1 编译原理概述

杨策

1 编译原理概述

- 本课程与计算机学科体系的关联
- 编译过程概述
- 调试技术入门

计算学科体系

科学 ⟷ 工程

信息论

计算理论

机器学习

可解释性

信息安全

算法

图形学

图像处理

编译原理

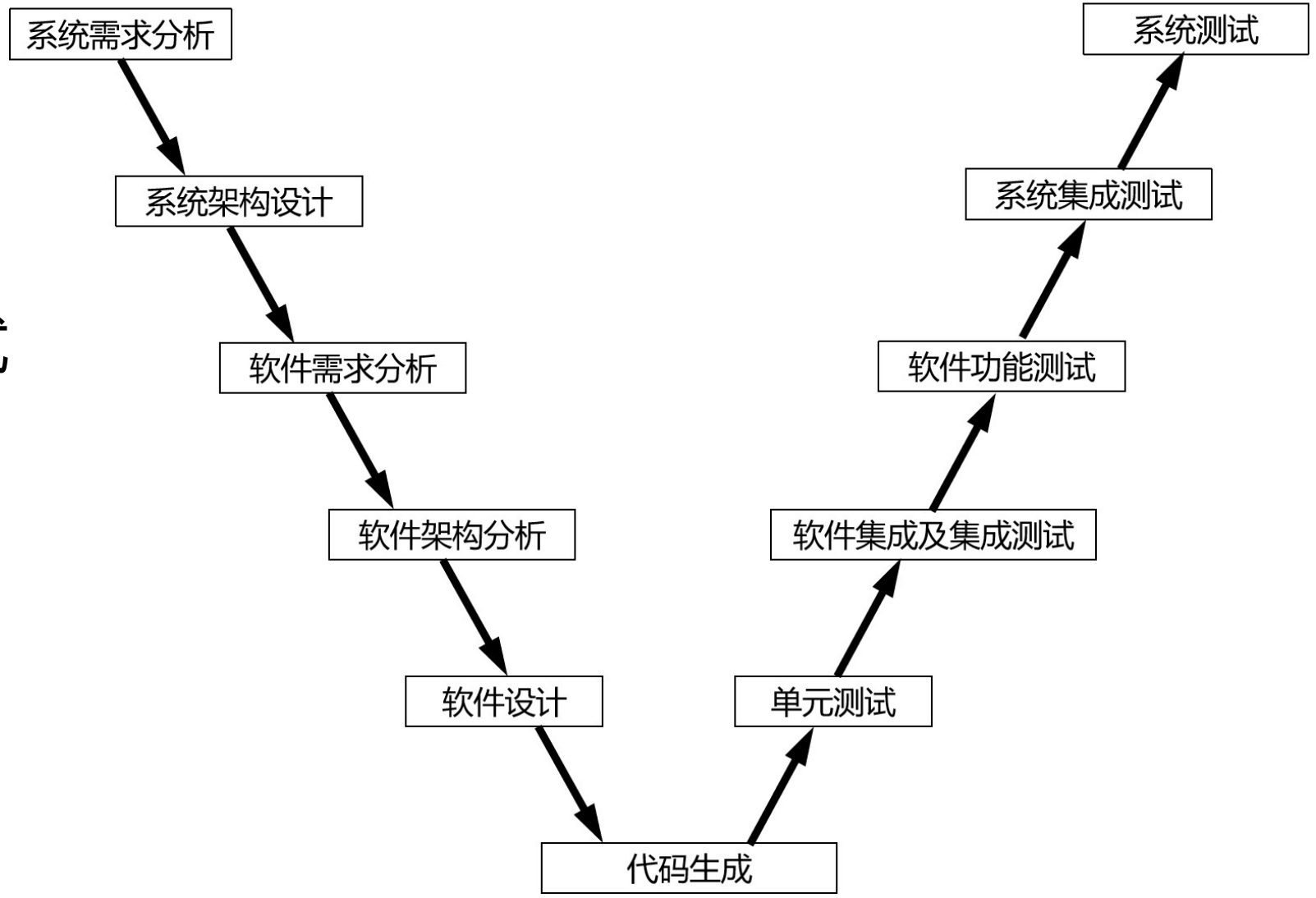
人机交互

软件工程

web开发

软件开发过程

- 设计
- 编码
- 测试/调试

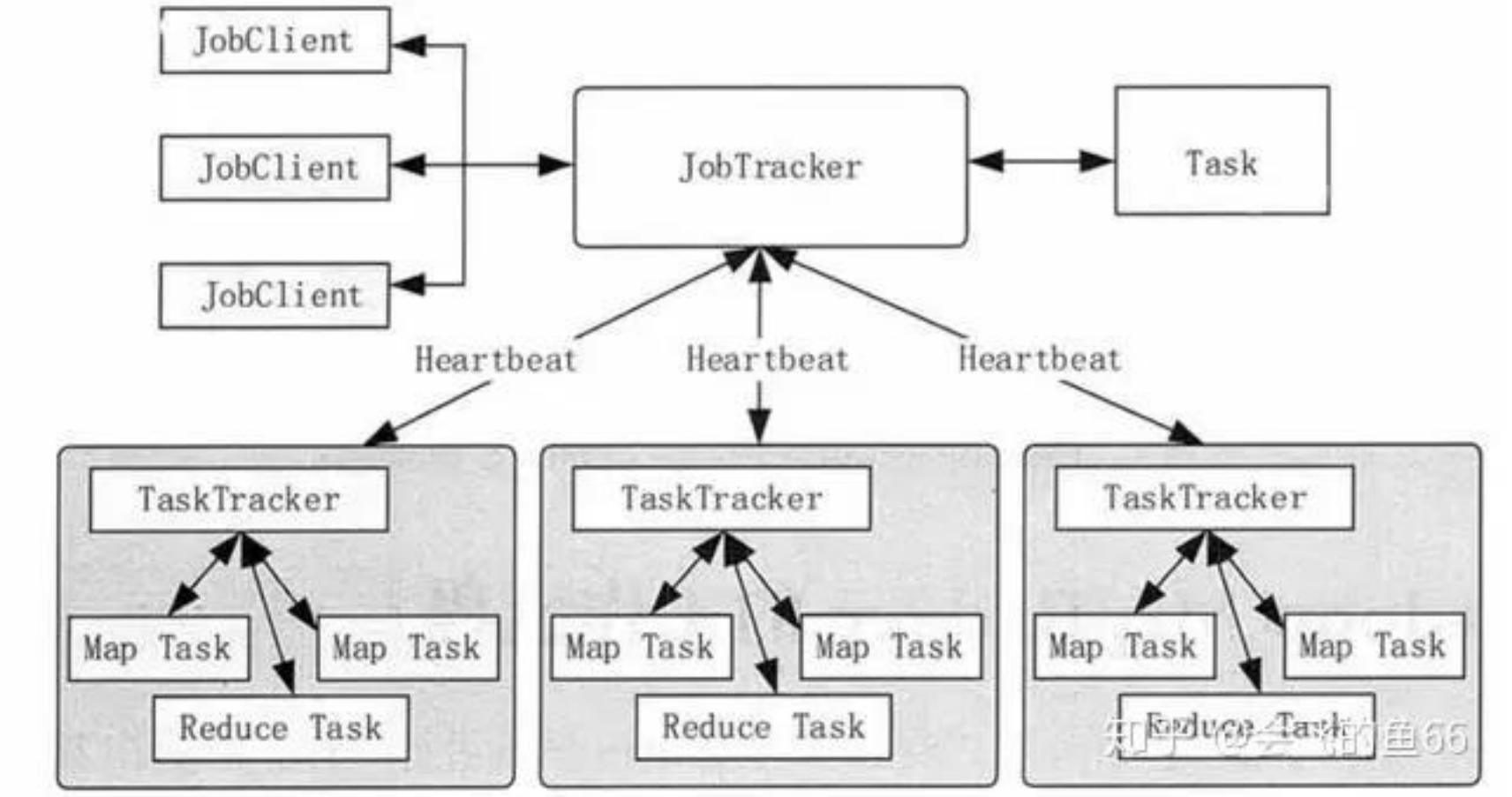


架构设计

- 黑客帝国 Matrix
  - 架构师 Architect
  - 先知 Oracle
- 高可用
  - 单机10小时故障1次
  - 系统10w小时1次

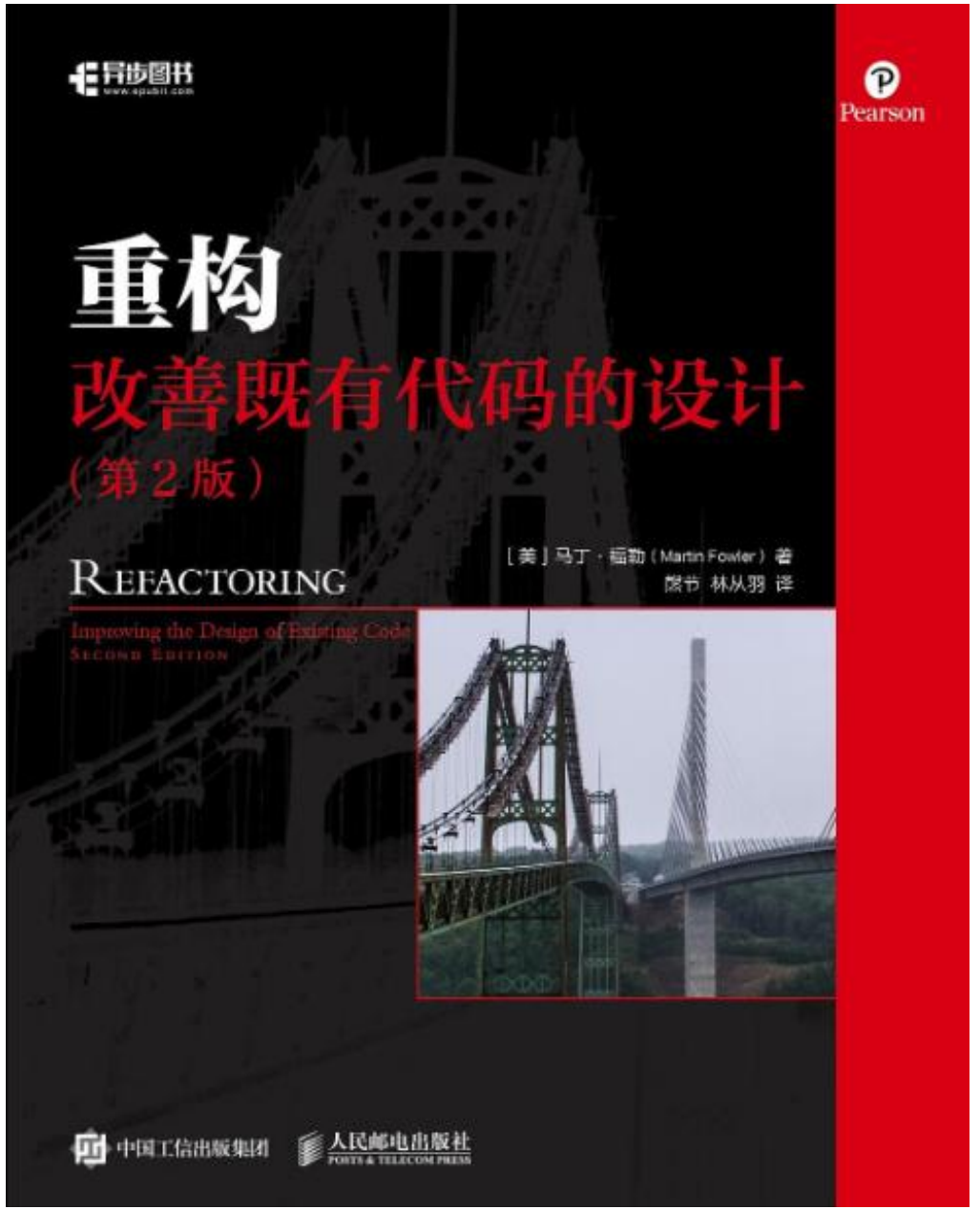
架构设计

- 高可用
  - 利用冗余提供容错

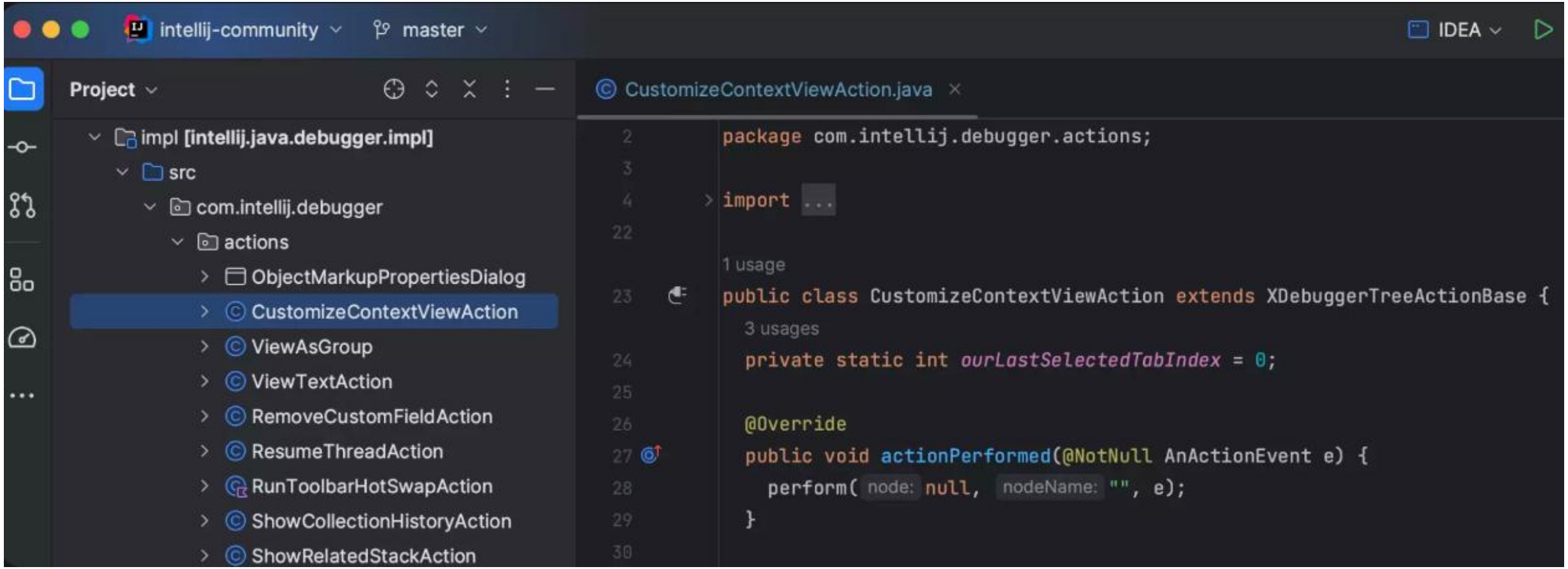


设计

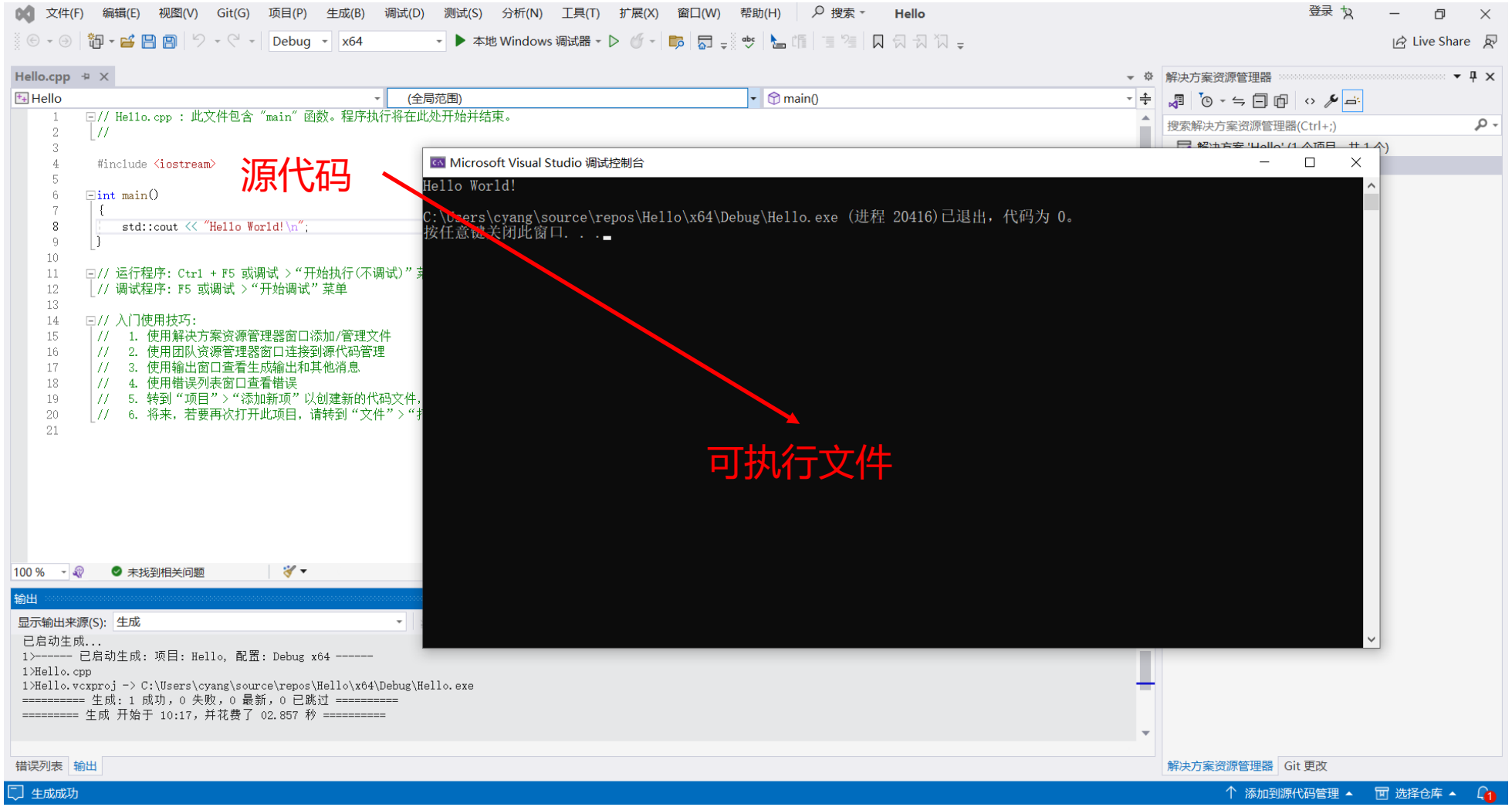
- 类与模块设计
  - 抽象数据类型
- 设计模式
  - 工厂模式
  - 控制反转
  - AOP



编写代码

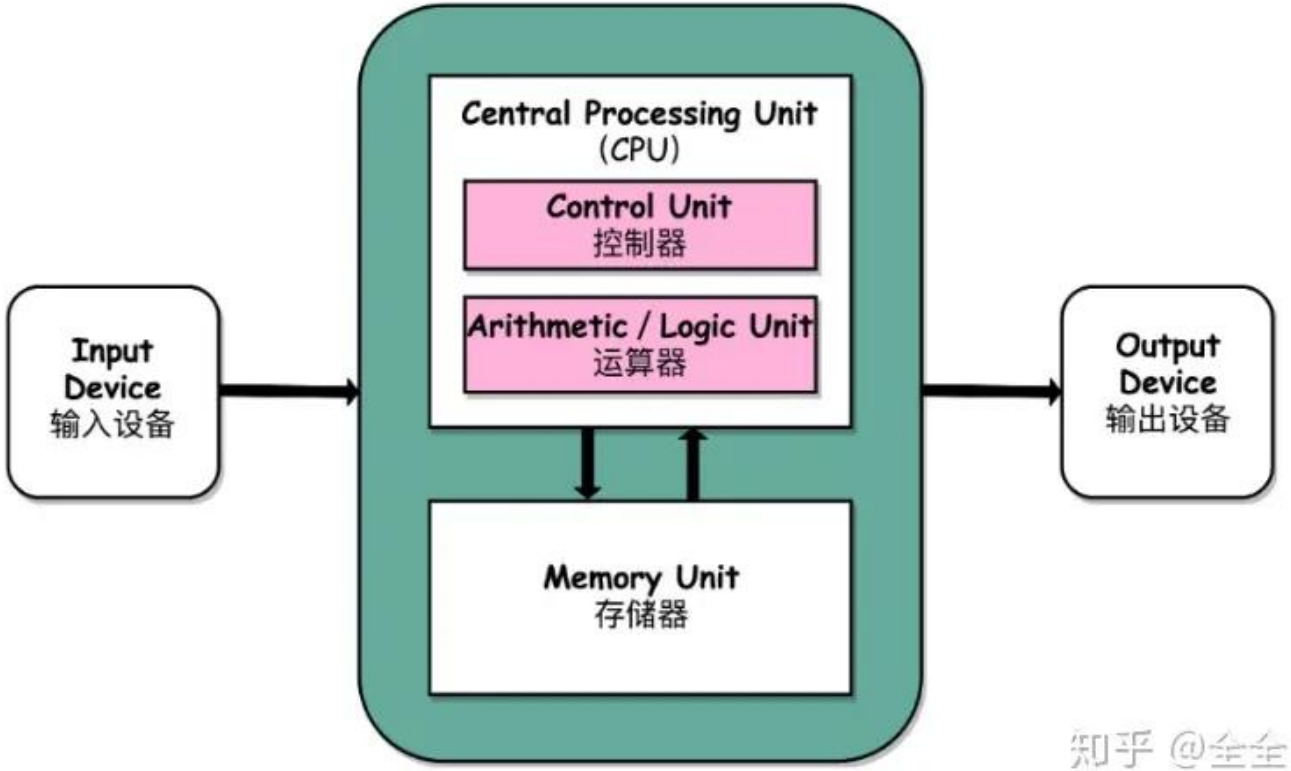

intellij-community
master
IDEA
Project
impl [intellij.java.debugger.impl]
