通过ProjectN构建RISC-V计算机系统 塑造学生的初步系统能力

余子濠
中科院计算所

袁春风
南京大学

2019.11.12@深圳

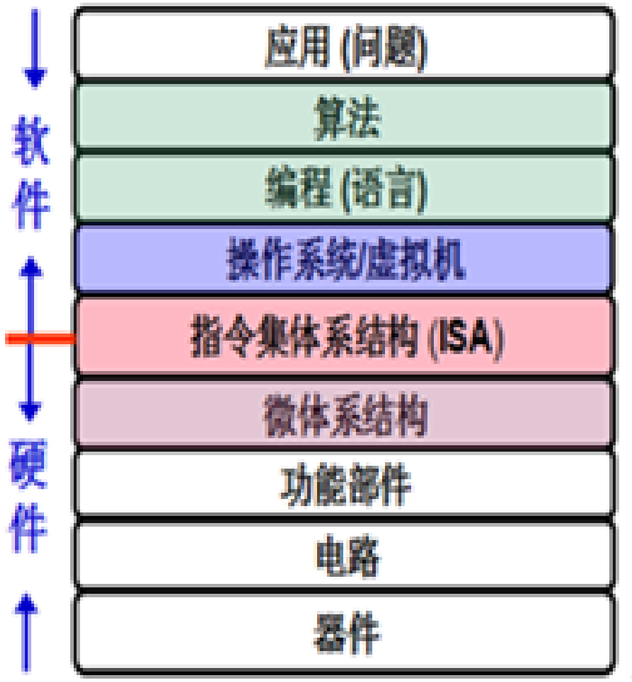
来自本科教育的灵魂拷问

- 计算机系统方向课程的终极问题

```
[23:46:56 ~]$ vim hello.c
[23:47:27 ~]$ gcc hello.c -o hello
[23:47:39 ~]$ ./hello
Hello World!
[23:47:42 ~]$
```

- 这个过程计算机都做了些什么？
  - **程序如何在计算机上运行的?**

计算机系统抽象层的转换



- 一个打通全栈的实验很重要
  - **只有亲自实现一个完整的计算机系统，并在其上运行真实的程序，才能明白系统栈每一个层次之间的关系，才会对“程序如何在计算机上运行”有深刻的认识**

南京大学的教学生态系统ProjectN

- 覆盖从处理器到应用程序
- 可适配多套指令集

PA简介

- 南京大学大二上“计算机系统基础”课程项目
- Programming Assignment
  - 指导学生实现一个功能完备的模拟器NEMU(NJU EMUlator)
    - 支持x86(教学版子集)/**riscv32**/mips32
  - 最终在NEMU上运行真实游戏“仙剑奇侠传”
  - 揭示“程序在计算机上运行”的基本原理

PA由ProjectN的部分组件构成

ProjectN组件	说明
Navy-apps	应用程序集
NCC	NJU C Compiler, C编译器
Newlib	嵌入式C库(从官方版本移植)
Nanos/Nanos-lite	NanJU OS, 操作系统/简化版
Nexus-AM	抽象计算机, ISA抽象层
NEMU	NJU EMUlator, 全系统模拟器
NPC	NJU Personal Computer, SoC
NOOP	NJU Out-of-Order Processor, 处理器

- PA包括6部分连贯的实验内容:
  - PA0: 开发环境配置(准备实验)
  - PA1: 简易调试器
  - PA2: 冯诺依曼计算机系统
  - PA3: 批处理系统
  - PA4: 分时多任务
  - PA5: 程序性能优化(选做)

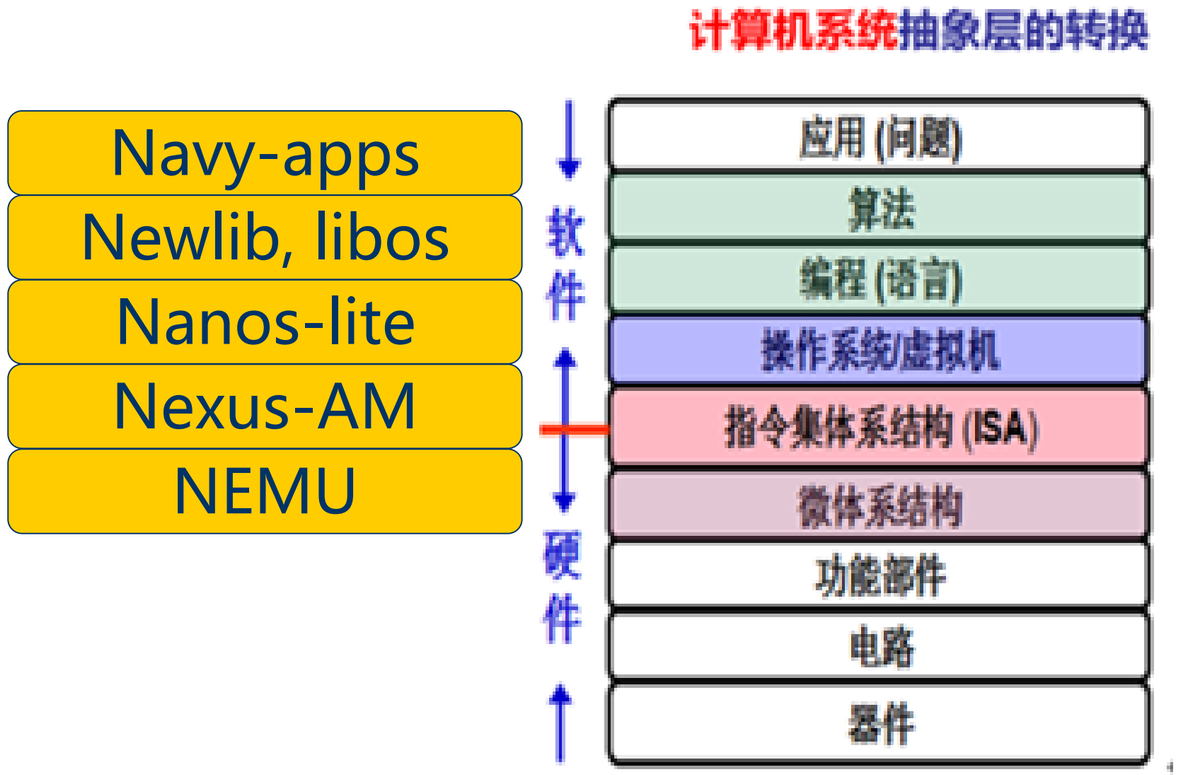
PA资源

- 实验平台与工具
  - GNU/Linux + gcc + C
  - 其它工具: gdb, make, git
- 实验讲义
  - https://nju-projectn.github.io/ics-pa-gitbook
- 框架代码
  - https://github.com/NJU-ProjectN/ics-pa

- 无需编写RTL
  - 因此无需FPGA开展实验

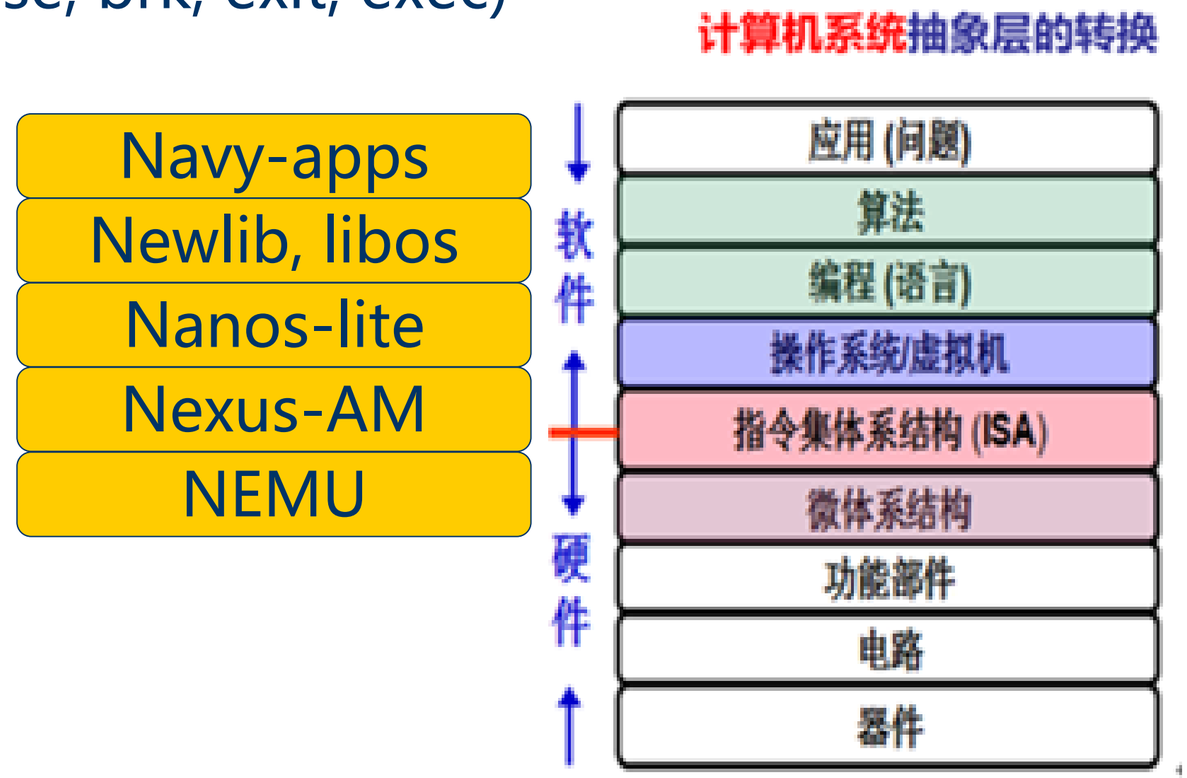
PA组成

- NEMU 全系统模拟器
  - 监控模块Monitor(单步执行, 打印寄存器/内存, 表达式求值, 监视点)
  - RV32IM指令的完整指令周期(取指, 译码, 执行, 更新PC)
  - 中断异常(ecall和M-mode时钟中断)
  - Sv32分页的MMU
  - 128MB物理内存
  - 串口, 时钟, 键盘, VGA(功能经过简化)
- Nexus-AM ISA抽象层(C语言API)
  - TRM 计算抽象
  - IOE 输入输出抽象
  - CTE 上下文管理抽象
  - VME 虚存抽象



PA组成

- Nanos-lite 简易多任务OS
  - ELF加载器
  - 中断异常事件分发
  - 8个系统调用(open, read, write, lseek, close, brk, exit, exec)
  - 简易VFS
    - 文件数量, 大小皆固定, 没有目录
    - 文件系统基于ramdisk, 无需持久化
    - 3个设备文件(映射到设备抽象层)
  - 简易分页存储管理(顺序分配, 不释放)
  - 两个进程的简易轮转调度
- Navy-apps 库和程序集
  - libos – 系统调用接口
  - Newlib – C库, 支持POSIX标准
  - hello, 仙剑奇侠传, LiteNES



PA1 - 简易调试器
[TRM]

```
No image is given. Use the default build-in image.
Welcome to NEMU!
[src/monitor/monitor.c,23,welcome] Build time: 17:39:41, Jul 26 2017
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100026
```

只有load/store
指令的程序

学生任务: 实现简易调试器

理解最简单计算机的基本构成

- 单步执行 - PC
- 打印寄存器 - Reg
- 扫描内存 - Mem

基础设施(帮助调试), 复习C语言程序设计

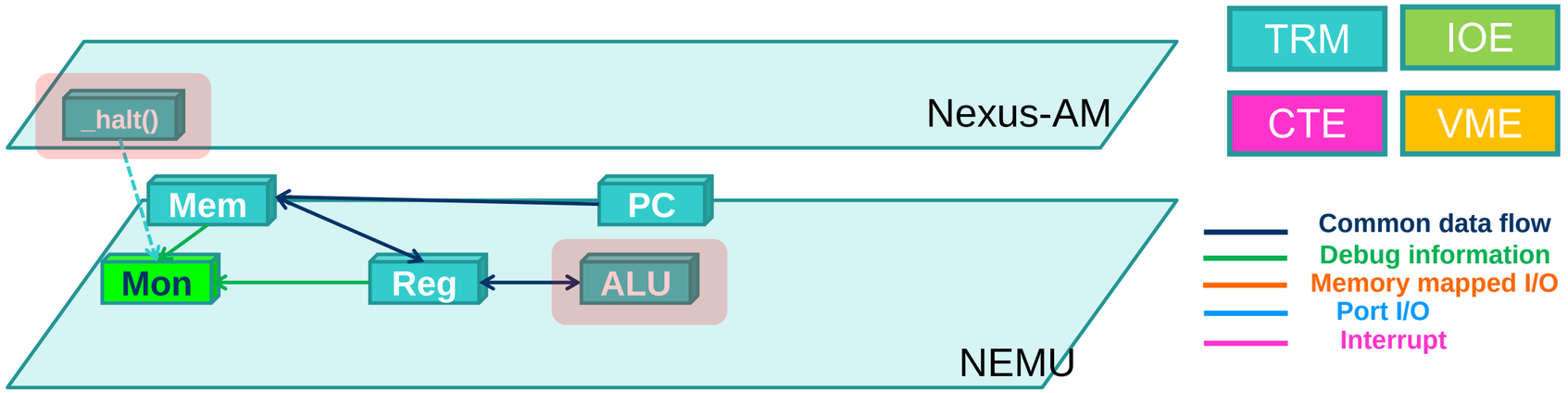
- 表达式求值(递归)
- 监视点(指针, 链表)

PA2 - 冯诺依曼计算机系统
[TRM(x86/riscv32/mips32指令集)]

学生任务:
添加常用指令 -> cputest测试集
- 用RTL实现指令

矩阵乘法

```
[src/monitor/cross-check/cross-check.c,97,init_check] Connect to QEMU successfully
The image is /home/yuzihao/NJUCS-ComputerSystemLab/nexus-am/tests/cputest/build/matrix-mul-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,23,welcome] Build time: 17:39:41, Jul 26 2017
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x0010005e
```

