

计算机系统基础
Programming Assignment

PA 0 – 在线实验平台与PA简介

2022年2月18日

南京大学《计算机系统基础》课程组

目录

- 课程基本信息
- 在线实验平台的使用
- PA课程内容和原理简介



目录

- 课程基本信息
- 在线实验平台的使用
- PA课程内容和原理简介



课程基本信息

- 上课时间地点

周五，5-6节，逸B-403
(前两周在线)

- 在线实验平台

<http://114.212.10.200>

使用方法接下来详细演示

- 助教（都在QQ群）

王文（负责答疑）

庄宇乾（负责平台问题）

- 以前学期的视频

- <https://space.bilibili.com/284613991/channel/detail?cid=103368>
- <https://box.nju.edu.cn/d/20fa8f3141f44f11938a/>

目录

- 课程基本信息
- 在线实验平台的使用
- PA课程内容和原理简介



在线实验平台的使用说明

开发

汪亮

Email : wl#nju.edu.cn (replace # with @)

QQ/微信: 715764994

庄宇乾

Email: zhuangyq#smail.nju.edu.cn

张明亚

Email: DG20330034#smail.nju.edu.cn

FA NEMU OL



- 无须自己部署环境!
- 更强的防作弊功能!
- 更好的学习体验?

Driven by ~~magic~~.
great passion and a few bugs

在线实验平台的使用说明



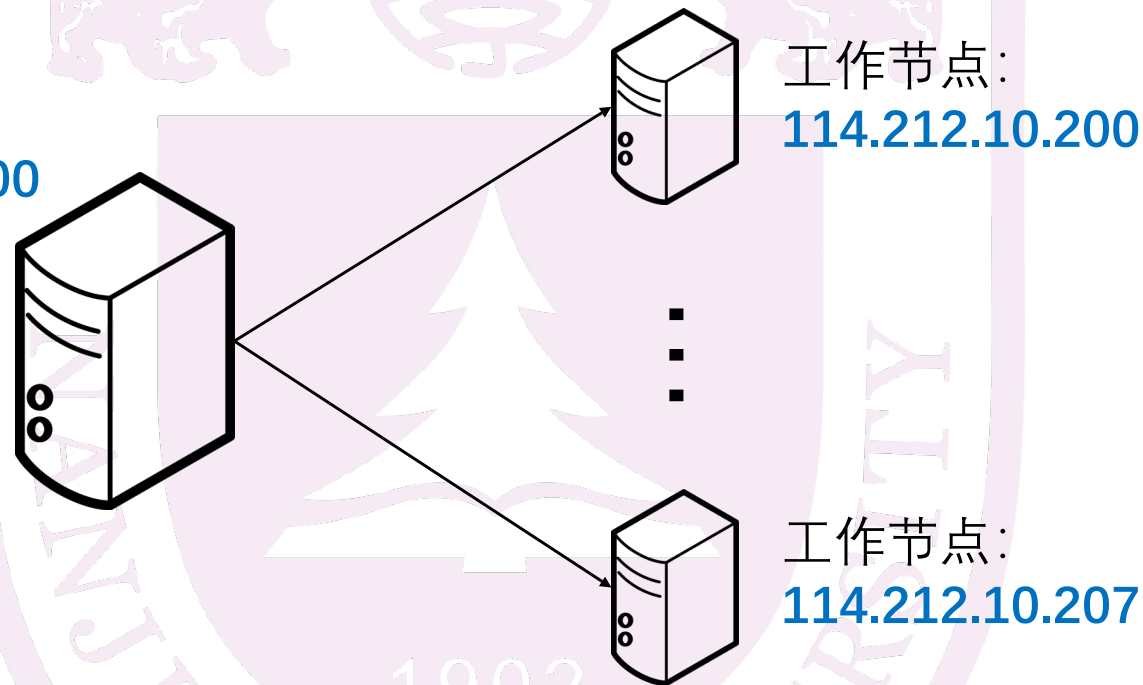
确保自己连接了校园网，如果在校外访问，则需要拨通南京大学VPN



使用我们推荐的浏览器访问在线实验平台，若使用的是firefox，按照教程首页上的说明进行设置

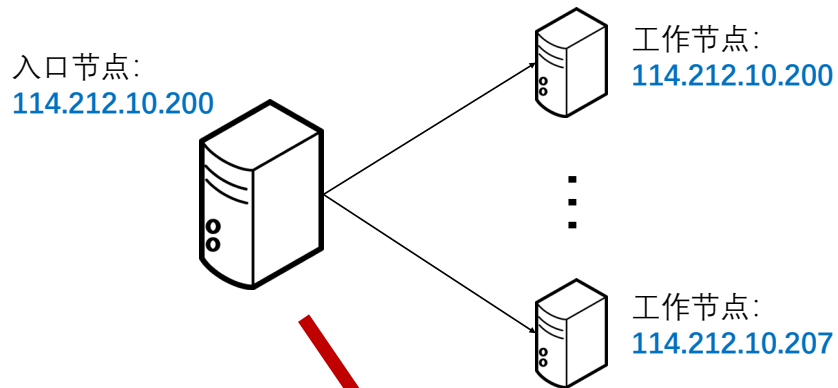
入口节点：
114.212.10.200

访问



1902
在线实验平台架构

(1) 访问平台



汪亮

Email: wl#nju.edu.cn (replace # with @)

QQ/微信: 715764994

庄宇乾

Email: zhuangyq#smail.nju.edu.cn

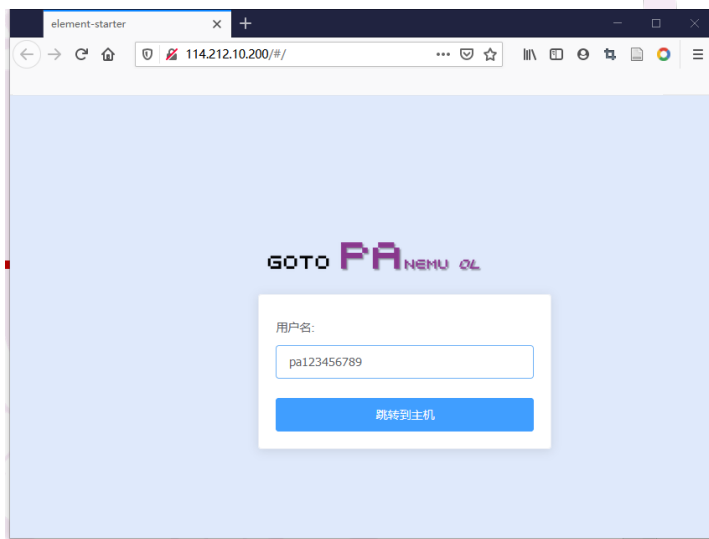
张明亚

Email: DG20330034#smail.nju.edu.cn

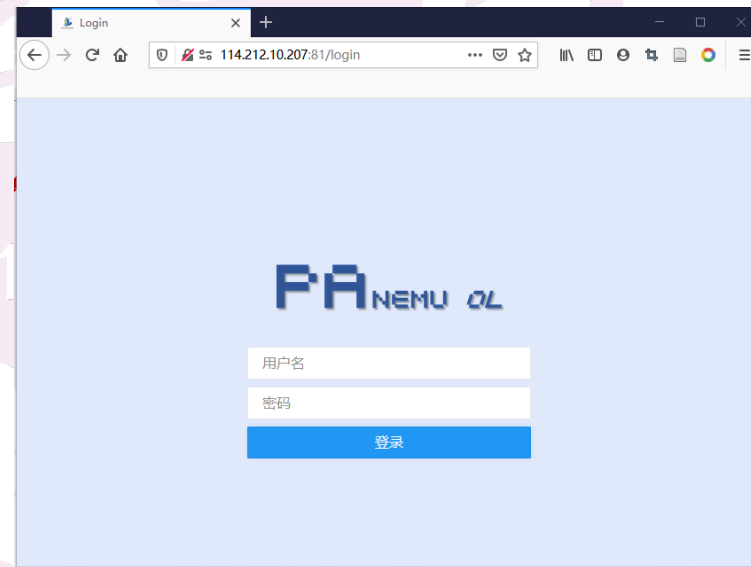
跳转失败:
群里找老师和助教



1. 访问入口节点
2. 输入**pa学号**作为用户名
3. 点击**跳转到主机**按钮

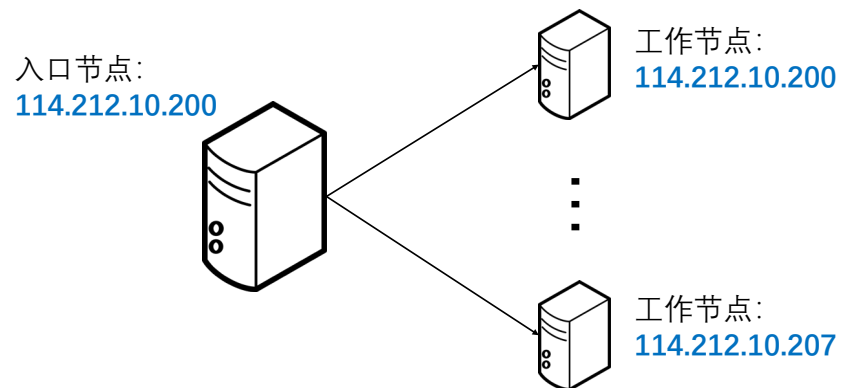


跳转成功:
来到工作节点

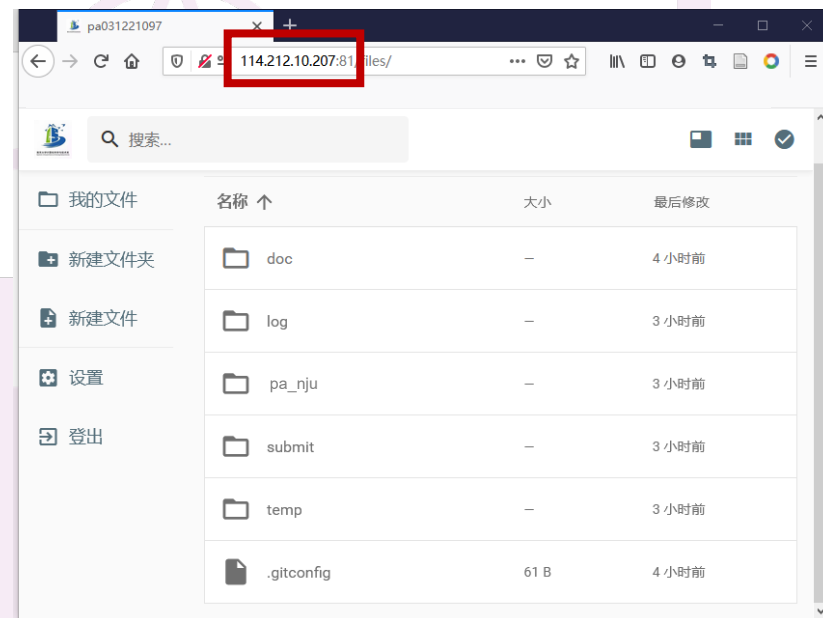
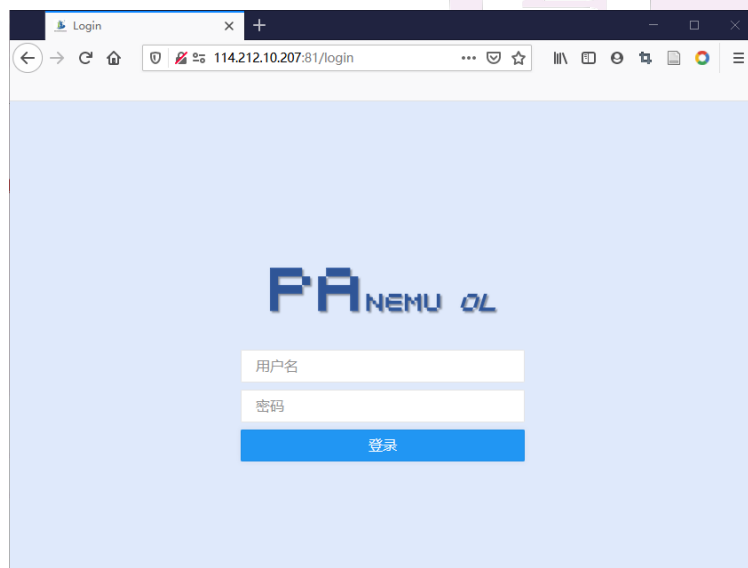


