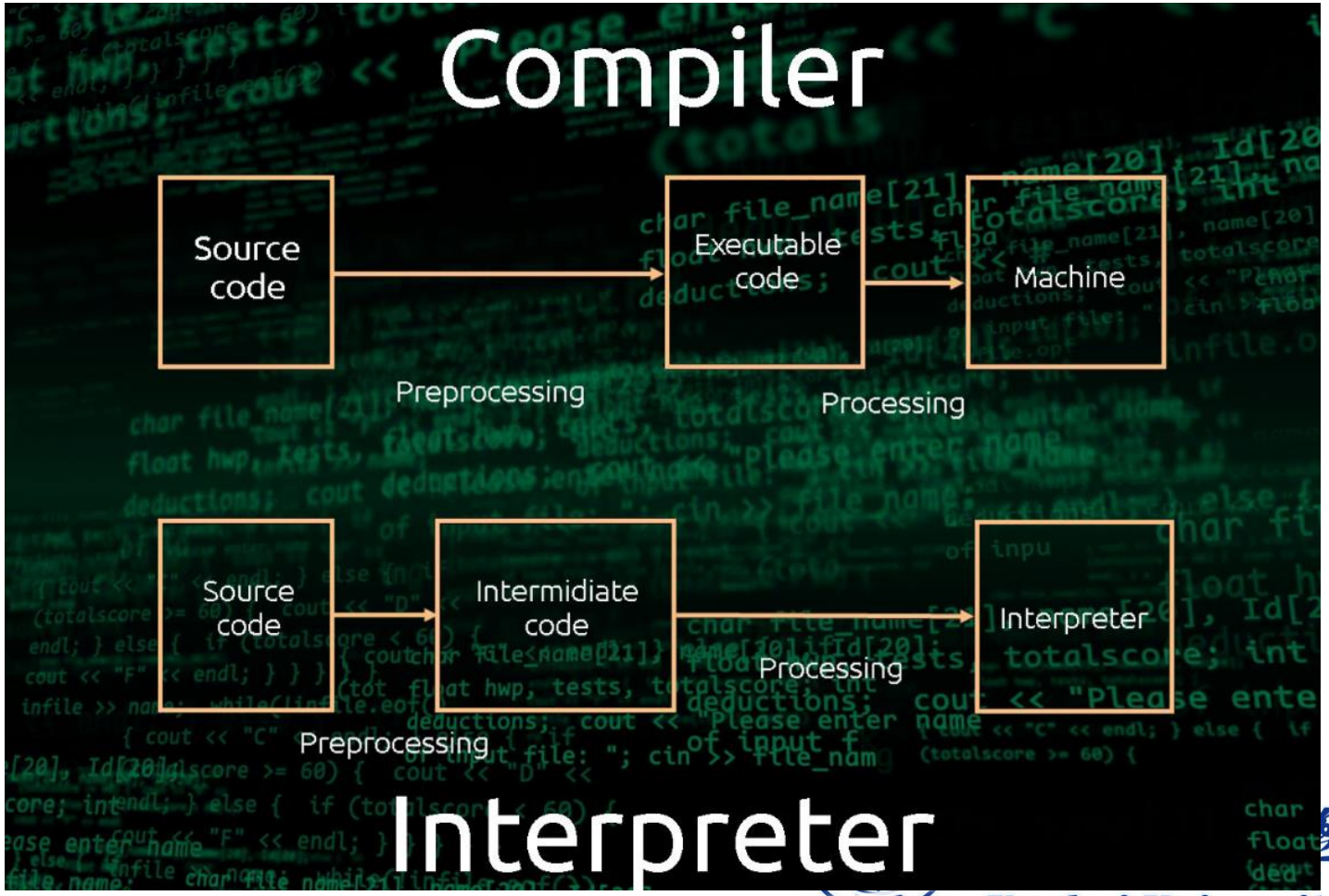


虚拟机的层次

- 第5层：高级语言 (功能强大)
- 第4层：汇编语言 (助记符、标签)
- 第3层：操作系统（定义交互命令的虚拟机）
- 第2层：指令集体系结构 （固化在处理器内部的指令集）
- 第1层：微结构（芯片上特殊的微结构指令）
- 第0层：数字逻辑

执行方式

- 翻译方式
  - 高层虚拟机的程序被整体翻译成底层虚拟机程序，然后在底层虚拟机上执行；
- 解释方式
  - 低层虚拟机对高层虚拟机的程序，逐条指令进行解码并执行；



单选题 10分

设置

下面哪种语言的程序是解释执行的?