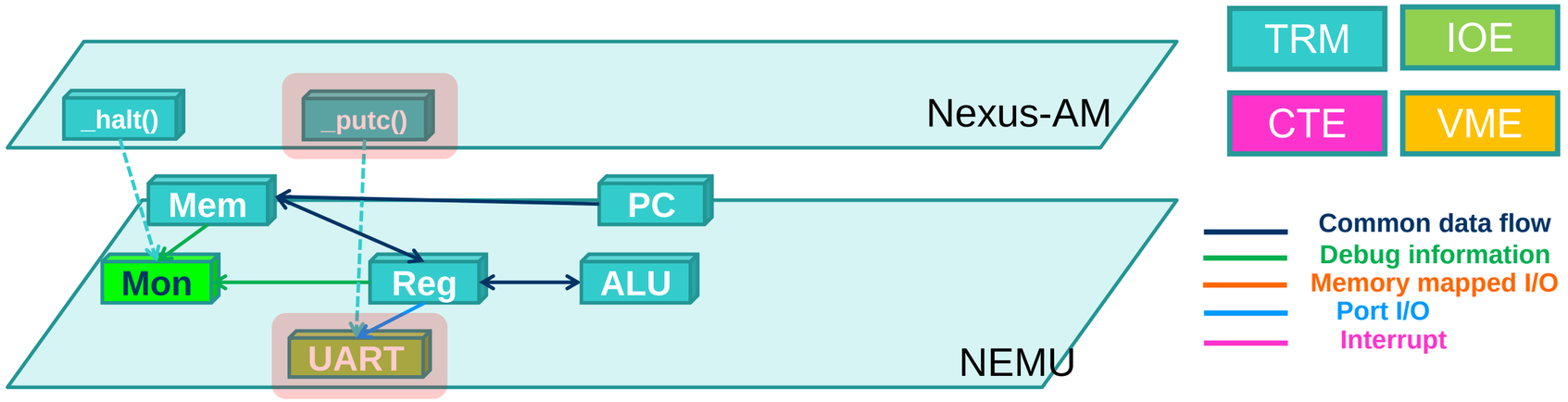


PA2 - 冯诺依曼计算机系统
[TRM(x86/riscv32/mips32指令集) + IOE]

```
The image is /home/yuzihao/NJUCS-Computer
Welcome to NEMU!
[src/monitor/monitor.c,23,welcome] Build
For help, type "help"
(nemu) c
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
nemu: HIT GOOD TRAP at eip = 0x0010006e
```

学生任务:
添加常用指令 -> cputest测试集
- 用RTL实现指令

添加IOE
- UART -> _putc() -> hello程序

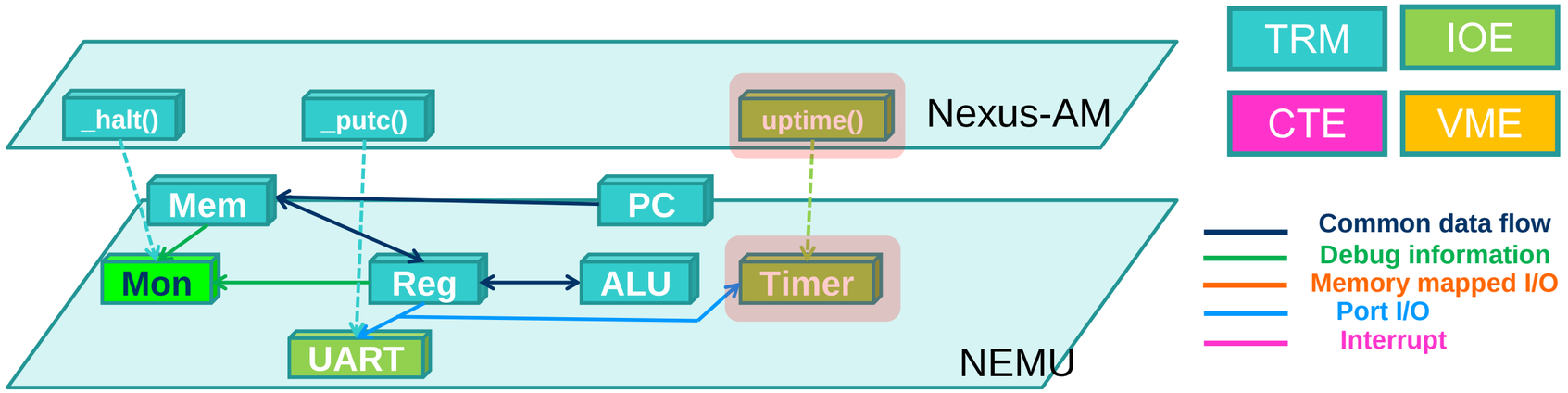


PA2 - 冯诺依曼计算机系统
[TRM(x86/riscv32/mips32指令集) + IOE]

```
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time:
For help, type "help"
(nemu) c
1 second.
2 seconds.
3 seconds.
4 seconds.
5 seconds.
6 seconds.
7 seconds.
8 seconds.
9 seconds.
10 seconds.
11 seconds.
12 seconds.
13 seconds.
```

学生任务:
添加常用指令 -> cputest测试集
- 用RTL实现指令

添加IOE
- UART -> _putc() -> hello程序
- Timer -> uptime() -> timertest程序



PA2 - 冯诺依曼计算机系统
[TRM(x86/riscv32/mips32指令集) + IOE]

```
The image is /home/yuzihao/NJUCS-ComputerSystemLab/nexus
Welcome to NEMU!
[src/monitor/monitor.c,23,welcome] Build time: 16:34:25,
For help, type "help"
(nemu) c
[qsort] Quick sort: * Passed.
  min time: 2891 ms [190]
[queen] Queen placement: * Passed.
  min time: 5205 ms [99]
[bf] Brainf**k interpreter: * Passed.
  min time: 29526 ms [88]
[fib] Fibonacci number: * Passed.
  min time: 57489 ms [49]
[sieve] Eratosthenes sieve: * Passed.
  min time: 48906 ms [86]
[15pz] A* 15-puzzle search: * Passed.
  min time: 9257 ms [62]
[dinic] Dinic's maxflow algorithm: * Passed.
  min time: 8477 ms [159]
[lzip] Lzip compression: * Passed.
  min time: 24113 ms [109]
[ssort] Suffix sort: * Passed.
  min time: 4714 ms [125]
[md5] MD5 digest: * Passed.
  min time: 45426 ms [43]
================================================
MicroBench PASS        101 Marks
                 vs. 100000 Marks (i7-6700 @ 3.40GHz)
nemu: HIT GOOD TRAP at eip = 0x00100032
```

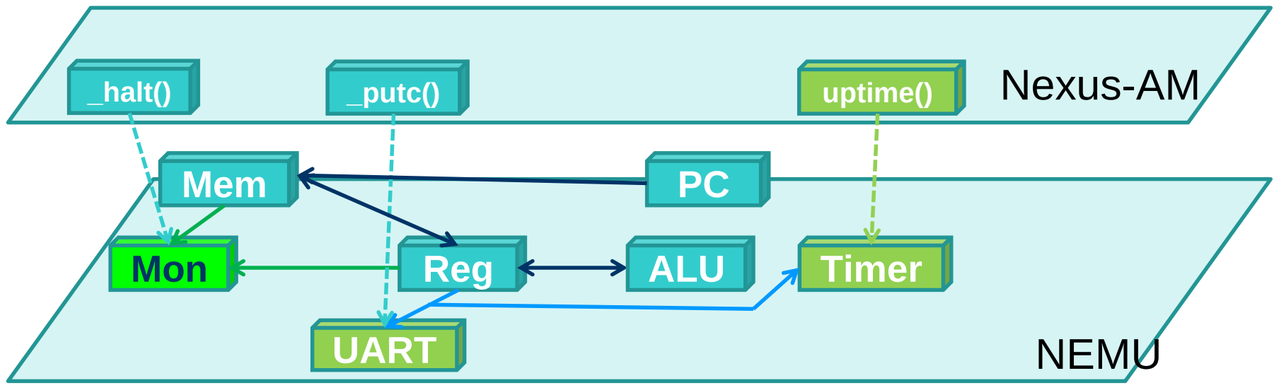
学生任务:

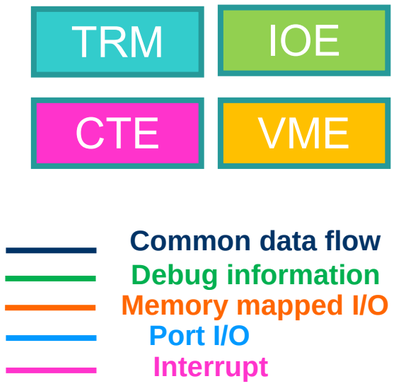
添加常用指令 -> cputest测试集

- 用RTL实现指令

添加IOE

- UART -> _putc() -> hello程序
- Timer -> uptime() -> timertest程序
  - 运行microbench



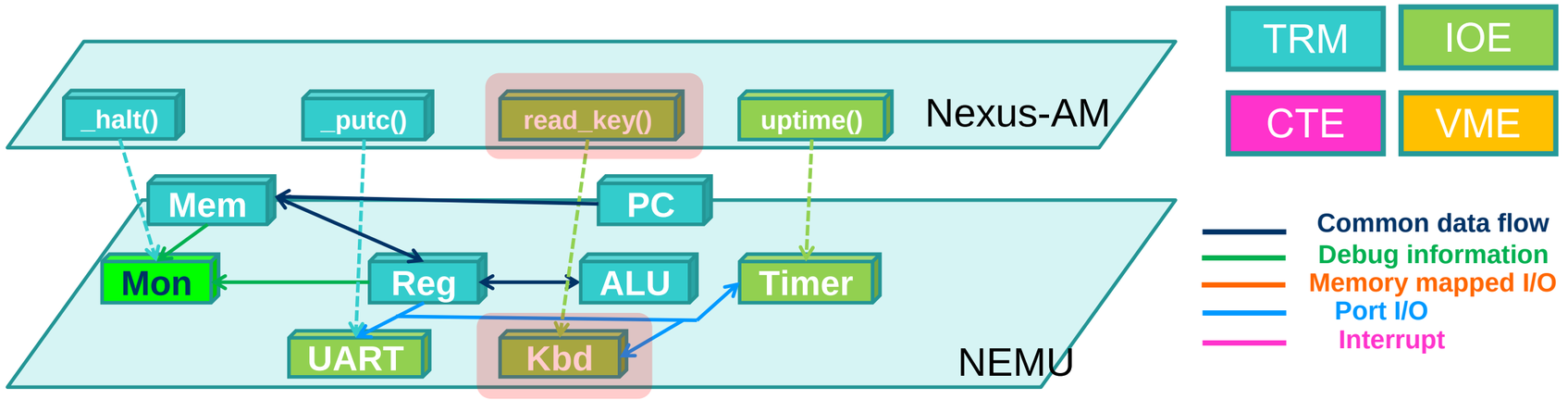


PA2 - 冯诺依曼计算机系统
[TRM(x86/riscv32/mips32指令集) + IOE]

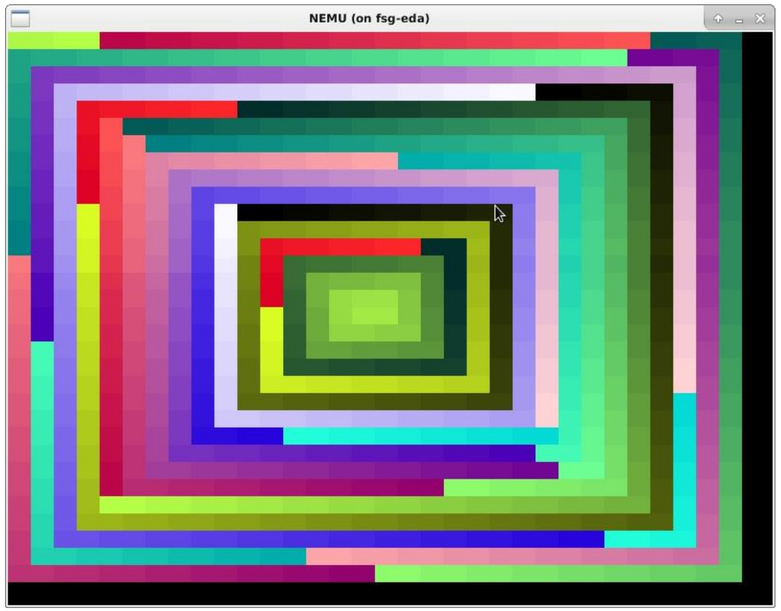
```
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome]
For help, type "help"
(nemu) c
Get key: 15 1 down
Get key: 15 1 up
Get key: 43 A down
Get key: 49 J down
Get key: 43 A up
Get key: 45 D down
Get key: 49 J up
Get key: 45 D up
Get key: 54 RETURN down
Get key: 54 RETURN up
Get key: 70 SPACE down
Get key: 70 SPACE up
```

学生任务:
添加常用指令 -> cputest测试集
- 用RTL实现指令

添加IOE
- UART -> _putc() -> hello程序
- Timer -> uptime() -> timertest程序
  - 运行microbench
- Kbd ->read_key() -> keytest程序

