计算机系统基础
Programming Assignment

PA 0 实验环境配置与实验总览

2018年9月5日

课程信息

- PA和实验课上课时间：
  - 一班（苏丰老师）：周三5-6节　仙II - 214
  - 二班（唐杰老师）：周五5-6节　仙I - 207
- 授课团队
  - 讲师：汪亮，助教：待定，QQ群里已经有几位
- 信息
  - 课程主页：http://114.212.189.154:2000
  - CMS系统：http://cslabcms.nju.edu.cn/
  - 课程QQ群：601286965
    - 验证消息输入：大班PA实验 学号 姓名
    - 修改群名片为：学号 姓名

实验环境配置 – 虚拟机

- 安装虚拟机软件
  - Vmware
  - VirtualBox
- 下载debian操作系统iso镜像
  - 32位（i386）
  - 最小化安装（network install）
- 新建虚拟机并安装32位debian操作系统
  - 语言选择US_en
  - 地区选择中国并选择国内的源（source）
    - Locale选择united states
  - 选择一个桌面环境，推荐MATE

实验环境配置 – 配置用户和环境

- 配置客户操作系统和开发环境
  - 将用户添加到sudoer list，不允许使用root用户做实验
  - 具体过程参见Guide
- 安装vmware-tools或virtualbox增强功能
  - 具体过程参见Guide或者网上自行搜索解决

实验环境配置 – 获取PA实验环境

- 连接校园网并访问http://114.212.189.154:2000
  - 点击下载安装脚本按钮
  - 输入学号获取 学号_install.sh 脚本
    - 报错的话在群里联系助教
  - 将这个脚本放在虚拟机里合适的位置，如，/home/username/
  - 以普通用户身份执行 ./学号_install.sh并根据提示输入sudo密码、个人的邮箱和姓名，并解决问题
- 注意！不要将自己的脚本泄露给他人或在自己机器上执行他人的脚本，因此产生的问题自己负责。
- 脚本执行时需要保持对校外网络的访问通畅
- 在git clone的步骤会消耗一定的时间，需要下载大约46MB的内容

实验环境配置 – 获取PA实验环境

- 脚本执行成功后即可开始实验
- 参考的软件版本信息如下

类型	名称	版本
虚拟机	VMware® Workstation 12 Player	12.5.6 build-5528349
客户操作系统	Debian GNU/Linux 9 (i386)	9.0
桌面环境	MATE	1.16.2
编译器	gcc	6.3.0
readline开发包	libreadline-dev	7.0-3
SDL开发包	libsdl1.2-dev	1.2.15+dfsg1-4
PA实验代码	pa2018_fall	pa2018_fall

请注意版本号：gcc 版本是否是6.x很关键，SDL库必须是1.x版本而非2.x版本

PA实验的过程和提交方式

- pa2018_fall/的组织结构

```
pa2018_fall/
├──── game                  // 包含游戏相关代码
├──── include               // PA整体依赖的一些文件
│     └──── config.h        // 一些配置用的宏
├──── kernel                // 一个微型操作系统内核
├──── libs                  // 库文件
├──── Makefile              // 帮助编译和执行工程的Makefile
├──── Makefile.git          // 和git以及提交信息有关的部分
├──── nemu                  // NEMU
│     └──── src
│           └──── main.c    // NEMU入口
├──── scripts               // 一些重要的脚本和数据
└──── testcase              // 测试用例
```

- 不要改动以下文件
  - main.c, parse_args.c, nemu_trap.c
  - scripts/和libs/文件夹下的任意文件

PA实验的过程和提交方式

- 参照Guide的提示完成各个阶段
- 运行本地测试，执行

make test_pa-阶段

- 所有的test系列目标已经列举在Makefile中

PA实验的过程和提交方式

- 提交PA实验代码
  - 在pa2018_fall/目录下输入以下命令自动打包和上传代码

make submit_pa-阶段

  - 所有submit系列目标已经列举在Makefile中，不要改动此部分
  - 若上传失败，则根据提示将压缩包上传到CMS备用提交窗口
- 在Guide中有对应阶段的书面小问题，在每个大阶段截止时，将该阶段所有小问题的答卷提交到CMS系统中对应的PA课后问题提交窗口
- PA各阶段的截止
  - 应该在规定的提交截止时间前提交代码和答卷
  - 若迟交，则根据迟交多久逐步增加扣分惩罚
- 不允许抄袭！我们会比对今年和往年的所有代码，若发现抄袭，则对应阶段0分。不要试图破解或攻击submit相关逻辑和服务器，若发现，提交校方进行处理。