A C++

B C

C Python

D 汇编语言

提交

数据的表示方法

数据的表示方法

- 十进制（Decimal）
- 二进制（Binary）
- 八进制（Octet）
- 十六进制（**Hex**adecimal）

Decimal	Binary	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

填空题 2分

设置

19的二进制表示是[填空1] ？

作答

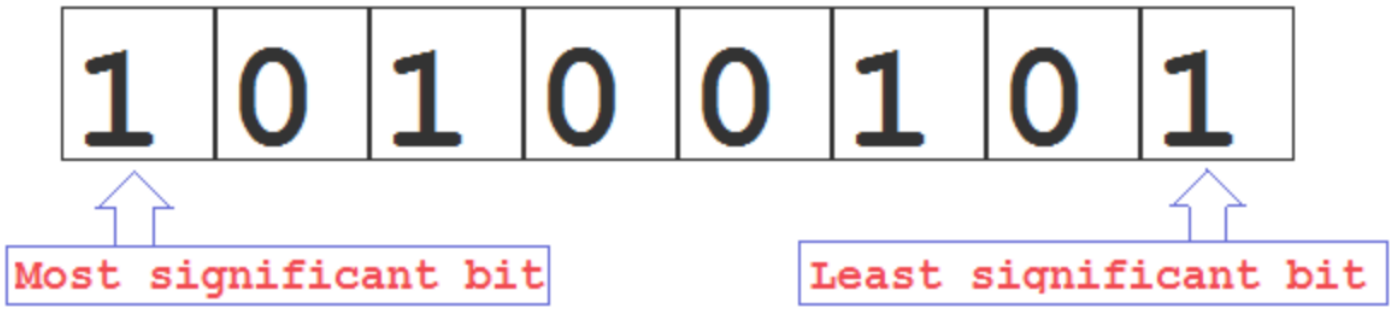
南開大學 Nankai University

二进制数

- 二进制数字以2为基数
- 每个二进制数字是一个比特位（bit）
- 位数从最右边的第0位开始计算，向左依次次递增

二进制数

- 最左边的位称为最高有效位（**MSB，** Most Significant Bit）
- 最右边的位称为最低有效位（**LSB，** Least Significant Bit)



单选题 1分

设置

01001101的最低有效位LSB是?

A 1

B 0

C 2

D 3

提交

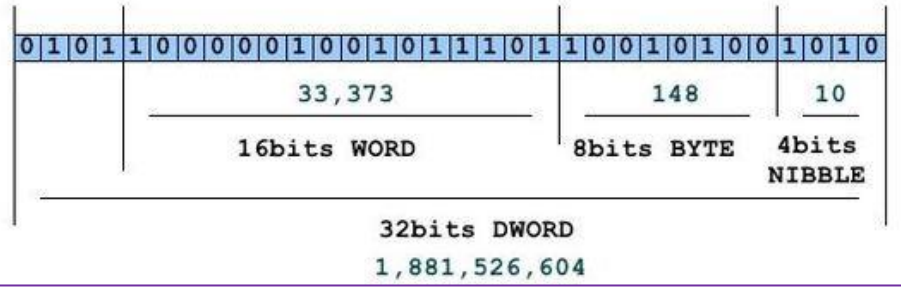
南開大學 Nankai University

数据的存储单位

- 字节（byte）：包含8个bit位
- 字（word）：包含两个字节
- 双字（doubleword）：包含两个字
- 八字节（quadword）：包含两个双字