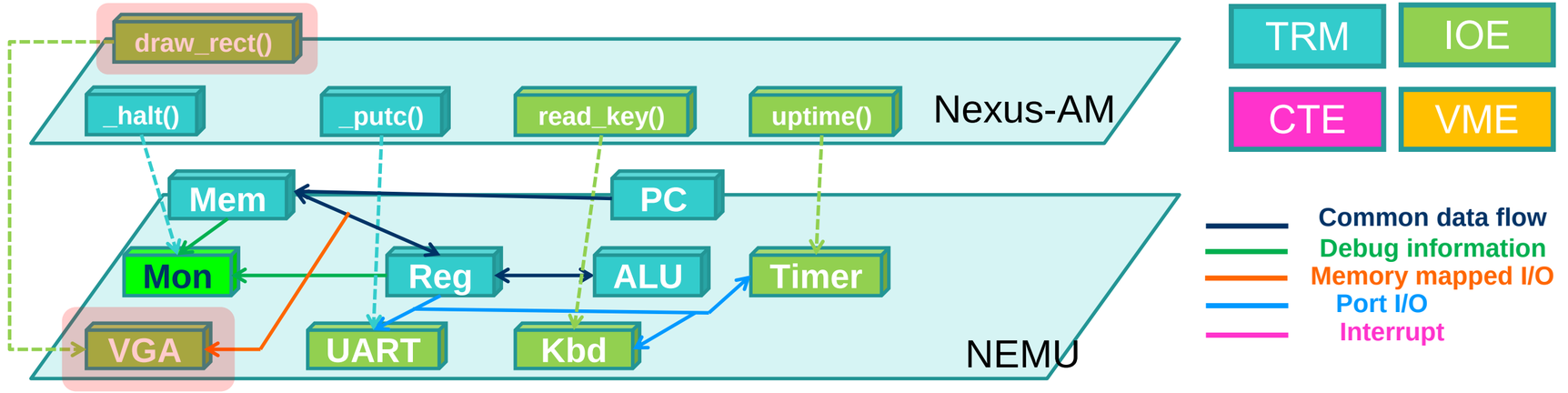


PA2 - 冯诺依曼计算机系统
[TRM(x86/riscv32/mips32指令集) + IOE]

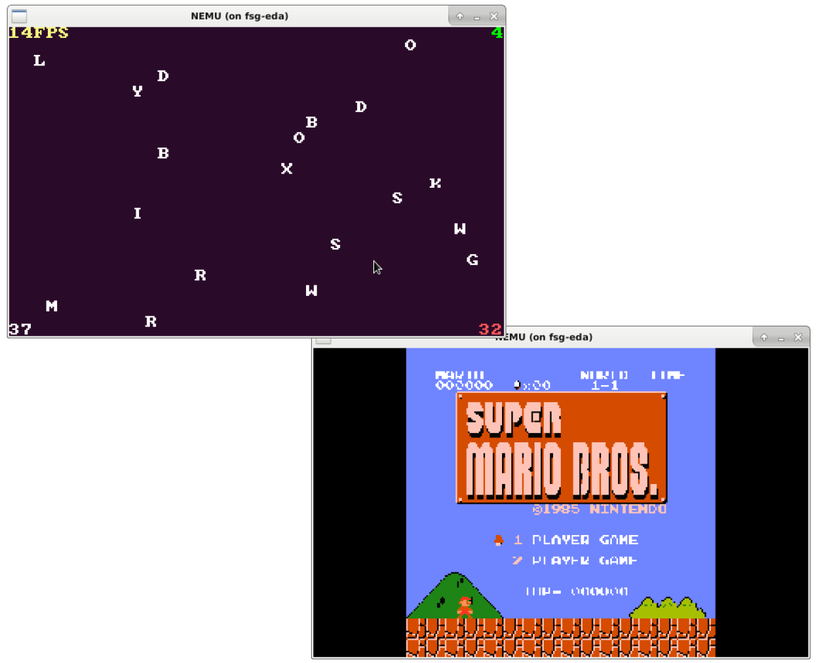


学生任务:
添加常用指令 -> cputest测试集
- 用RTL实现指令

添加IOE
- UART -> _putc() -> hello程序
- Timer -> uptime() -> timertest程序
  - 运行microbench
- Kbd ->read_key() -> keytest程序
- VGA -> draw_rect() -> videotest程序



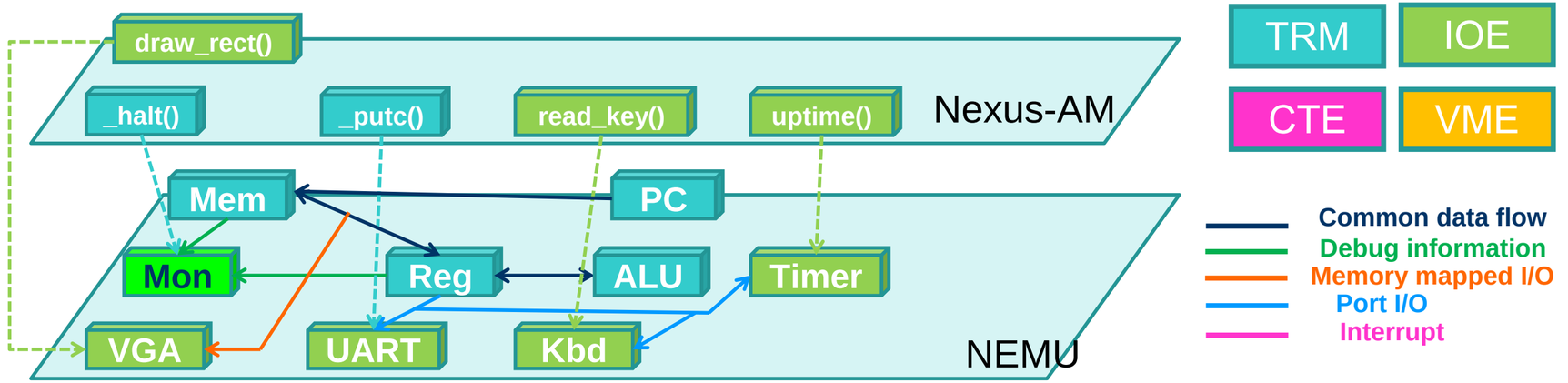
PA2 - 冯诺依曼计算机系统
[TRM(x86/riscv32/mips32指令集) + IOE]



学生任务:
添加常用指令 -> cputest测试集
- 用RTL实现指令

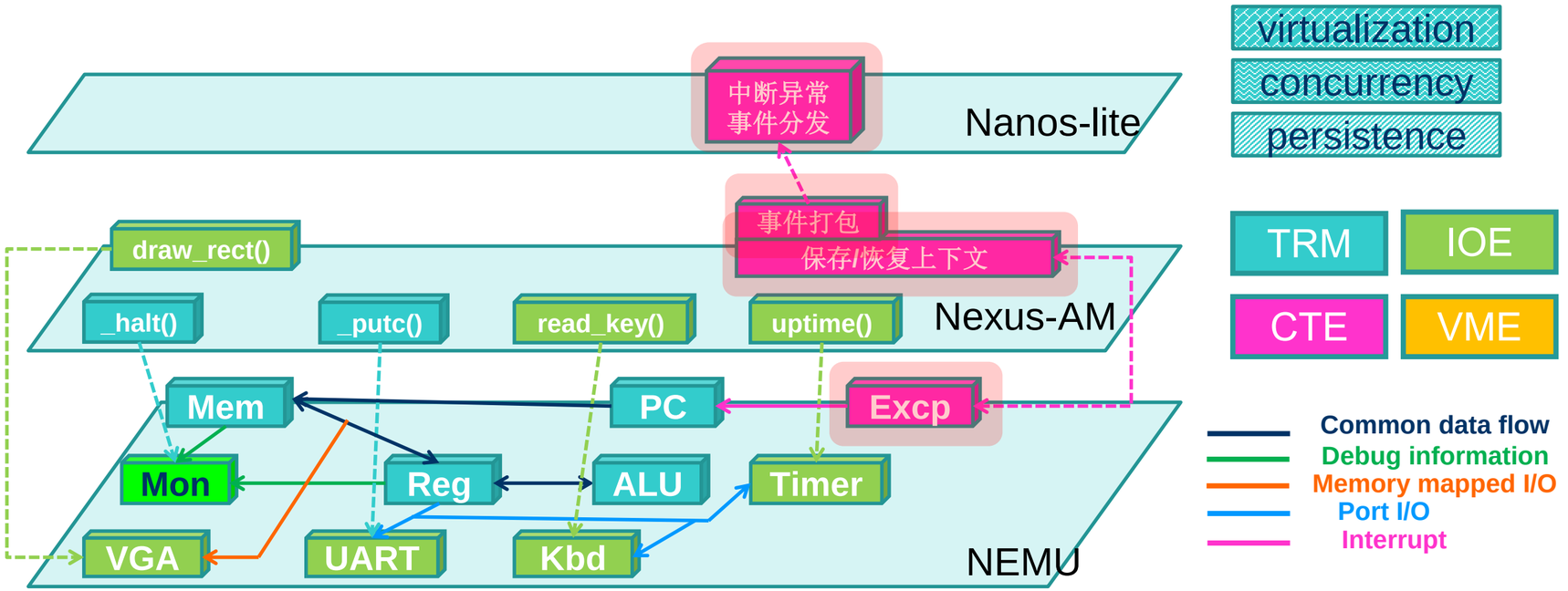
添加IOE
- UART -> _putc() -> hello程序
- Timer -> uptime() -> timertest程序
  - 运行microbench
- Kbd ->read_key() -> keytest程序
- VGA -> draw_rect() -> videotest程序

运行打字游戏, Litenes



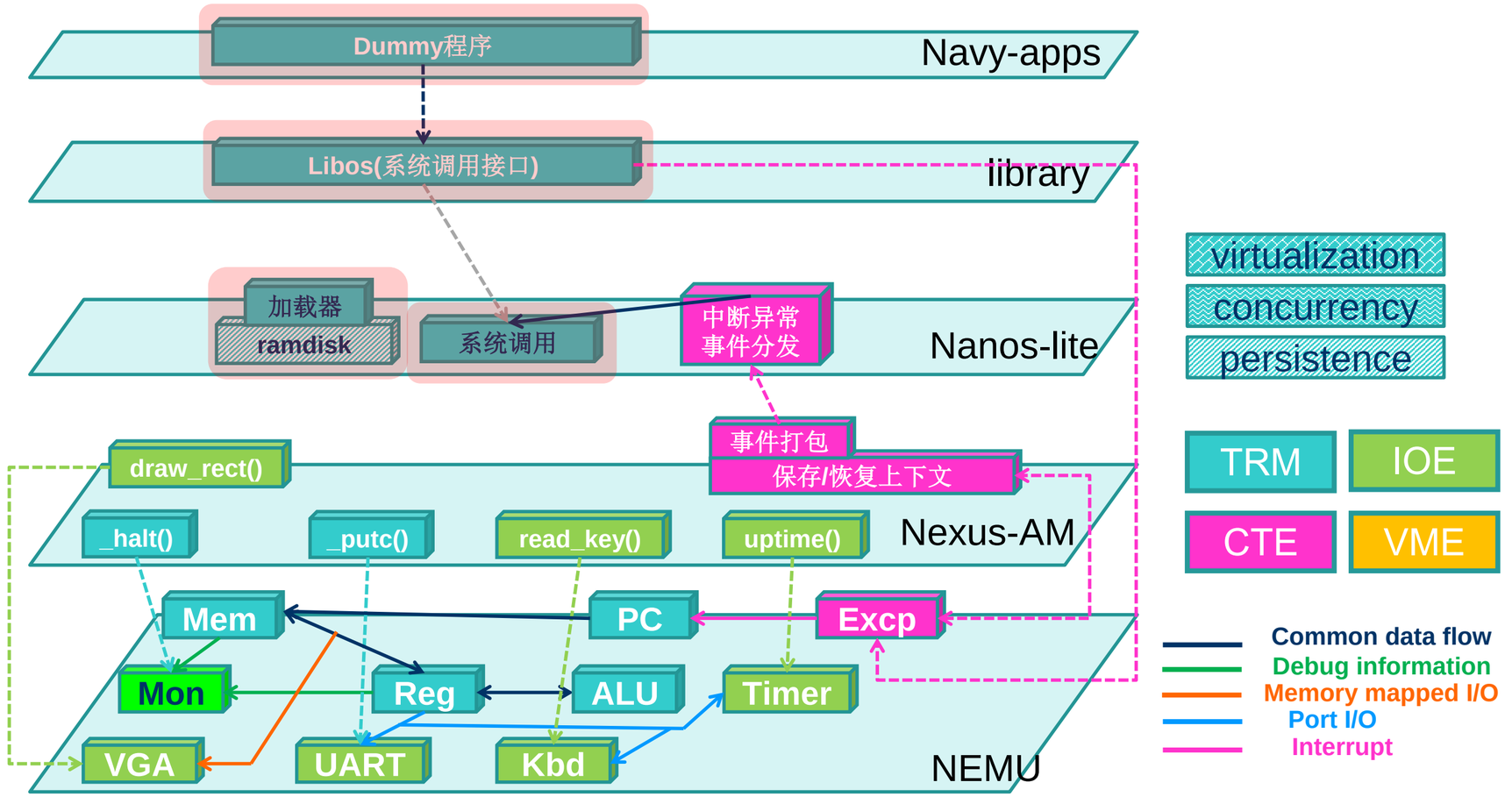
PA3 - 批处理系统
[TRM(指令集) + IOE + CTE]

学生任务:
添加CTE -> _yield()



PA3 - 批处理系统
[TRM(指令集) + IOE + CTE]