src
com.intellij.debugger
actions
ObjectMarkupPropertiesDialog
CustomizeContextViewAction
ViewAsGroup
ViewTextAction
RemoveCustomFieldAction
ResumeThreadAction
RunToolbarHotSwapAction
ShowCollectionHistoryAction
ShowRelatedStackAction
CustomizeContextViewAction.java
package com.intellij.debugger.actions;
import ...
1 usage
public class CustomizeContextViewAction extends XDebuggerTreeActionBase {
3 usages
private static int ourLastSelectedTabIndex = 0;
@Override
public void actionPerformed(@NotNull AnActionEvent e) {
perform(node: null, nodeName: "", e);
}

编译：计算机程序是如何从代码生成的



可执行文件

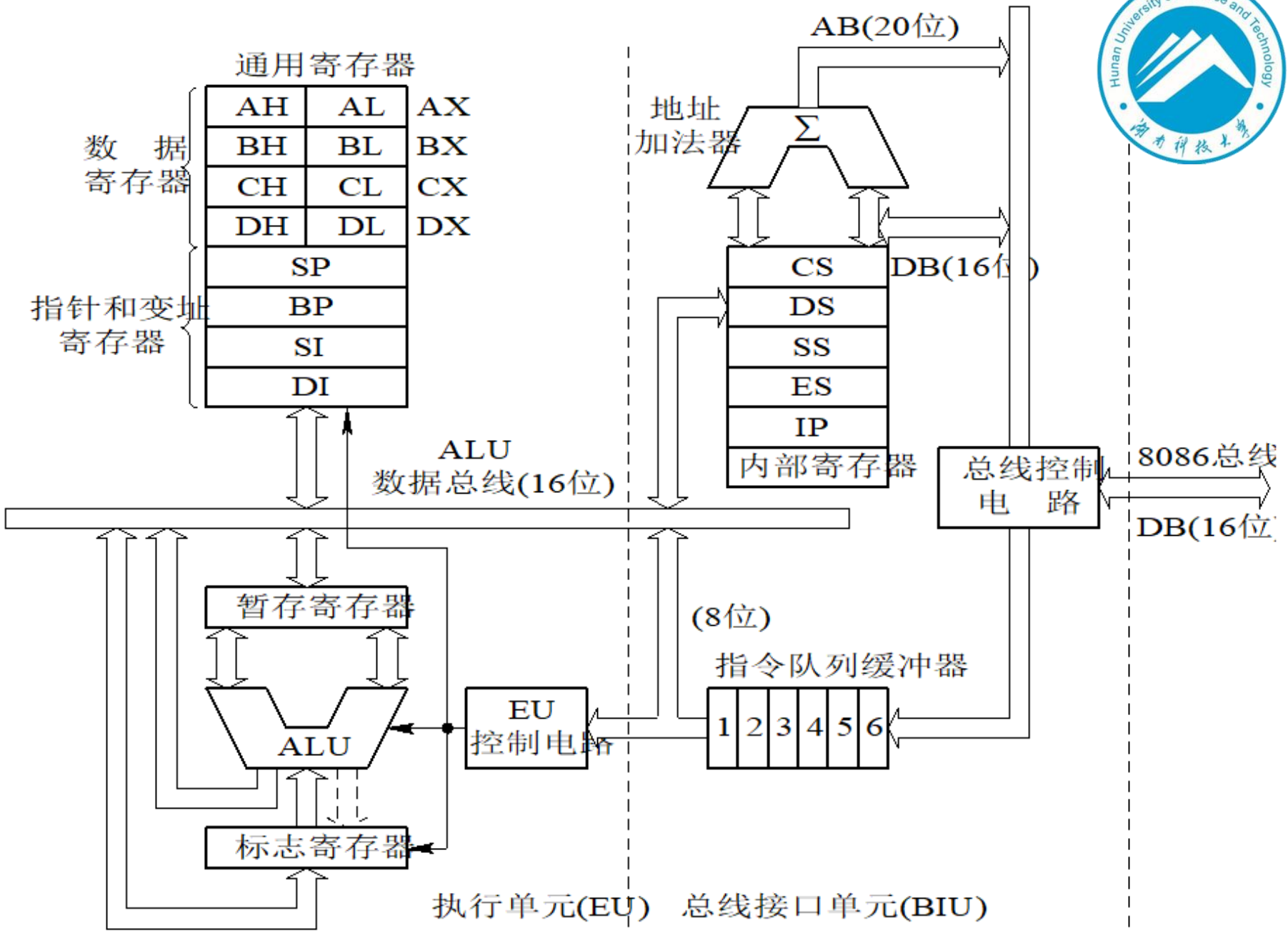
- 冯·诺依曼架构
  - 控制器
  - 算术逻辑单元
  - 存储器
  - IO设备