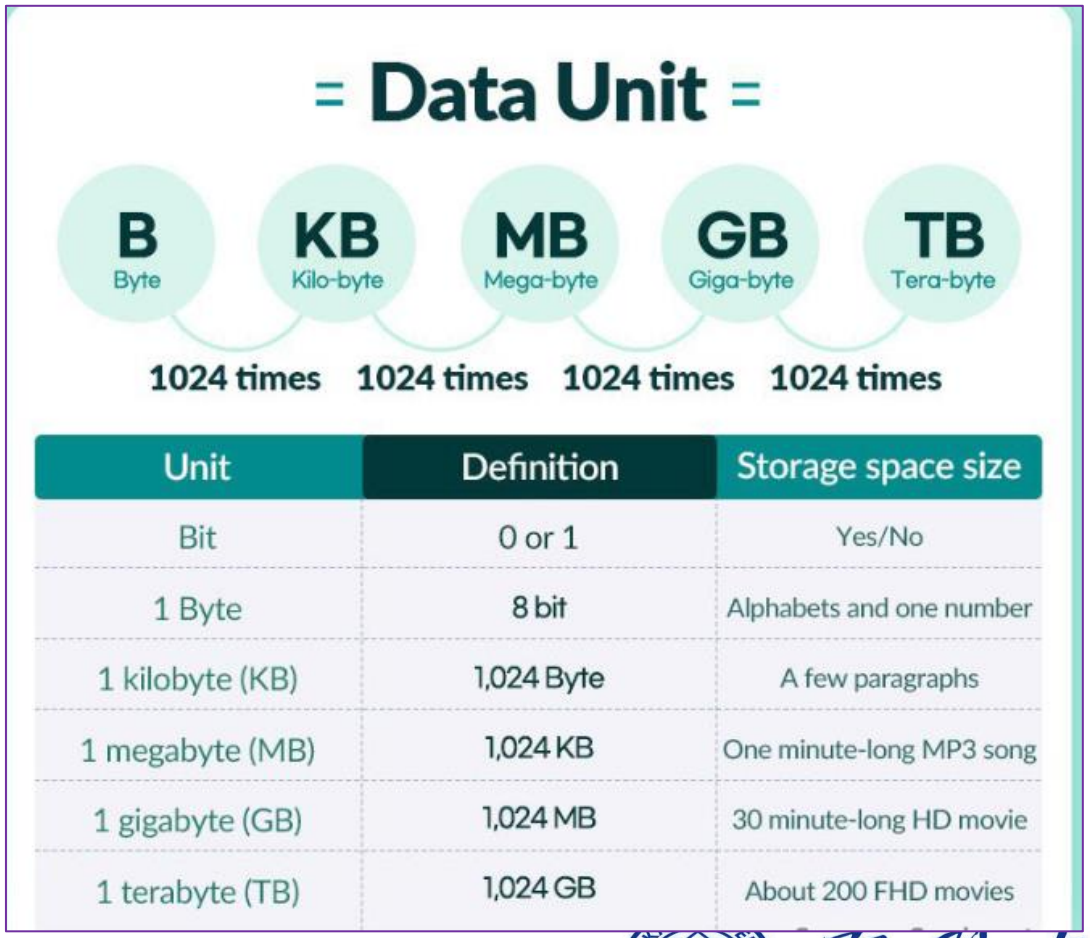
Data Representations

Bit:	1 bit (0/1)
Nibble:	4 bits (0-15)
Byte:	8 bits (0-255)
Word:	16 bits (0-65535)
Double Word(DWORD):	32 bits (0-4294967295)
Quad Word(QWORD):	64 bits (0-18446744073709551615)



