调试器的基本原理

lazyparser@gmail.com

2016-10-14

大道理：程序猿(媛)的自我修养

真技术：现场给各位写个调试器

老八卦：计算机领域有趣的典故

大道理

测试 | 调试 | 修复

查明错误原因 | 定位错误根源

调试这个事情，很难的

谁调试过谁知道 ☺

“Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.”

– Brian Kernighan

来源: 维基quotes百科

调试能力：稀缺资源

难度大 | 功底深 | 套路少 | 没人教

调试经验作为面试依据

简历普遍浮夸，调试经验无法造假

听了这么多大道理

你依然调不好手头的代码 :)

持之以恒，耐住寂寞。

多喝点热水，加油!

那么，来写一个调试器吧

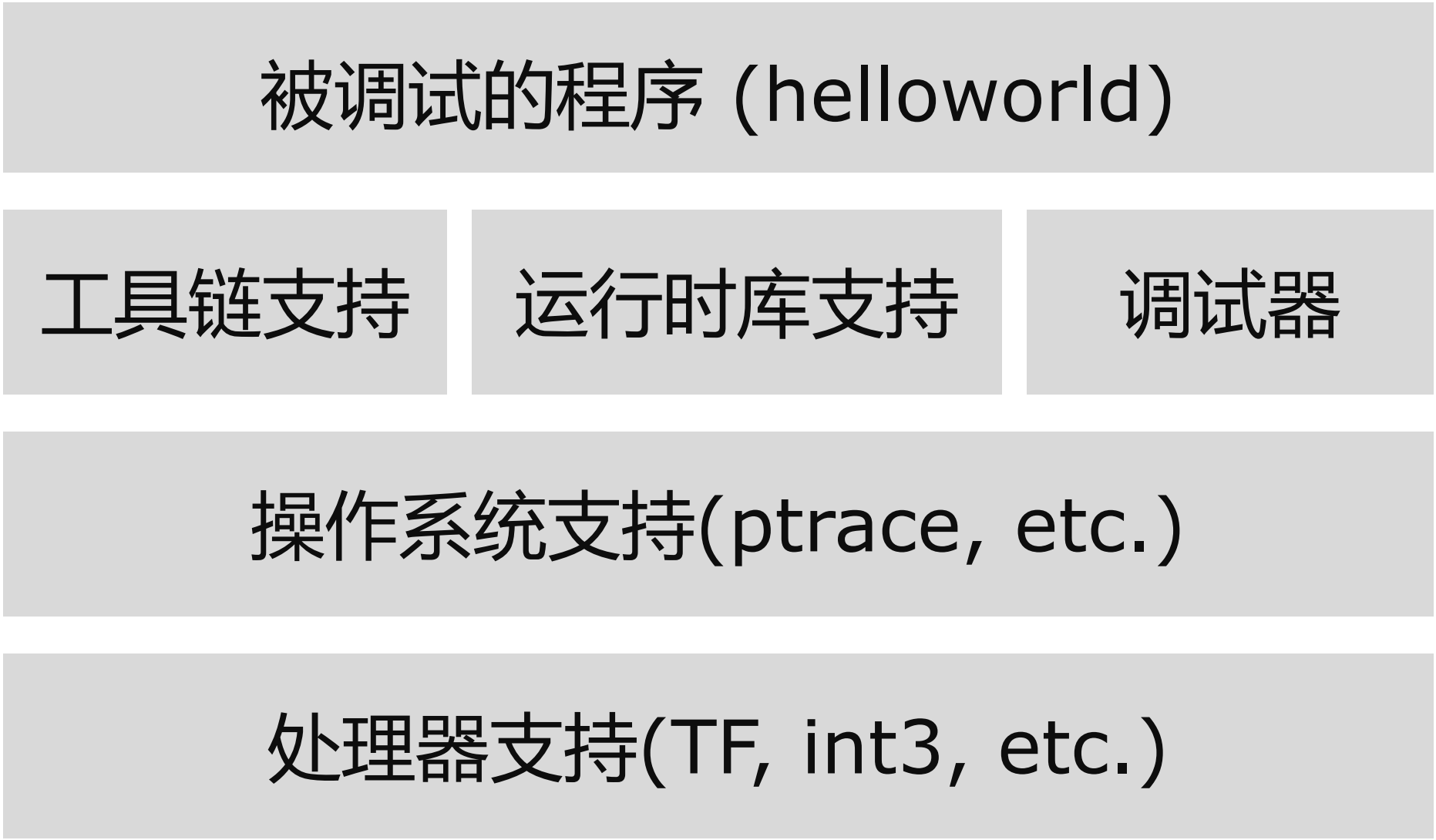
不难的不难的真的不难的

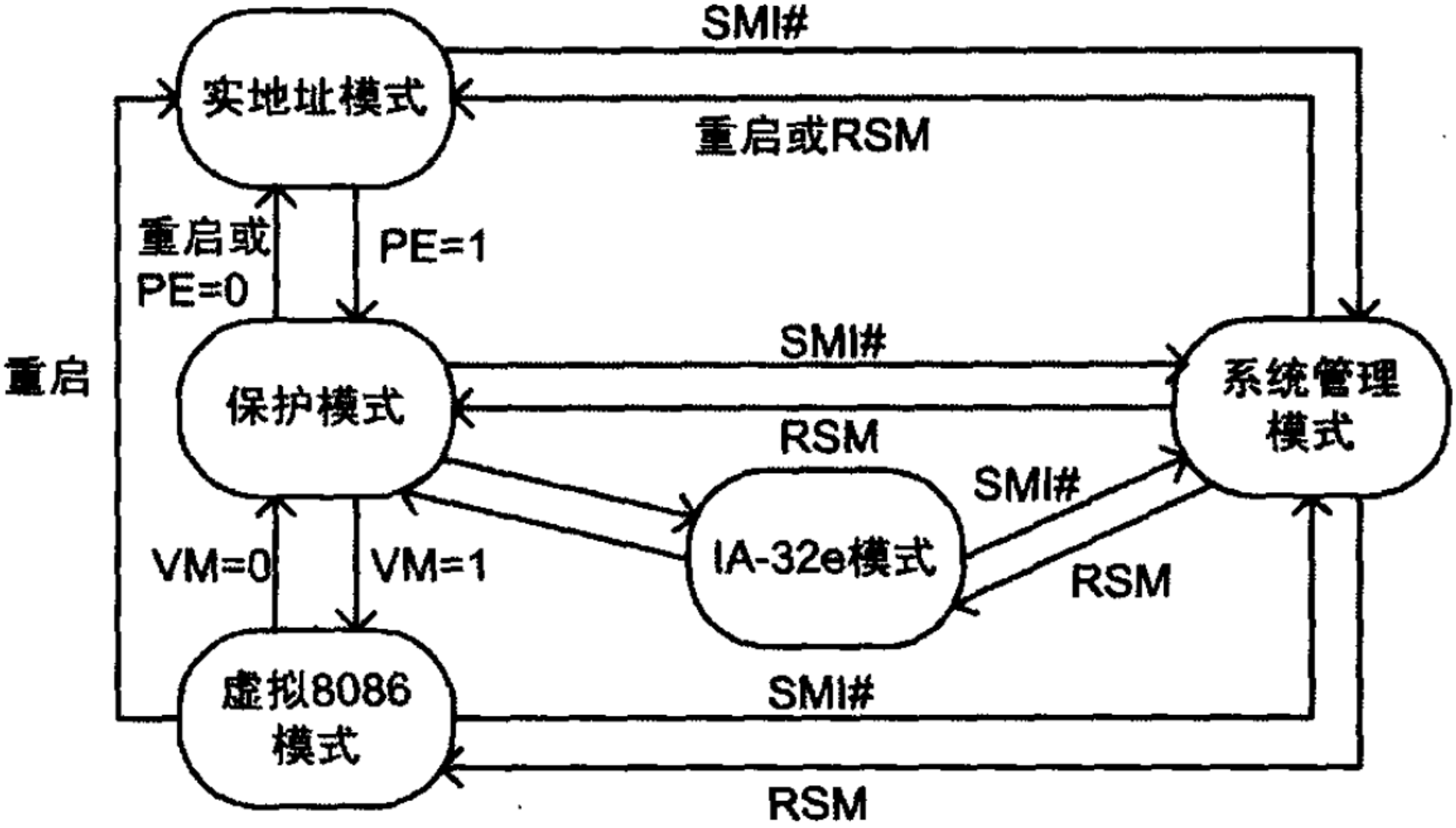
停下来 | 看看 | 戳戳

我们先给自己设定一个小目标：
调试器三大基础功能

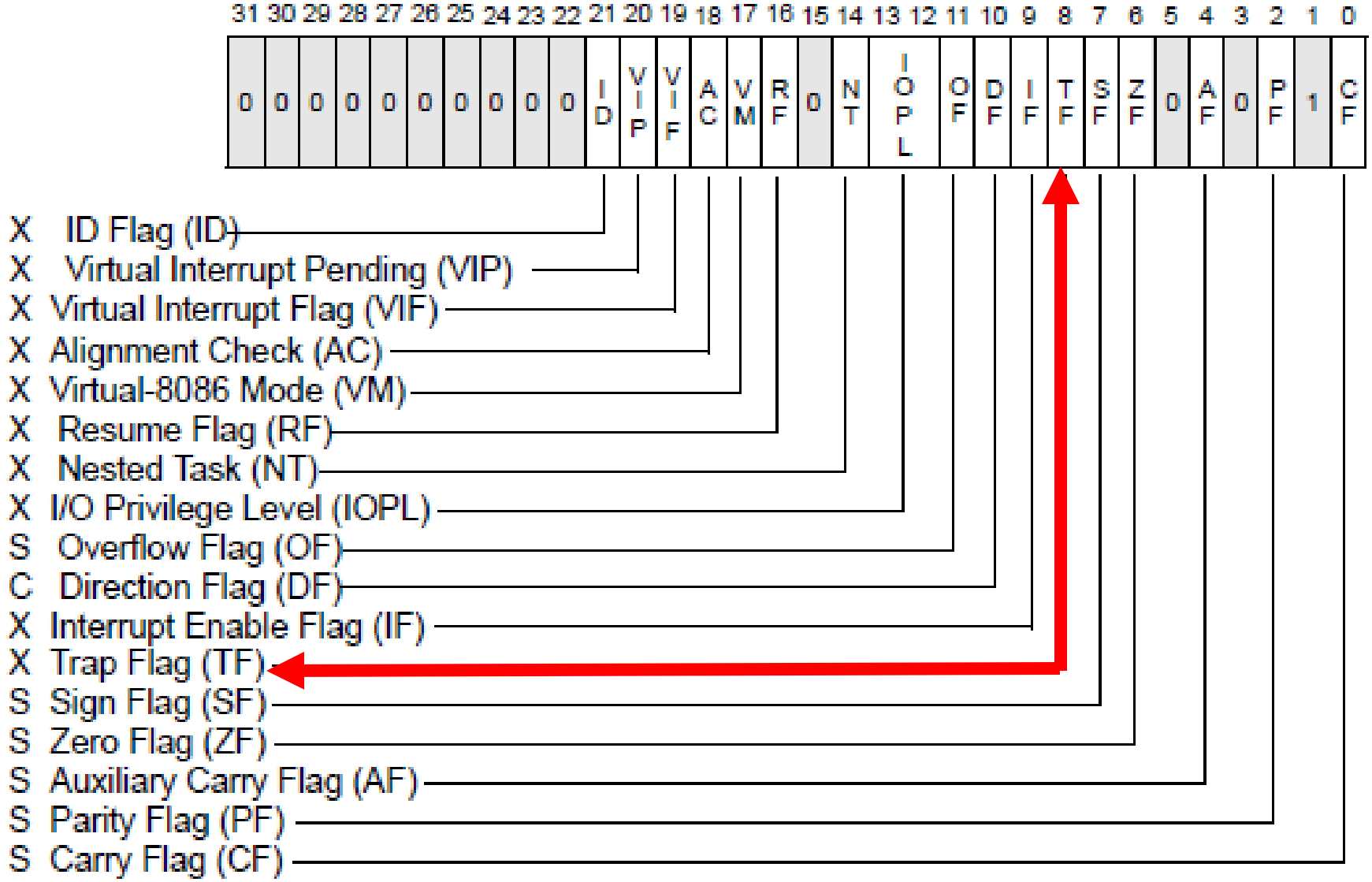
一点点预备知识

CPU | 汇编 | 进程 | 编译 | 链接 | 加载


被调试的程序 (helloworld)
工具链支持
运行时库支持
调试器
操作系统支持(ptrace, etc.)
处理器支持(TF, int3, etc.)


SMI#
实地址模式
重启或RSM
重启或
PE=0
PE=1
重启
SMI#
保护模式
系统管理
模式
RSM
SMI#
VM=0
VM=1
IA-32e模式
RSM
虚拟8086
模式
SMI#
RSM

来源：张银奎《软件调试》


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0 0 0 ID VIP VIF AC VM RF 0 NT IOPL OF DF IF TF SF ZF 0 AF 0 PF 1 CF
X ID Flag (ID)
X Virtual Interrupt Pending (VIP)
X Virtual Interrupt Flag (VIF)
X Alignment Check (AC)
X Virtual-8086 Mode (VM)
X Resume Flag (RF)
X Nested Task (NT)
X I/O Privilege Level (IOPL)
S Overflow Flag (OF)
C Direction Flag (DF)
X Interrupt Enable Flag (IF)
X Trap Flag (TF)
S Sign Flag (SF)
S Zero Flag (ZF)
S Auxiliary Carry Flag (AF)
S Parity Flag (PF)
S Carry Flag (CF)
S Indicates a Status Flag
C Indicates a Control Flag
X Indicates a System Flag

来源：张银奎《软件调试》

```
  400521:     e9 7a ff ff ff          jmpq   4004a0 <register_tm_clones>

0000000000400526 <main>:
  400526:     55                      push   %rbp
  400527:     48 89 e5                mov    %rsp,%rbp
  40052a:     bf c4 05 40 00          mov    $0x4005c4,%edi
  40052f:     b8 00 00 00 00          mov    $0x0,%eax
  400534:     e8 c7 fe ff ff          callq  400400 <printf@plt>
  400539:     b8 00 00 00 00          mov    $0x0,%eax
  40053e:     5d                      pop    %rbp
  40053f:     c3                      retq
```