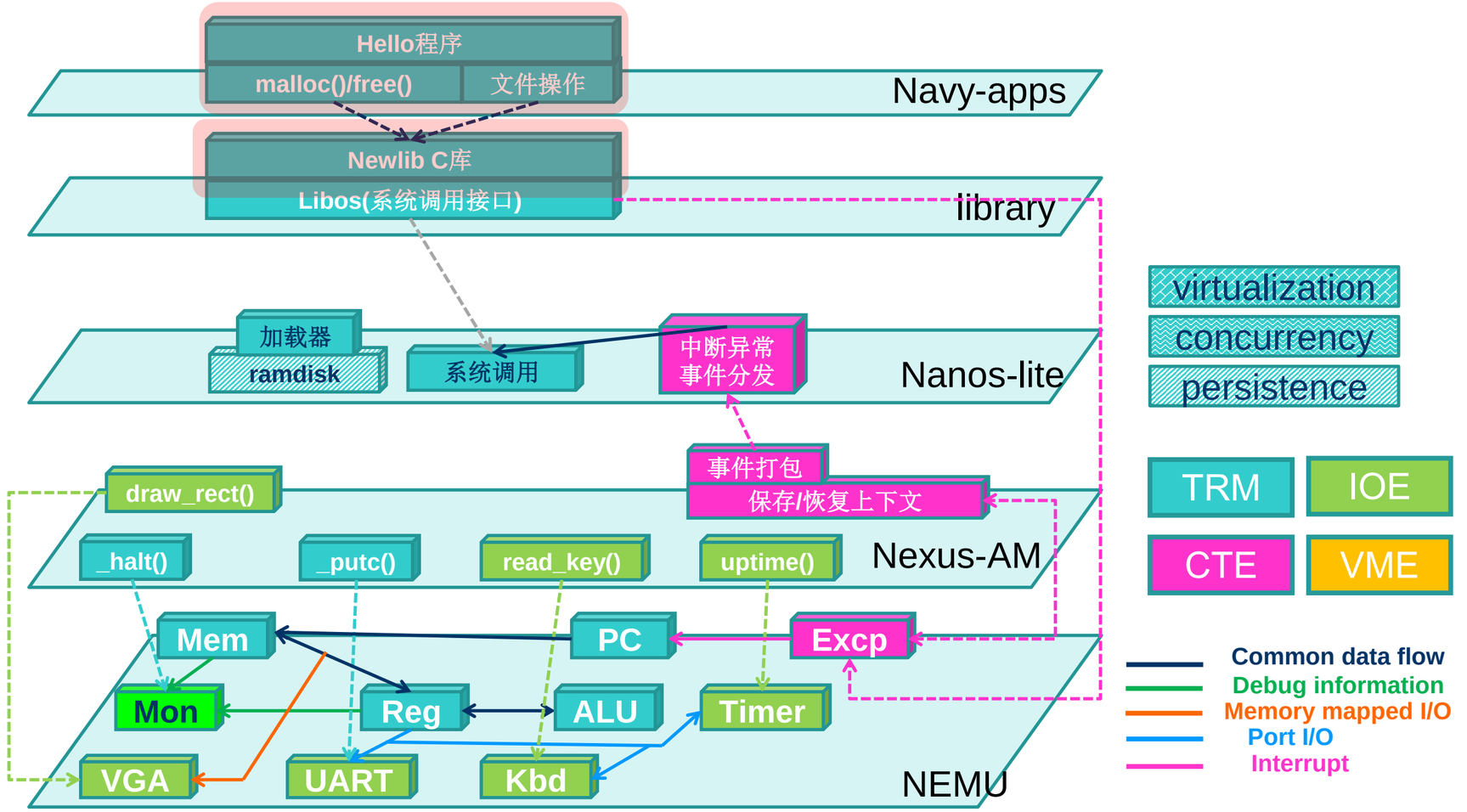
学生任务:
添加CTE -> _yield()
实现简易加载器
->系统调用SYS_yield



PA3 - 批处理系统
[TRM(指令集) + IOE + CTE]

学生任务:
添加CTE -> _yield()
实现简易加载器
->系统调用SYS_yield
添加SYS_write
-> hello程序

```
Hello World!
Hello World for the 2th time
Hello World for the 3th time
Hello World for the 4th time
Hello World for the 5th time
```



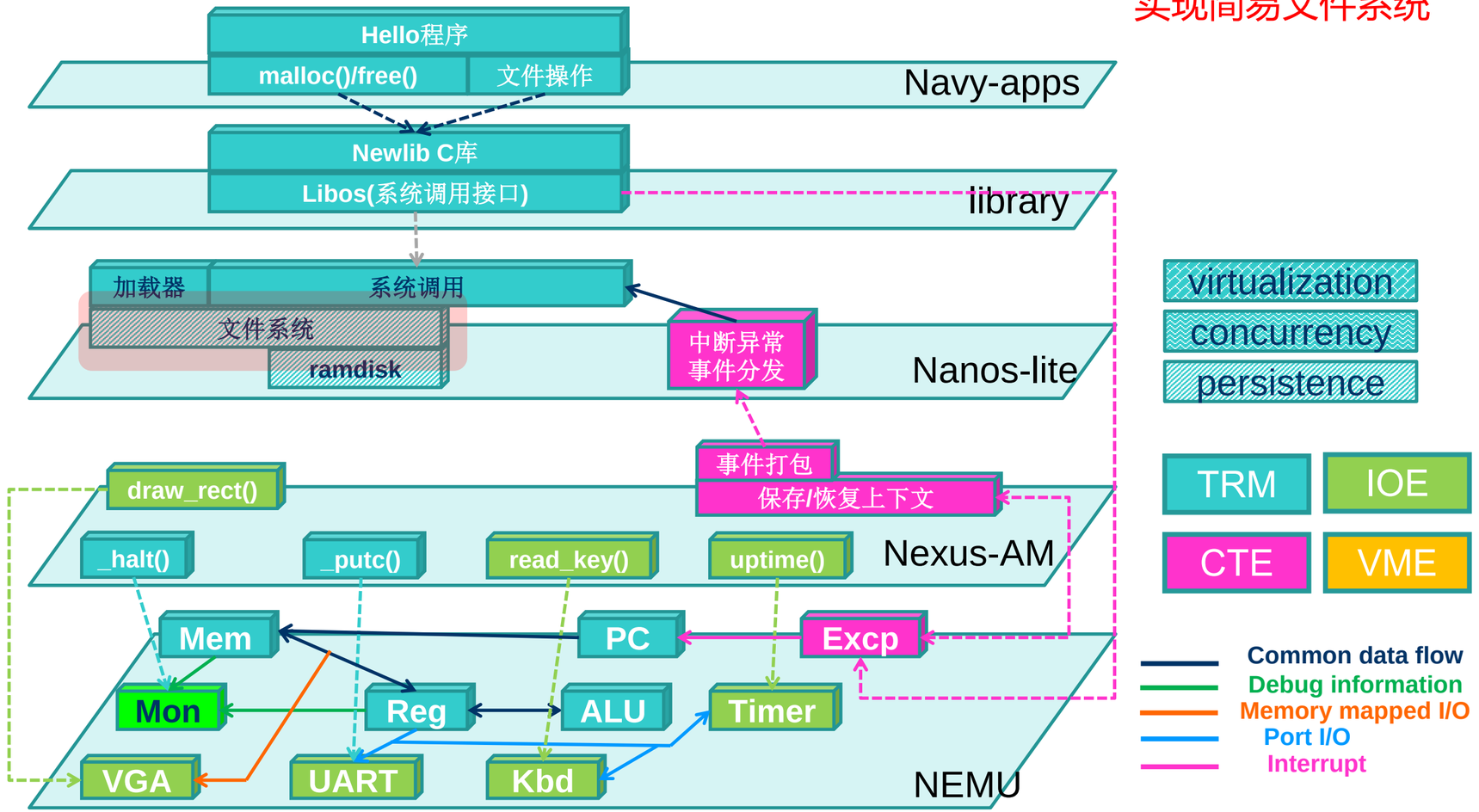
virtualization
concurrency
persistence

TRM
IOE
CTE
VME

Common data flow
Debug information
Memory mapped I/O
Port I/O
Interrupt

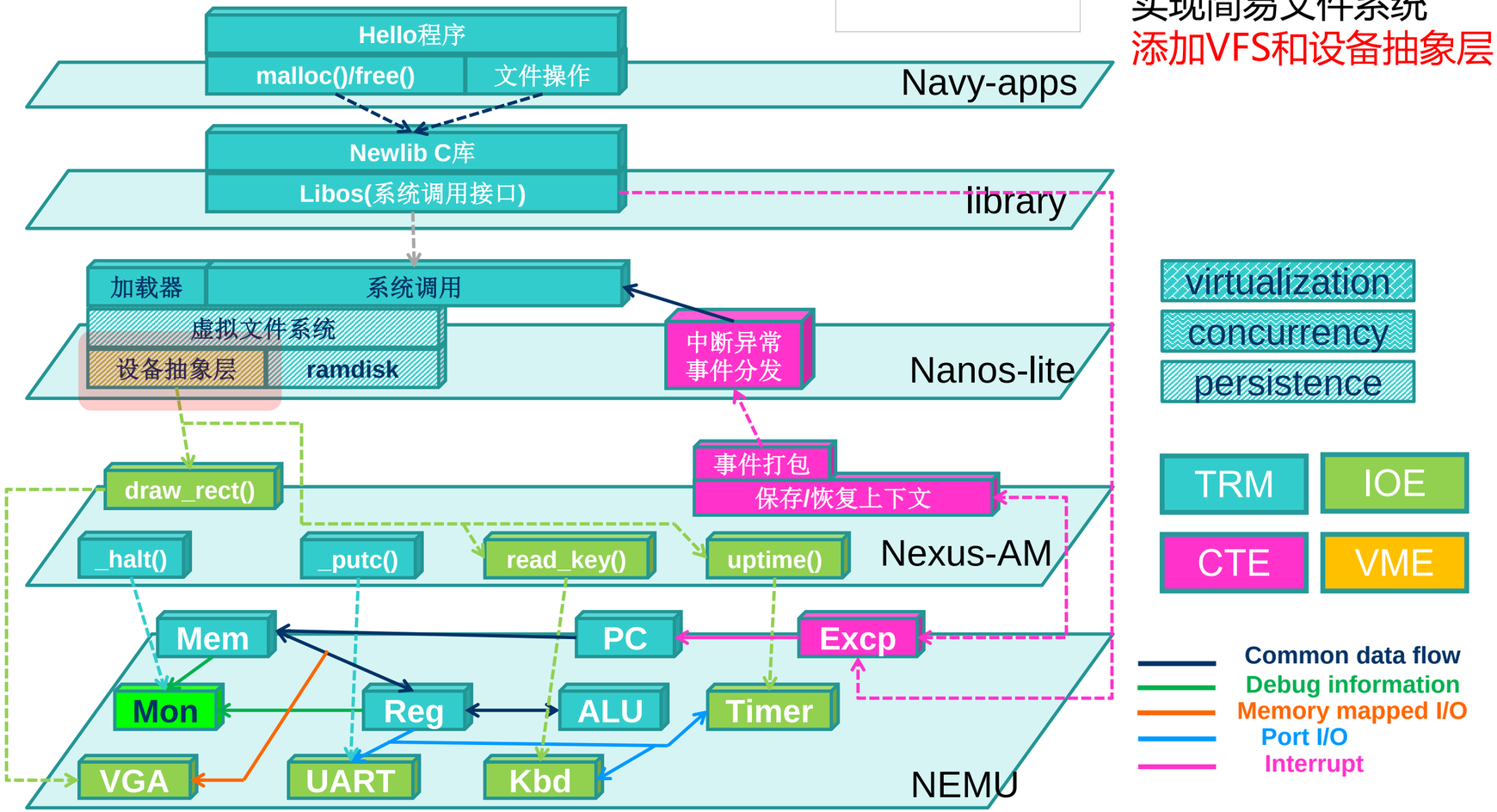
PA3 - 批处理系统
[TRM(指令集) + IOE + CTE]

学生任务:
添加CTE -> _yield()
实现简易加载器
->系统调用SYS_yield
添加SYS_write
-> hello程序
实现简易文件系统



PA3 - 批处理系统
[TRM(指令集) + IOE + CTE]

学生任务:
添加CTE -> _yield()
实现简易加载器
->系统调用SYS_yield
添加SYS_write
-> hello程序
实现简易文件系统
添加VFS和设备抽象层



virtualization
concurrency
persistence

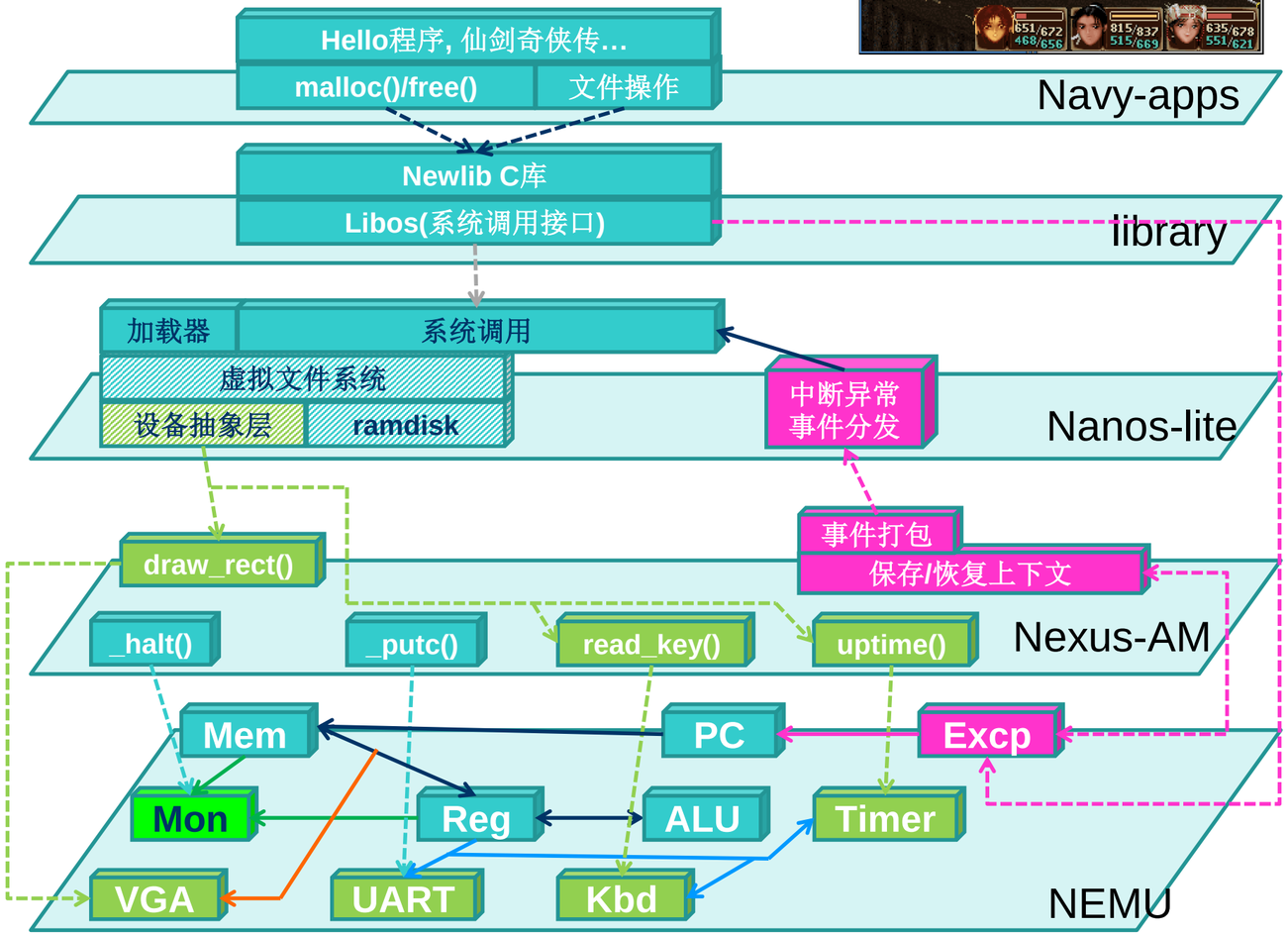
TRM
IOE
CTE
VME

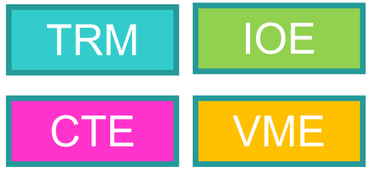
Common data flow
Debug information
Memory mapped I/O
Port I/O
Interrupt