数据的存储单位

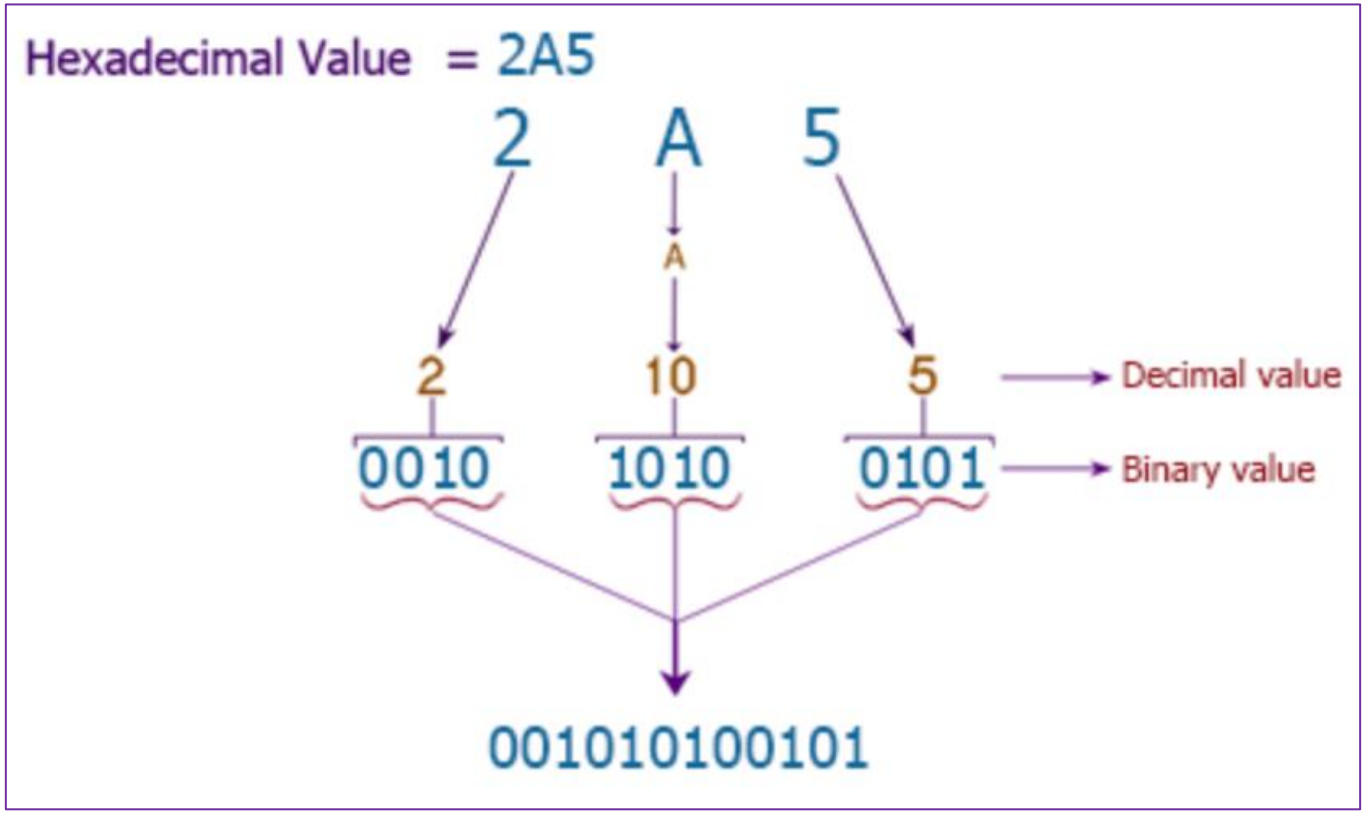
- 1 kB = 1024 byte
- 1 MB = 1024 kB
- 1 GB = 1024 MB
- 1 TB = 1024 GB
- 1 PB = 1024 TB



Unit	Definition	Storage space size
Bit	0 or 1	Yes/No
1 Byte	8 bit	Alphabets and one number
1 kilobyte (KB)	1,024 Byte	A few paragraphs
1 megabyte (MB)	1,024 KB	One minute-long MP3 song
1 gigabyte (GB)	1,024 MB	30 minute-long HD movie
1 terabyte (TB)	1,024 GB	About 200 FHD movies

十六进制

- 二进制数字的阅读不方便，十六进制使用更加方便
- 十六进制的每个数据可以表示4个二进制位
- 两个十六进制位就可以表示1个字节



主观题 10分

设置

01001101的十六进制表示是?