CPU只能执行机器语言

汇编语言

- 数据传送
- 算术运算
- 逻辑运算
- 跳转
- 函数调用



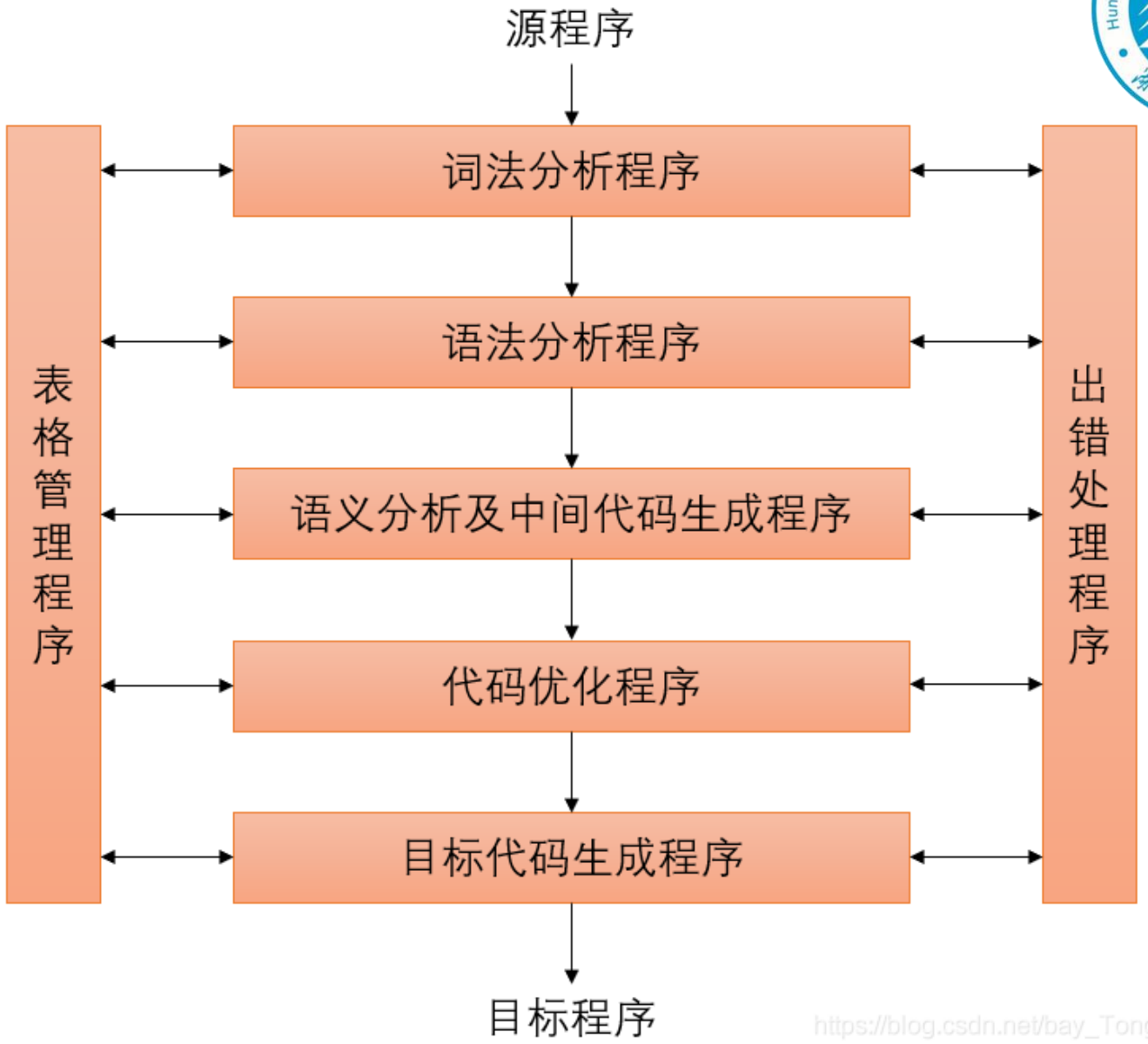
8086CPU内部结构框图

编译


#include <stdio.h>
int main()
{
printf("Hello World!\n");
}
高级语言
编译
汇编语言
机器语言
00007FF735D91880 40 55 push rbp
00007FF735D91882 57 push rdi
00007FF735D91883 48 81 EC E8 00 00 00 sub rsp,0E8h
00007FF735D9188A 48 8D 6C 24 20 lea rbp,[rsp+20h]
00007FF735D9188F 48 8D 0D 72 F7 00 00 lea rcx,[__F7EED072_Hello@cpp (07FF735DA1008h)]
00007FF735D91896 E8 D0 FA FF FF call __CheckForDebuggerJustMyCode (07FF735D9136Bh)
00007FF735D9189B 48 8D 0D 86 83 00 00 lea rcx,[string "Hello World!\n" (07FF735D99C28h)]
00007FF735D918A2 E8 EE F8 FF FF call printf (07FF735D91195h)
00007FF735D918A7 33 C0 xor eax,eax
00007FF735D918A9 48 8D A5 C8 00 00 00 lea rsp,[rbp+0C8h]
00007FF735D918B0 5F pop rdi
00007FF735D918B1 5D pop rbp
00007FF735D918B2 C3 ret

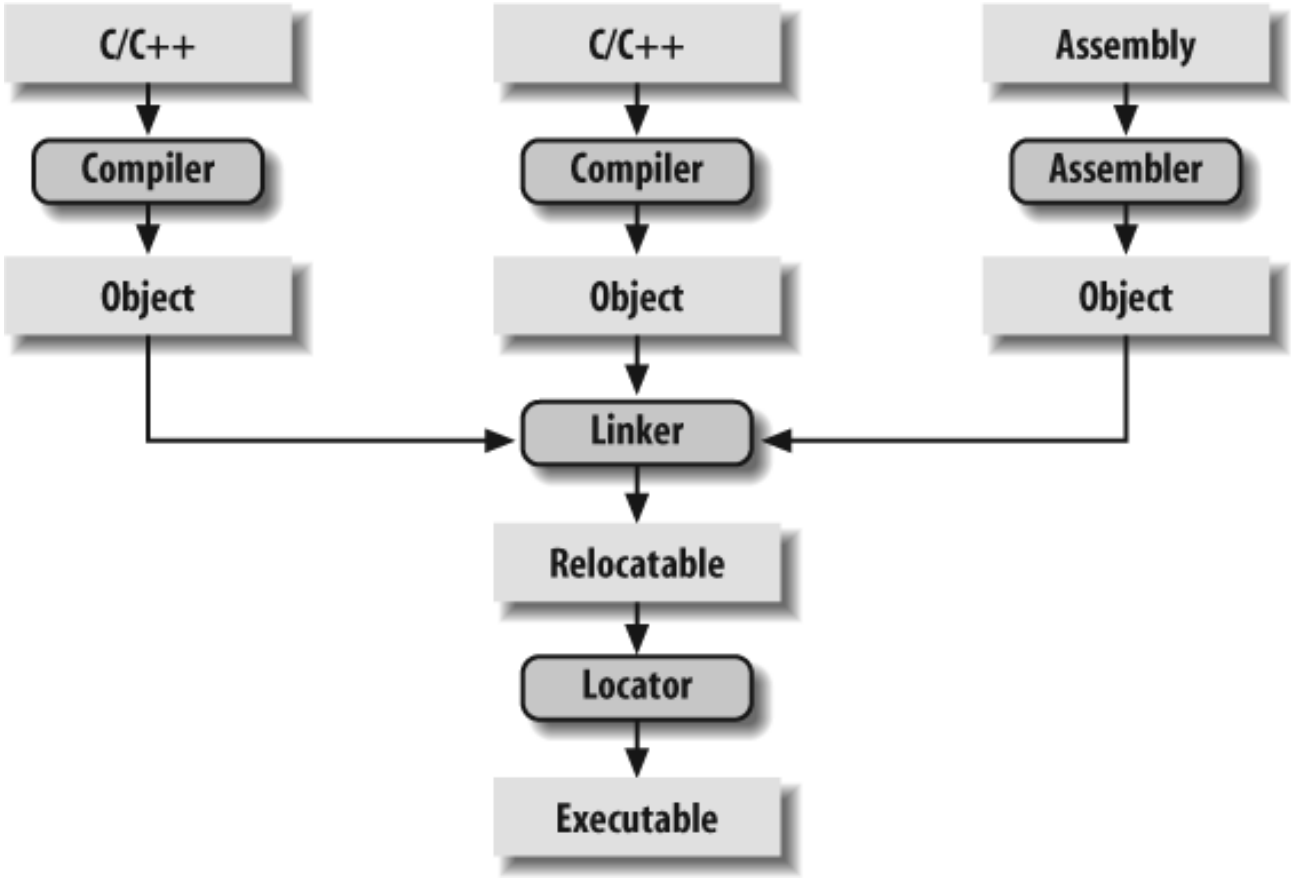
编译原理

- 编译过程中的原理
  - 现代主要是c语言
  - 算法为主
  - 工程化内容较少
  - 部分步骤有工具可用



C++ compiler

- 预处理
- 编译
- 汇编
- 链接
- 加载



C++ compiler

- 链接

```
// file a.h
include <b.h>
class ClassA {
private:
    ClassB b;
};

// file a.cpp
Class ClassA { ... }
```

编译

```
a.obj call ClassBImpl::Run()
```

链接

```
// file b.h
class ClassBImpl;
class ClassB {
private:
    ClassBImpl* b_impl;
};

// file b.cpp
Class ClassB { ... }
Class ClassBimpl
```

编译

```
b.obj ClassBImpl::Run()
```

C++ compiler

- 外部链接

节省编译资源

二进制兼容

```
g++ main.cpp
    -I/usr/local/include/opencv4        头文件目录
    -L/usr/local/lib                    链接库目录
    -lopencv_core                       链接库名称
    -lopencv_imgproc                    链接库名称
```

C++ compiler

- 静态链接
- 动态链接
  - DLL
  - 隐式链接
  - 显式链接

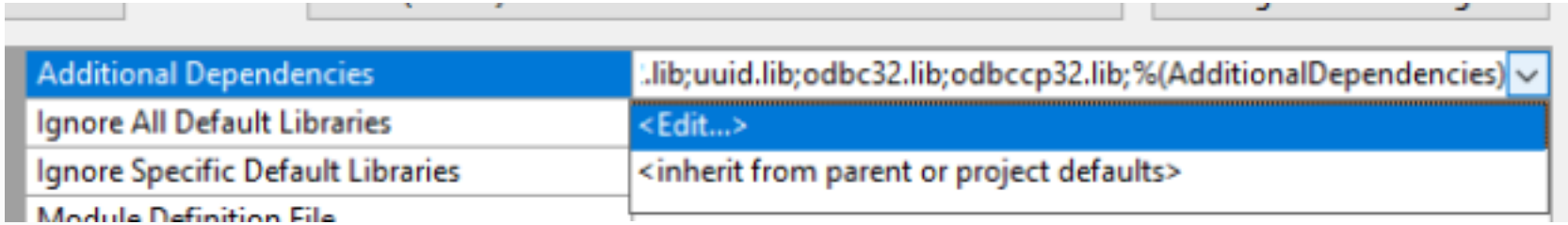
```
push        rbp
push        rdi
sub         rsp,0E8h
lea         rbp,[rsp+20h]
lea         rcx,[__F7EED072_Hello@cpp (07FF735DA1008h)]
call        __CheckForDebuggerJustMyCode (07FF735D9136Bh)

lea         rcx,[string "Hello World!\n" (07FF735D99C28h)]
call        printf (07FF735D91195h)

xor         eax,eax
lea         rsp,[rbp+0C8h]
pop         rdi
pop         rbp
ret
```

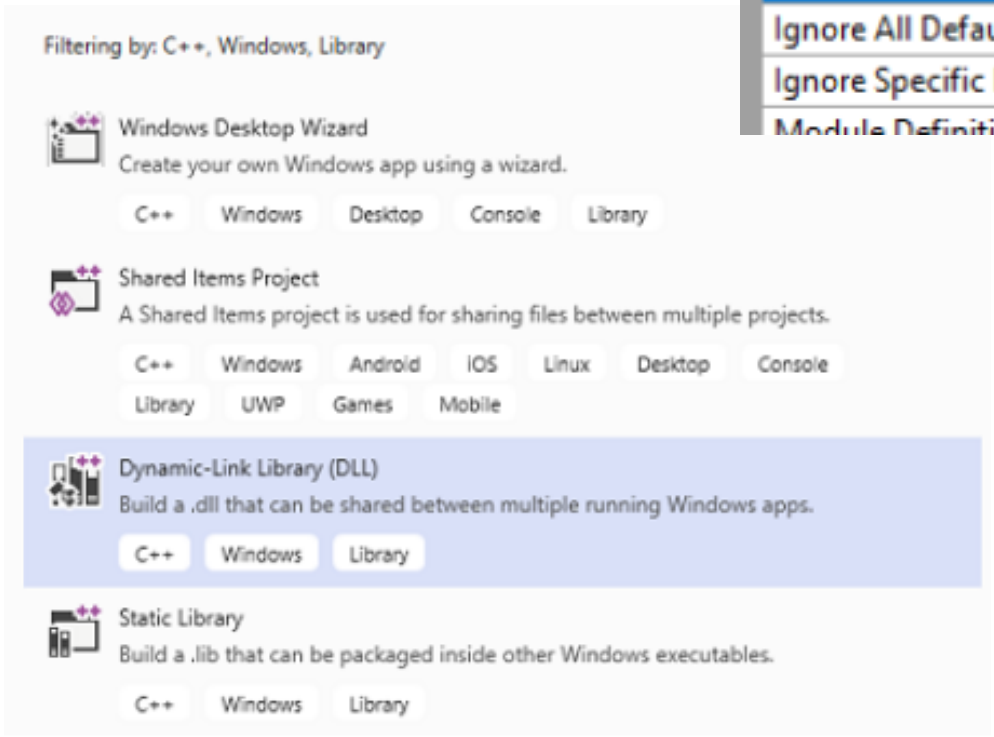
https://learn.microsoft.com/zh-cn/cpp/build/linking-an-executable-to-a-dll?view=msvc-170