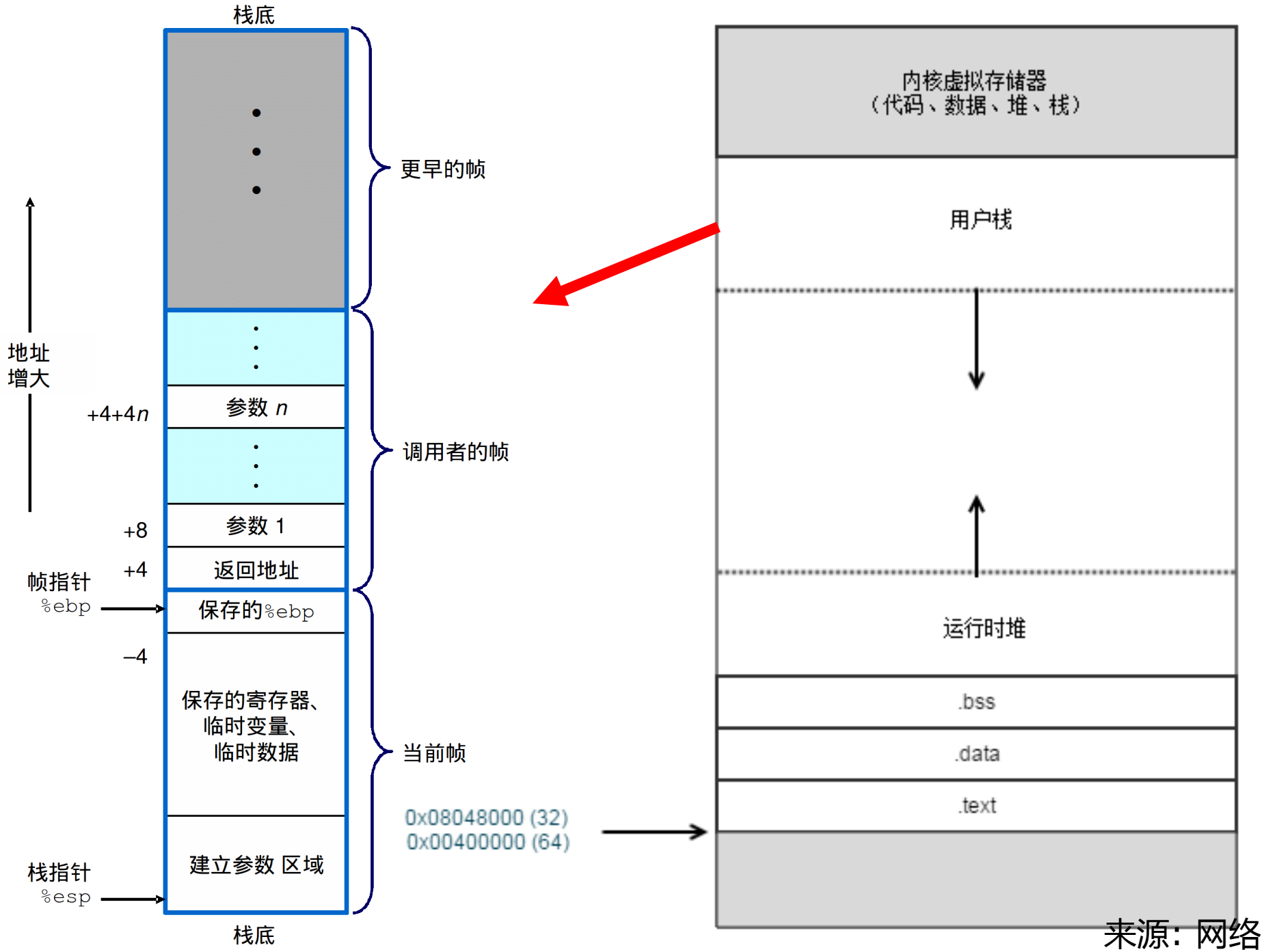
```
/* Hello World program */


#include<stdio.h>

main() { printf("Hello World"); }
```

int3

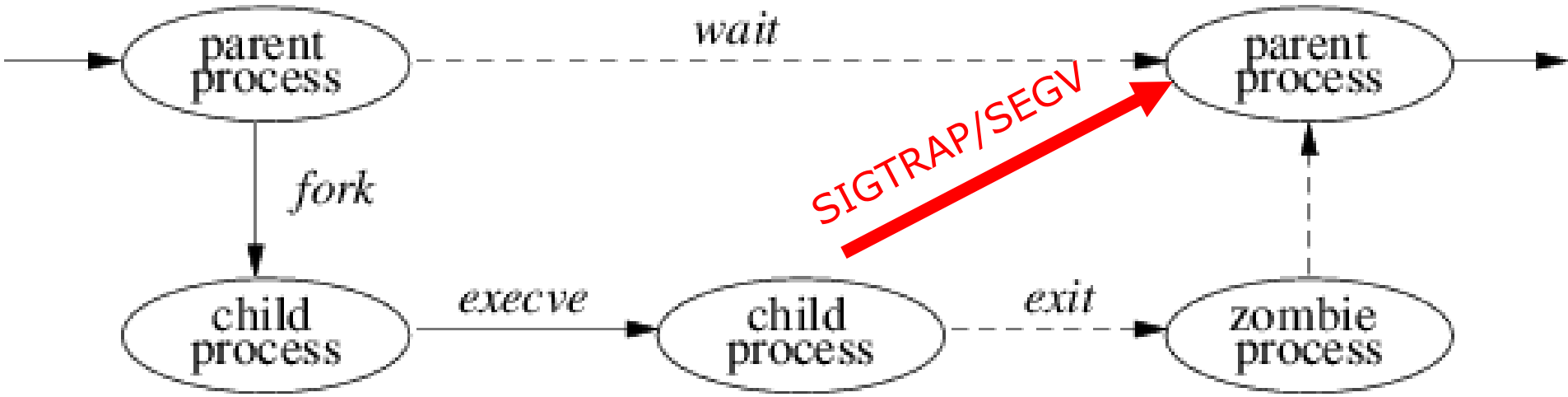
0xCC

int dd

0xCD dd


栈底
更早的帧
地址
增大
+4+4n
参数 n
调用者的帧
+8
参数 1
+4
返回地址
帧指针
%ebp
保存的%ebp
–4
保存的寄存器、
临时变量、
临时数据
当前帧
栈指针
%esp
建立参数 区域
栈底
0x08048000 (32)
0x00400000 (64)
内核虚拟存储器
（代码、数据、堆、栈）
用户栈
运行时堆
.bss
.data
.text

来源：网络


parent process
wait
parent process
fork
SIGTRAP/SEGV
child process
execve
child process
exit
zombie process