PA3 - 批处理系统

[TRM(指令集) + IOE + CTE]

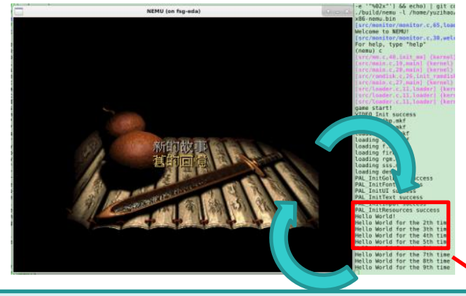
学生任务:
添加CTE -> _yield()
实现简易加载器
->系统调用SYS_yield
添加SYS_write
-> hello程序
实现简易文件系统
添加VFS和设备抽象层
运行仙剑奇侠传

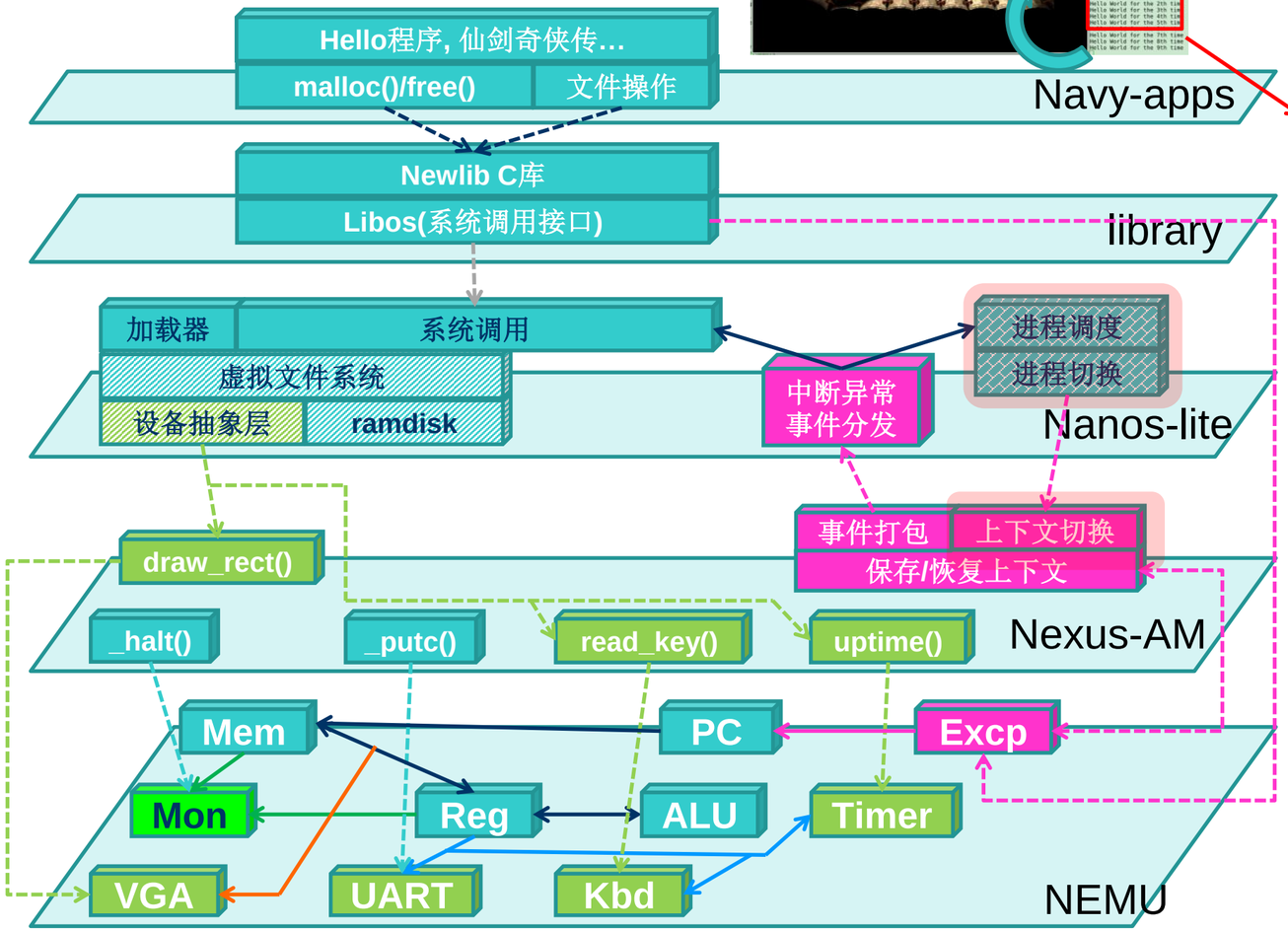




PA4 - 分时多任务
[TRM(指令集) + IOE + CTE + VME]

学生任务:
实现上下文切换
- 多道程序, 并发运行hello程序和仙剑






```
Hello World!
Hello World for the 2th time
Hello World for the 3th time
Hello World for the 4th time
Hello World for the 5th time
```

virtualization
concurrency
persistence

TRM
IOE
CTE
VME

Common data flow
Debug information
Memory mapped I/O
Port I/O
Interrupt

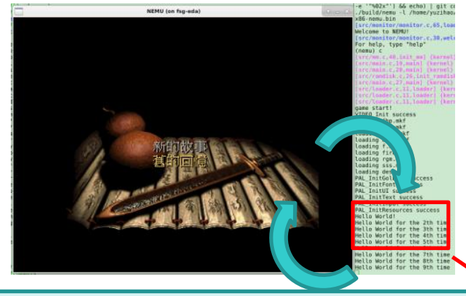
PA4 - 分时多任务

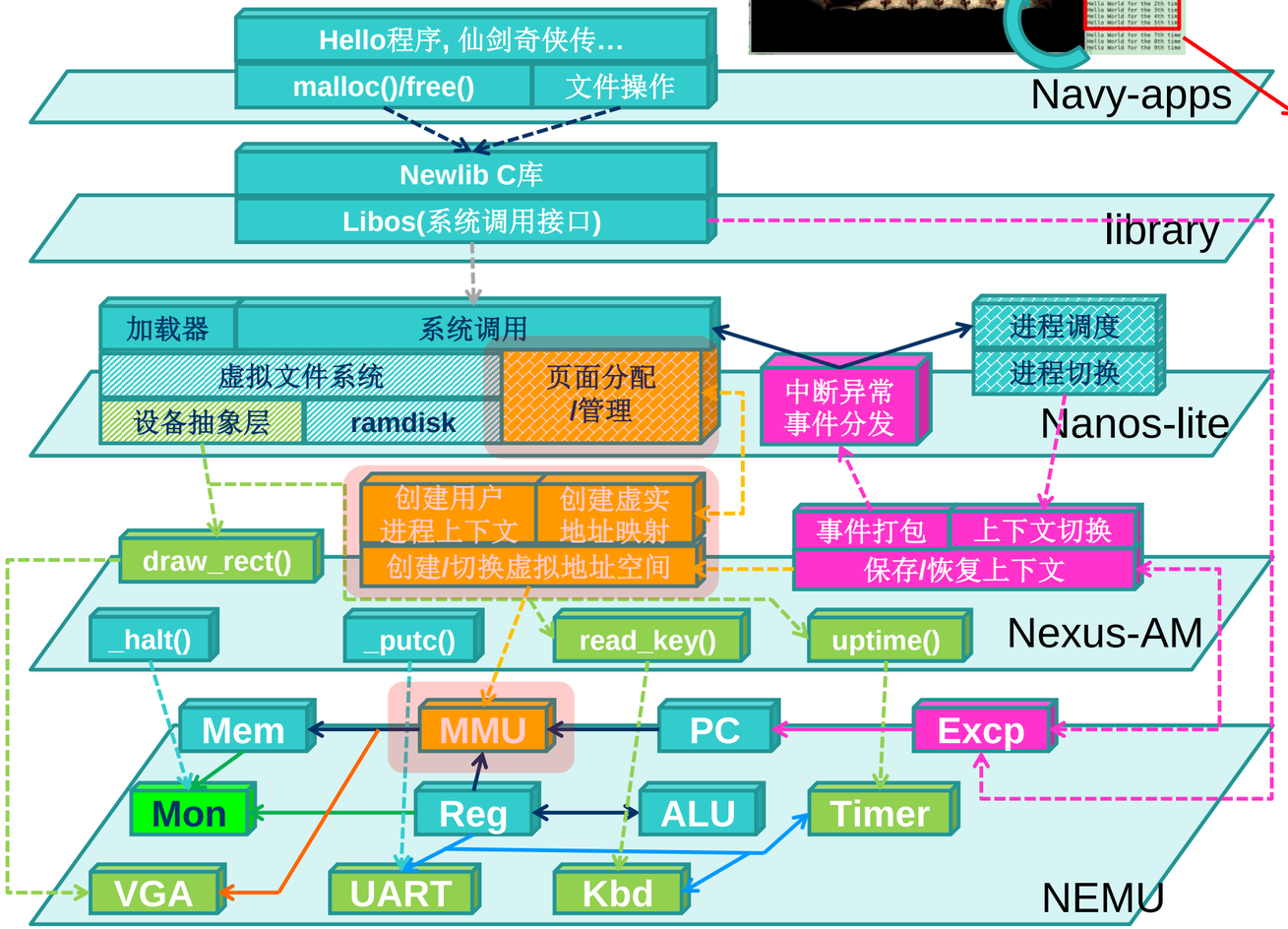
[TRM(指令集) + IOE + CTE + VME]

学生任务:
实现上下文切换

- 多道程序, 并发运行hello程序和仙剑

实现VME -> 支持虚存管理的多道程序






```
Hello World!
Hello World for the 2th time
Hello World for the 3th time
Hello World for the 4th time
Hello World for the 5th time
```