(1) 访问平台

记住自己的工作节点IP地址和端口号（81），下次可无需再访问入口节点



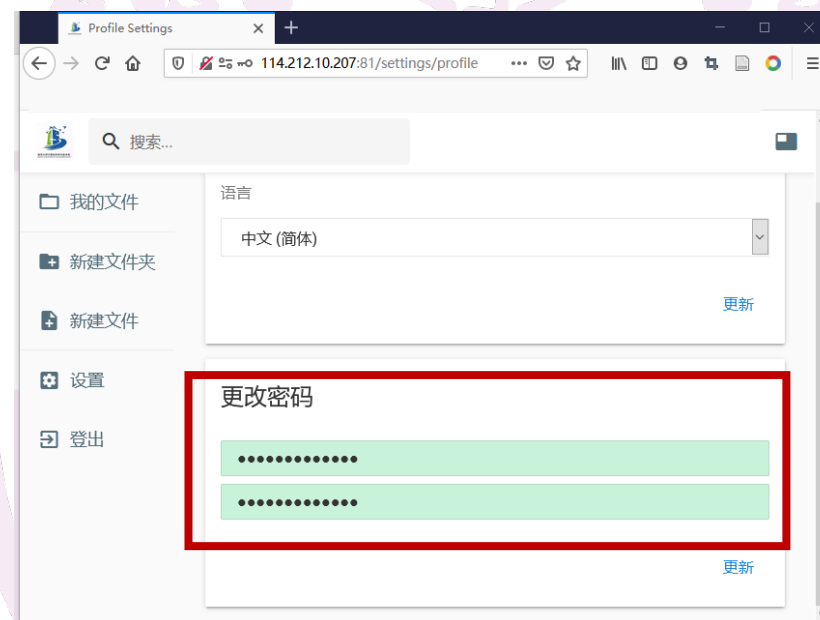
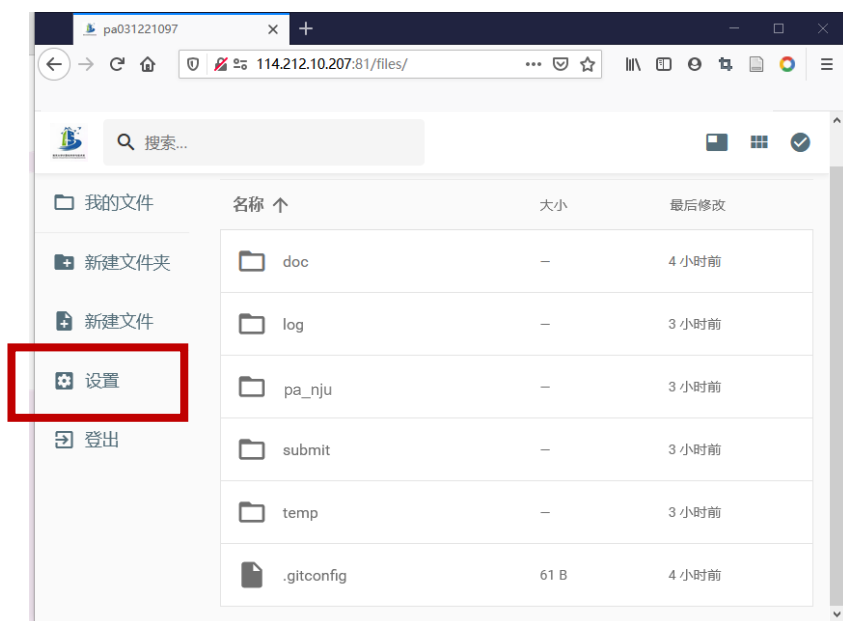
1. 输入pa学号作为用户名
2. 输入pa学号作为密码
3. 点击‘登陆’按钮
4. 如登陆有问题，找老师和助教



登陆成功!



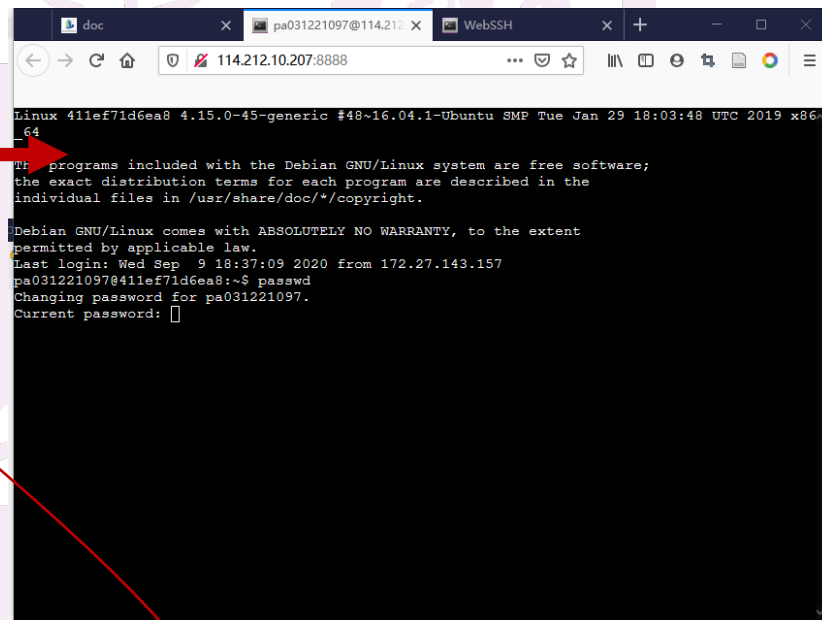
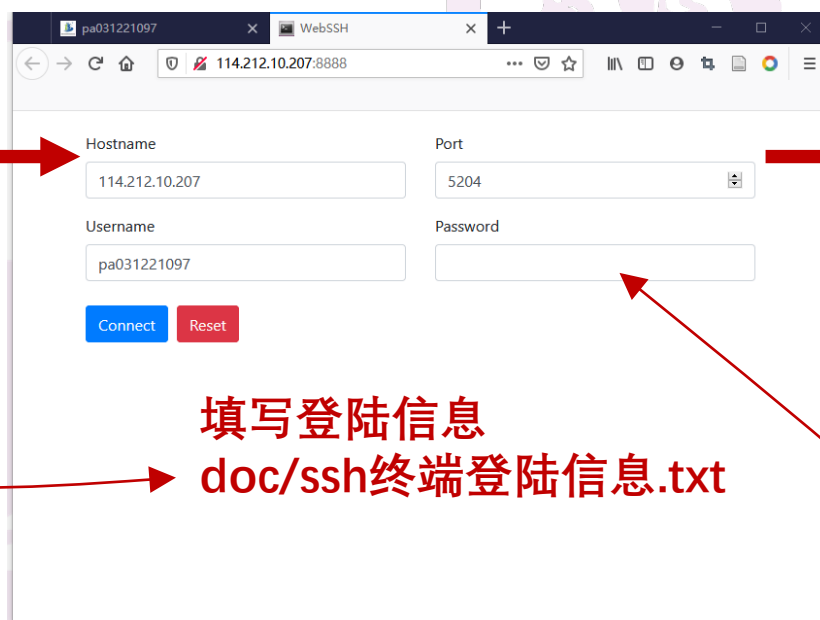
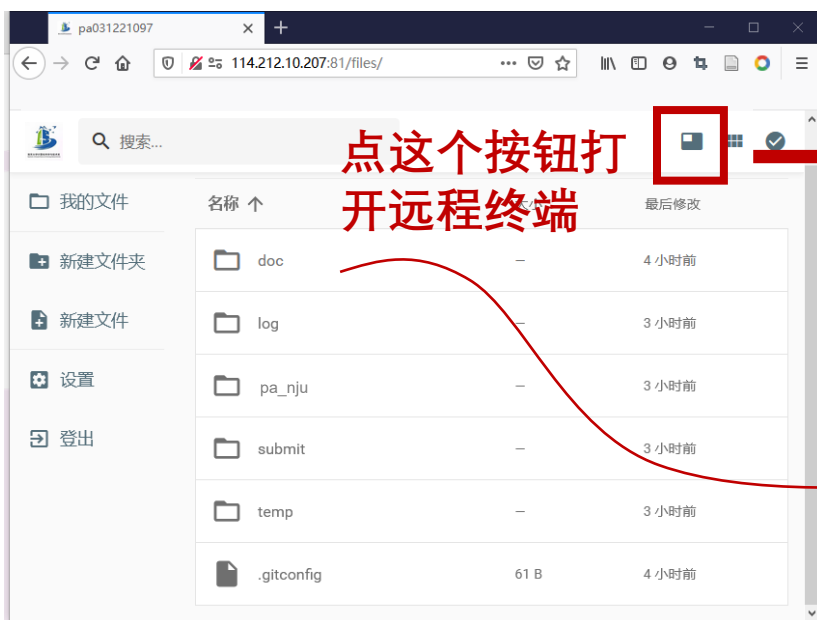
(2) 更改密码