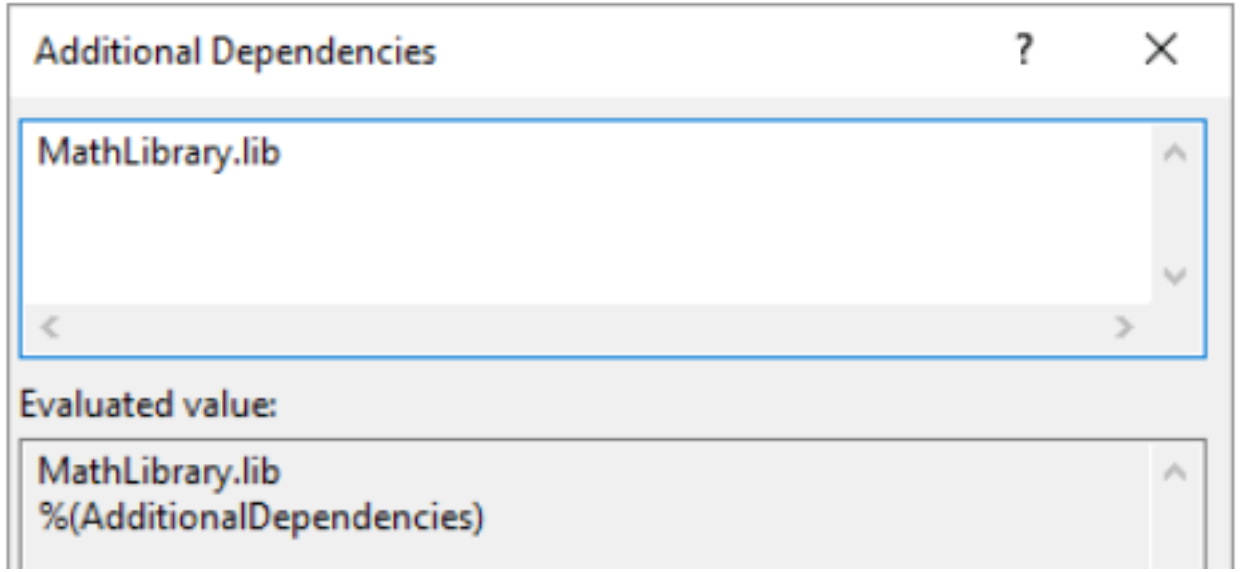
C++ compiler

```
#ifdef MATHLIBRARY_EXPORTS
#define MATHLIBRARY_API __declspec(dllexport)
#else
#define MATHLIBRARY_API __declspec(dllimport)
#endif
```

- 动态链接







https://learn.microsoft.com/zh-cn/cpp/build/walkthrough-creating-and-using-a-dynamic-link-library-cpp?view=msvc-170

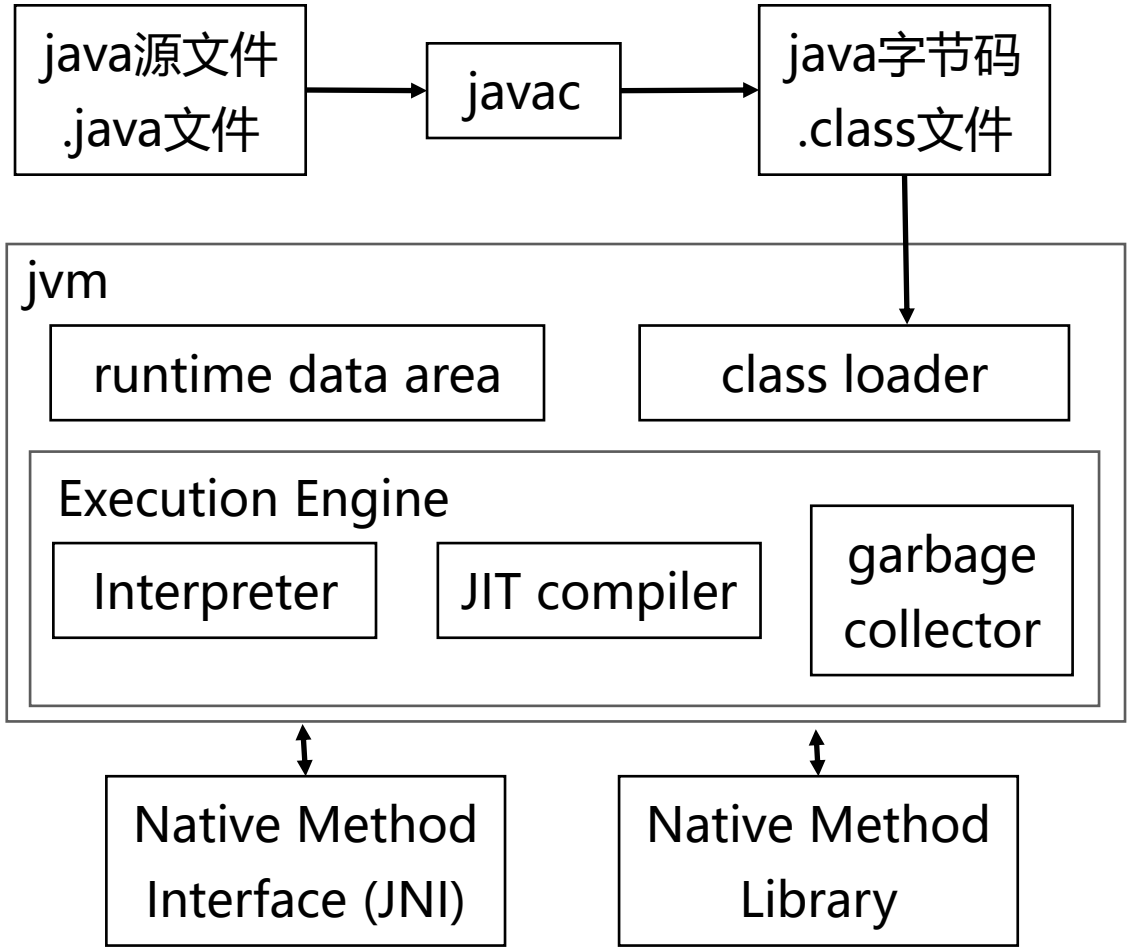
C++ compiler

- 常用编译器
  - MSVC
  - GCC
  - Clang-LLVM
- Make
  - 解决依赖与链接问题

```
edit : main.o kbd.o command.o display.o \
        insert.o search.o files.o utils.o
    cc -o edit main.o kbd.o command.o display.o \
        insert.o search.o files.o utils.o

main.o : main.c defs.h
    cc -c main.c
kbd.o : kbd.c defs.h command.h
    cc -c kbd.c
command.o : command.c defs.h command.h
    cc -c command.c
display.o : display.c defs.h buffer.h
    cc -c display.c
insert.o : insert.c defs.h buffer.h
    cc -c insert.c
```

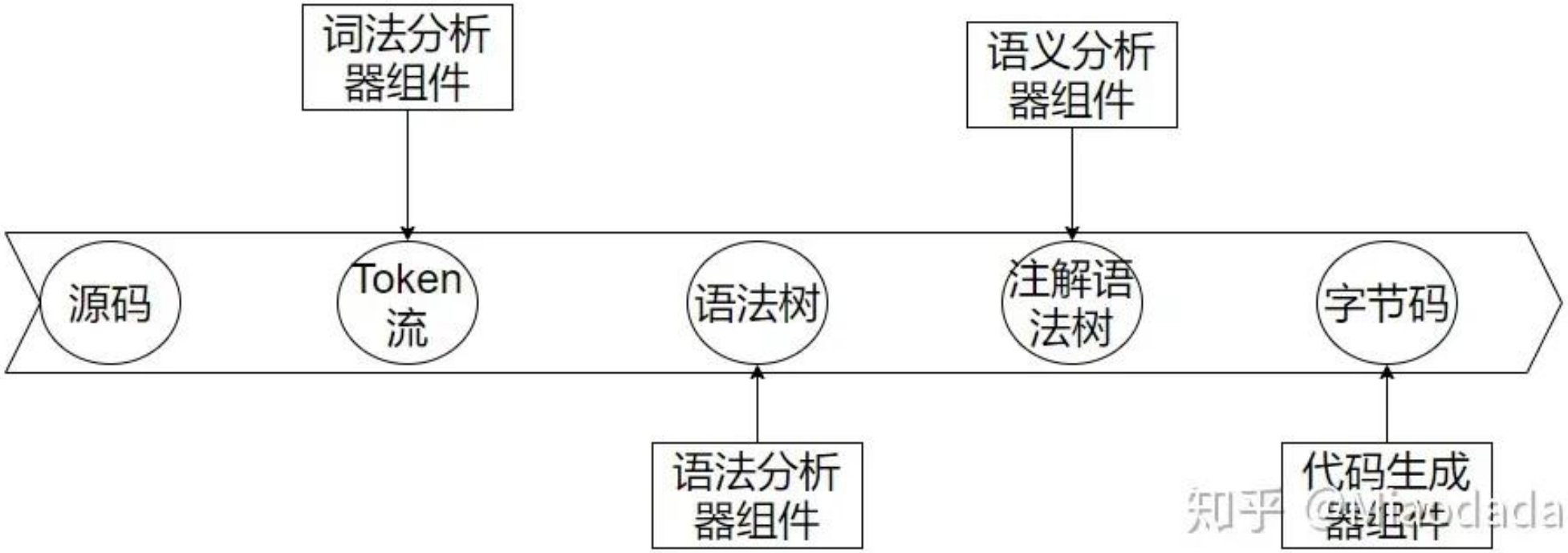
Java Compiler

- 混合结构
  - java源代码
  - 字节码:jvm上的汇编
- JIT compiler
  - Tiered Compilation
  - C1 compiler
  - C2 compiler