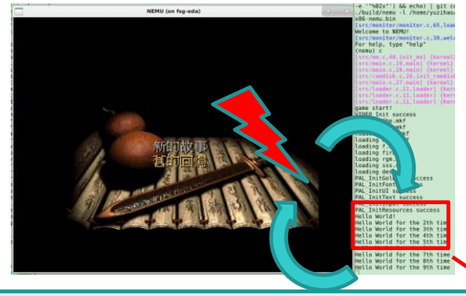
virtualization
concurrency
persistence

TRM IOE
CTE VME

Common data flow
Debug information
Memory mapped I/O
Port I/O
Interrupt

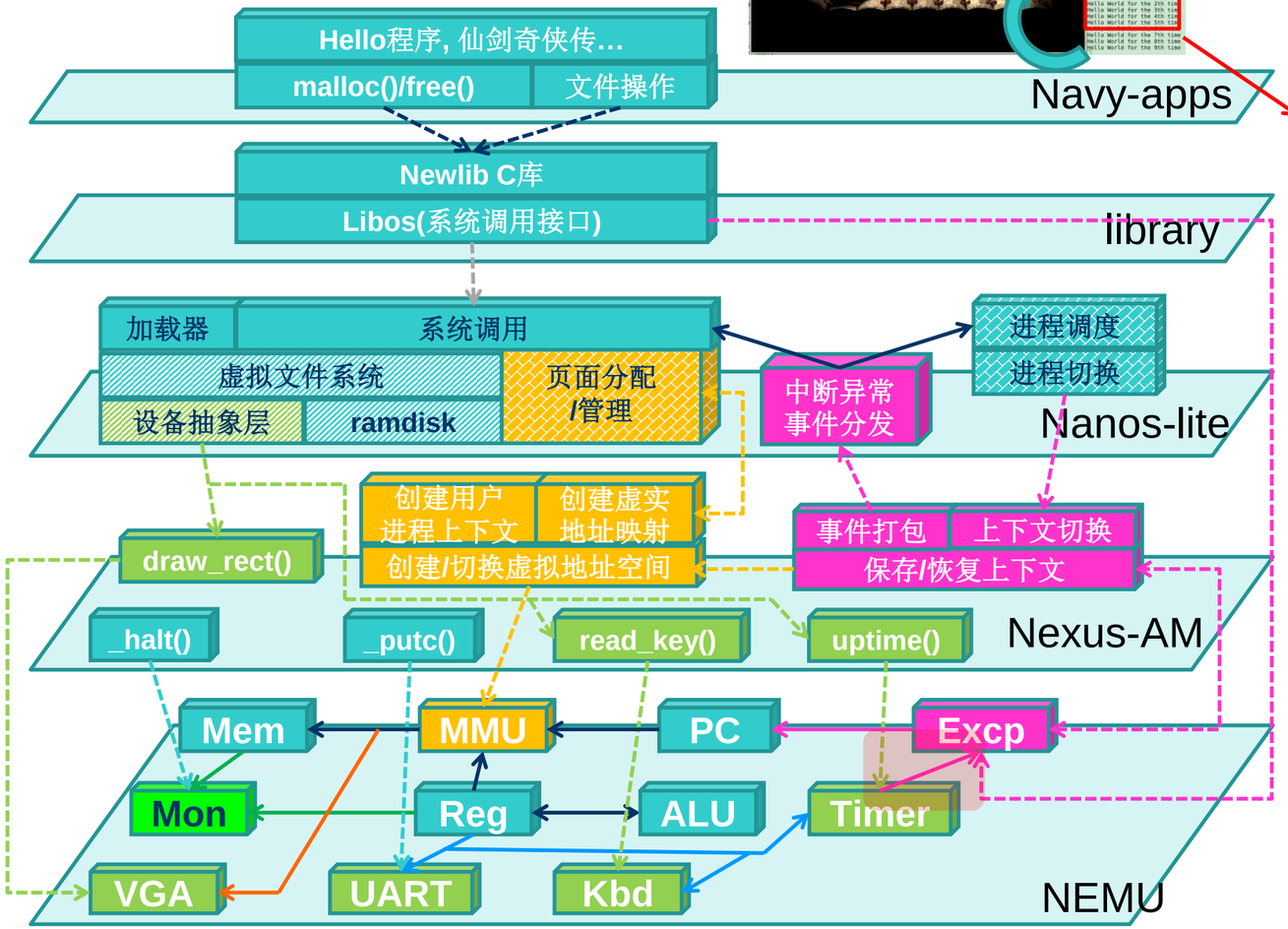
PA4 - 分时多任务

[TRM(指令集) + IOE + CTE + VME]



学生任务:
实现上下文切换
- 多道程序, 并发运行hello程序和仙剑

实现VME -> 支持虚存管理的多道程序
添加中断
-> 抢占式分时多任务




```
Hello World!
Hello World for the 2th time
Hello World for the 3th time
Hello World for the 4th time
Hello World for the 5th time
```

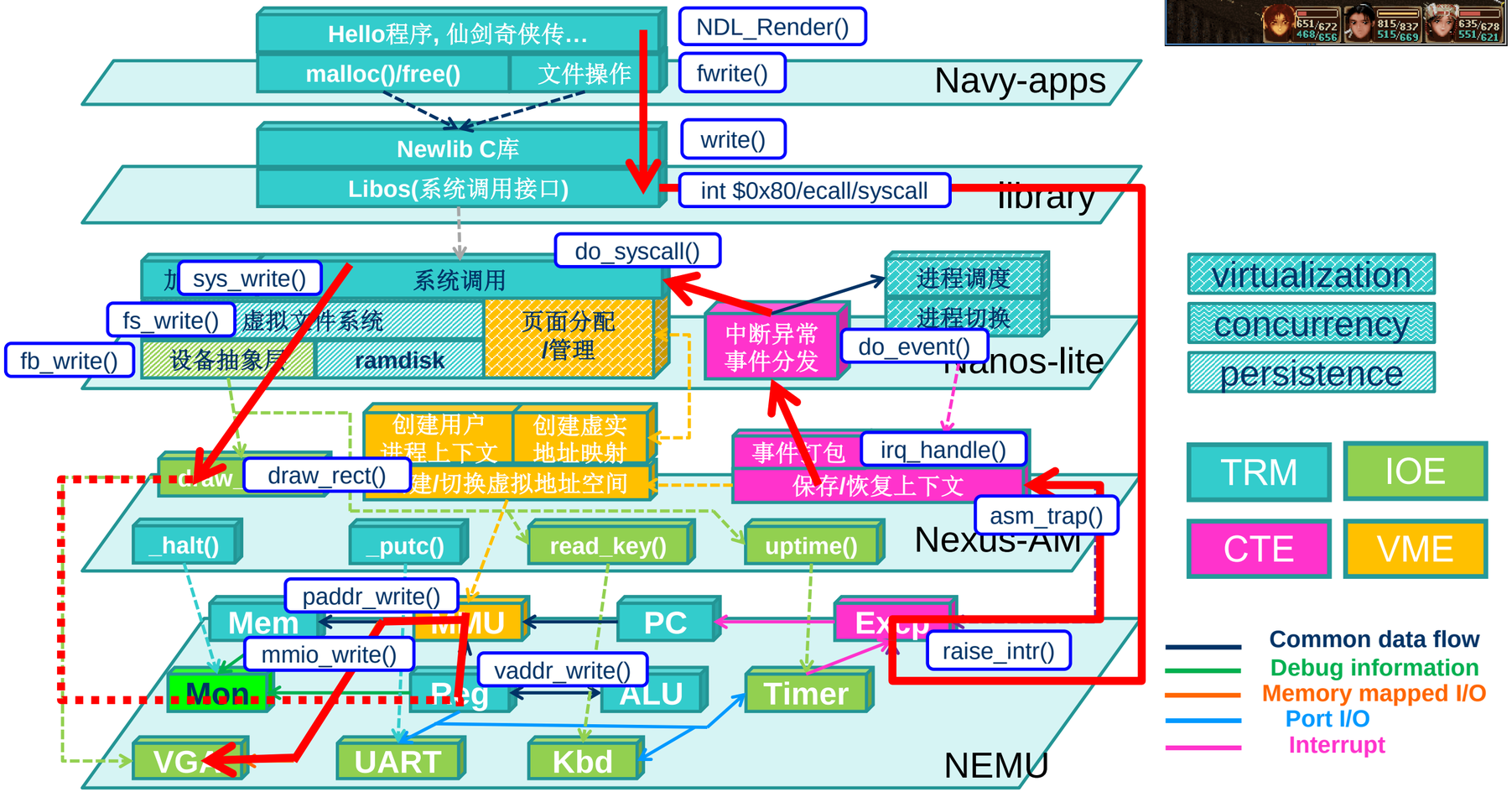
virtualization
concurrency
persistence

TRM IOE
CTE VME

Common data flow
Debug information
Memory mapped I/O
Port I/O
Interrupt

例子 - 仙剑奇侠传更新屏幕

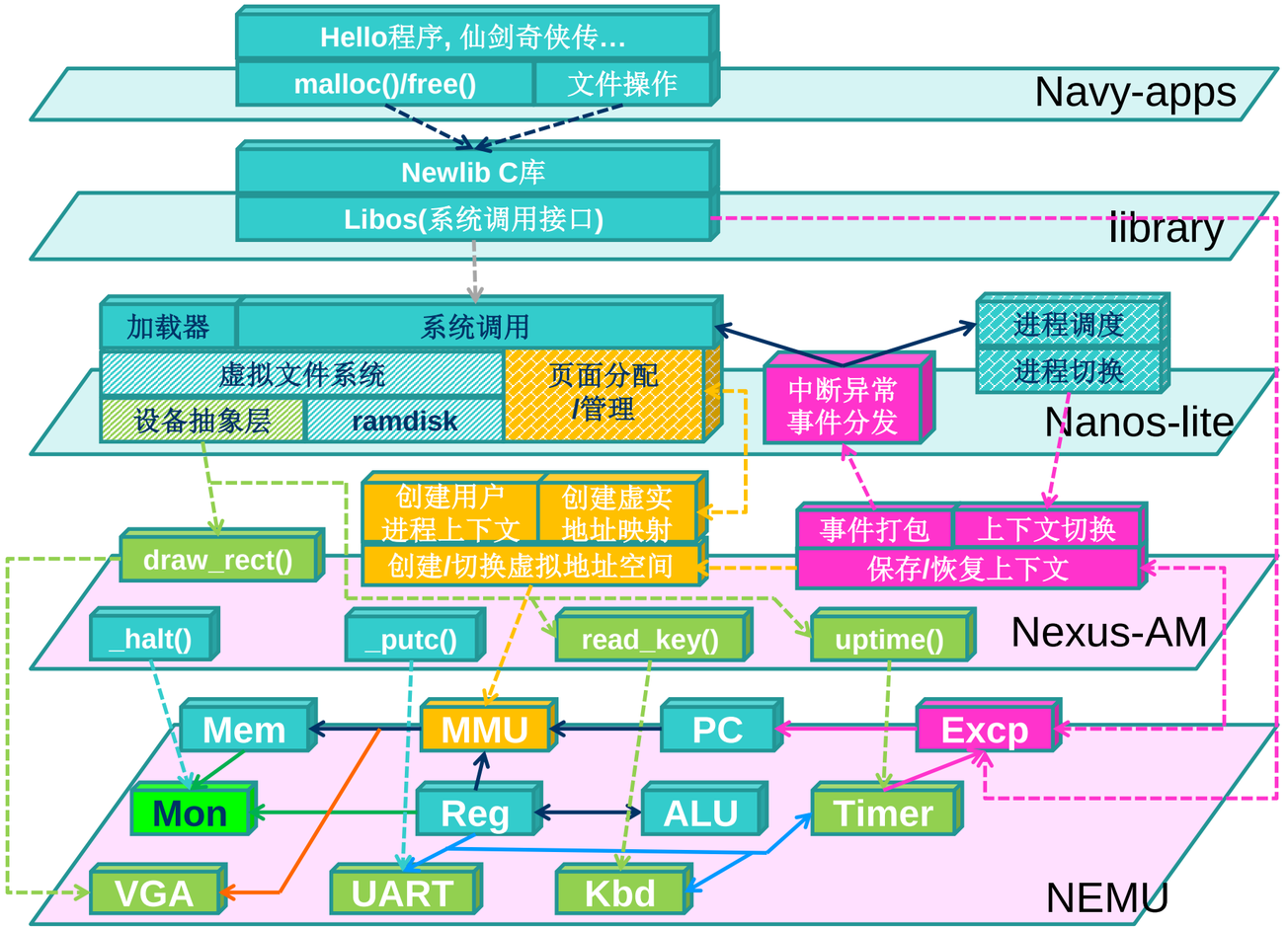
学生亲手构建这个计算机系统
可以白盒理解 **“程序如何运行”**



PA与系统方向其他课程实验的衔接

组成原理实验 = 硬件版的NEMU

- 集中关注微结构的实现细节(PA已经打通全栈)



virtualization
concurrency
persistence

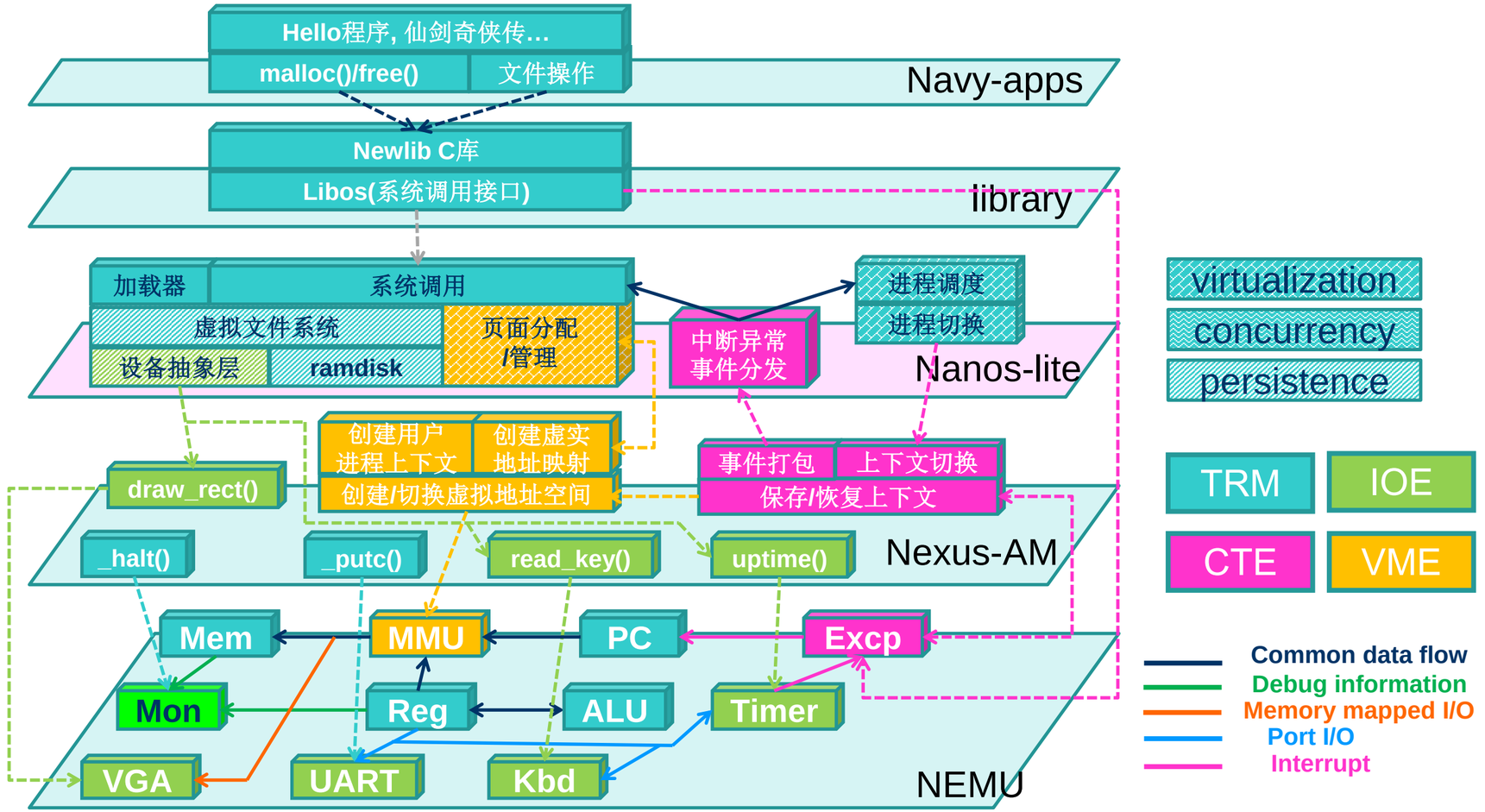
TRM
IOE
CTE
VME

Common data flow
Debug information
Memory mapped I/O
Port I/O
Interrupt

PA与系统方向其他课程实验的衔接(2)