正常使用主观题需2.0以上版本雨课堂

作答

南開大學 Nankai University

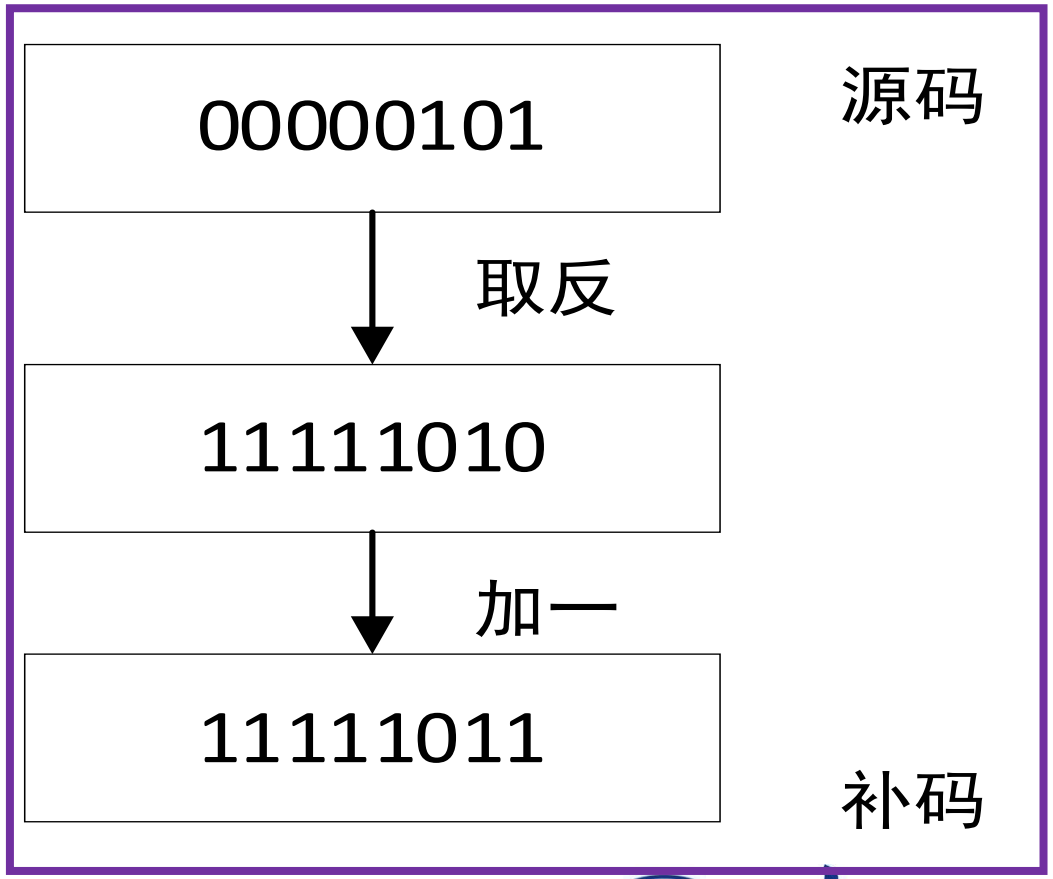
有符号整数

- 有符号整数在计算机科学中既可以表示正数也可表示负数。
- 与无符号整数不同，有符号整数的存储结构专门留出一个位来表示数值的符号，通常是最高位（**MSB**）。
  - 如果最高位是0，则表示该数是正数；
  - 如果是1，则表示为负数。

补码表示法

•补码表示法是计算机中表示负整数的常用方法。补码（two's complement）的定义可以用数学术语来描述：

•一个整数的补码是其相反数的表示，即一个数与其补码相加的和为零。



十六进制数补码

•考虑十六进制数2F4B，要计算它的补码，我们首先将每个数字位取反。在十六进制中，取反可以通过用F（即15）减去每一位实现。因此：

- •2（即0010）取反后变为D（即1101）
- •F（即1111）保持不变，因为它已经是最大值
- •4（即0100）取反后变为B（即1011）
- •B（即1011）取反后变为4（即0100）
- •2F4B的每位取反后得到D0B4。对这个结果加1，即D0B4 + 1，得到D0B5。因此，2F4B的补码是D0B5。

有符号二进制数到十进制数的转换

- 如果MSB是0，表示该数是正数，转换过程与无符号二进制数相同。
- 如果MSB是1，表示该数是负数，转换过程需要考虑补码表示法。
- **正数的转换**
  - 直接将二进制数转换为十进制数。
  - 例如，二进制数 0101（MSB为0）可以直接转换为十进制数5。
- **负数的转换**
  - 首先计算其补码，然后再转换为十进制。
  - 补码计算方法是先对该二进制数除符号位外的其余位取反，然后加1。
  - 例如，二进制数 1101（MSB为1，表示是负数），先取反得到 0010，再加1得到 0011，即十进制的3。由于原二进制数是负数，所以最终结果是-3。

有符号十进制数到二进制数的转换

•转换正数

•对于正的十进制数，转换过程与将无符号十进制数转换为二进制数相同。简单地将十进制数转换为其直接的二进制等价形式即可。例如，十进制数5转换为二进制是 0101。

有符号十进制数到二进制数的转换

- **转换负数**
  - **1. 取绝对值并转换为二进制：**首先忽略负号，将数的绝对值转换为二进制。例如，对于-5，首先将5转换为二进制得到0101。
  - **2. 求补码：**接着求这个二进制数的补码。这包括将除符号位外的所有位取反（0变为1，1变为0），然后在结果上加1。
  - 在上述例子中，0101取反得到1010，再加1得到1011。因此，-5在二进制补码表示中为1011。