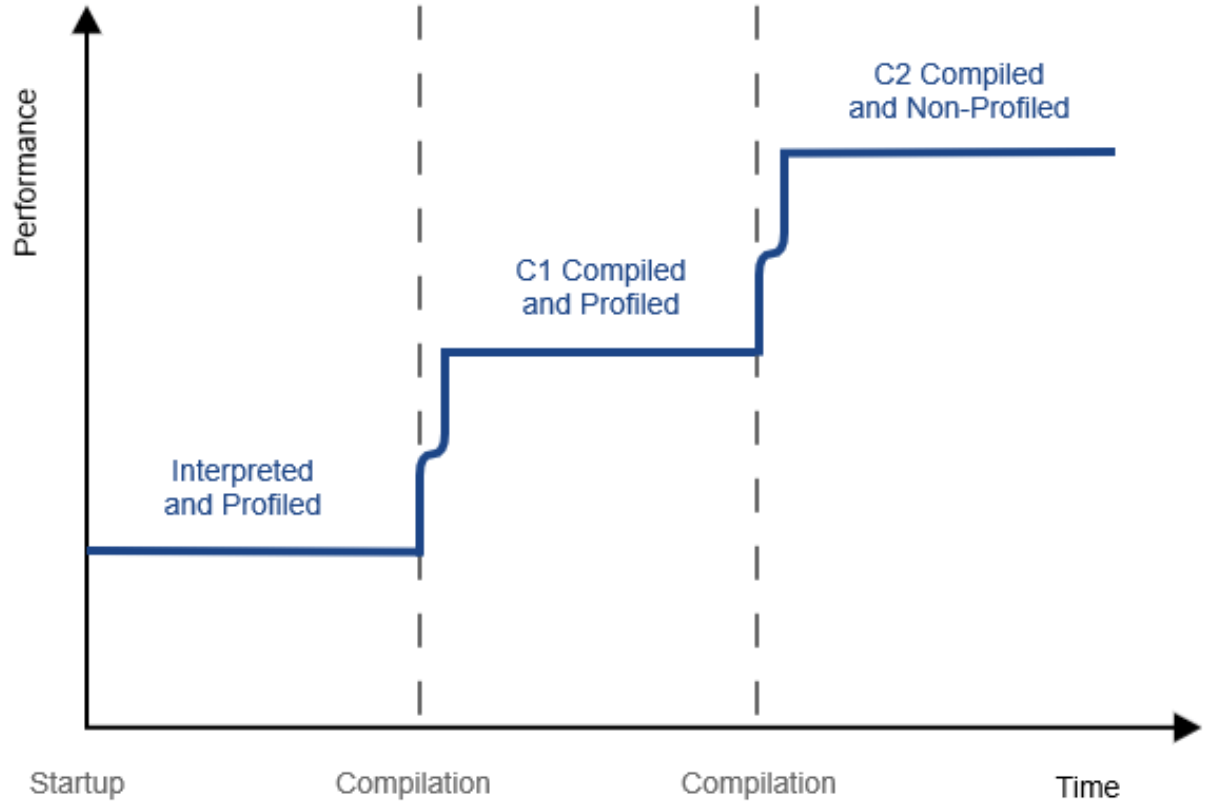
Java Compiler

- Javac 编译过程



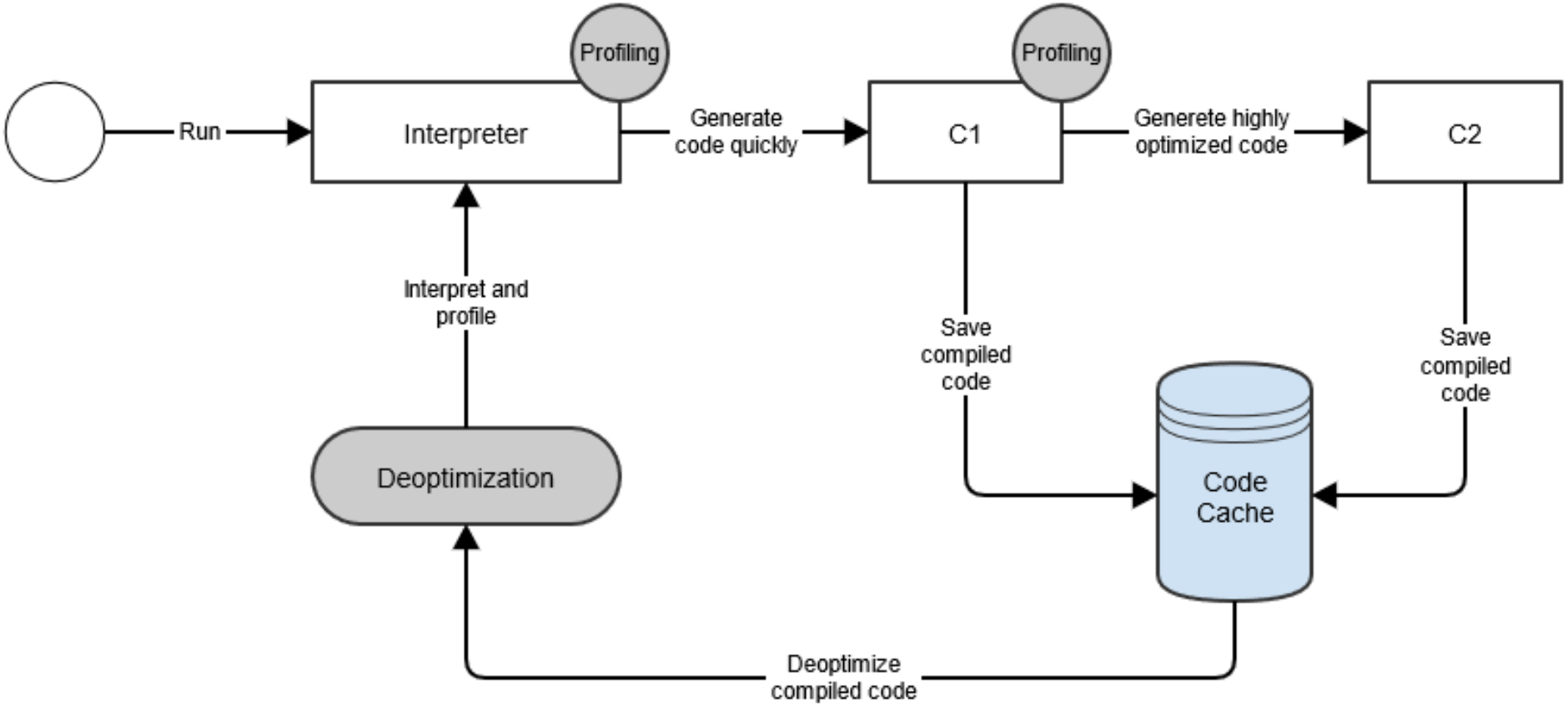
Java Compiler

- Hotspot
  - 优化多次调用的函数
  - C1-client
  - C2-server
  - 分层编译
    - 先使用C1编译
    - 热点使用C2编译

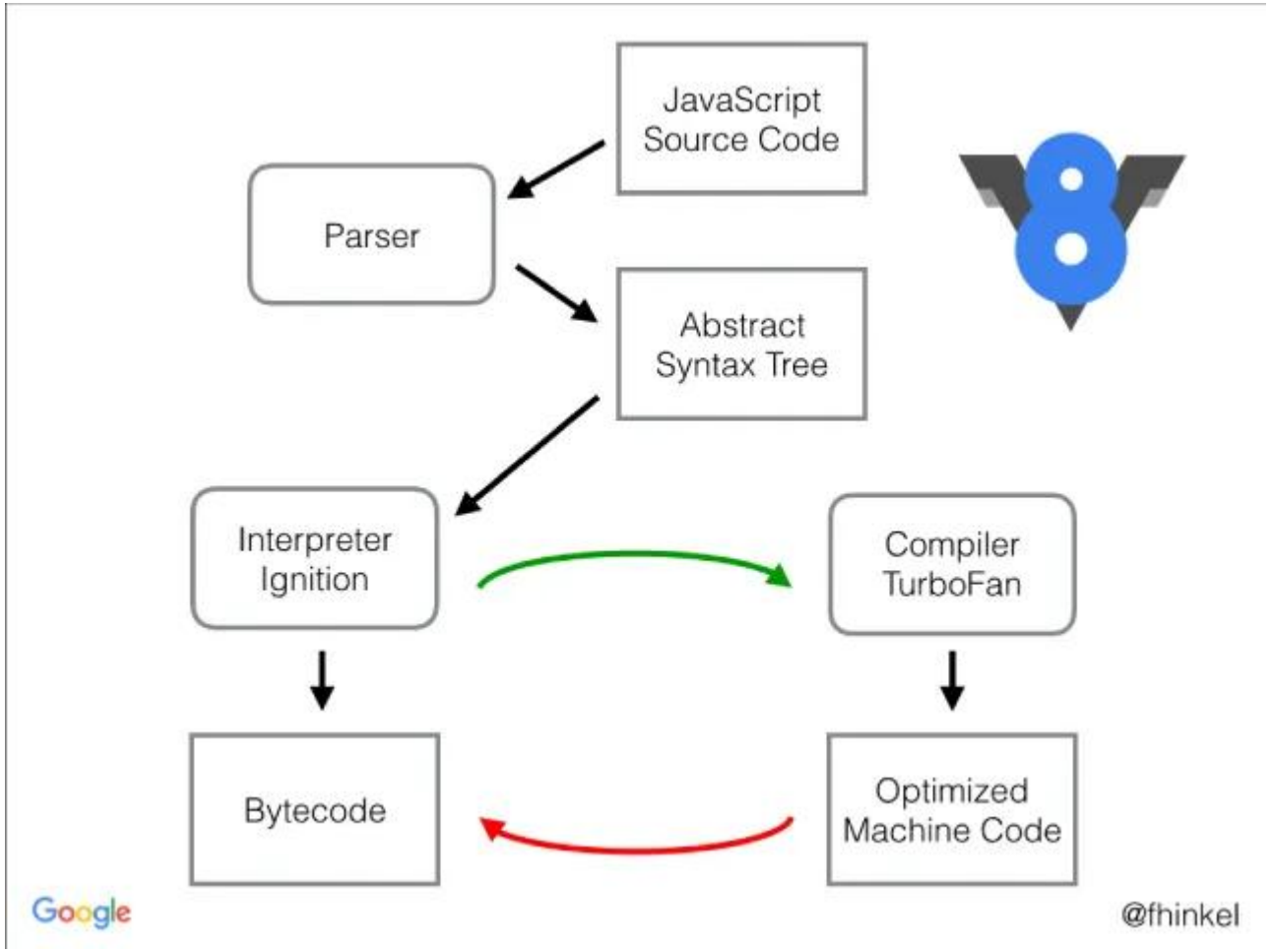


由于硬件资源增加，JAVA 7开始增加分层编译，而不再区分client server

Java Compiler



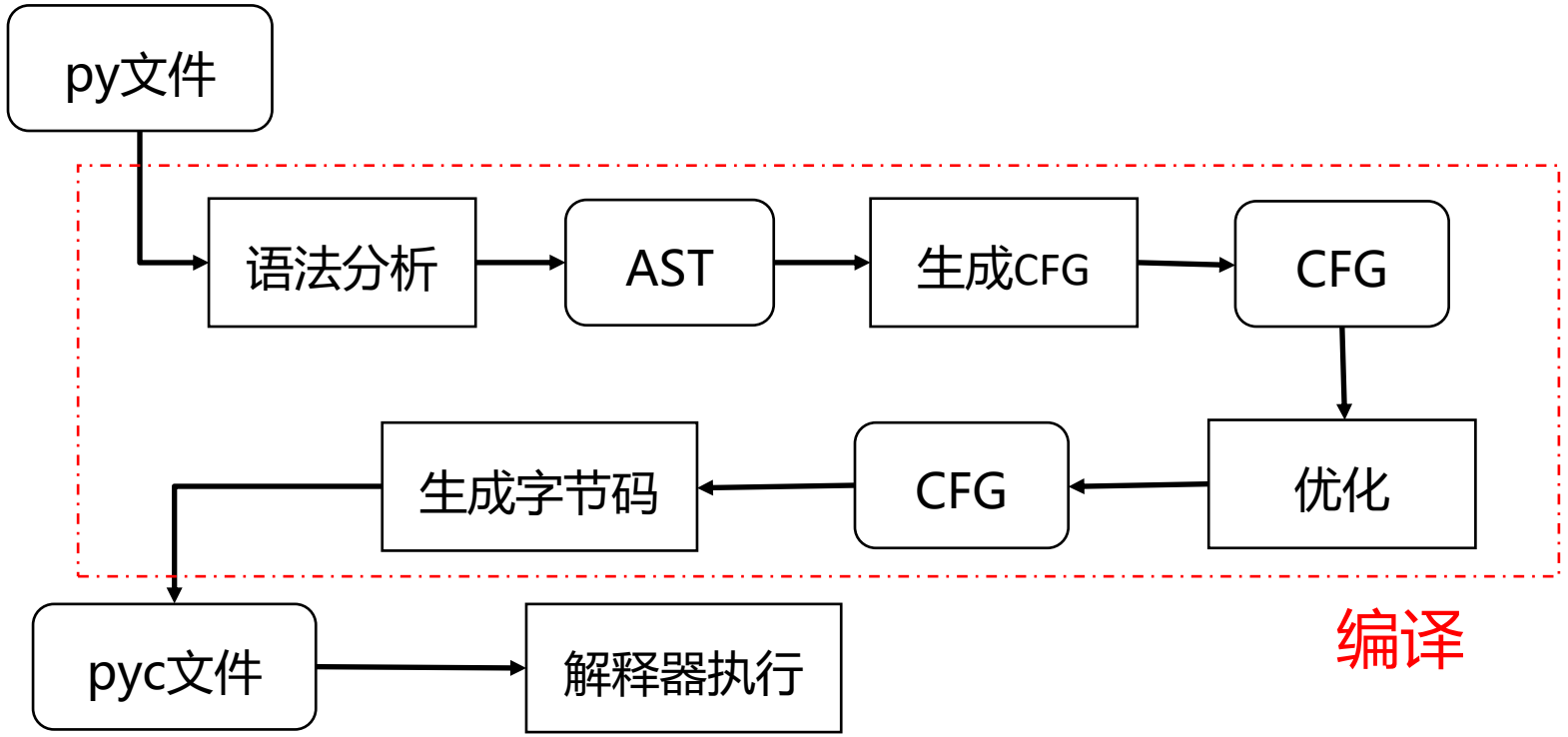
Javascript Engine



- Parser
  - 将JS代码转换成AST
- Ignition
  - 解释执行字节码
- TurboFan
  - 进行JIT编译
  - 编译成机器语言