操作系统实验 = 完整版的Nanos-lite

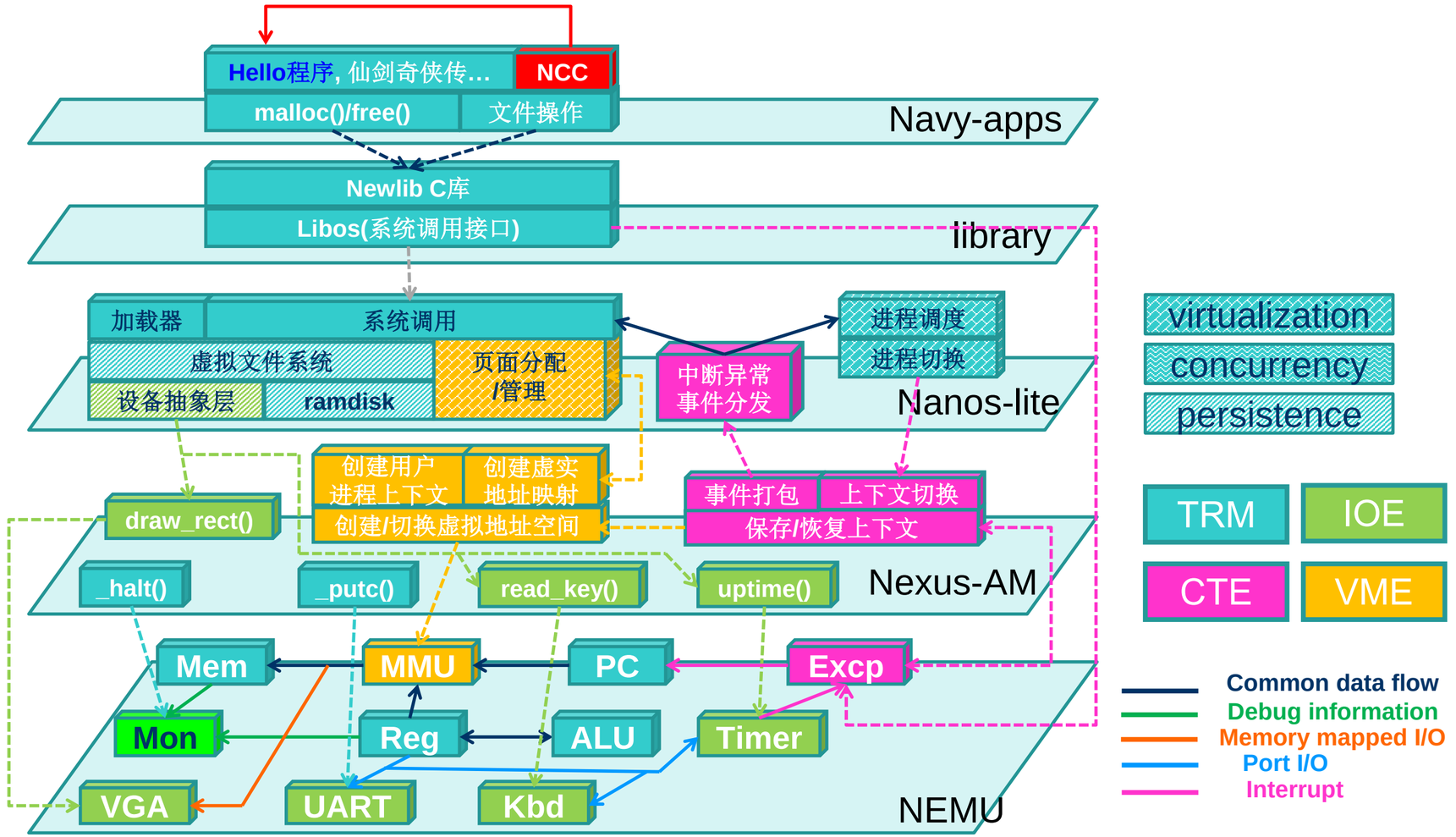
- 集中关注OS的核心概念 - 虚拟化, 并发, 持久化 (PA已经打通全栈)



PA与系统方向其他课程实验的衔接(3)

编译原理实验可归到Navy-apps中(相对独立)

- 可考虑如何生成运行在教学生态系统栈上的可执行文件(简化ABI)



Nexus-AM的意义

- **按照计算机发展史的顺序(更容易理解)揭示程序与计算机的关系**

图灵机(1936) -> 冯诺依曼机(1945) -> GM-NAA I/O(1956) -> CTSS(1961)

运行程序
^
|
提供抽象
^
|
实现功能

	TRM	IOE	CTE	VME
Apps	-	文件操作	-	malloc, free
库	-	Newlib C库		
		Libos(系统调用接口)		
Nanos-lite	系统调用			
	文件系统		进程调度	页面分配/管理
	加载器	设备抽象层	上下文切换	
Nexus-AM	指令, 堆 _putc() _halt()	_uptime() _read_key() _draw_rect()	保存/恢复现场 事件打包	创建/切换虚拟地址空间 创建虚实地址映射 创建用户进程上下文
NEMU	RV32IM	内存映射I/O	中断异常处理 (ecall等)	Sv32分页

NEMU的好处

- 有人说: 不写RTL, 太假了
- 但不写RTL恰恰是NEMU的优势
- RTL的问题: (相对软件)难写难调试
  - 分时多任务仙剑(PA一学期) vs 五级流水线跑???(组原实验一学期)
- 更多的问题: 中断异常和虚存
  - 传统的组原课只讲硬件行为, 不讲软件协同, 也不做, 学生无法深入理解
  - 操作系统课把底层硬件的具体行为当做黑盒, 学生同样难以理解"神秘"的硬件行为

- NEMU是个C模拟器
  - 学生可以一边在NEMU中实现MMU, 一边在Nanos-lite中使用MMU
    - 完全明白OS与MMU交互的每一个细节
  - **软"硬"协同理解计算机系统**
- 和组原课相比, PA更适合当OS的前导课

对riscv32的裁剪

- 实现标准的riscv32费时费力
  - 课时有限(一学期), 学生还要上其它课
- 程序如何在计算机上运行 -> **正确的**程序如何在计算机上运行
  - 假设用户程序不会有非法行为

- 裁剪
  - 指令(TRM) -> RV32IM, 指令模块化方便裁剪
    - x86和mips32无法给出指令的某个子集, 使得编译器仅生成该子集内的指令
  - 输入输出(IOE) -> 简化设备本身的功能
  - 中断异常(CTE) -> 异常只保留系统调用ecall, 中断只保留时钟中断
    - 轮询其它外设
  - 虚存管理(VME) -> 只保留S-mode的Sv32分页, 无需实现权限检查

PA方案耗时与代码量

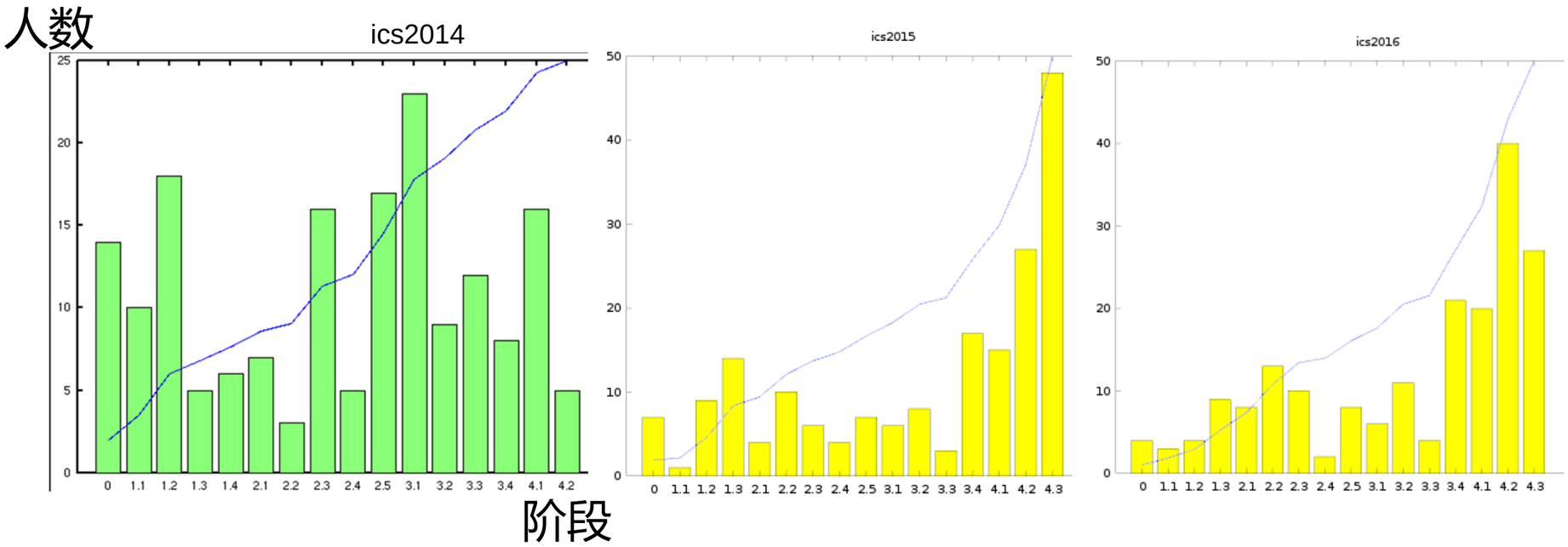
实验内容	持续时间/周	预计耗时/小时	代码量/行
PA0 – 开发环境配置	1	10	无
PA1 – 简易调试器	3	30	400
PA2 – 冯诺依曼计算机系统	4	30	300
课时不足可选择完成到PA2 [小计]	(8)	(70)	(700)
PA3 – 批处理系统	3	30	200
PA4 – 分时多任务	3	30	200
PA5 – 程序性能[选做]	-	-	-
总计	14	130	1100

- 预计耗时以中等水平学生为准
  - 中等水平: 编过规模约500行代码的单个程序, 懂得调试

学生完成情况(2014-2016)

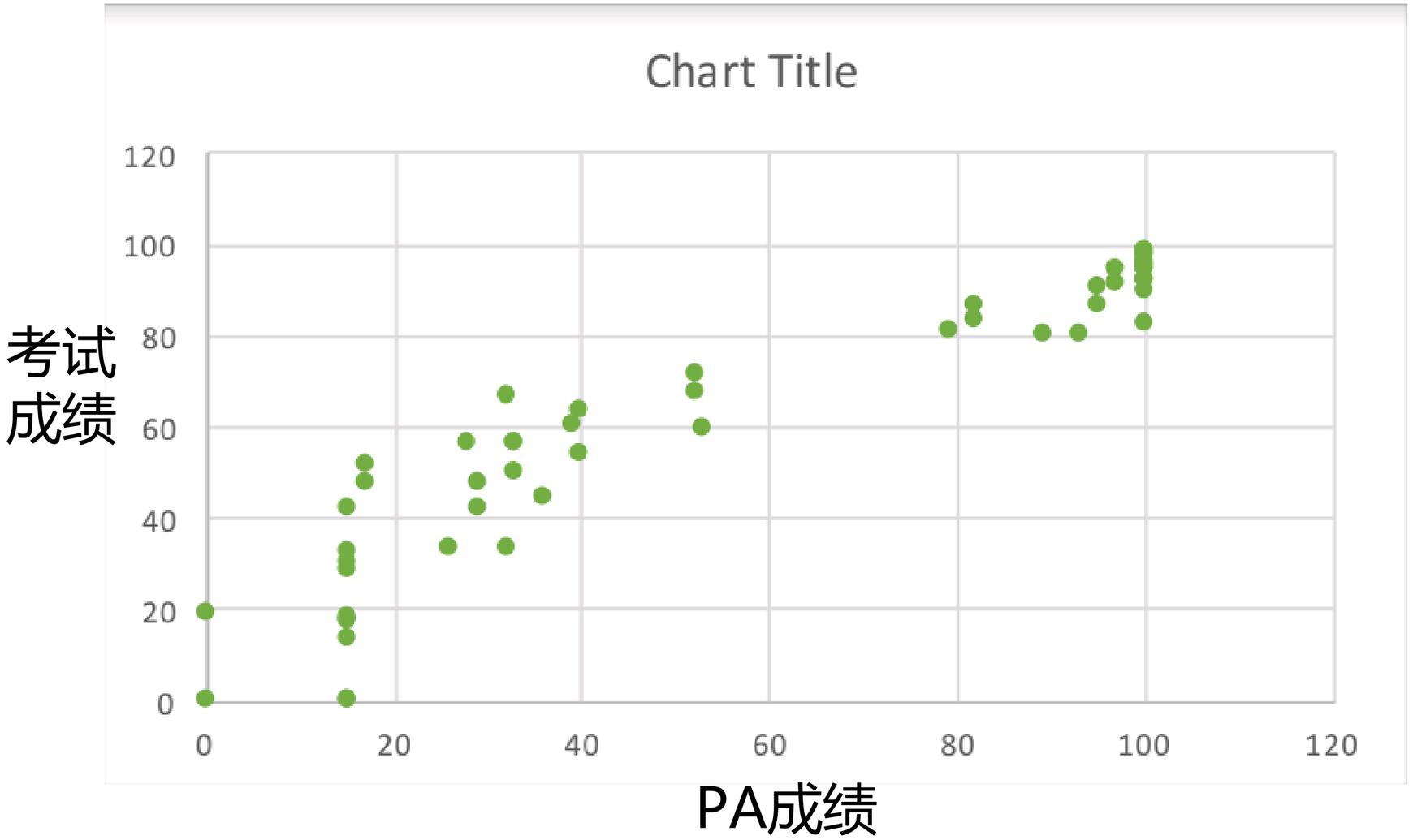
- 第1年缺少有效的指导, 第2, 3年情况好转

学期	人数	平均分/40	完成PA2	完成PA3	开始移植仙剑
2014秋	174	23.07	51.72%	10.92%	2.87%
2015秋	186	29.50	70.43%	57.53%	25.81%
2016秋	190	30.88	72.11%	56.84%	14.21%