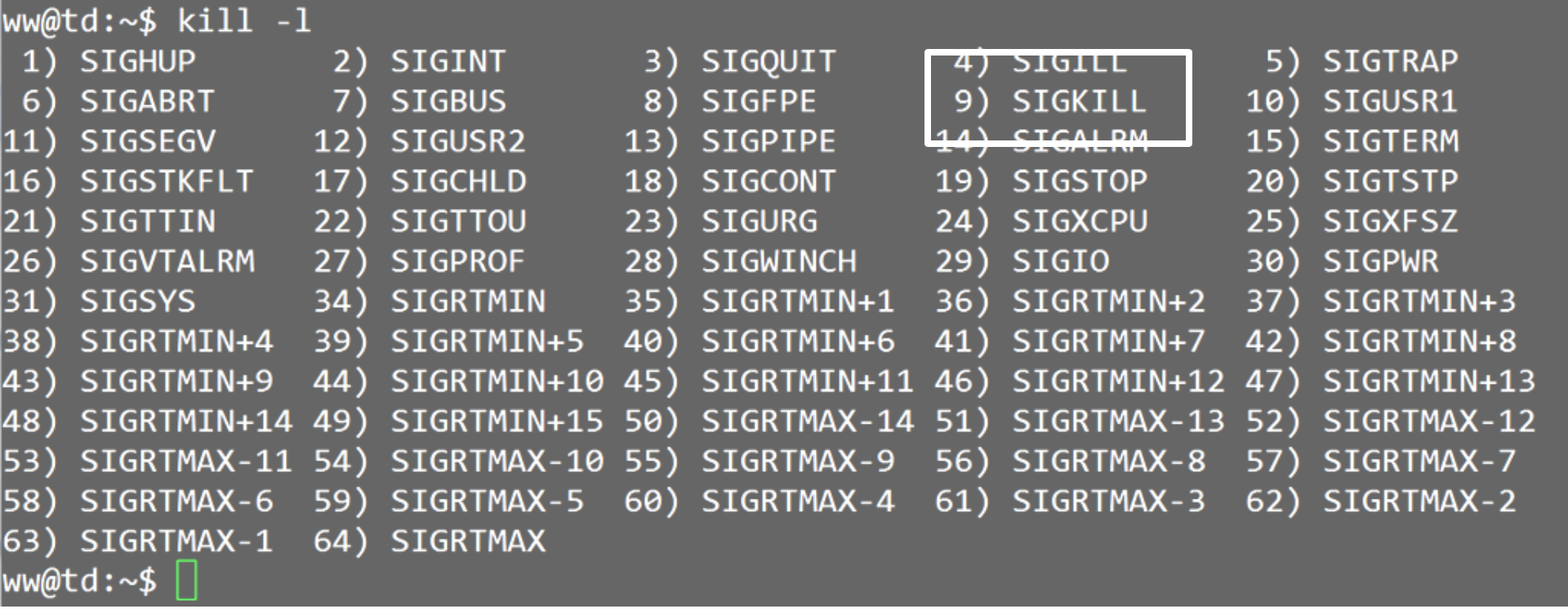
来源：网络

```
ww@td:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
ww@td:~$
```

SIGnals – 进程间消息传递

```
ww@td:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
ww@td:~$
```

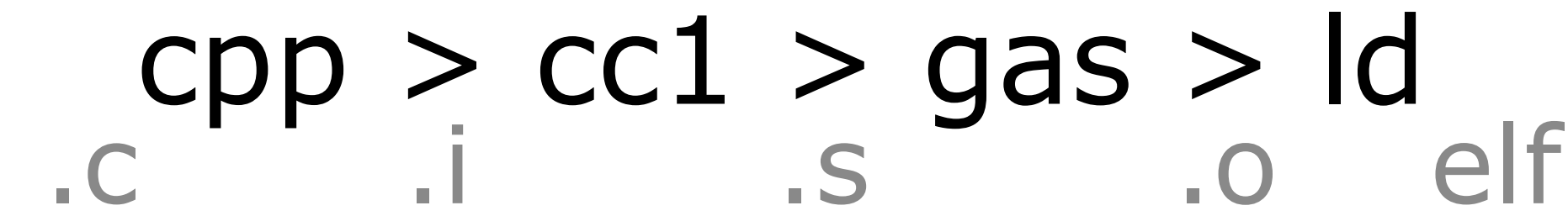
SIGnals – 进程间消息传递



SIGnals – 进程间消息传递

```
ww@td:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
ww@td:~$
```

SIGnals – 进程间消息传递


cpp > cc1 > gas > ld
.c
.i
.s
.o
elf

ptrace

Tiny-Debugger的实现基础

process trace

同时也是GDB的实现基础

```
#include <sys/ptrace.h>

long
ptrace(enum __ptrace_request request,
       pid_t pid_of_your_child,
       void *addr,
       void *data);
```

```
ptrace(PTRACE_TRACEME, 0, 0, 0); //on child proc

ptrace(PTRACE_SINGLESTEP, …)
ptrace(PTRACE_CONT, …)

ptrace(PTRACE_PEEK{TEXT,DATA,USER}, pid, addr, 0);
ptrace(PTRACE_POKE{TEXT,DATA,USER}, pid, addr, val);
ptrace(PTRACE_GET{REGS,PREGS}, pid, 0, &struct);
ptrace(PTRACE_SET{REGS,PREGS}, pid, 0, &struct);
ptrace(PTRACE_GETREGSET, pid, NT_foo, &iov);
ptrace(PTRACE_SETREGSET, pid, NT_foo, &iov);
ptrace(PTRACE_GETSIGINFO, pid, 0, &siginfo);
ptrace(PTRACE_SETSIGINFO, pid, 0, &siginfo);
ptrace(PTRACE_GETEVENTMSG, pid, 0, &long_var);
ptrace(PTRACE_SETOPTIONS, pid, 0, PTRACE_O_flags);
```