PA实验的过程和提交方式

- PA环境配置了两个追踪功能
  - 针对实验代码开发过程的追踪，必须启用，用于辅助判断抄袭和科研目的
  - 针对实验过程中的心理调查问卷，自愿参加，用于教学过程管理和科研目的
  - 相关数据仅用于教学和科研目的，若需要公开发表，则会先进行匿名处理

下面开始讲PA

PA 0 基础知识部分

- PA的目标
- PA的阶段构成
- PA的原理
- PA的成果

PA 0 基础知识部分

- PA的目标
- PA的阶段构成
- PA的原理
- PA的成果

PA的目标

- 要创建NEMU（一个简化的x86模拟器）
  - 由C语言编写
  - 以用户软件的形态运行
  - 能够执行通过交叉编译得到的i386指令集程序

PA 0 基础知识部分

- PA的目标
- PA的阶段构成
- PA的原理
- PA的成果

PA的阶段构成

- PA 1 – 数据的表示、存取和运算
  - PA 1-1 – 数据的存储、寄存器模拟
  - PA 1-2 – 整数运算、ALU模拟
  - PA 1-3 – 浮点数运算、FPU模拟
- PA 2 – 程序的执行
  - PA 2-1 – 指令解码与执行
  - PA 2-2 – 装载程序的Loader
  - PA 2-3 – 可选任务：完善调试器
- PA 3 – 存储管理
  - 分三个阶段
- PA 4 – 异常、中断与I/O
  - 分三个阶段