第一处：更改网页登陆密码

(2) 更改密码

命令passwd



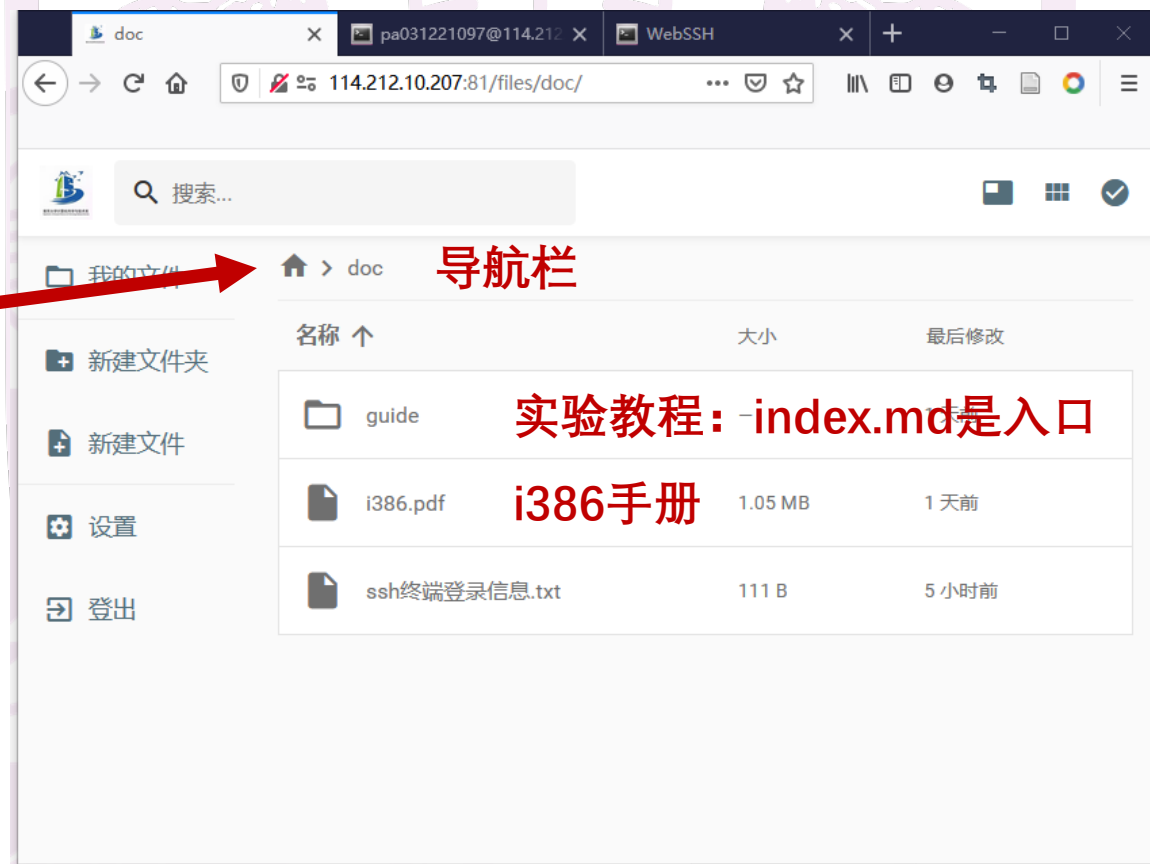
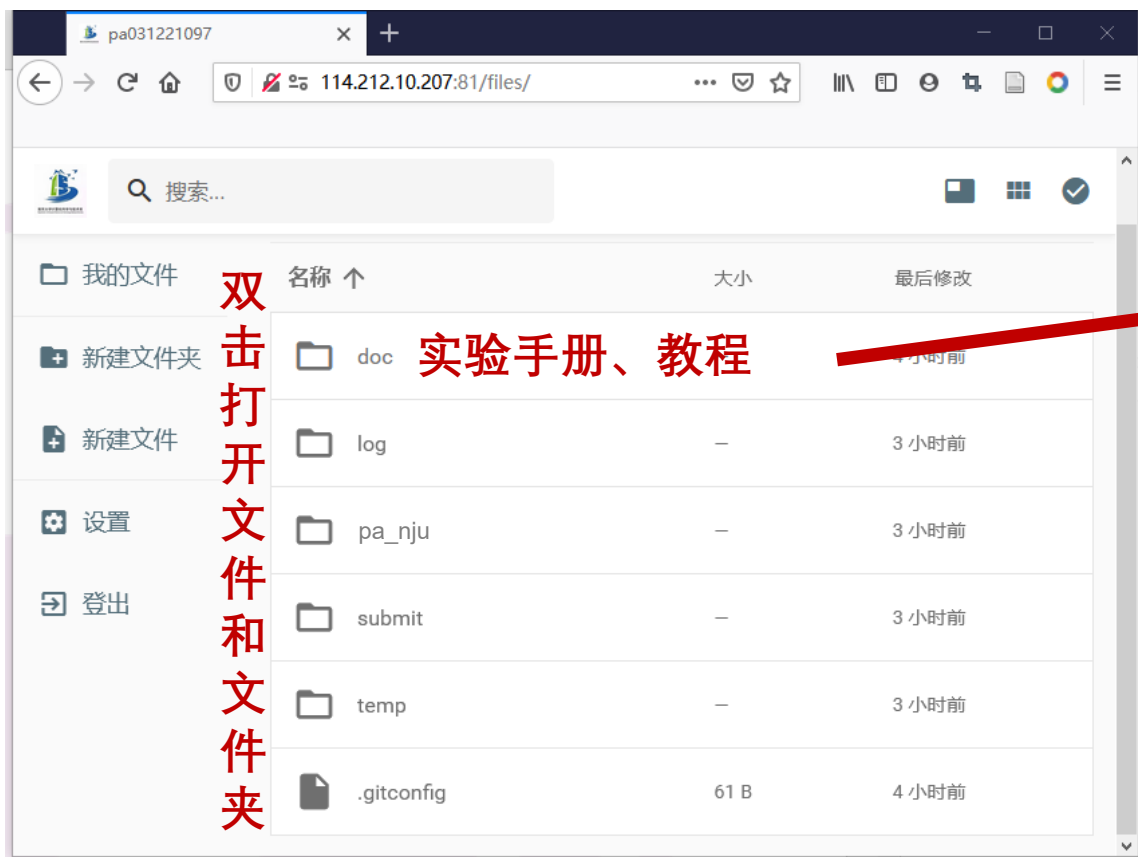
第二处：更改个人docker密码