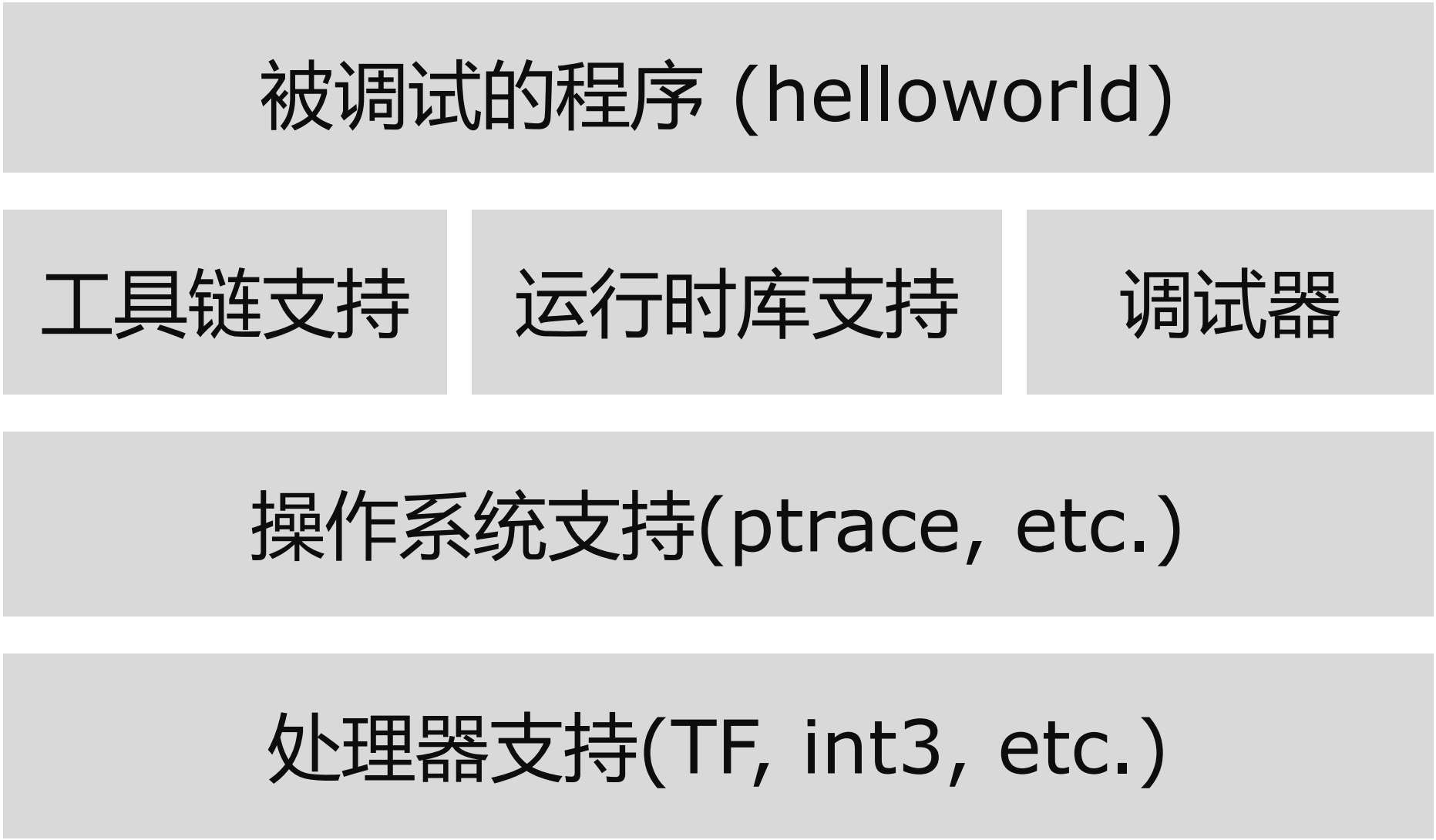
准备完毕，开始写调试器

从 Hello World 开始

```
/* Hello World program */

#include<stdio.h>
main() { printf("Hello World"); }
```

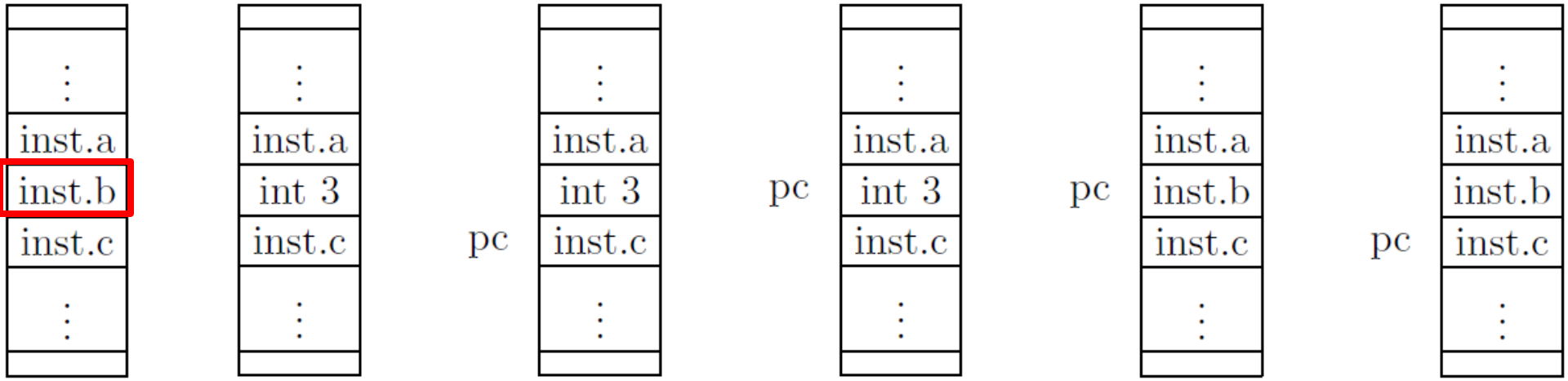
simple-debugger.c

码农(妇)们激动不已的时刻到来了!!

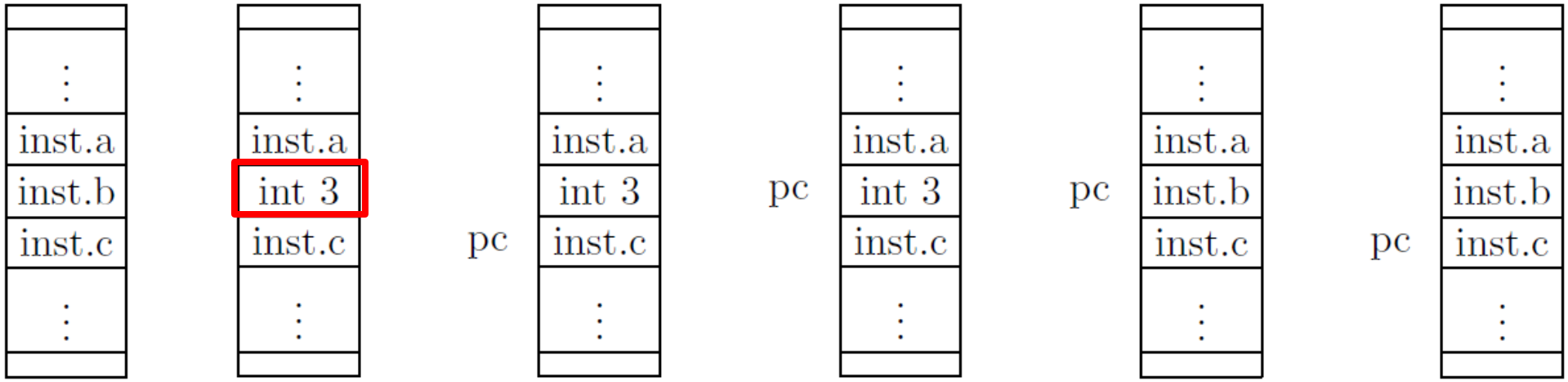

被调试的程序 (helloworld)
工具链支持
运行时库支持
调试器
操作系统支持(ptrace, etc.)
处理器支持(TF, int3, etc.)