PA的阶段构成

NEMU

大致对应的器件

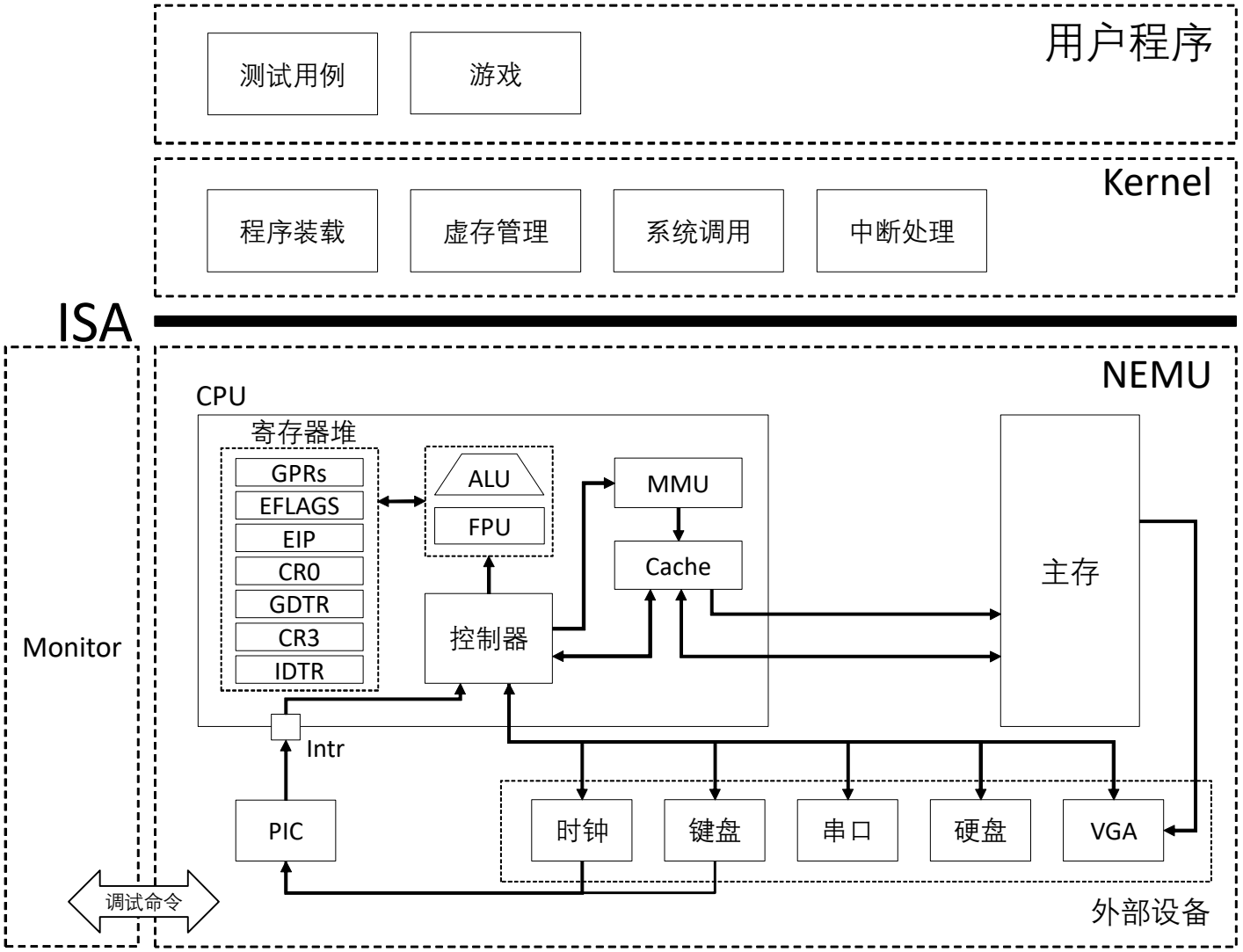
PA的四个大阶段

PA 1 数据的表示、存取和运算
PA 2 程序的执行

PA 3 存储管理

PA 4 异常、中断与 I/O

PA的阶段构成


用户程序
测试用例
游戏
Kernel
程序装载
虚存管理
系统调用
中断处理
ISA
NEMU
CPU
寄存器堆
GPRs
EFLAGS
EIP
CR0
GDTR
CR3
IDTR
ALU
FPU
MMU
Cache
控制器
主存
Monitor
Intr
PIC
时钟
键盘
串口
硬盘
VGA
调试命令
外部设备

PA 0 基础知识部分

- PA的目标
- PA的阶段构成
- PA的原理
- PA的成果

PA的原理

- 以一个软件来模拟硬件可行吗？
  - 答案是肯定的，有好多现成的例子

虚拟机

安卓模拟器

NES模拟器

PA的原理

- 以一个软件来模拟硬件可行吗?
  - 答案是肯定的，安装模拟器后的系统栈

模拟器上执行的**程序**（OS, Hello World, Super Mario, …）
模拟器/虚拟机（**NEMU**, Vmware, x Emulator, …）
操作系统（Windows, GNU/Linux, Mac, …）
机器硬件（Lenovo, Dell, Mac, …）

从程序的角度看，什么是一个计算机?

PA的原理

- Hello World究竟是怎么运行起来的?

hello_world.c

```
#include<stdio.h>

int main() {
        printf("Hello World!\n");
        return 0;
}
```