https://zhuanlan.zhihu.com/p/28590489

Python Compiler



https://devguide.python.org/internals/compiler/

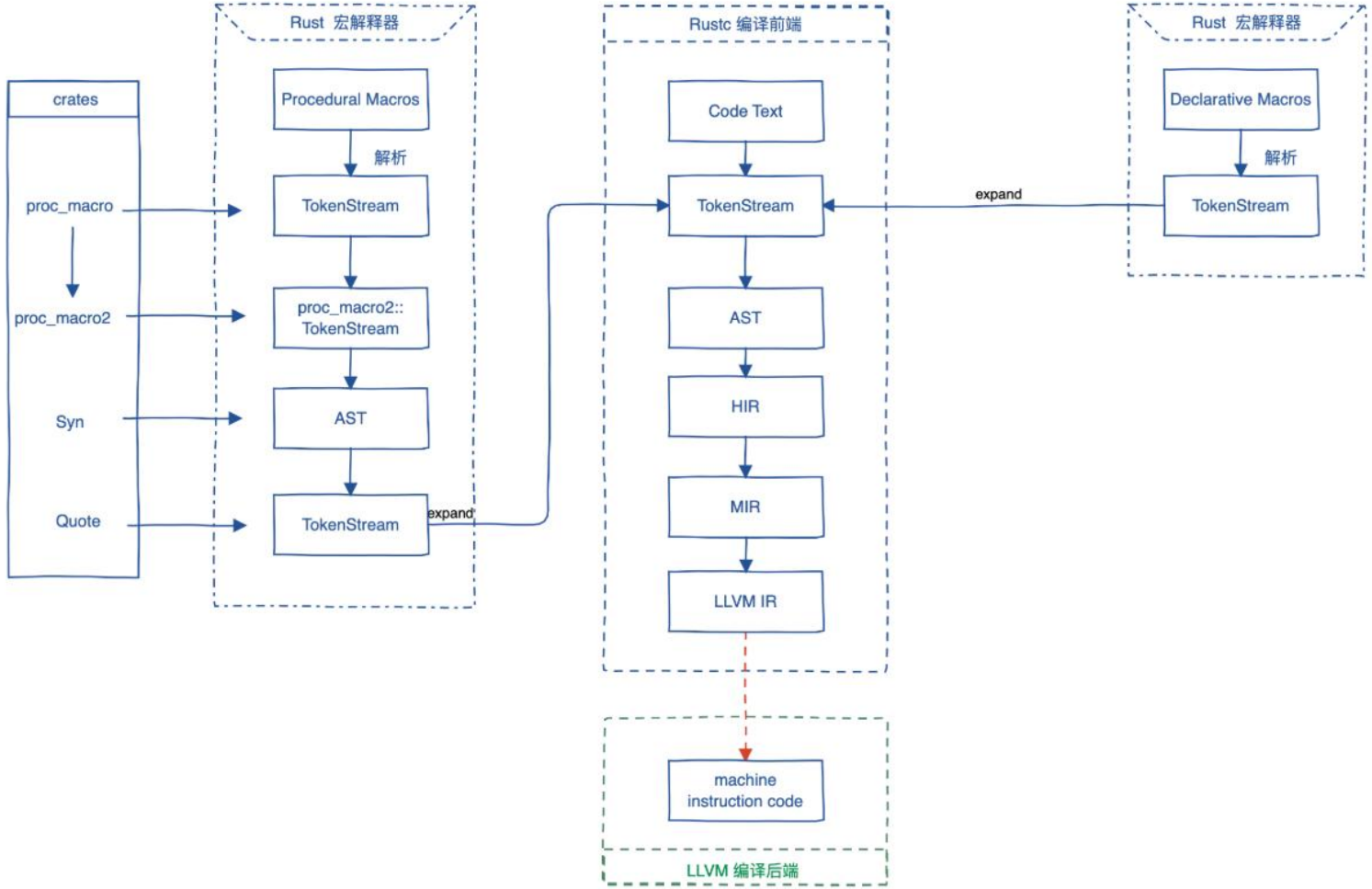
https://nanguage.gitbook.io/inside-python-vm-cn/

Python Compiler

- Python解释器
  - 执行PyCodeObject
  - Global Interpreter Lock - CPython
  - 垃圾回收：引用计数
    - 基本上是学的Java

Rust Compiler

- 基于LLVM后端
- Rustc前端



测试

- Google Test
- Junit

```
// 测试正数
TEST(FactorialTest, Positive) {
  EXPECT_EQ(1, Factorial(1));
  EXPECT_EQ(2, Factorial(2));
  EXPECT_EQ(6, Factorial(3));
  EXPECT_EQ(40320, Factorial(8));
}
```

```
public class CalculatorTest {
    private Calculator calculator;

    @BeforeEach
    public void setUp() {
        calculator = new Calculator();
    }

    @Test
    public void testAdd() {
        assertEquals(5, calculator.add(2, 3));
    }
```

调试

- 瞪眼法
  - 北大韦神
  - 神经网络 Chatgpt
- 加日志
- 使用调试器

调试

```java
public static boolean cmpQuit(String str) { return (str.toLowerCase() == "quit"); }

public static void main(String[] args) {
    System.out.printf("Input until read Quit:\n");

    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
    String str = null;
    try {
        str = bufferedReader.readLine();
        while (!cmpQuit(str)) {
            str = bufferedReader.readLine();
        }
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

```
Input until read Quit:
abc
Quit
quit
not quit?
```

调试

printf

logger

glog

log4j

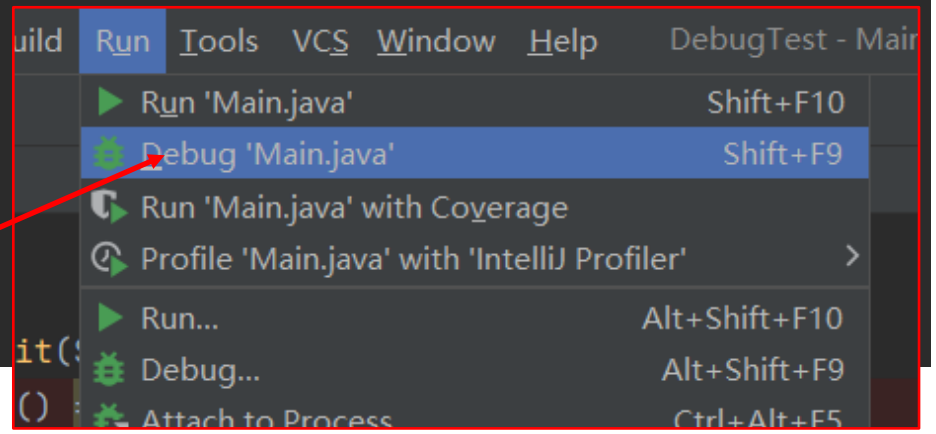
```java
public static Logger logger;
1 usage
public static boolean cmpQuit(String str) {
    logger.info(String.format("length: %d, content: %s", str.length(), str));
    return (str.toLowerCase() == "quit");
}

public static void main(String[] args) {
    logger = Logger.getLogger( name: "test logger");
    System.out.printf("Input until read Quit:\n");

    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
    String str = null;
    try {
        str = bufferedReader.readL
        while (!cmpQuit(str)) {
            str = bufferedReader.r
        }
    } catch (IOException e) {
        throw new RuntimeException
    }
}
```

```
Input until read Quit:
abc
1月 17, 2024 1:34:41 下午 Main cmpQuit
信息: length: 3, content: abc
quit
1月 17, 2024 1:34:49 下午 Main cmpQuit
信息: length: 4, content: quit
Quit
1月 17, 2024 1:34:51 下午 Main cmpQuit
信息: length: 4, content: Quit
```

调试

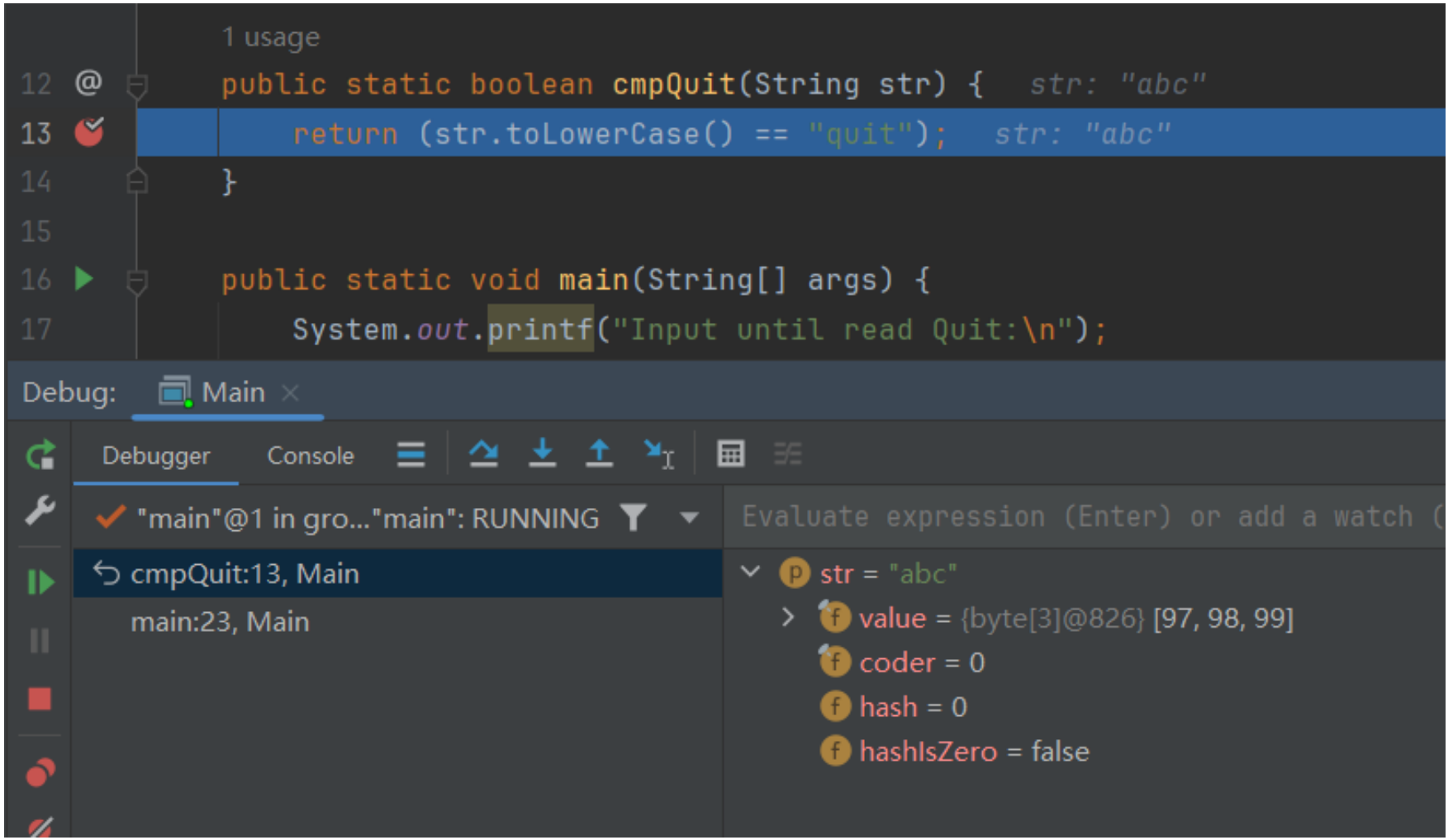
一般单击行的左侧设置断点

```
1 usage
public static boolean cmpQuit(String str) {
    return (str.toLowerCase() == "quit");
}

public static void main(String[] args) {
    System.out.printf("Input until read Quit:\n");

    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
    String str = null;
    try {
        str = bufferedReader.readLine();
        while (!cmpQuit(str)) {
            str = bufferedReader.readLine();
        }
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