改完后用新密码登陆

(3) 熟悉环境

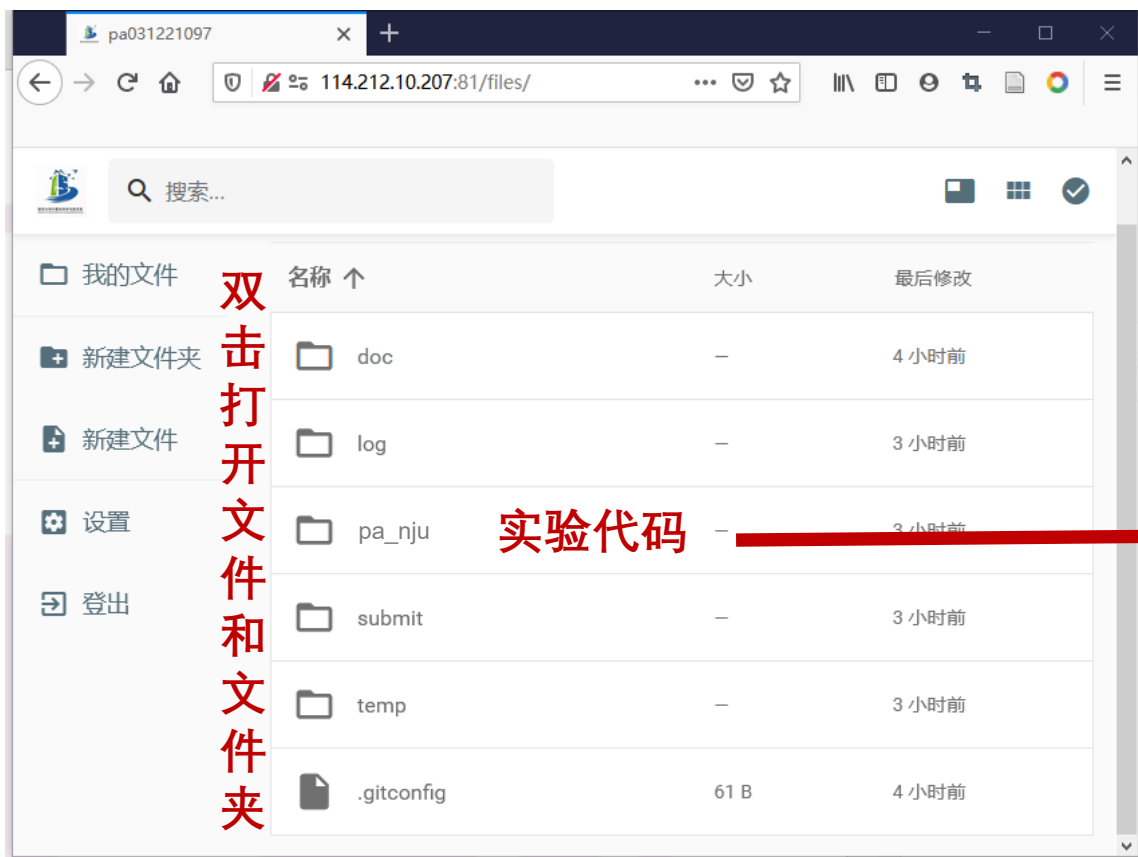


(3) 熟悉环境



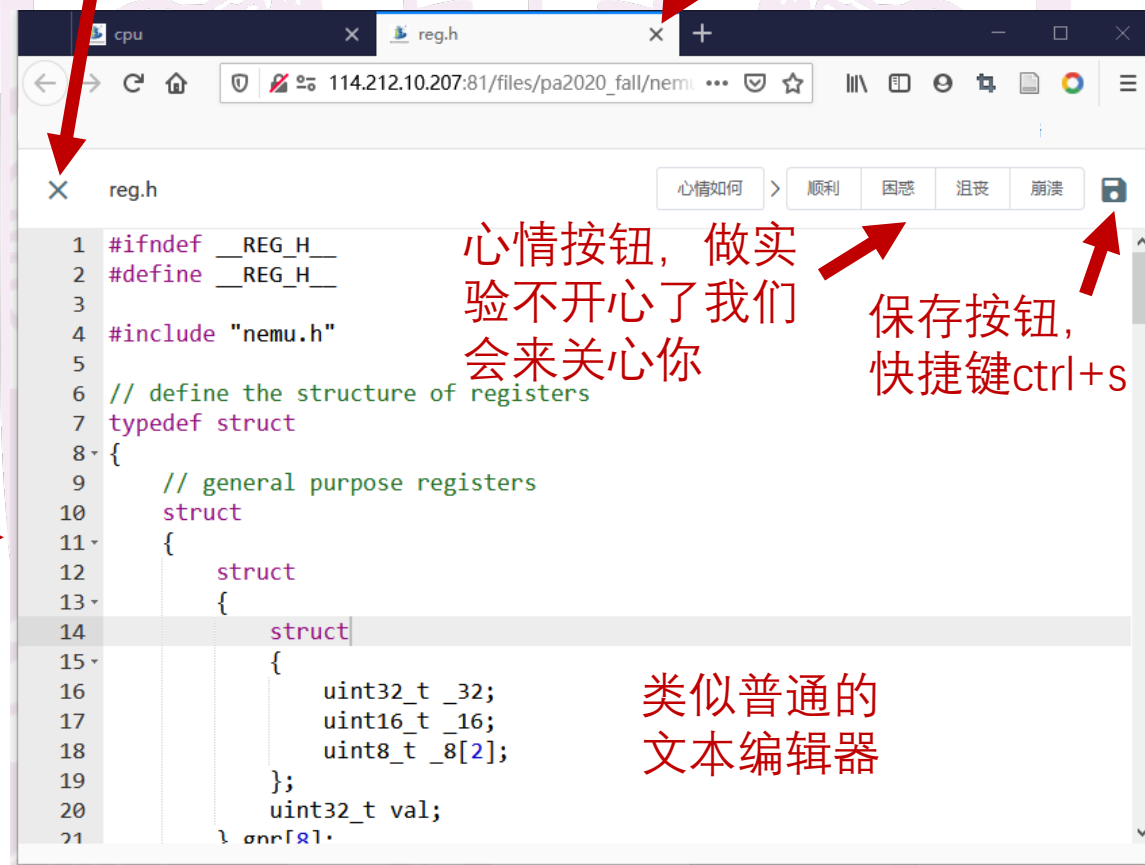
找到实验的相关手册和教程

(3) 熟悉环境



关闭按钮2: 保存了
也会再弹个窗提醒你

关闭按钮1: 没有保存
也不会有任何提示

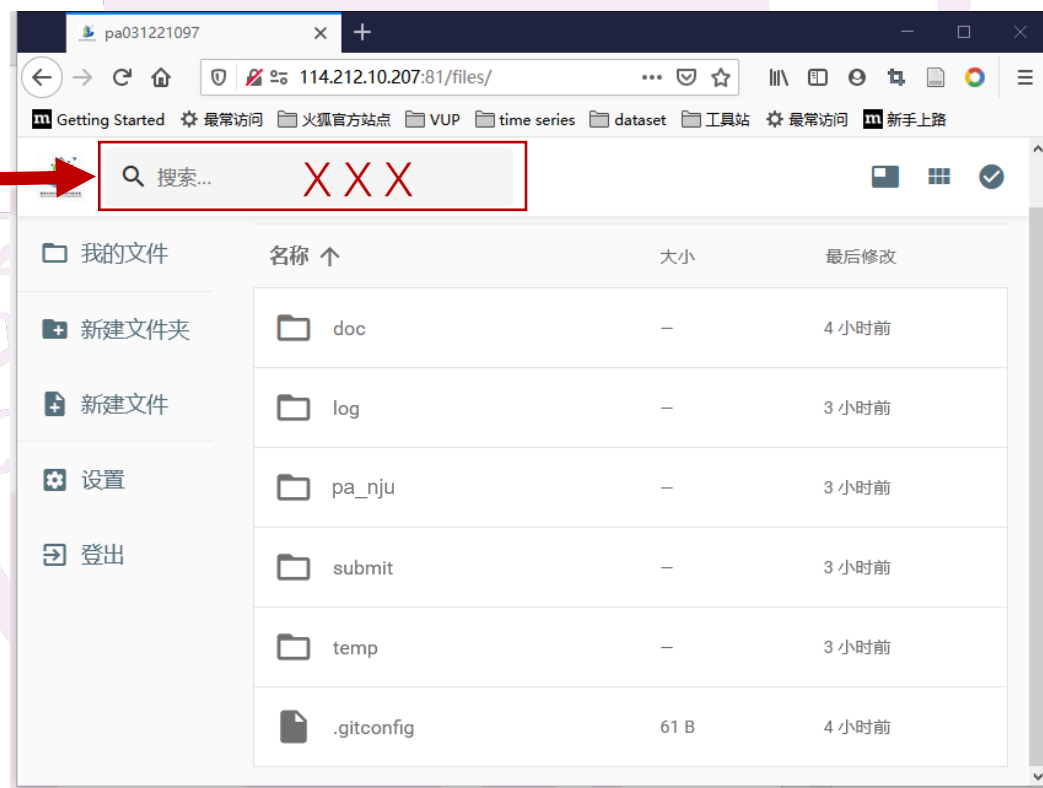


熟悉代码编辑界面

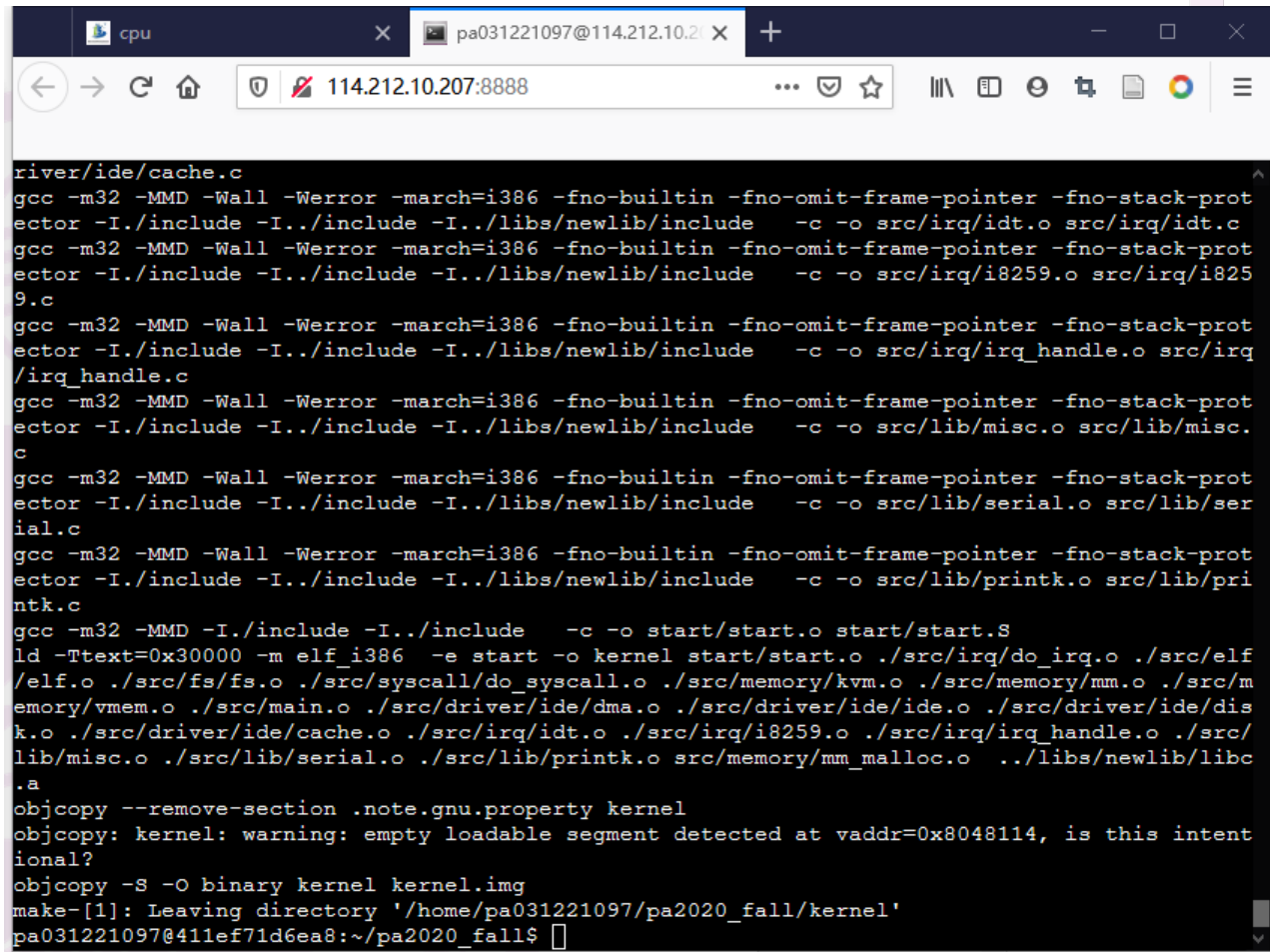
文本编辑窗口的已知缺陷

- 不要同时打开多个编辑窗口，内容会互相覆盖
- 不要使用系统内的搜索功能，内容会莫名改变
- 经常保存

不要用搜索，也不要
用Ctrl+F出来的搜索功能



(4) 尝试一下make



```
river/ide/cache.c
gcc -m32 -MMD -Wall -Werror -march=i386 -fno-builtin -fno-omit-frame-pointer -fno-stack-protector -I./include -I../include -I../libs/newlib/include -c -o src/irq/idt.o src/irq/idt.c
gcc -m32 -MMD -Wall -Werror -march=i386 -fno-builtin -fno-omit-frame-pointer -fno-stack-protector -I./include -I../include -I../libs/newlib/include -c -o src/irq/i8259.o src/irq/i8259.c
gcc -m32 -MMD -Wall -Werror -march=i386 -fno-builtin -fno-omit-frame-pointer -fno-stack-protector -I./include -I../include -I../libs/newlib/include -c -o src/irq/irq_handle.o src/irq/irq_handle.c
gcc -m32 -MMD -Wall -Werror -march=i386 -fno-builtin -fno-omit-frame-pointer -fno-stack-protector -I./include -I../include -I../libs/newlib/include -c -o src/lib/misc.o src/lib/misc.c
gcc -m32 -MMD -Wall -Werror -march=i386 -fno-builtin -fno-omit-frame-pointer -fno-stack-protector -I./include -I../include -I../libs/newlib/include -c -o src/lib/serial.o src/lib/serial.c
gcc -m32 -MMD -Wall -Werror -march=i386 -fno-builtin -fno-omit-frame-pointer -fno-stack-protector -I./include -I../include -I../libs/newlib/include -c -o src/lib/printk.o src/lib/printk.c
gcc -m32 -MMD -I./include -I../include -c -o start/start.o start/start.S
ld -Ttext=0x30000 -m elf_i386 -e start -o kernel start/start.o ./src/irq/do_irq.o ./src/elf/elf.o ./src/fs/fs.o ./src/syscall/do_syscall.o ./src/memory/kvm.o ./src/memory/mm.o ./src/memory/vmem.o ./src/main.o ./src/driver/ide/dma.o ./src/driver/ide/ide.o ./src/driver/ide/disk.o ./src/driver/ide/cache.o ./src/irq/idt.o ./src/irq/i8259.o ./src/irq/irq_handle.o ./src/lib/misc.o ./src/lib/serial.o ./src/lib/printk.o src/memory/mm_malloc.o ../libs/newlib/libc.a
objcopy --remove-section .note.gnu.property kernel
objcopy: kernel: warning: empty loadable segment detected at vaddr=0x8048114, is this intentional?
objcopy -S -O binary kernel kernel.img
make-[1]: Leaving directory '/home/pa031221097/pa2020_fall/kernel'
pa031221097@411ef71d6ea8:~/pa2020_fall$
```

> cd pa_nju/
> make clean
> make

❀❀ \ (° ▽ °) / ❀

warning: empty loadable segment detected at vaddr=0x8048114, is this intentional? 2