hellodebugger.c

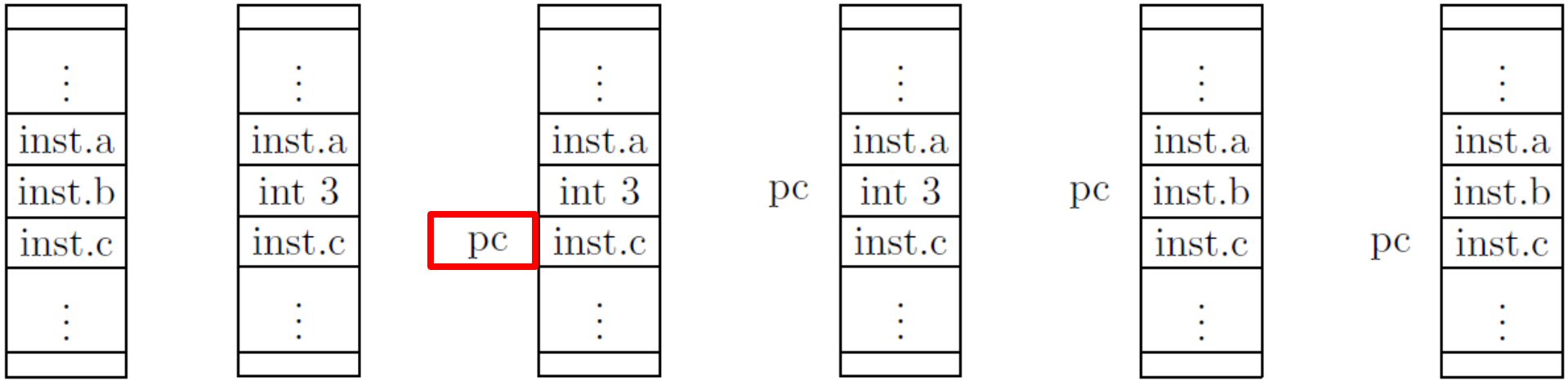
父知子，子不知父



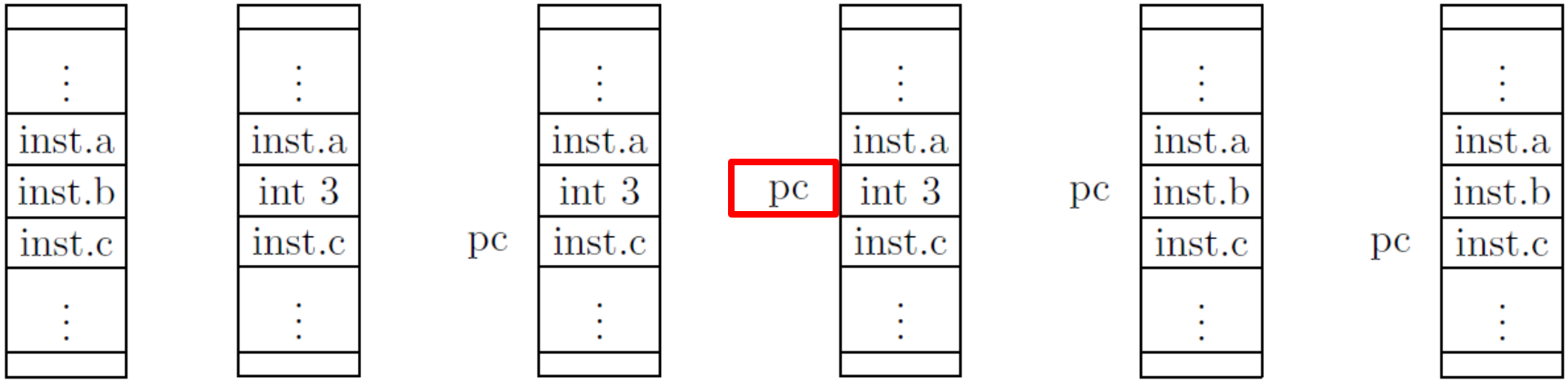
来源: 齐尧《Debugger Not in Depth》



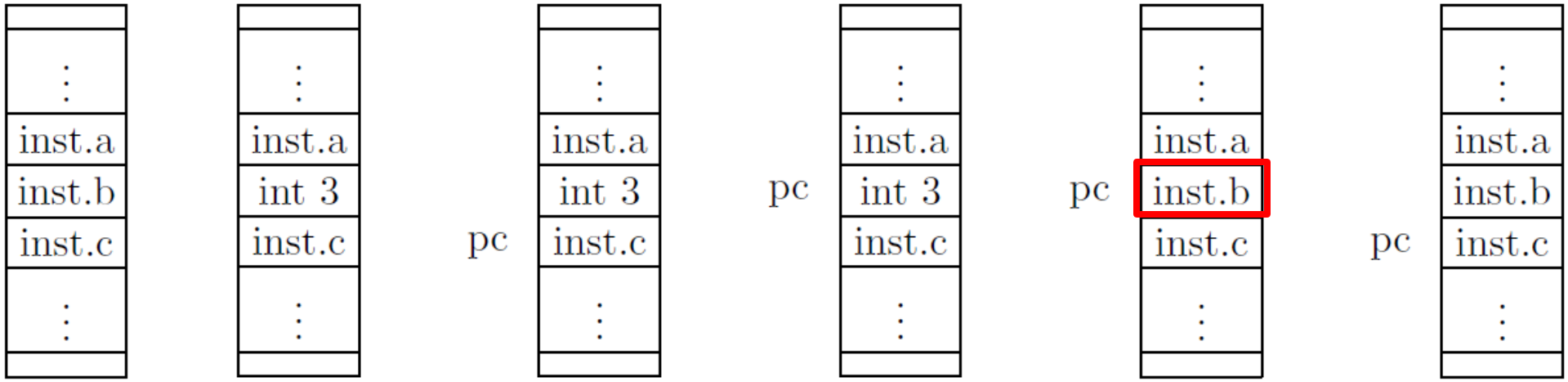
来源: 齐尧《Debugger Not in Depth》



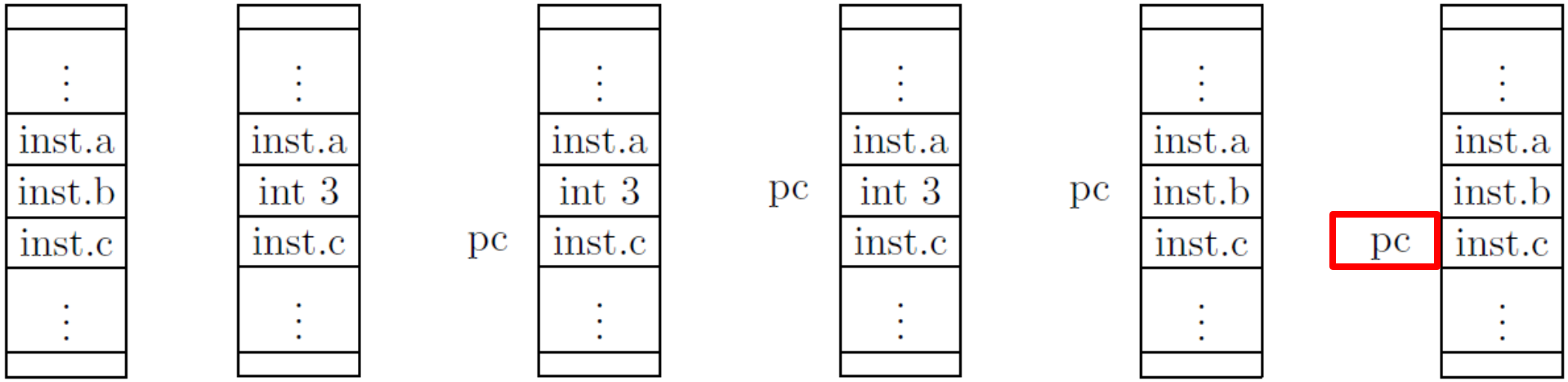
来源: 齐尧《Debugger Not in Depth》



来源: 齐尧《Debugger Not in Depth》



来源: 齐尧《Debugger Not in Depth》



来源: 齐尧《Debugger Not in Depth》

tiny-debugger.c

码农(妇)们激动不已的时刻终于到来了 ☺

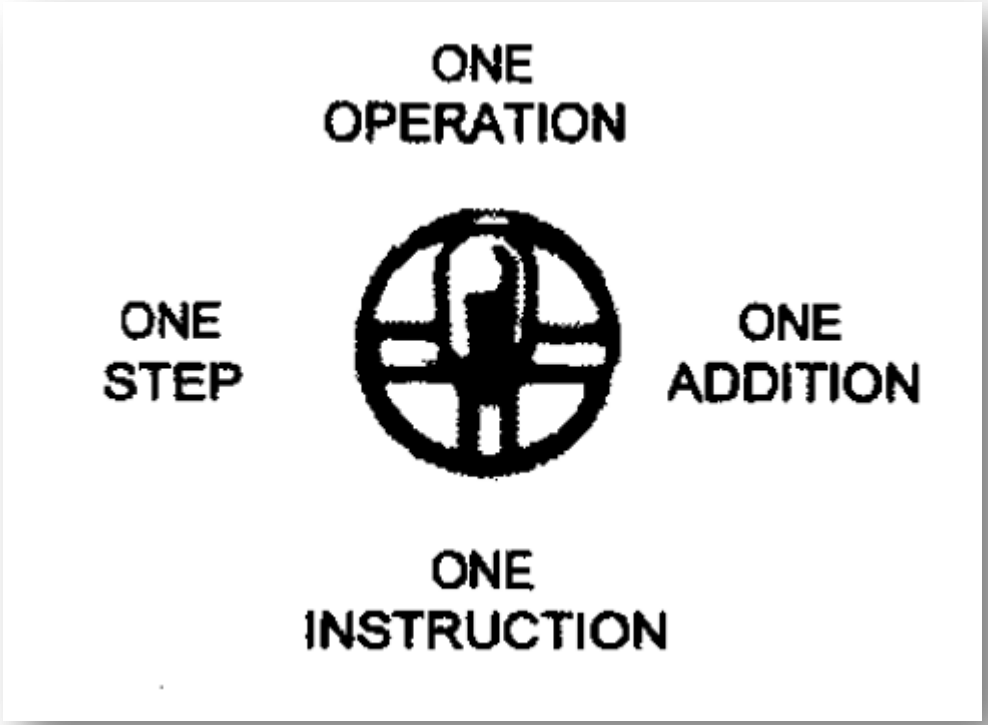
我们已经实现了一个调试器

运功试试. 您有没有觉得功力大增?

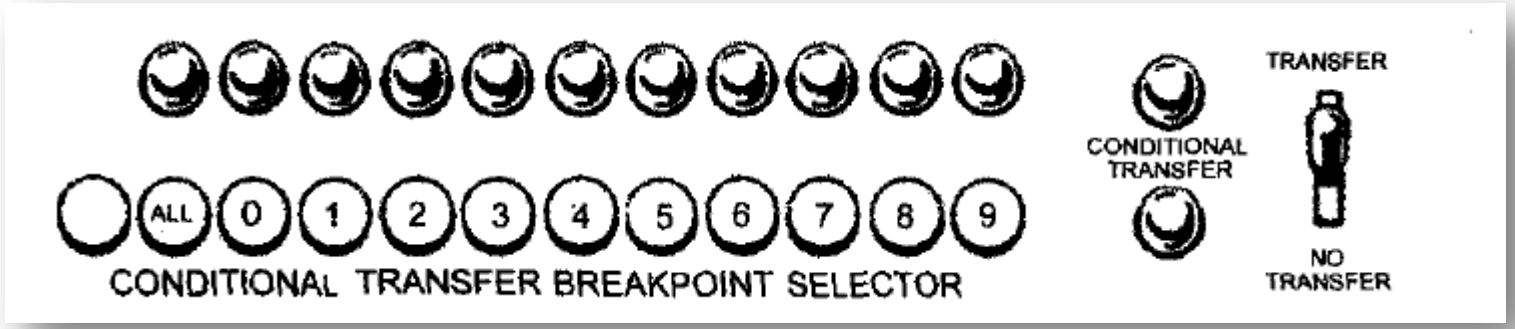