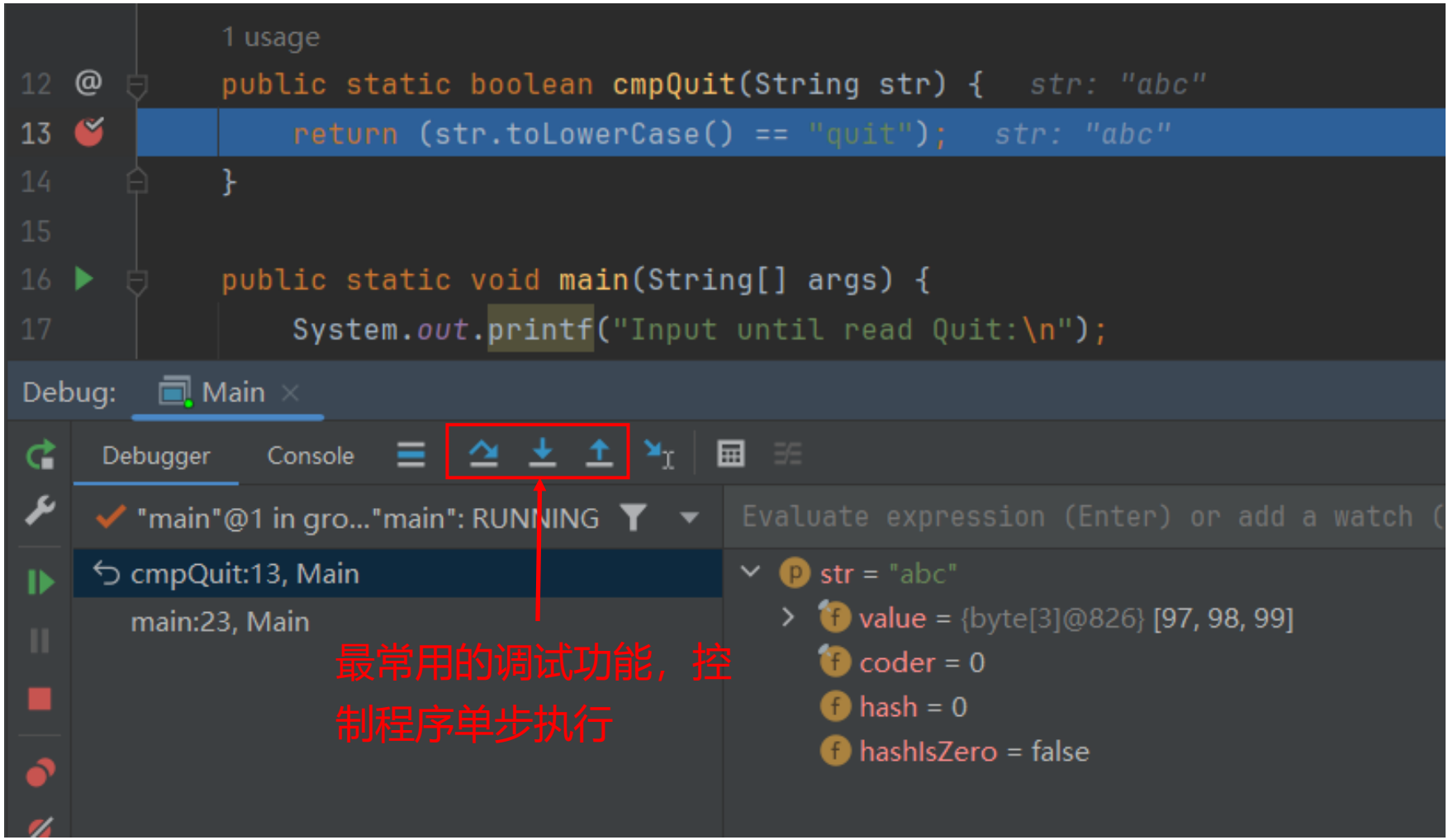


通过调试模式运行

调试


1 usage
12 @ public static boolean cmpQuit(String str) { str: "abc"
13 return (str.toLowerCase() == "quit"); str: "abc"
14 }
15
16 public static void main(String[] args) {
17 System.out.printf("Input until read Quit:\n");
Debug: Main
Debugger
Console
"main"@1 in gro..."main": RUNNING
Evaluate expression (Enter) or add a watch (
cmpQuit:13, Main
main:23, Main
str = "abc"
value = {byte[3]@826} [97, 98, 99]
coder = 0
hash = 0
hashIsZero = false

调试

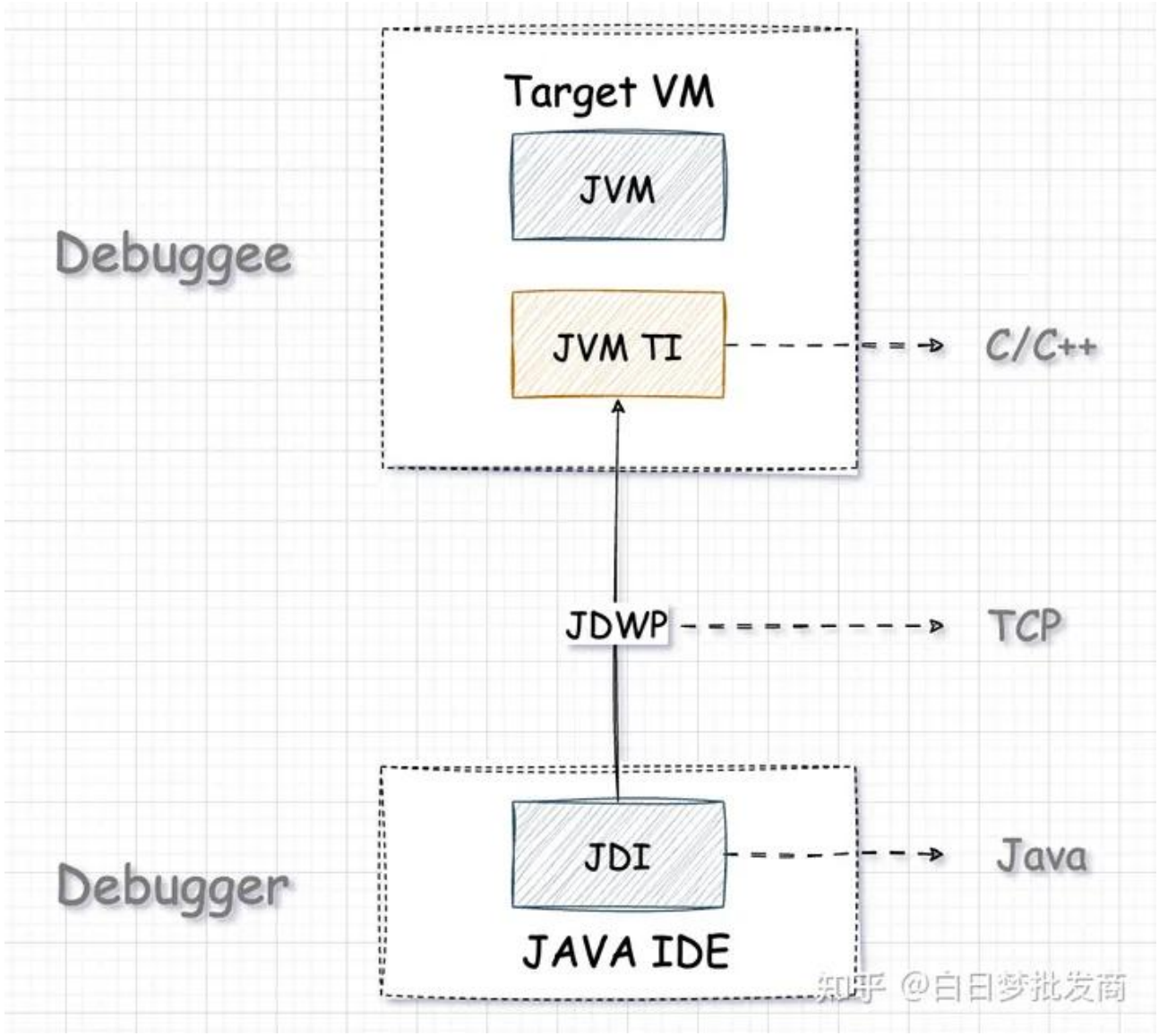

1 usage
12 @ public static boolean cmpQuit(String str) { str: "abc"
13 return (str.toLowerCase() == "quit"); str: "abc"
14 }
15
16 public static void main(String[] args) {
17 System.out.printf("Input until read Quit:\n");
Debug: Main
Debugger
Console
"main"@1 in gro..."main": RUNNING
Evaluate expression (Enter) or add a watch
cmpQuit:13, Main
main:23, Main
str = "abc"
value = {byte[3]@826} [97, 98, 99]
coder = 0
hash = 0
hashIsZero = false
最常用的调试功能，控制程序单步执行

调试器

查询：栈帧信息

动作：设置断点

通知：断点命中



https://zhuanlan.zhihu.com/p/674064216

在线调试

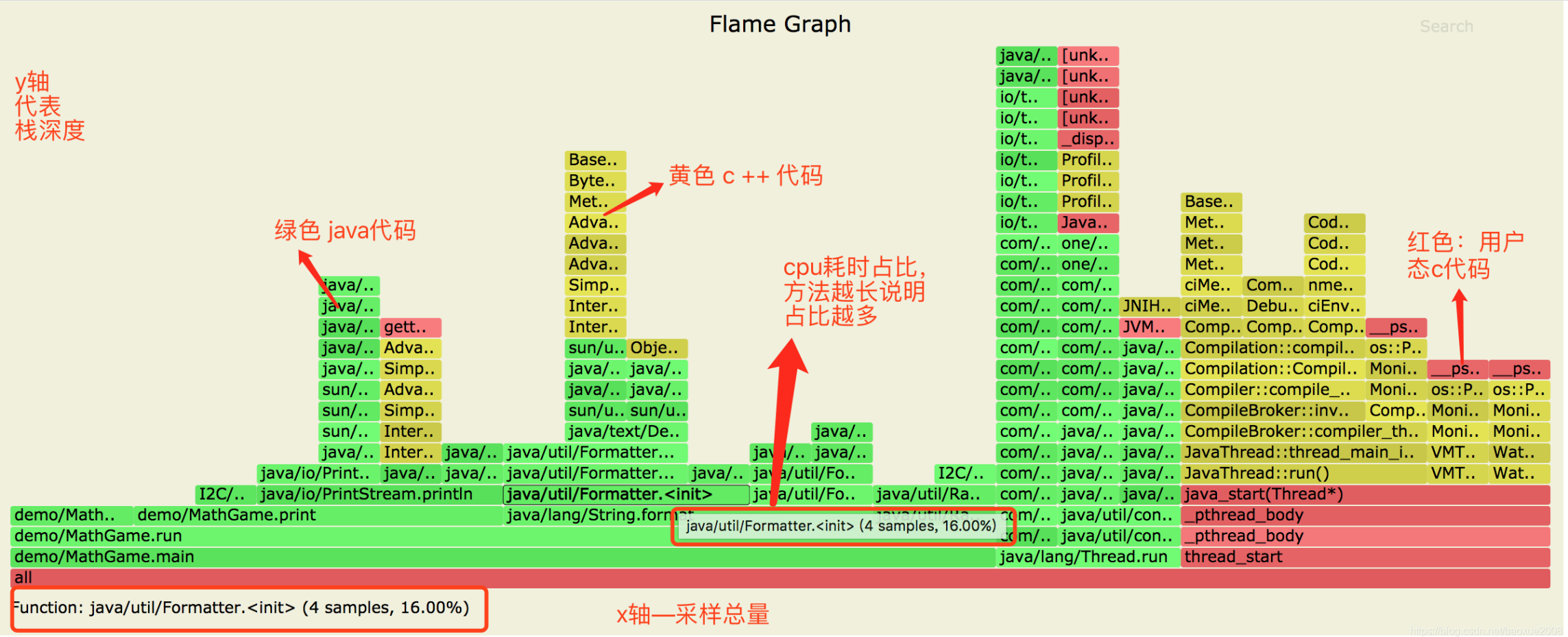
- Java: Arthas
  - debug attach
  - watch/trace

Arthas Trace

```
`---ts=2021-08-17 16:28:00;thread_name=http-nio-8080
    `---[4046.398447ms] xxxService.controller.Helios
        +---[0.022259ms] xxxService.model.helios.Hel
        +---[0.007132ms] xxxService.model.helios.Hel
```

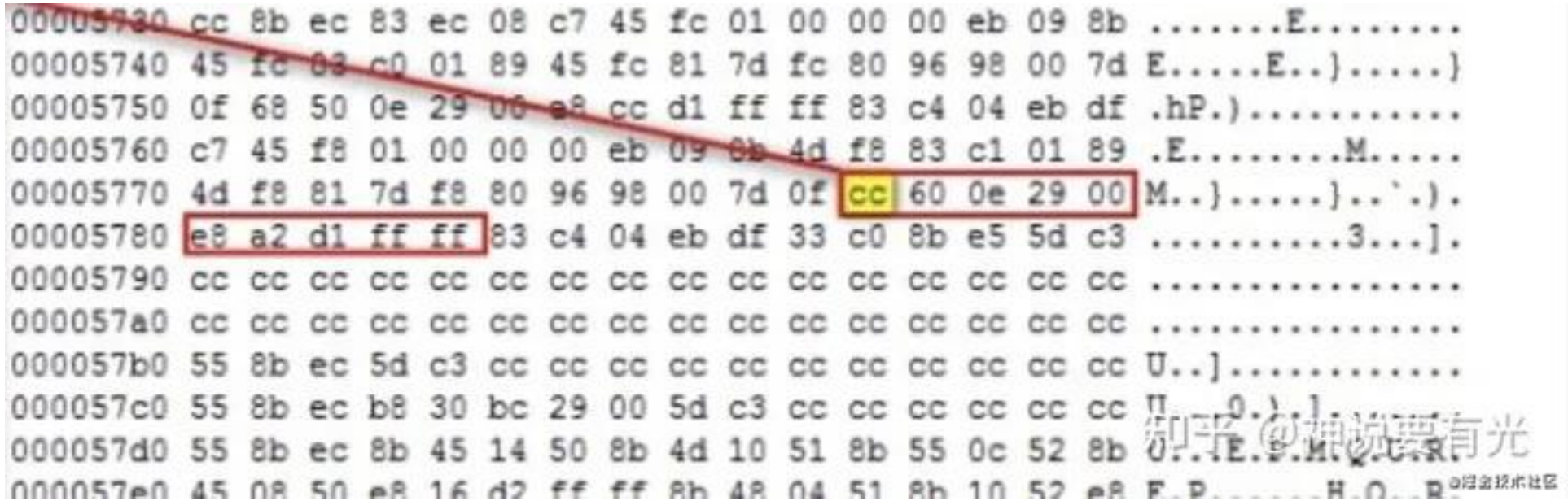
```
061003ms,total=94.294061ms,count=175680] xxxService.model.helios.HeliosGetScoreRequest:getEndTime() #16
038792ms,total=74.505056ms,count=175680] java.util.Date:compareTo() #168
036548ms,total=87.693935ms,count=175680] xxxService.helios.entity.HeliosScore:getTimeFrom() #171
.068413ms,total=391.739063ms,count=175680] xxxService.utils.DateUtils$yyyyMMddHHmm:formatDate() #171
037904ms,total=108.107714ms,count=175680] java.util.Set:add() #171
031555ms,total=88.173857ms,count=175680] xxxService.helios.entity.HeliosScore:getScores() #172
033584ms,total=84.689466ms,count=175680] xxxService.helios.entity.HeliosScore:getScores() #176
```