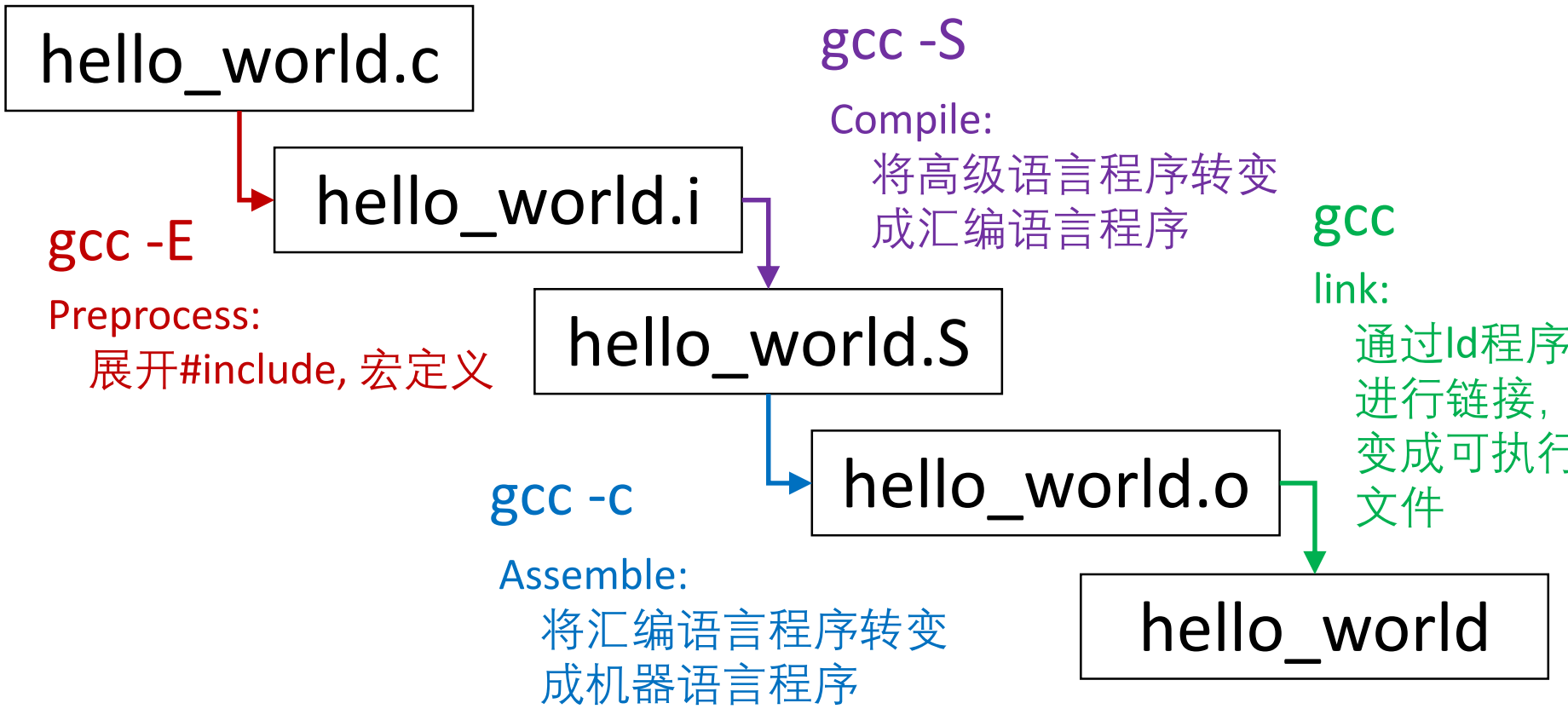
```
$ gcc –o hello_world hello_world.c
$ ./hello_world
Hello World!
```

编译
运行

PA的原理

```
$ gcc –o hello_world hello_world.c
```

一条命令执行了四个步骤



PA的原理

```
$ gcc –c –o hello_world.o hello_world.S
$ hexdump hello_world.o | less
```

hello_world.o 查看其内容

```
0000000 457f 464c 0101 0001 0000 0000 0000 0000
0000010 0001 0003 0001 0000 0000 0000 0000 0000
0000020 02fc 0000 0000 0000 0034 0000 0000 0028
0000030 000f 000e 0001 0000 0007 0000 4c8d 0424
0000040 e483 fff0 fc71 8955 53e5 e851 fffc ffff
0000050 0105 0000 8300 0cec 908d 0000 0000 8952
0000060 e8c3 fffc ffff c483 b810 0000 0000 658d
0000070 59f8 5d5b 618d c3fc 6548 6c6c 206f 6f57
0000080 6c72 2164 8b00 2404 00c3 4347 3a43 2820
0000090 6544 6962 6e61 3620 332e 302e 312d 2938
```

偏移量　　数据，两字节一组，小端

机器语言程序!

PA的原理

```
$ gcc –c –o hello_world.o hello_world.S
$ objdump -d hello_world.o | less
```

hello_world.o 反汇编其内容

```
hello_world.o:     file format elf32-i386


Disassembly of section .text:

00000000 <main>:
   0:   8d 4c 24 04             lea    0x4(%esp),%ecx
   4:   83 e4 f0                and    $0xfffffff0,%esp
   7:   ff 71 fc                pushl  -0x4(%ecx)
   a:   55                      push   %ebp
   b:   89 e5                   mov    %esp,%ebp
   d:   53                      push   %ebx
   e:   51                      push   %ecx
   f:   e8 fc ff ff ff          call   10 <main+0x10>
```

PA的原理

```
$ gcc –c –o hello_world.o hello_world.S
$ objdump -d hello_world.o | less
```