每次编译会看到这个warning，这是正常的，不用担心。其它的任何warning和error都要引起警觉！

目录

- 课程基本信息
- 在线实验平台的使用
- PA课程内容和原理简介

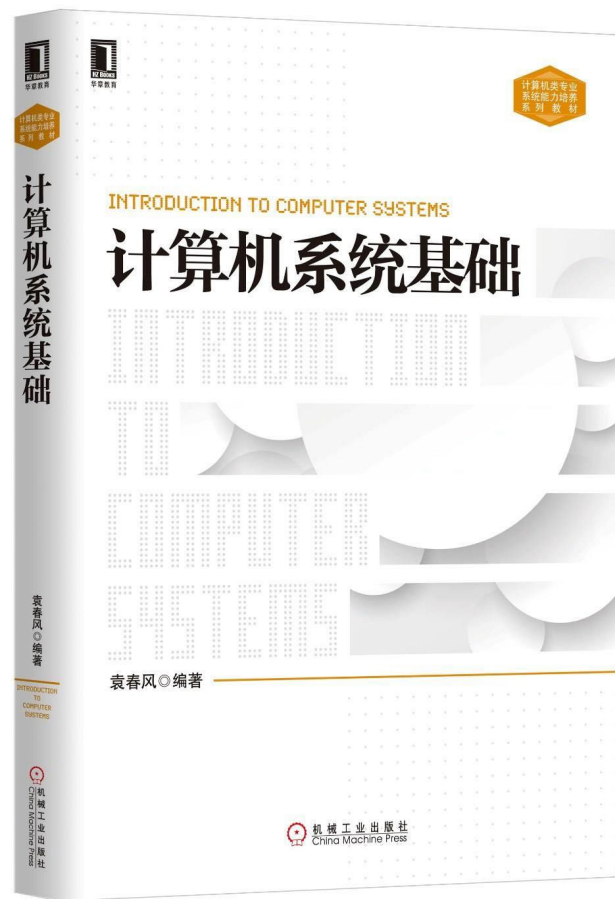


PA简介



计算机系统基础（第二版）

PA简介



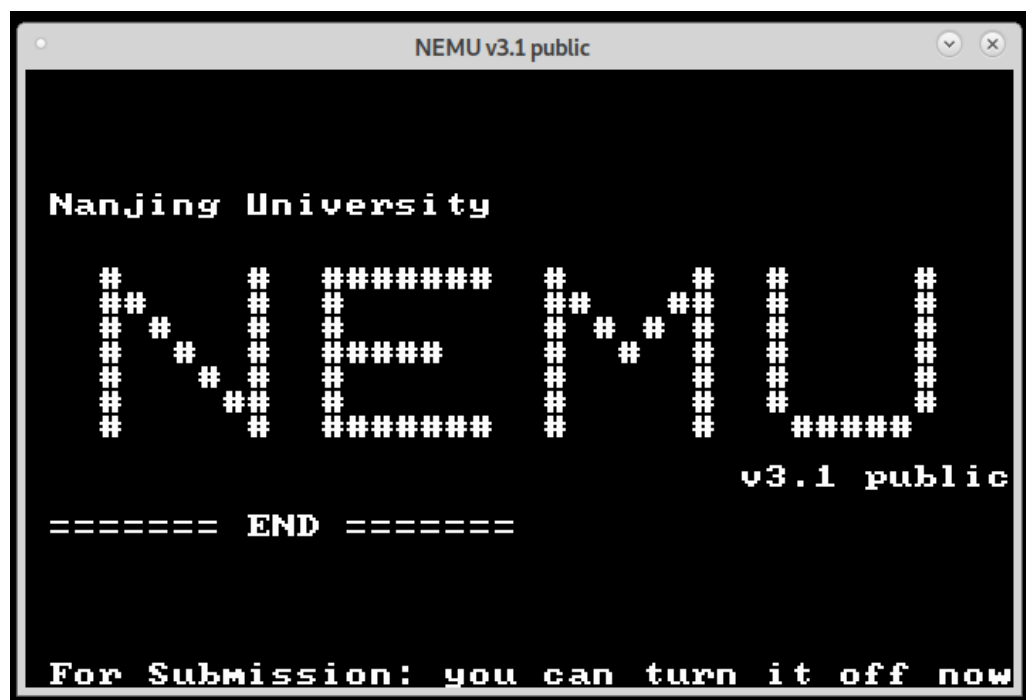
Programming Assignment, 简称PA

编程实践部分



PA简介

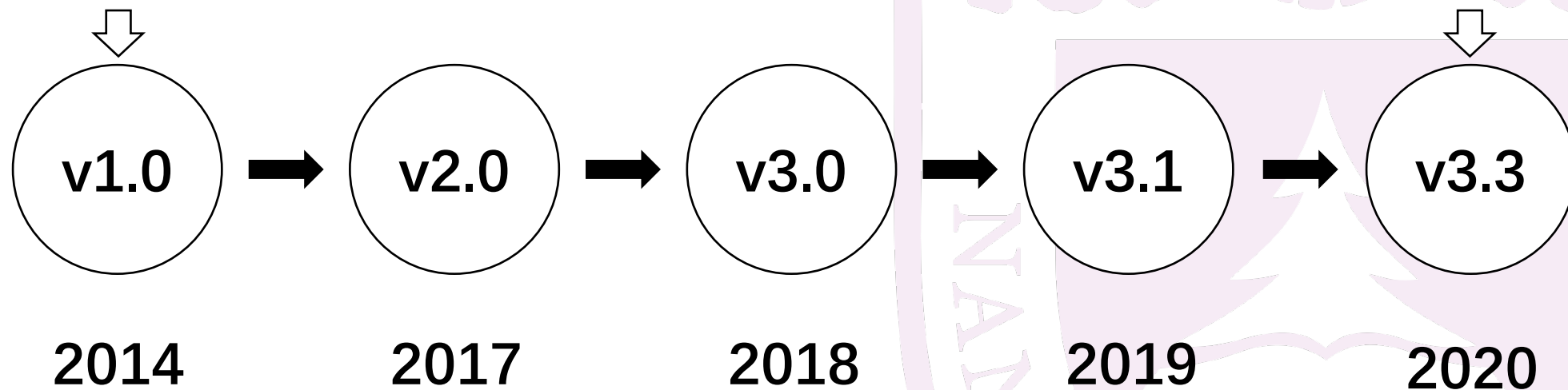
PA实验的目的：实现NEMU，一个简化的i386模拟器



- 由C语言编写
- 以用户软件的形式运行
- 能够执行i386指令集程序

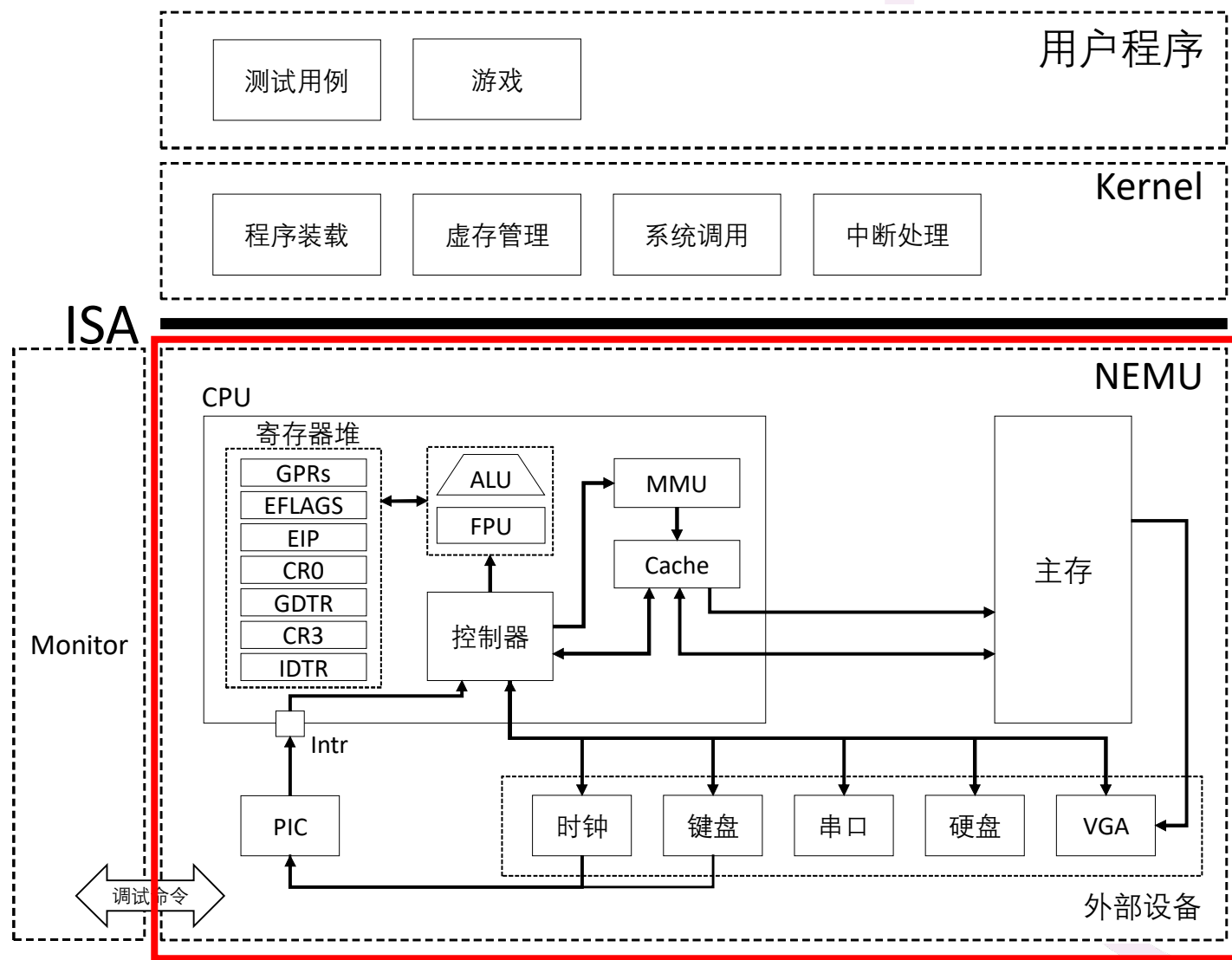
PA简介

余子濠 开发完成



PA的简要历史

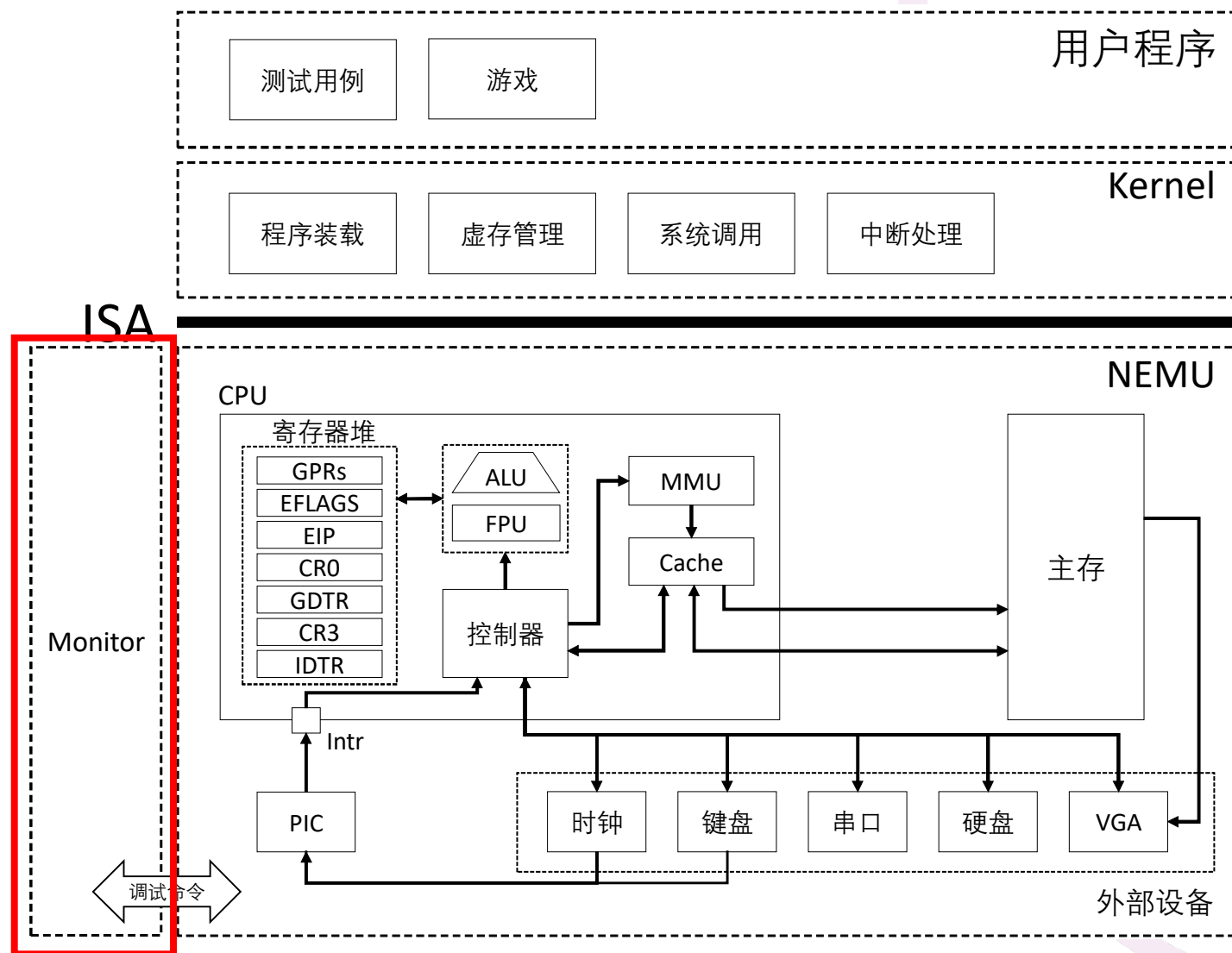
PA简介



NEMU模拟器
(NJU Emulator)