有符号十进制数到十六进制数的转换

- **1. 确定符号：**首先判断十进制数是正数还是负数。对于正数，直接转换为十六进制即可；对于负数，则需要先转换为补码形式。
- **2. 转换正数：**对于正的十进制数，将其直接转换为十六进制的形式。例如，十进制数27转换为十六进制是1B。
- **3. 转换负数：**
  - 转换为二进制：首先将十进制数的绝对值转换为二进制形式。
  - 计算补码：接着对这个二进制数求补码。这包括取反所有位（除了符号位），然后加1。
  - 转换为十六进制：将得到的二进制补码转换为十六进制形式。

有符号十进制数到十六进制数的转换

- 以-30为例，其转换过程如下：

  30 的二进制表示为11110。

  取反（除了符号位）得到 00001，并加1得到 00010。

  将00010转换为十六进制，得到02。因此，-30 的十六进制补码表示为 02。

有符号十六进制数到十进制数的转换

- **1. 确定符号位：**首先识别十六进制数的最高位（MSB）。在固定长度的十六进制数中，MSB作为符号位，0表示正数，1表示负数。
- **2. 转换正数：**如果十六进制数是正数（MSB为0），直接将其转换为十进制数即可。
- **3. 转换负数：**
  - **求补码：**对于负数（MSB为1），首先需要将十六进制数转换为二进制补码。
  - **转换为十进制：**然后将二进制补码转换为对应的十进制数。这通常涉及补码的逆运算——先减1，然后取反，得到原数的二进制表示，最后将此二进制数转换为十进制，并添加负号。

n位有符号整数所能表示的最大值和最小值

- 有符号字节（8位）：最大值为$2^{7}-1=127$，最小值为$-2^{7}=-128$。
- 有符号节（16位）：最大值为$2^{15}-1=32767$，最小值为$-2^{15}=-32768$。
- 有符号双字（32位）：最大值为$2^{31}-1=2147483647$，最小值为$-2^{31}=-2147483648$。
- 有符号八字节（64位）：最大值为$2^{63}-1$，最小值为-2^{63}。