聊点八卦，冷静一下

让时间退回到1940年代

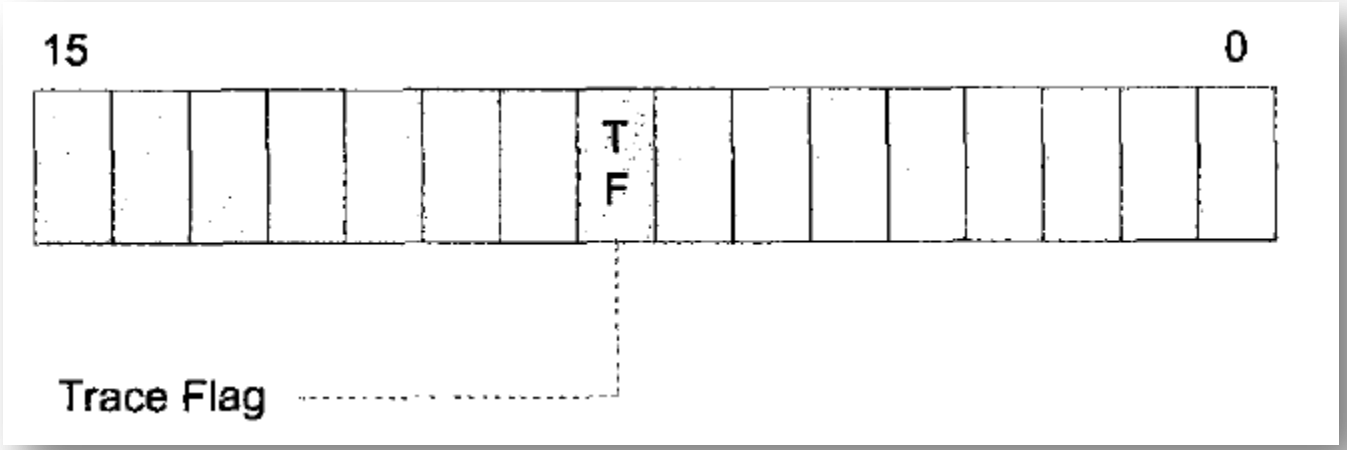

来源：维基百科



来源：张银奎《软件调试》


ALL
0
1
2
3
4
5
6
7
8
9
CONDITIONAL TRANSFER BREAKPOINT SELECTOR
CONDITIONAL TRANSFER
TRANSFER
NO TRANSFER

来源：张银奎《软件调试》



来源：张银奎《软件调试》

来源：维基百科

9/9

0800 Arctan started

1000 " stopped - arctan ✓ { 1.2700 9.037 847 025
9.037 846 ?95 correct

13" vc (032) MP - MC ~~1.9821447000~~ ~~2.13047641 5 (-3)~~ 4.615925059(-2)

(033) PRO 2 2.130476415

correct 2.130676415

Relays 6-2 in 033 failed special speed test

In Relay " 11.00 test.