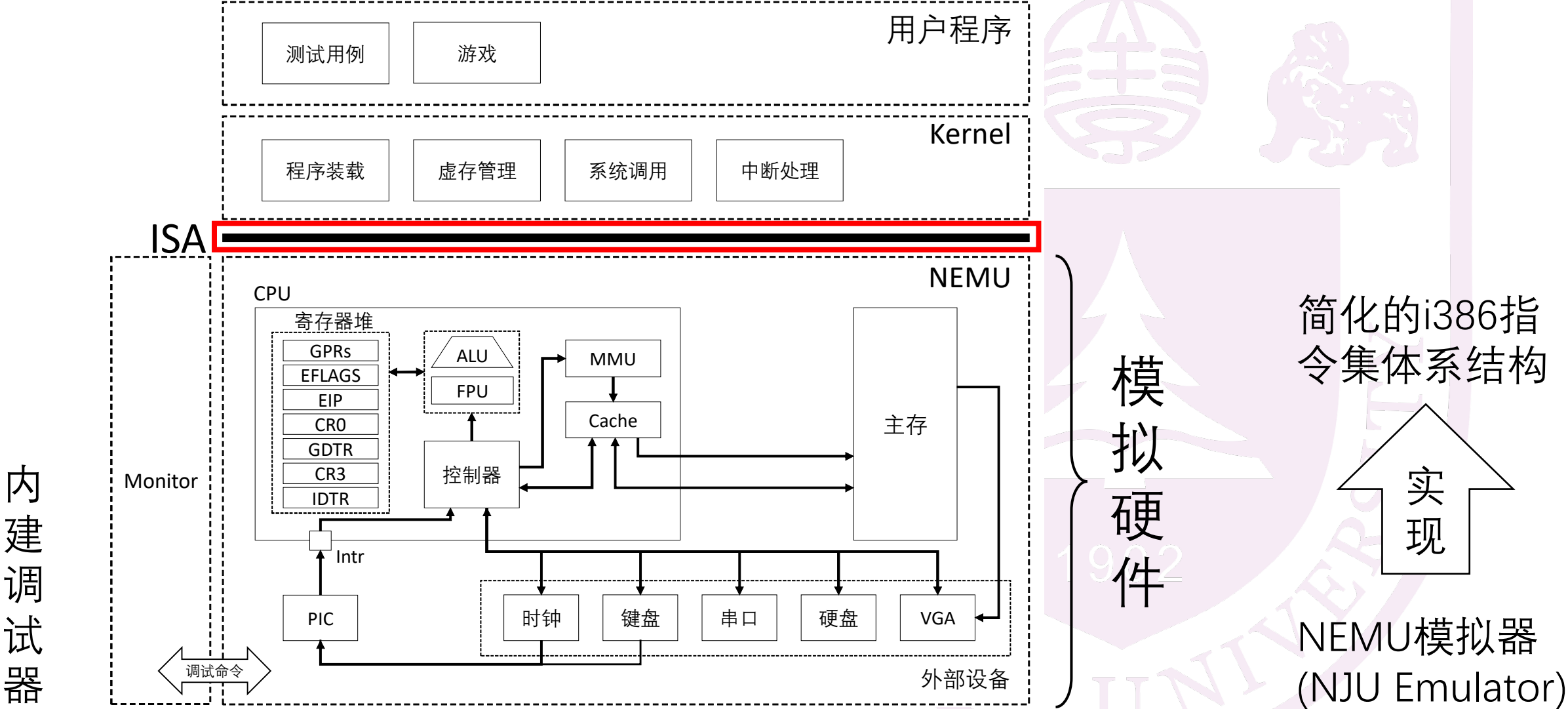
PA简介

内建调试器

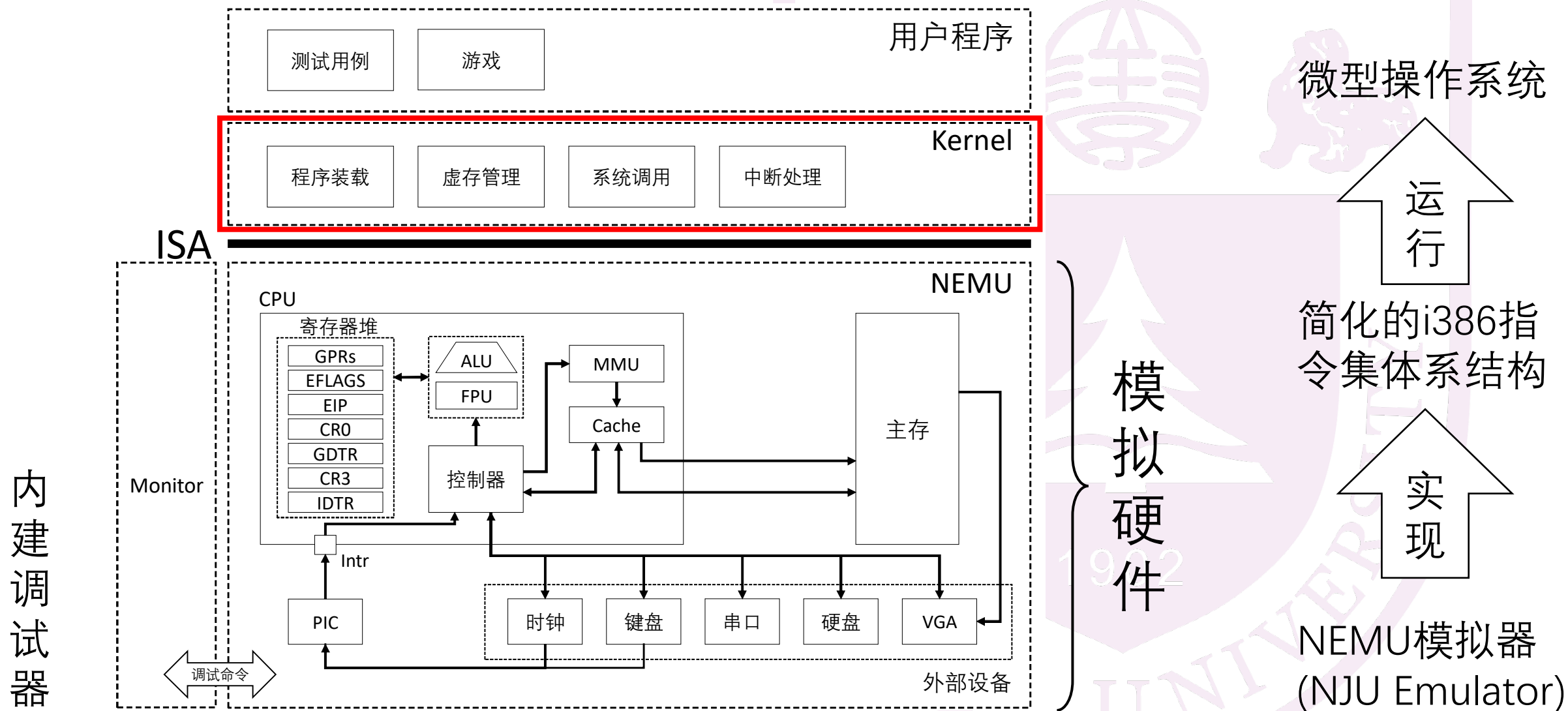


NEMU模拟器
(NJU Emulator)