字符集

主观题 10分

设置

计算机只能存储二进制数据，那么如何表示字符呢？

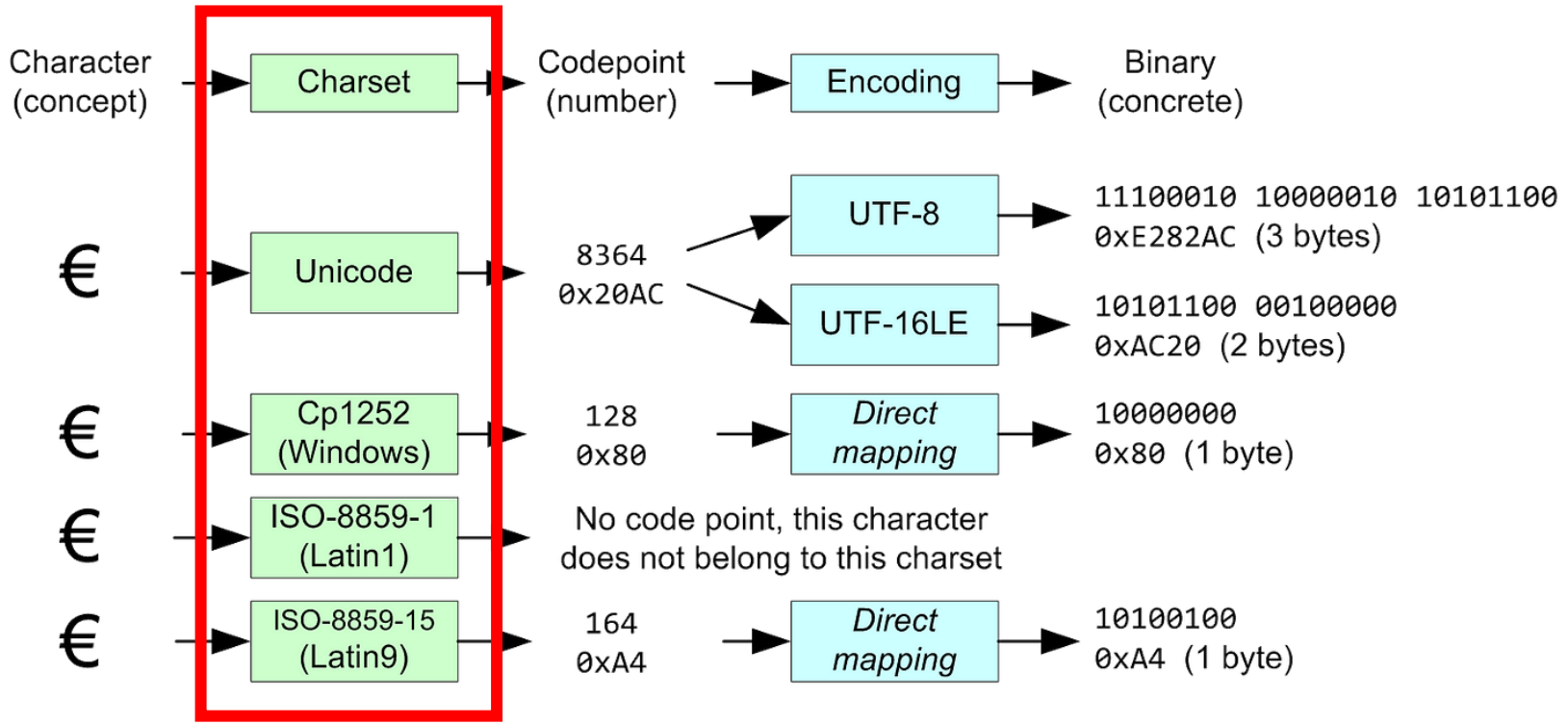
正常使用主观题需2.0以上版本雨课堂

作答

南開大學 Nankai University

字符集

- 字符集是一个系统支持的所有抽象字符的集和



多选题 10分

设置

下列哪些是字符集?

A ASCII

B Unicode

C UTF-8

D GB2312

提交

南開大學 Nankai University

ASCII字符集

- American Standard Code for Information Interchange
- 美国信息交换标准码
  - ASCII出现于20世纪50年代后期，于1967年定案

字符	ASCII值		字符	ASCII值		字符	ASCII值		字符	ASCII值		字符	ASCII值	
	DEC	HEX		DEC	HEX		DEC	HEX		DEC	HEX		DEC	HEX
Esc	27	1B	1	49	31	E	69	45	Y	89	59	m	109	6D
CR	13	0D	2	50	32	F	70	46	Z	90	5A	n	110	6E
LF	10	0A	3	51	33	G	71	47	[	91	5B	o	111	6F
Space	32	20	4	52	34	H	72	48	\	92	5C	p	112	70
!	33	21	5	53	35	I	73	49	]	93	5D	q	113	71
"	34	22	6	54	36	J	74	4A	^	94	5E	r	114	72
#	35	23	7	55	37	K	75	4B	_	95	5F	s	115	73
$	36	24	8	56	38	L	76	4C	`	96	60	t	116	74
&	37	25	9	57	39	M	77	4D	a	97	61	u	117	75
%	38	26	:	58	3A	N	78	4E	b	98	62	v	118	76
'	39	27	;	59	3B	O	79	4F	c	99	63	w	119	77
(	40	28	<	60	3C	P	80	50	d	100	64	x	120	78
)	41	29	=	61	3D	Q	81	51	e	101	65	y	121	79
*	42	2A	>	62	3E	R	82	52	f	102	66	z	122	7A
+	43	2B	?	63	3F	S	83	53	g	103	67	{	123	7B
,	44	2C	@	64	40	T	84	54	h	104	68	|	124	7C
-	45	2D	A	65	41	U	85	55	i	105	69	}	125	7D
.	46	2E	B	66	42	V	86	56	j	106	6A	~	126	7E
/	47	2F	C	67	43	W	87	57	k	107	6B	Del	127	7F
0	48	30	D	68	44	X	88	58	l	108	6C			

ASCII字符集

- ASCII是一个7位的编码标准，编码的取值范围实际上是00h-7Fh
  - 26个小写字母
  - 26个大写字母
  - 10个数字
  - 32个符号
  - 33个控制代码和空格