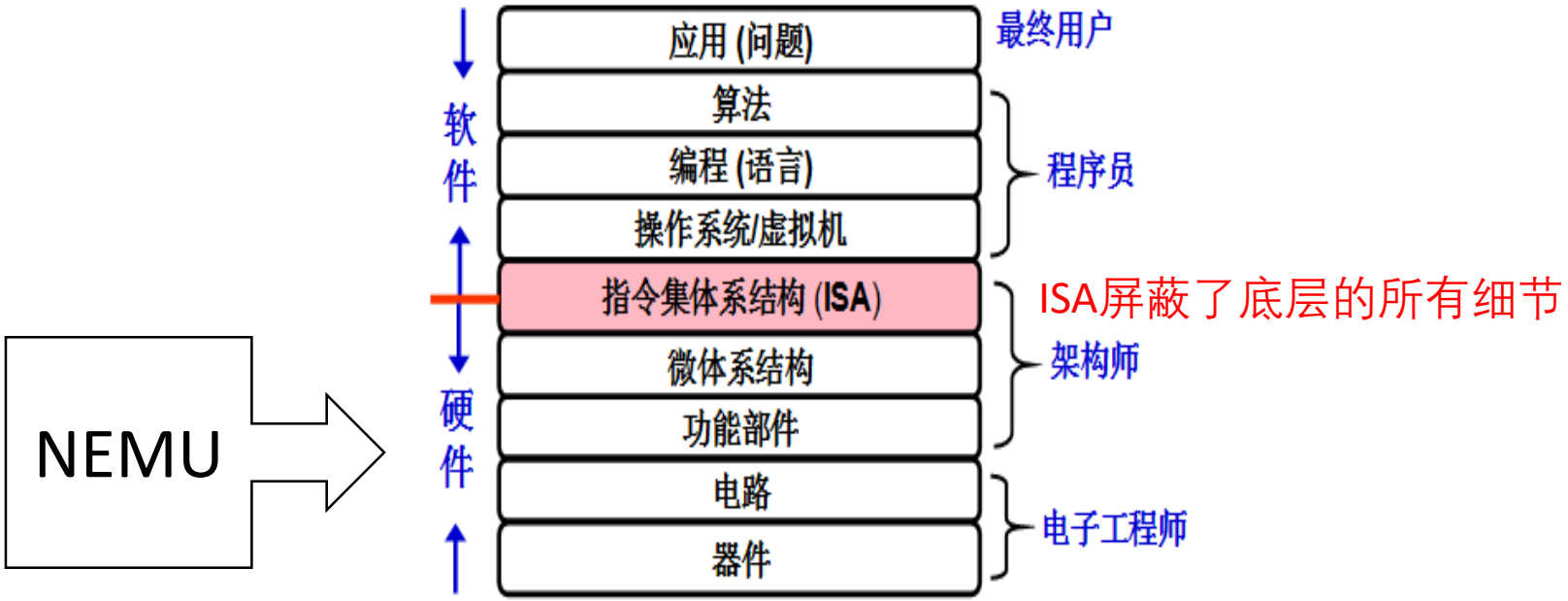
hello_world.o 反汇编其内容

```
hello_world.o:

Disassembly of

00000000 <main>:
   0:   8d 4c 24 04             lea    0x4(%esp),%ecx
   4:   83 e4 f0                and    $0xfffffff0,%esp
   7:   ff 71 fc                pushl  -0x4(%ecx)
   a:   55                      push   %ebp
   b:   89 e5                   mov    %esp,%ebp
   d:   53                      push   %ebx
   e:   51                      push   %ecx
   f:   e8 fc ff ff ff          call   10 <main+0x10>
```

一系列的指令
对程序而言能执行指令的就是计算机

PA的原理

- 为什么可以用软件来模拟计算机执行程序?
  - 所谓程序，在形式上就是一串指令的序列
  - 硬件设计者和软件开发者约定好有哪些指令可以用（ISA）
  - 只要能够读懂指令并正确完成对应动作（运算、数据操作、输入输出等）的东西就是一台合格的计算机



PA的原理

```
$ gcc –c –o hello_world.o hello_world.S
$ objdump -d hello_world.o | less
```

hello_world.o 反汇编其内容

```
hello_world.o:     file format elf32-i

Disassembly of section .text:

00000000 <main>:
   0:   8d 4c 24 04             lea    0x4(%esp),%ecx
   4:   83 e4 f0                and    $0xfffffff0,%esp
   7:   ff 71 fc                pushl  -0x4(%ecx)
   a:   55                      push   %ebp
   b:   89 e5                   mov    %esp,%ebp
   d:   53                      push   %ebx
   e:   51                      push   %ecx
   f:   e8 fc ff ff ff          call   10 <main+0x10>
```