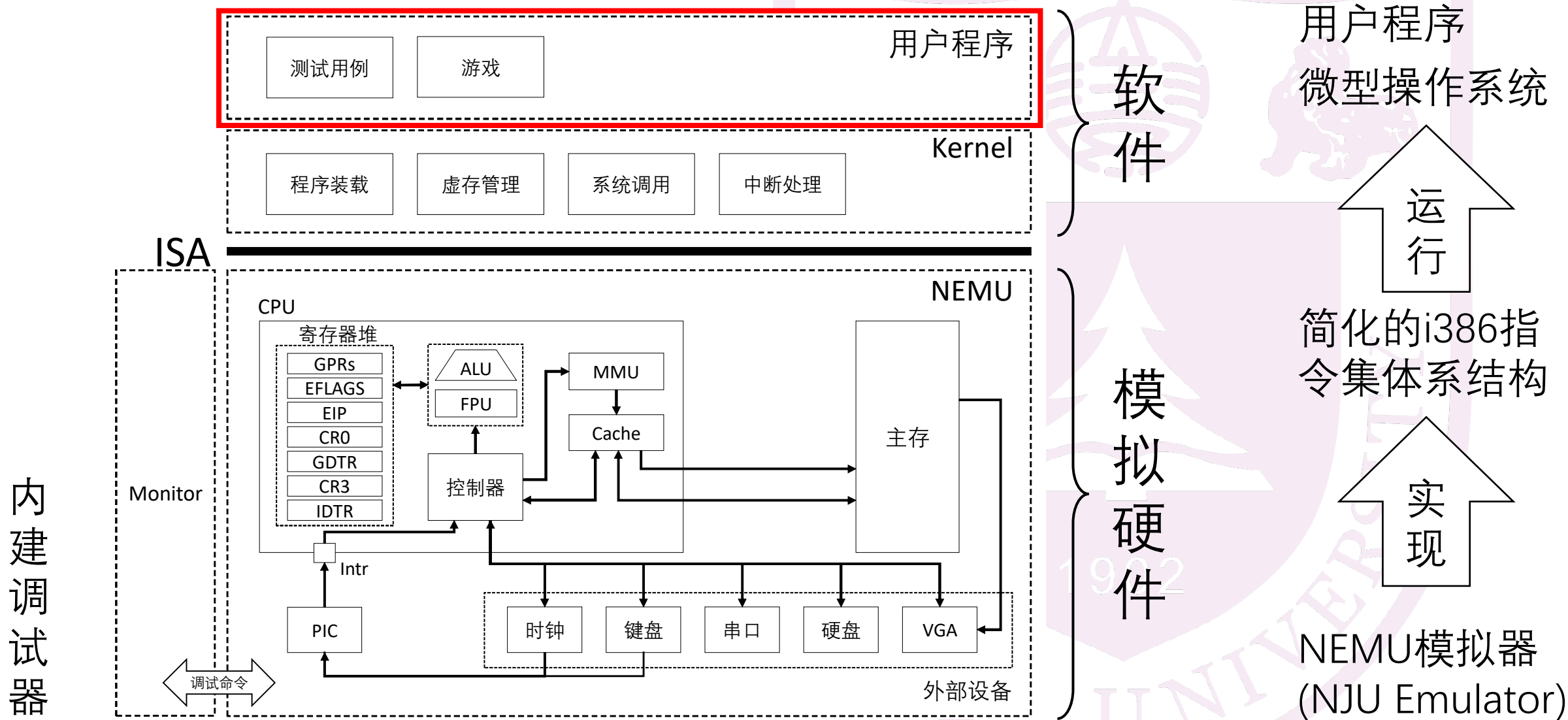
PA简介



PA简介



PA简介



PA简介

PA分为四个大的阶段，每个大阶段又拆分为三个小阶段

- 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的原理

如何编写软件模拟一台计算机？

PA的原理

可以使用软件模拟一台计算机！

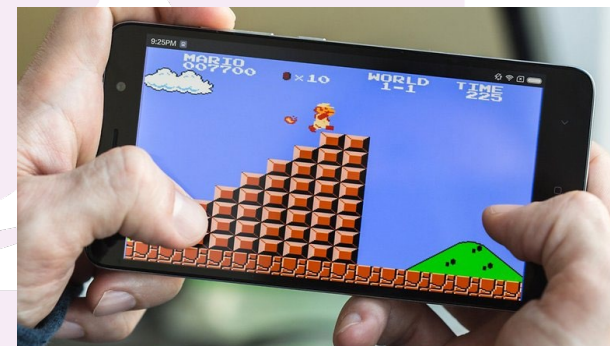


✓ 模拟PC机



ANDROID

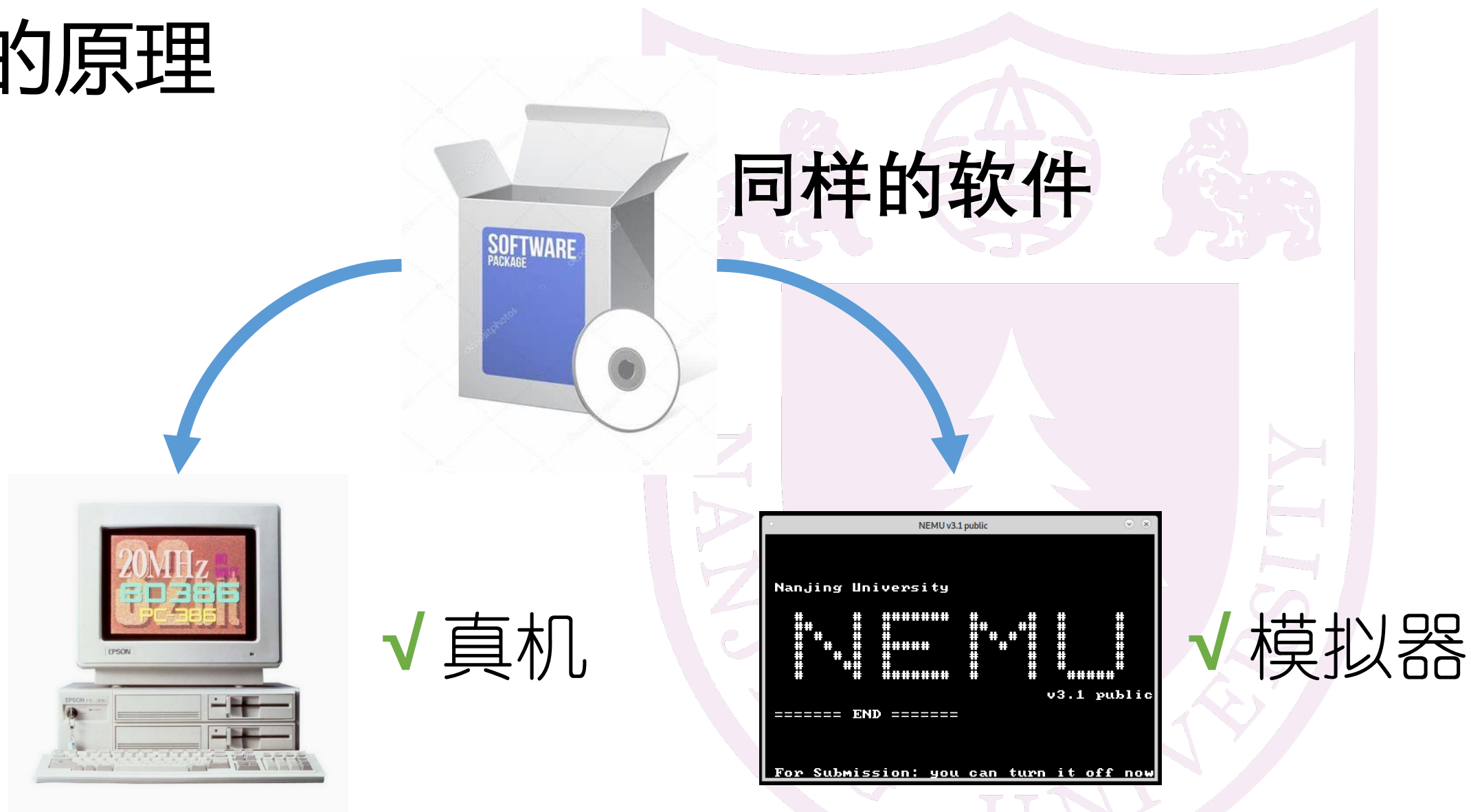
✓ 模拟手机



1902

✓ 模拟游戏机

PA的原理



PA的原理

- 从机器的视角来看软件

hello_world.c

```
#include<stdio.h>

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

编译得到
可执行文件

机器运行
可执行文件

得到
执行结果

```
$ gcc -o hello_world hello_world.c
```

```
$ ./hello_world
```

```
Hello World!
```

PA的原理

- 从机器的视角来看软件

hello_world.c

```
#include<stdio.h>

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

编译得到
可执行文件

机器运行
可执行文件

得到
执行结果

\$ gcc -o hello_world hello_world.c

\$./hello_world

软件

Hello World!

PA的原理

- 从机器的视角来看软件

hello_world.c

```
#include<stdio.h>

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

编译

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

hello_world.o 查看其内容

00000000	457f	464c	0101	0001	0000	0000	0000	0000
00000010	0001	0003	0001	0000	0000	0000	0000	0000
00000020	02fc	0000	0000	0000	0034	0000	0000	0028
00000030	000f	000e	0001	0000	0007	0000	4c8d	0424
00000040	e483	fff0	fc71	8955	53e5	e851	fffc	ffff
00000050	0105	0000	8300	0cec	908d	0000	0000	8952
00000060	e8c3	fffc	ffff	c483	b810	0000	0000	658d
00000070	59f8	5d5b	618d	c3fc	6548	6c6c	206f	6f57
00000080	6c72	2164	8b00	2404	00c3	4347	3a43	2820
00000090	6544	6962	6e61	3620	332e	302e	312d	2938

偏移量

数据，两字节一组，小端

```
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
```

```
$ 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	\$0xfffffffff0,%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>

```
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
```

```
$ 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	\$0xfffffffff0,%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>

```

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

```

```

$ 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
4: 83 e4 f0
7: ff 71 fc
a: 55
b: 89 e5
d: 53
e: 51
f: e8 fc ff ff ff

```

```

lea    0x4(%esp),%ecx
and     $0xfffffffff0,%esp
pushl   -0x4(%ecx)
push    %ebp
mov     %esp,%ebp
push    %ebx
push    %ecx
call    10 <main+0x10>

```

交给机器执行的
就是指令的序列

PA的原理



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    $0xffffffff0,%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>
```

编写

软硬件遵守共同的规范

指令集体系结构
Instruction Set Architecture, ISA

用硬件实现ISA



实现

PA的原理



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    $0xffffffff0,%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>
```

编写

软硬件遵守共同的规范

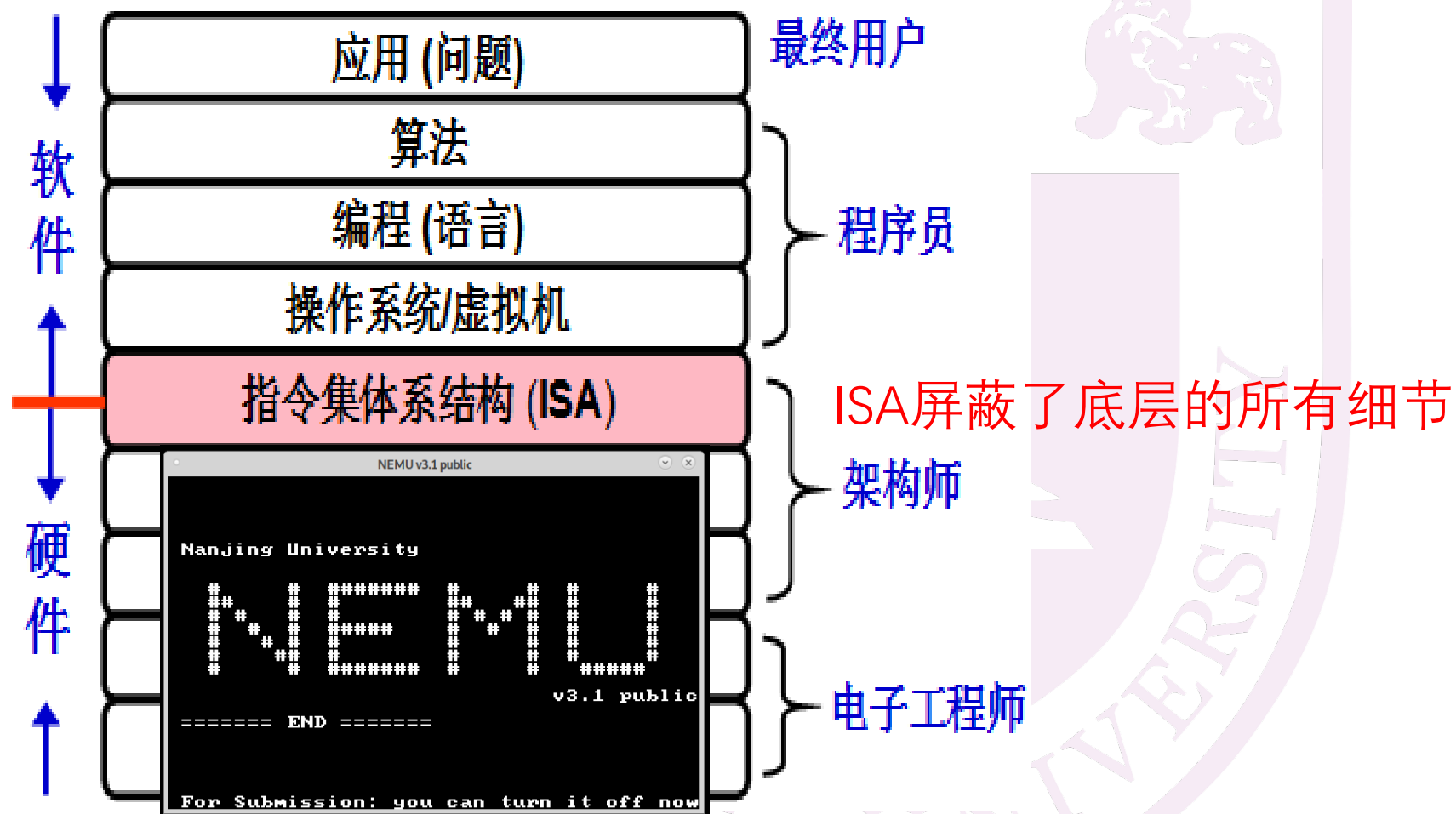
指令集体系结构
Instruction Set Architecture, ISA

替换ISA实现方案，用软件模拟



实现

PA的原理



软件



编译得到指令的序列