字符	ASCII值		字符	ASCII值		字符	ASCII值		字符	ASCII值		字符	ASCII值	
	DEC	HEX		DEC	HEX		DEC	HEX		DEC	HEX		DEC	HEX
Esc	27	1B	1	49	31	E	69	45	Y	89	59	m	109	6D
CR	13	0D	2	50	32	F	70	46	Z	90	5A	n	110	6E
LF	10	0A	3	51	33	G	71	47	[	91	5B	o	111	6F
Space	32	20	4	52	34	H	72	48	\	92	5C	p	112	70
!	33	21	5	53	35	I	73	49	]	93	5D	q	113	71
"	34	22	6	54	36	J	74	4A	^	94	5E	r	114	72
#	35	23	7	55	37	K	75	4B	_	95	5F	s	115	73
$	36	24	8	56	38	L	76	4C	`	96	60	t	116	74
&	37	25	9	57	39	M	77	4D	a	97	61	u	117	75
%	38	26	:	58	3A	N	78	4E	b	98	62	v	118	76
'	39	27	;	59	3B	O	79	4F	c	99	63	w	119	77
(	40	28	<	60	3C	P	80	50	d	100	64	x	120	78
)	41	29	=	61	3D	Q	81	51	e	101	65	y	121	79
*	42	2A	>	62	3E	R	82	52	f	102	66	z	122	7A
+	43	2B	?	63	3F	S	83	53	g	103	67	{	123	7B
,	44	2C	@	64	40	T	84	54	h	104	68	|	124	7C
-	45	2D	A	65	41	U	85	55	i	105	69	}	125	7D
.	46	2E	B	66	42	V	86	56	j	106	6A	~	126	7E
/	47	2F	C	67	43	W	87	57	k	107	6B	Del	127	7F
0	48	30	D	68	44	X	88	58	l	108	6C			

ASCII字符集

- 不同的计算机厂 商对ASCII进行了扩充，增加了128个附加字符，它们的值在**127以上的部分不是统一的**
  - ANSI字符集
  - Symbol字符集
  - OEM字符集

填空题 10分

设置

4D 5A在ASCII编码中对应的字符是 [填空1]

正常使用填空题需3.0以上版本雨课堂

作答

布尔运算

布尔表达式

•布尔表达式是一种返回布尔值（True或False）的表达式，由布尔变量和布尔运算符组成。例如，表达式 (A AND B) OR C 是一个典型的布尔表达式，用于描述逻辑关系。

NOT运算符

•NOT运算符用于反转一个布尔值的状态，通常表示为 ¬X 或 !X。

下面的真值表显示了NOT运算的结果：

X	¬X
T	F
F	T

AND运算符

•AND运算符用于比较两个布尔值，只有当两个值都为真时，结果才为真。通常表示为 X AND Y。下面的真值表显示了AND运算的结果：

X	Y	X AND Y
T	T	T
T	F	F
F	T	F
F	F	F

OR运算符

•OR运算符也比较两个布尔值，如果至少一个值为真，结果就是真。通常表示为 X OR Y。下面的真值表显示了OR运算的结果：

X	Y	X OR Y
T	T	T
T	F	T
F	T	T
F	F	F

运算符的优先级

•在涉及多个运算符的布尔表达式中，理解各运算符的优先级是关键。如下表所示，NOT运算符拥有最高的优先级，其次分别是AND和OR运算符。

•为了消除歧义并确保表达式按照预期的顺序求值，建议使用小括号来明确指定求值的顺序

优先级	运算符
1	NOT
2	AND
3	OR

布尔函数的真值表

- 布尔函数的真值表是一种表格方法，用于展示对于不同输入组合的输出结果。
- 要构建布尔函数的真值表，需要列出所有可能的输入组合及其对应的输出结果。
- 真值表通常分为两部分：
  - **输入列：**列出了所有变量的所有可能组合。
  - **输出列：**显示了对应于每种输入组合的布尔函数的结果。

布尔函数的真值表

•F(X,Y, S) =(Y ∧S) ∨(X ∧¬S)

X	Y	S	Y ∧S	¬S	X ∧¬S	F(X,Y, S)
F	F	F	F	T	F	F
F	T	F	F	T	F	F
T	F	F	F	T	T	T
T	T	F	F	T	T	T
F	F	T	F	F	F	F
F	T	T	T	F	F	T
T	F	T	F	F	F	F
T	T	T	T	F	F	T

汇编语言与逆向技术

第1章 课程介绍

王志
zwang@nankai.edu.cn

南开大学 网络空间安全学院

2024-2025学年