Relays changed

Relay 2145
Relay 3370

1100 Started Cosine Tape (Sine check)

1525 Started Multy Adder Test.

1545 

Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

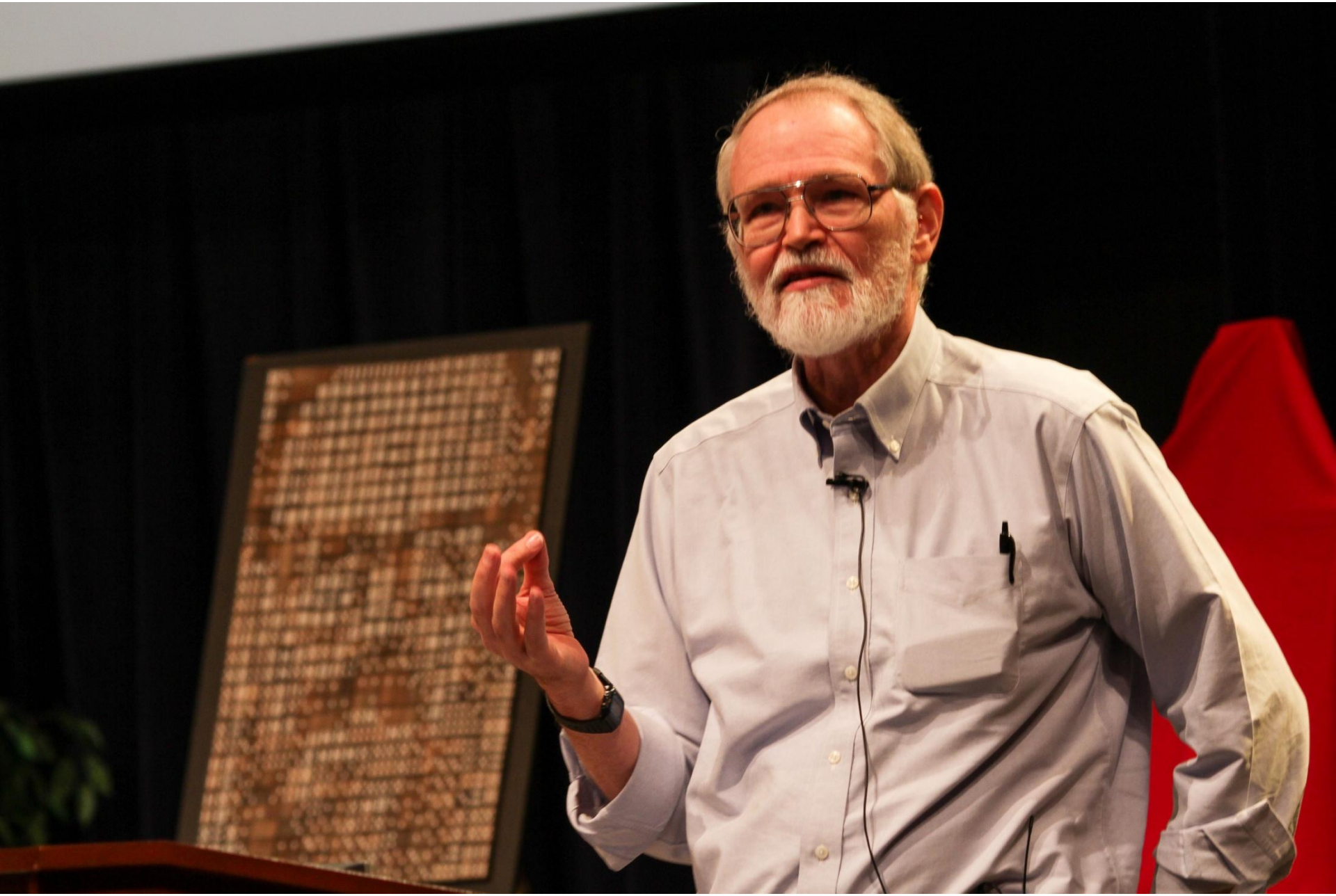
~~1645~~ 1630 Arctangent started.

1700 closed down.

“From then on, when anything went wrong with a computer, we said it had bugs in it.”

– Grace M. Hopper

来源: 维基quotes百科



来源：维基百科

“Hello, world!\n”

SECOND EDITION

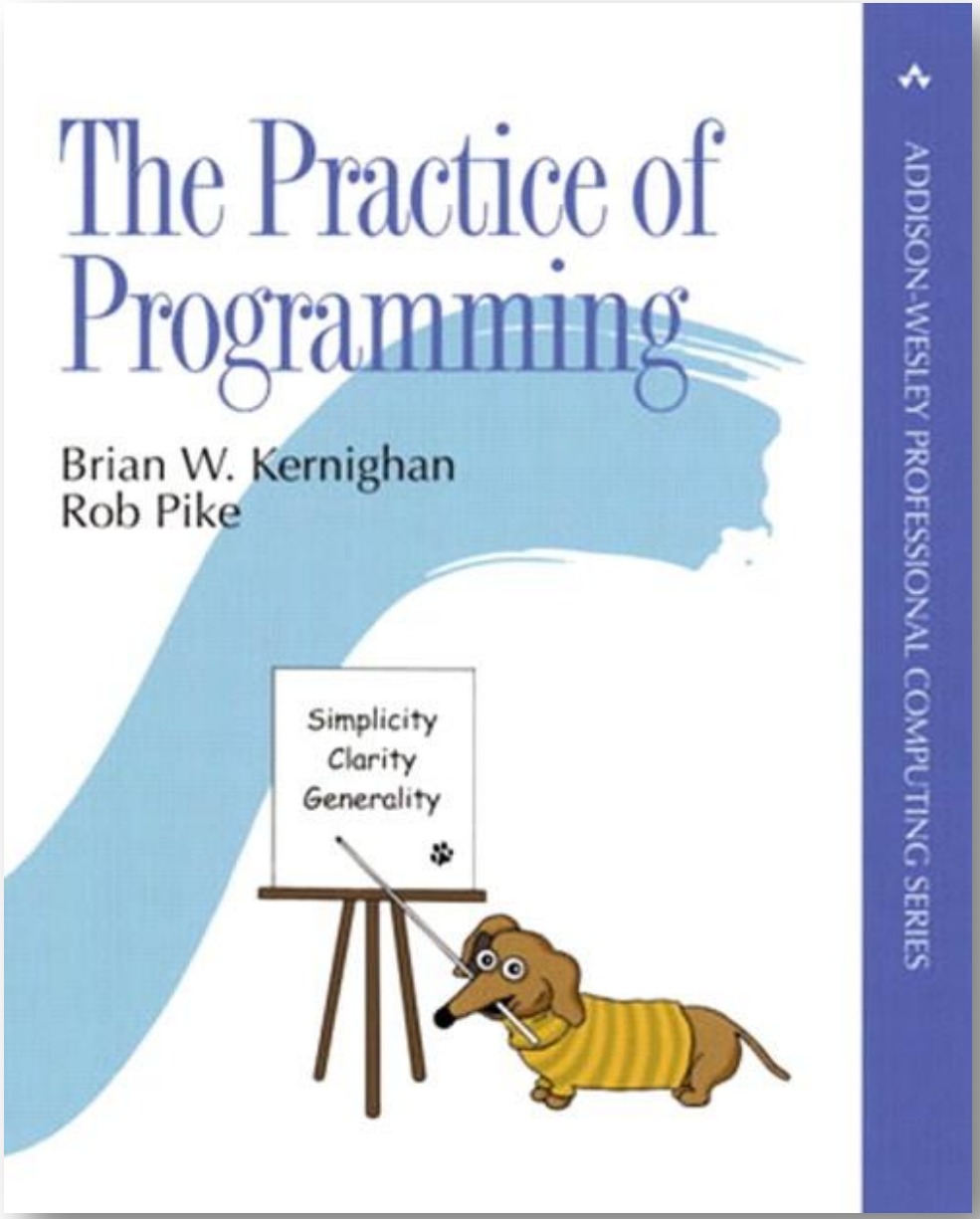
THE C PROGRAMMING LANGUAGE

BRIAN W. KERNIGHAN
DENNIS M. RITCHIE

PRENTICE HALL SOFTWARE SERIES

The AWK Programming Language

Alfred V. Aho
Brian W. Kernighan
Peter J. Weinberger


The Practice of Programming
Brian W. Kernighan
Rob Pike
Simplicity
Clarity
Generality
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

齐尧，国内首个GDB global maintainer。

谢谢各位坚持到最后

运功试试. 您有没有觉得功力大增?

致谢 | Credits

- 调试器代码改编自齐尧先生著作
  - 《Debugger not in depth》
- 八卦历史及部分大道理来自张银奎先生著作
  - 《软件调试》
- 本文八卦内容及人物图片来自维基百科和谷歌