```
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     call   10 <main+0x10>
```

PA的原理

pa_nju/

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

trap.h

// trap相关定义, 不可改动

kernel

// 一个微型操作系统内核

Makefile

// 帮助编译和执行工程的Makefile

Makefile.git

// 和git有关的部分

nemu

// NEMU

src

main.c

// NEMU入口

testcase

// 测试用例

scripts

// 框架代码功能脚本, 不可改动

libs

// 框架代码所使用的库, 不可改动

软件



编译得到指令的序列

```
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      call   10 <main+0x10>
```

ISA: Intel i386

pa_nju/

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

trap.h

// trap相关定义, 不可改动

kernel

// 一个微型操作系统内核

Makefile

// 帮助编译和执行工程的Makefile

Makefile.git

// 和git有关的部分

nemu

// NEMU

src

main.c

// NEMU入口

testcase

// 测试用例

scripts

// 框架代码功能脚本, 不可改动

libs

// 框架代码所使用的库, 不可改动

PA的原理

软件



编译得到指令的序列

```
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     call   10 <main+0x10>
```

ISA: Intel i386



模拟
实现

pa_nju/

game

include

config.h

trap.h

kernel

Makefile

Makefile.git

nemu

src

main.c

testcase

scripts

libs

// 包含游戏相关代码

// PA整体依赖的一些文件

// 一些配置用的宏

// trap相关定义, 不可改动

// 一个微型操作系统内核

// 帮助编译和执行工程的Makefile

// 和git有关的部分

// NEMU

// NEMU入口

// 测试用例

// 框架代码功能脚本, 不可改动

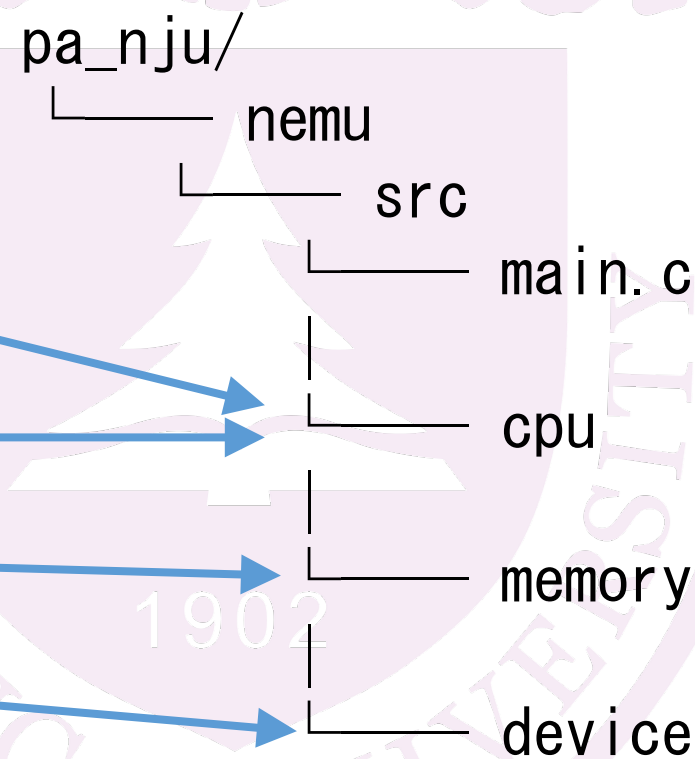
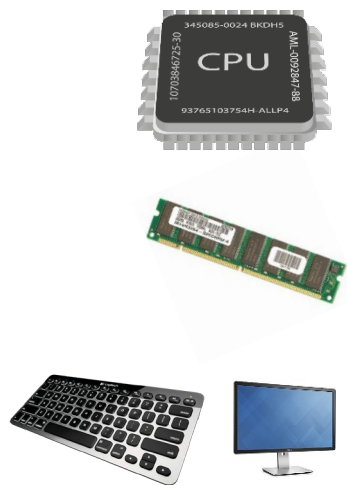
// 框架代码所使用的库, 不可改动

可以解释执行i386指令

PA的原理

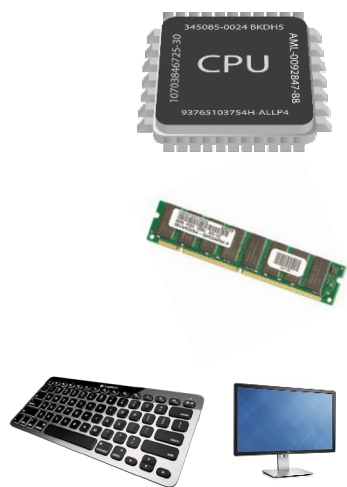
PA分为四个大的阶段，每个大阶段又拆分为三个小阶段

- 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

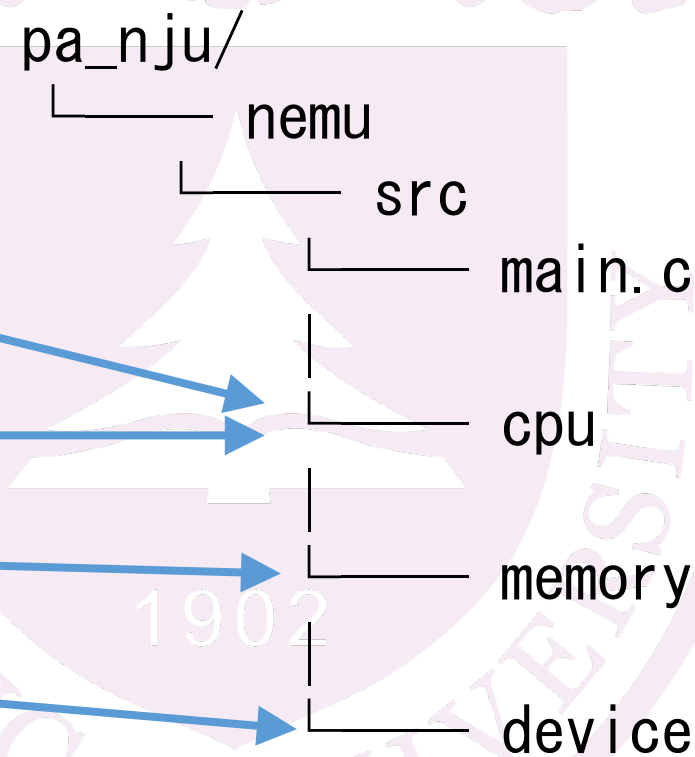


PA的原理

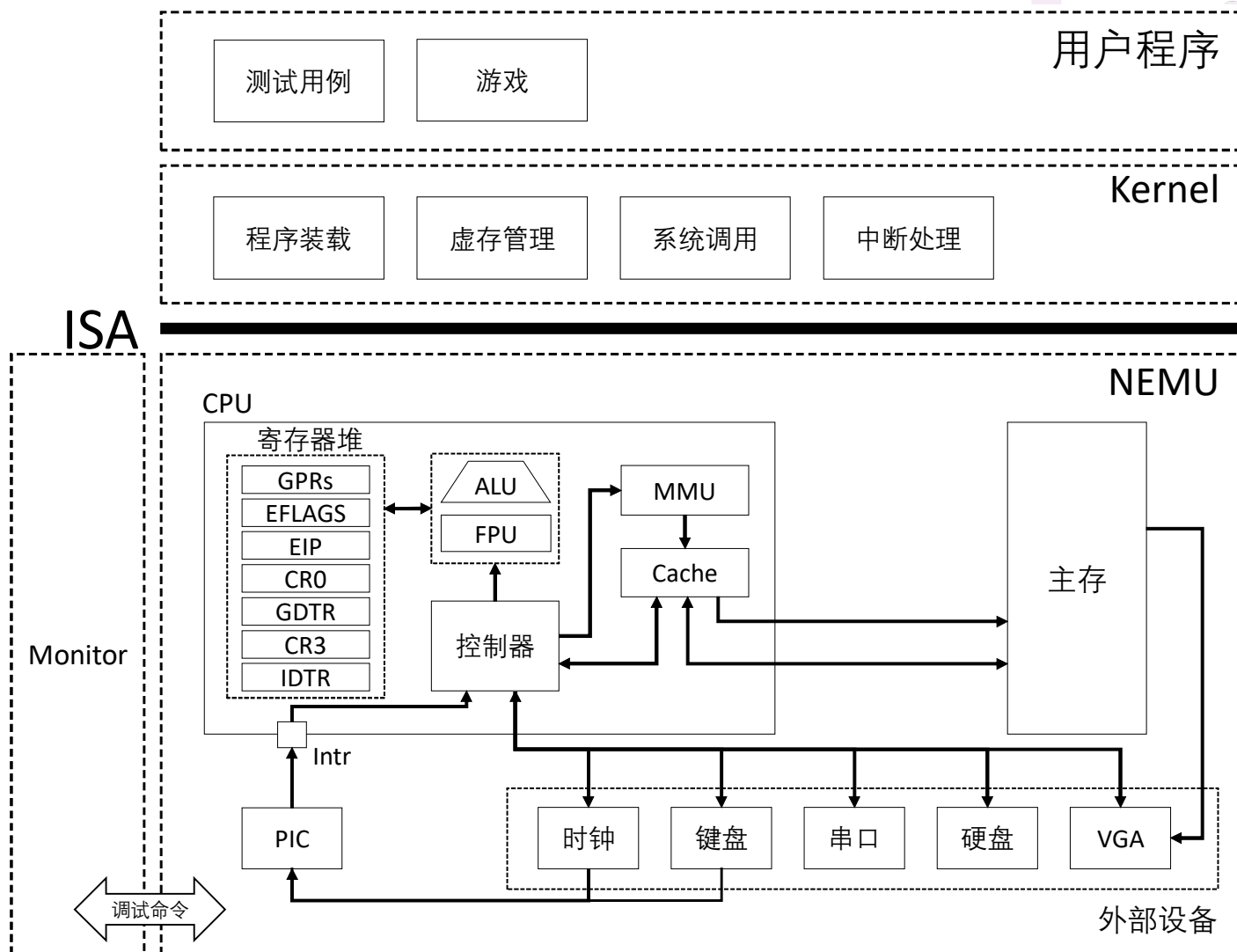
PA分为四个大的阶段，每个大阶段又拆分为三个小阶段



- PA 1
- PA 2
- PA 3
- PA 4

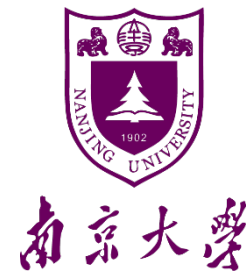


PA简介



完整的PA





PA 0 完成

重要事项

- 不许抄袭！ 必须通过在线平台完成整个实验！
- 记得时常 `git commit` 保存进度
- 一次只运行一个 `nemu` 进程，防止服务器过载
- 有问题找老师和助教

今天需要完成的内容

- 保证自己能够登陆在线实验平台
- 修改网站和SSH两处密码
- 熟悉网页的操作