学生完成情况(2017年秋)

- 第4, 5年引入Nexus-AM, 按计算机发展史重新组织, 效果更好



学生匿名反馈(2018年秋季)

- 35份有效回答

问题	平均得分
课程开始前对”程序如何在计算机上运行”的理解	2.97 (10-完全理解)
课程结束后对”程序如何在计算机上运行”的理解	6.94 (10-完全理解)
PA对理解”程序如何在计算机上运行”的帮助	7.88 (10-非常有帮助)
做PA前的编码能力	3.68 (10-非常强)
做PA后的编码能力	6.06 (10-非常强)
完成PA后对处理器设计的兴趣	5.00 (10-非常感兴趣)
完成PA后对操作系统, 编译器等系统软件的兴趣	7.12 (10-非常感兴趣)

问题1 – 学生不适应PA的训练

基本原理	做事方案	正确性风险	代表例子
阐述	明确	基本正确	高中物理实验
阐述	明确	可能出错	程序设计作业
阐述	需要思考	基本正确	数学/算法题
阐述	需要思考	可能出错	PA, OSLab
需要探索	需要思考	可能出错	业界和科研的真实问题

- 学生反馈
  - 我不知道应该在哪个文件中写代码, 因为讲义没说
  - 我不想RTFM, 能不能直接告诉我具体要做什么
  - 我懂分页的工作原理, 但我写不出相应的代码
- PA = 最靠近真实问题的训练
  - 独立完成PA -> 具备了解决真实问题的基本素质
    - 开源社区, 科研问题, 业界需求
- **挑战 - 如何更好引导学生踏入探索性/开源项目?**

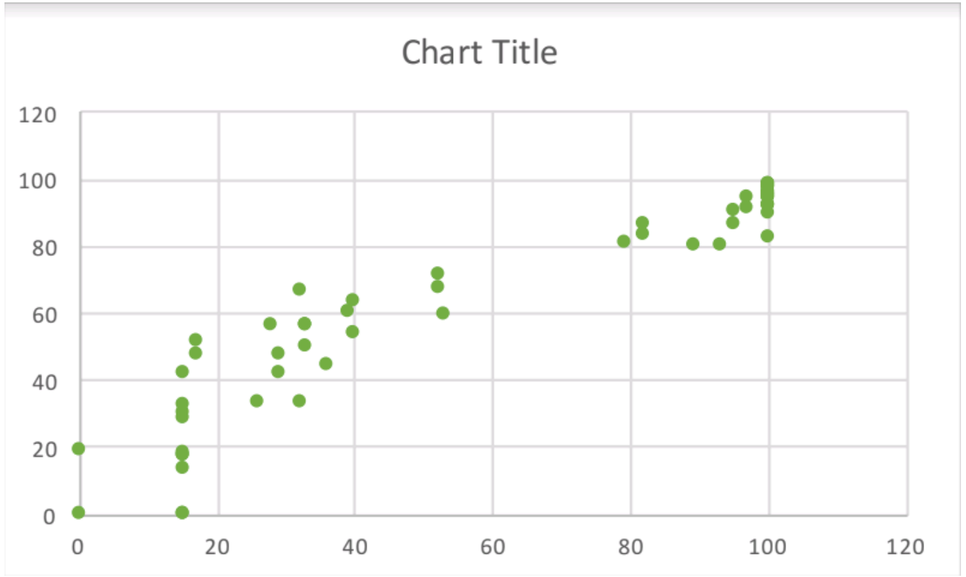
问题2 – 心态

- 不想使用新工具(gdb的bt命令等)
  - 又要RTFM, STFW, 太麻烦了, 我还是直接看自己写的代码吧
- 不去尝试彻底理解代码的每一处细节
  - 这部分代码看不大懂/不是我写的, 随便吧, 反正和成绩没关系
- 遇到bug
  - 我盯着代码看了半个小时都看不出什么问题, 还是找个大腿吧
    - 也反映出程序设计课训练不足, 学生编程基础不扎实

- 这样的学生错过了很多锻炼的机会
  - **调bug是厘清系统每个模块之间关系的好机会**(可惜学生不一定理解)
  - 随着实验的推进, 不同抽象层的交互越来越复杂, 学生的能力越来越跟不上

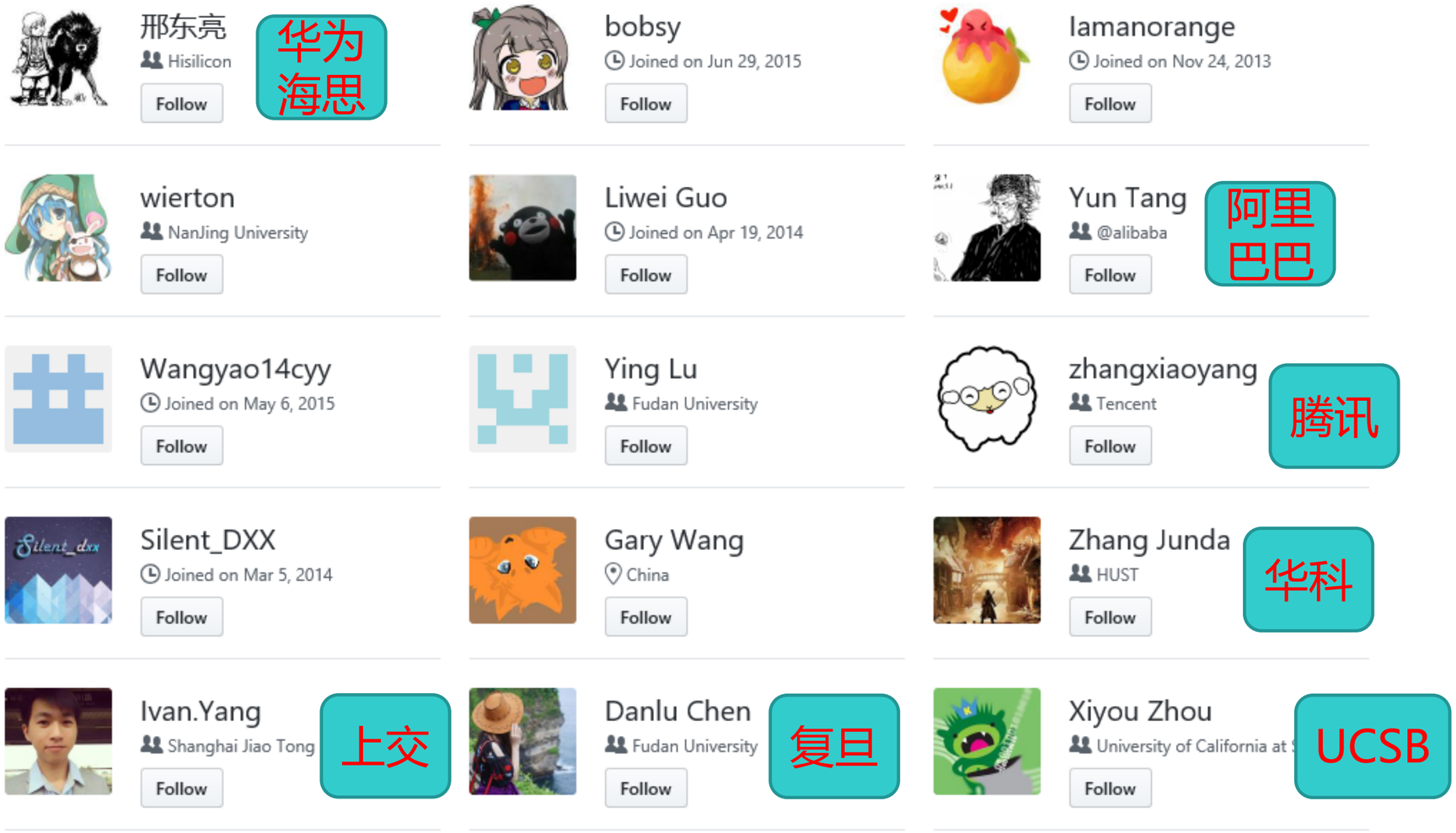
问题3 – 分数两极分化

- 分数两极分化, 但能力总体仍然呈正态分布
  - 大神型(10%) - 真正能吸收PA所传递的价值观
    - 有能力从设计者的角度去思考问题
  - 大众型(40%) - 分数至上, 仅仅是完成任务, 忽视方法的锻炼
    - 无法胜任新的探索性项目
  - 伸手党(30%) - 遇到问题就紧抱大神的大腿
    - 就算跑起了仙剑, 也没有得到该有的技能训练
    - 无法独立重新完成一遍PA
  - 学渣型(20%) - 做到PA1就直接放弃



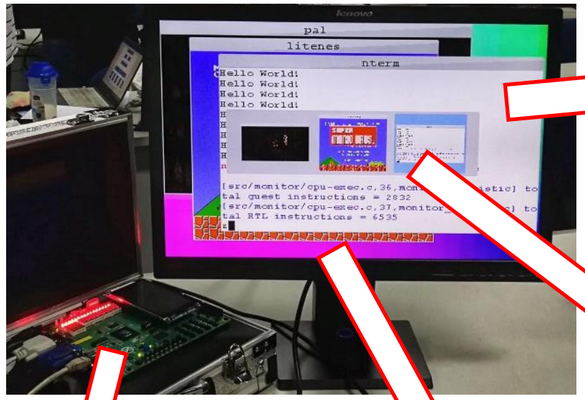
实验方案受到了各方的关注

- PA实验方案已被南航, 复旦, 华科, 天大, 南开等高校相继采用
- 社区关注:



第二届龙芯杯比赛南京大学作品

- 展示完整的教学生态系统ProjectN



x86
模拟器

nterm
终端模拟器

图形应用程序
NES模拟器、仙剑

应用

NWM 窗口管理器

libos, libc, libndl (图形)

运行库

功能测试

基准程序

打字游戏

Nanos
操作系统

操作系统

AM (抽象计算机) = TRM + IOE + [CTE] + [VME]

抽象层

mips32

ISA

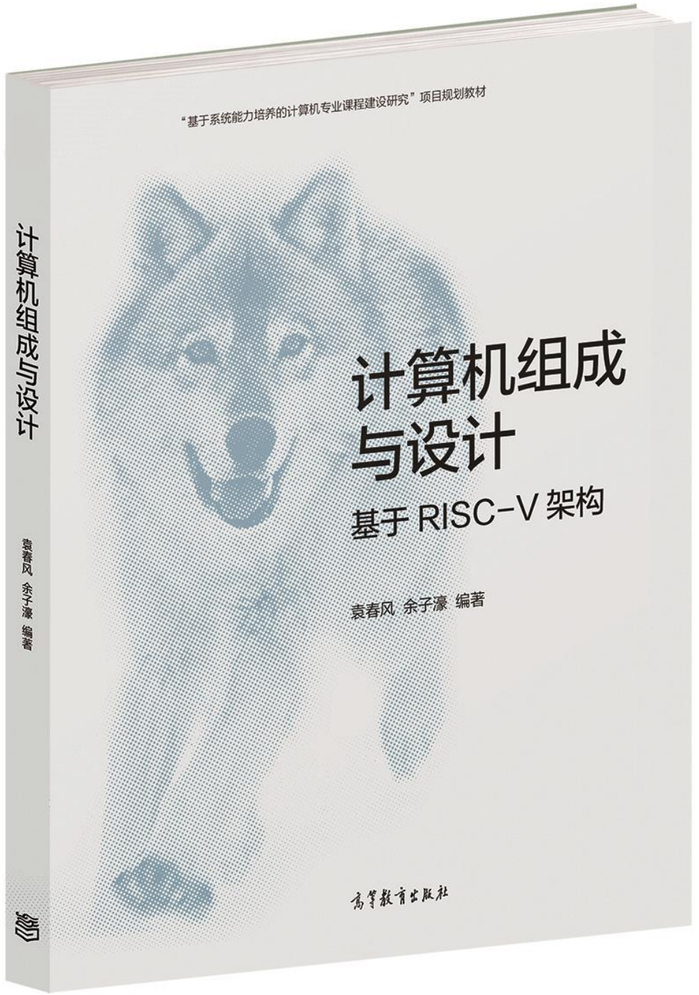
NOOP 乱序处理器

NEMU全系统模拟器

微结构

欢迎挑战

- 今年新特性: 多主线
  - 可选择x86/riscv32/mips32来攻略
  - 二周目可以选择不同的ISA
    - 体会不同的ISA设计对系统的影响
- 实验讲义
  - https://nju-projectn.github.io/ics-pa-gitbook
- 框架代码
  - https://github.com/NJU-ProjectN/ics-pa
- 欢迎挑战PA, 对系统能力进行充电!



基于RISC-V架构的计算机组成与设计教材
袁春风 余子濠 著
(正在出版)