指令里有各种数据移动、运算、控制操作

PA的原理

```
$ gcc –c –o hello_world.o hello_world.S
$ objdump -d hello_world.o | less
```

hello_world.o 反汇编其内容

```
hello_world.o:     file format elf32-i386

Disass

000000
   0:                                    0x4(%esp),%ecx
   4:                                    $0xfffffff0,%esp
   7:                                 hl -0x4(%ecx)
   a:   55                            push   %ebp
   b:   89 e5                         mov    %esp,%ebp
   d:   53                            push   %ebx
   e:   51                            push   %ecx
   f:   e8 fc ff ff ff                call   10 <main+0x10>
```

指令中涉及寄存器、立即数、内存地址等操作对象，相对应的就有寄存器和内存这样的设备对程序可见

PA的原理

```
#include<stdio.h>

int main() {
        printf("Hello World!\n");
        return 0;
}
```

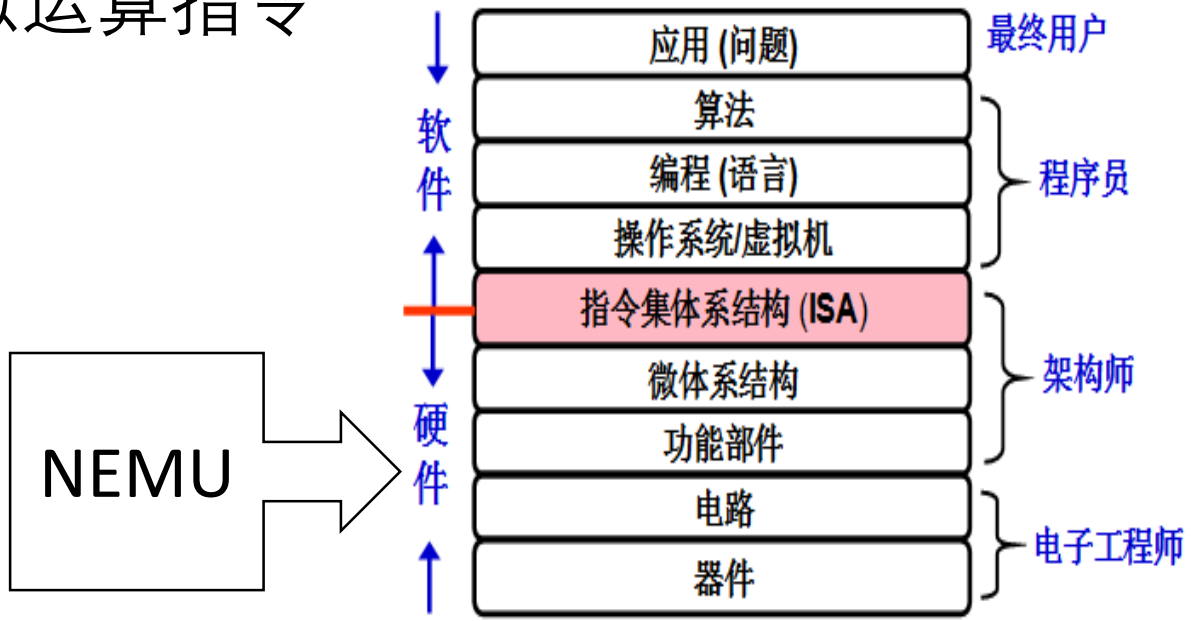
```
$ gcc –o hello_world hello_world.c
$ ./hello_world
Hello World!
```

指令中也包含对键盘、屏幕等外部设备的读写指令，因此外部设备也对程序可见

PA的原理

用C语言模拟机器的功能，实现指令的解释执行，模拟实现所有对程序可见的机器组件和功能：

- 变量模拟寄存器
- 数组模拟内存
- 变量赋值模拟数据移动指令
- 运算操作模拟运算指令
- ...



PA 0 基础知识部分

- PA的目标
- PA的阶段构成
- PA的原理
- PA的成果

PA的成果

当然，最重要的是掌握了计算机系统的重要知识！