在线调试


Flame Graph
Search
y轴
代表
栈深度
黄色 c ++ 代码
绿色 java代码
cpu耗时占比，
方法越长说明
占比越多
红色：用户
态c代码
java/..
[unk..
io/t..
_disp..
Profil..
Base..
Byte..
Met..
Adva..
Simp..
Inter..
Java..
Cod..
one/..
com/..
JNIH..
JVM..
ciMe..
Com..
Debu..
nme..
ciEnv..
Comp..
__ps..
os::P..
Moni..
sun/u..
Obje..
gett..
sun/..
java/text/De..
java/io/Print..
I2C/..
java/io/PrintStream.println
java/util/Formatter...
java/util/Formatter.<init>
java/util/Fo..
java/util/Ra..
Compilation::compil..
Compilation::Compil..
Compiler::compile_..
CompileBroker::inv..
CompileBroker::compiler_th..
JavaThread::thread_main_i..
JavaThread::run()
java_start(Thread*)
_pthread_body
java/util/con..
demo/Math..
demo/MathGame.print
java/lang/String.format
demo/MathGame.run
demo/MathGame.main
java/lang/Thread.run
thread_start
all
java/util/Formatter.<init> (4 samples, 16.00%)
Function: java/util/Formatter.<init> (4 samples, 16.00%)
x轴—采样总量

调试器

- int 3
- 0xCC



https://segmentfault.com/a/1190000030685251

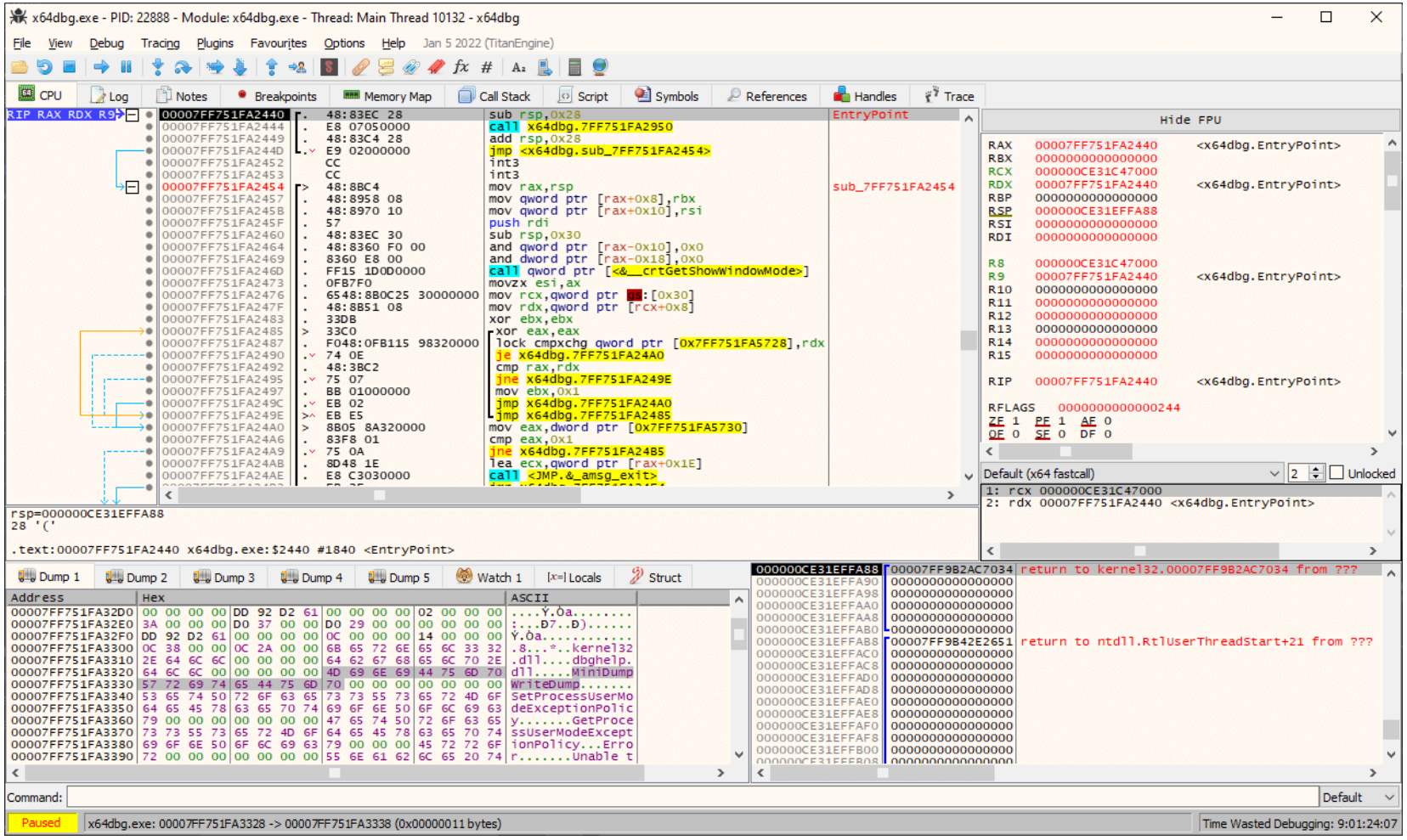
https://zhuanlan.zhihu.com/p/372135871

调试器

```
(gdb) b 12
Breakpoint 1 at 0x400547: file test.c, line 12.
(gdb) run
Starting program: /home/wanghe/test.exe
Breakpoint 1, main (argc=1, argv=0x7fffffffdff8) at test.c:12
12          int b = -1;
Missing separate debuginfos, use: debuginfo-install glibc-2.17-260.el7_6.5.x86_64
(gdb) watch b
Hardware watchpoint 2: b
(gdb) continue
Continuing.
Hardware watchpoint 2: b

Old value = 0
New value = 1
main (argc=1, argv=0x7fffffffdff8) at test.c:17
17          printf("%d\n",b);
```

调试二进制文件/core文件

- x64dbg
  - adb无法运行问题
- windbg
- gdb
  - linux下可用



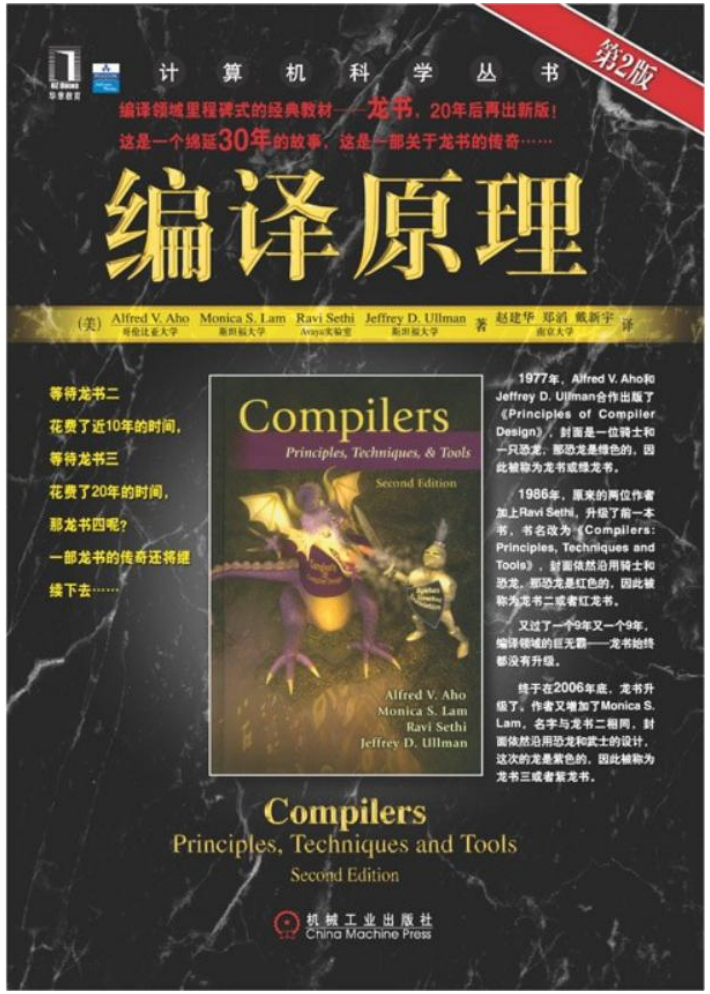
总结

- 编译原理
  - 根据源代码生成可执行文件
  - 考试内容
    - 讲课内容与教材的交集
    - 主要是几个算法
  - 平时成绩30%
  - 考试成绩70%
  - QQ群
- 成绩
  - 大实验 10%
    - 主要决定是否能上90分
    - 今年试行，少部分同学参加
  - 作业考勤 20%
  - 考试 70%
  - 及格60=35+25=50*0.7+25

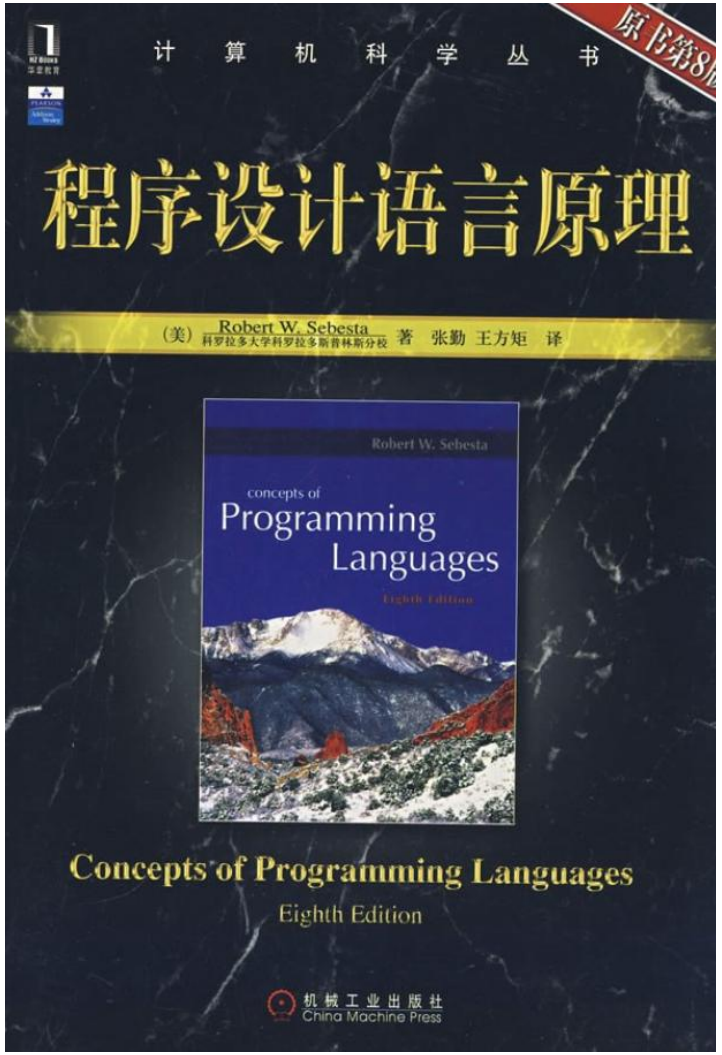
学科前沿

- 深度学习
  - Nvidia Cuda
- FPGA
  - 并行程序编译
- 量子计算
  - 量子编译器

参考书籍


计 算 机 科 学 丛 书
第2版
编译领域里程碑式的经典教材——龙书，20年后再出新版！
这是一个绵延30年的故事，这是一部关于龙书的传奇……
编译原理
(美) Alfred V. Aho Monica S. Lam Ravi Sethi Jeffrey D. Ullman 著 赵建华 郑滔 戴新宇 译
等待龙书二
花费了近10年的时间，
等待龙书三
花费了20年的时间，
那龙书四呢？
一部龙书的传奇还将继续下去……
Compilers
Principles, Techniques, & Tools
Second Edition
Alfred V. Aho
Monica S. Lam
Ravi Sethi
Jeffrey D. Ullman
1977年，Alfred V. Aho和Jeffrey D. Ullman合作出版了《Principles of Compiler Design》，封面是一位骑士和一只恐龙，那恐龙是棕色的，因此被称为龙书或棕龙书。
1986年，原来的两位作者加上Ravi Sethi，升级了前一本书，书名改为《Compilers: Principles, Techniques and Tools》，封面依然沿用骑士和恐龙，那恐龙是红色的，因此被称为龙书二或者红龙书。
又过了一个9年又一个9年，编译领域的巨无霸——龙书始终都没有升级。
终于在2006年底，龙书升级了，作者又增加了Monica S. Lam，名字与龙书二相同，封面依然沿用恐龙和武士的设计，这次的龙是紫色的，因此被称为龙书三或者紫龙书。
Compilers
Principles, Techniques and Tools
Second Edition
机械工业出版社
China Machine Press


TURING 图灵程序设计丛书
ELSEVIER
编译器设计
（第2版）
Engineering a Compiler
Second Edition
[美] Keith D. Cooper Linda Torczon 著 郭旭 译
人民邮电出版社
POSTS & TELECOM PRESS


计 算 机 科 学 丛 书
原书第8版
程序设计语言原理
(美) Robert W. Sebesta 著 张勤 王方矩 译
Robert W. Sebesta
concepts of
Programming
Languages
Eighth Edition
Concepts of Programming Languages
Eighth Edition
机械工业出版社
China Machine Press