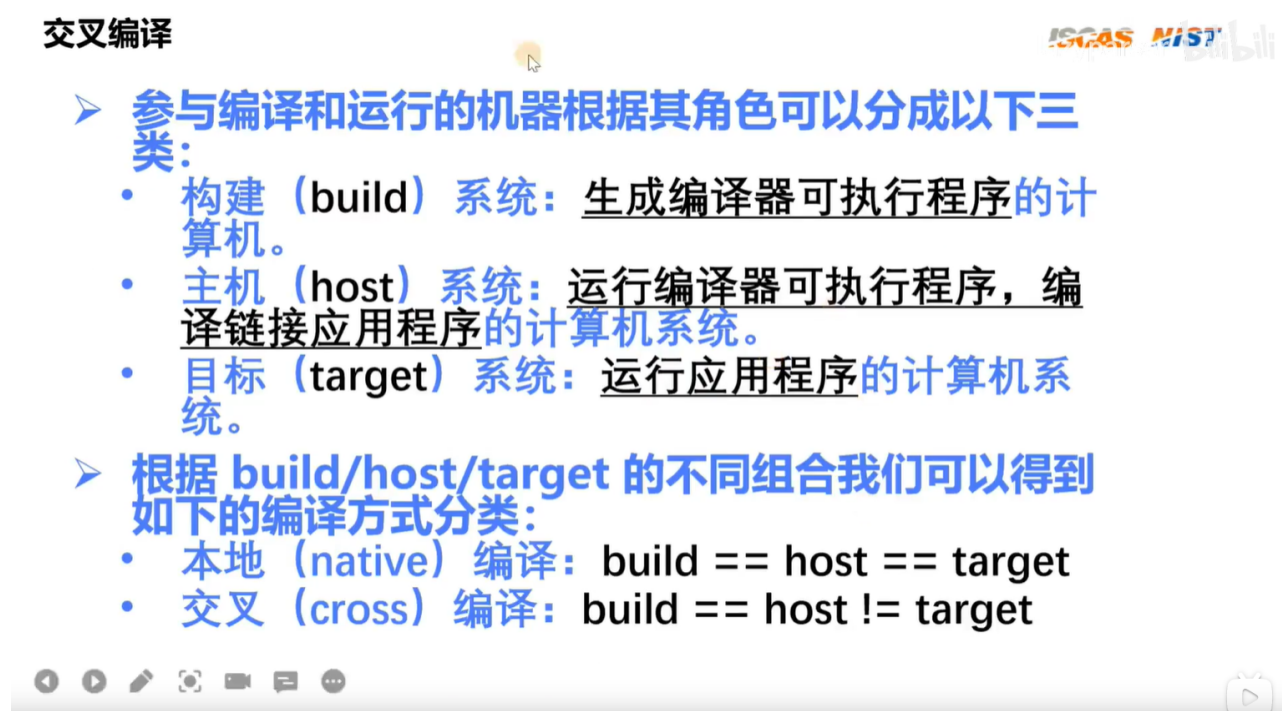


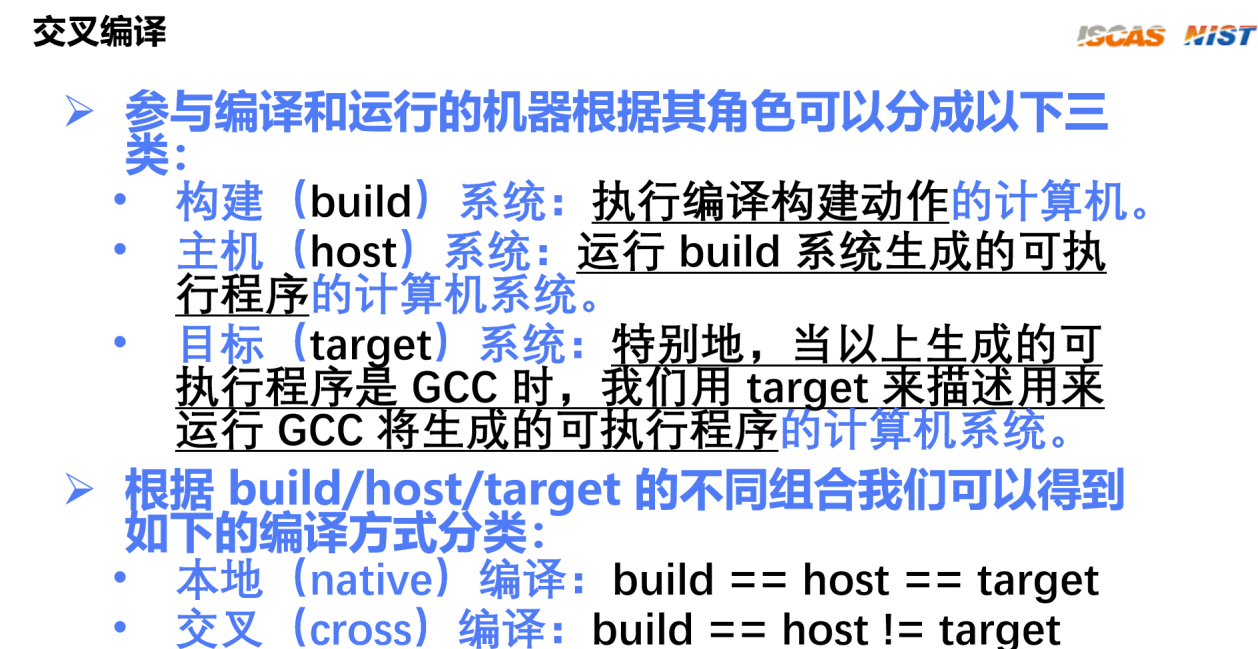
课程视频勘误表：

- **第4章-嵌入式开发介绍** 视频 03:09 左右处的课件描述有错误，详细介绍参考 [issue 描述](#)。

错误的讲稿页视频截图（[视频 P6](#) 播放第 3 分钟左右处）：



改正后正确的讲稿页截图：



- **第5章-part7-RISC-V 汇编语言编程**：[asm/code/cc_nested](#) 那个例子中对 “jal square” 这条语句解释说 “是尾调用”，这个解释不准确，而且在本例子的场景下也不涉及 “尾调用” 的概念，比较恰当的解释应该是 leaf call，这里仅仅想表达 `aa_bb()` 这个函数调用了 `square()` 函数，而 `square()` 函数内部不会再调用其他的函数了。
- **第6章-RVOS 介绍**：“课程项目简介” 那一页的 git 仓库路径有误

错误的讲稿页视频截图（[视频 P15](#) 播放第 11 分钟左右处）：

课程项目简介

ISCAS NIST

RVOS

RVOS (<https://www.rt-thread.org/>) 是一个用于教学演示的操作系统内核。诞生于 2021 年。采用 BSD 2-Clause 许可证发布。

- 设计小巧，整个核心有效代码不超过 1000 行；
- 可读性强，易维护，绝大部分代码为 C 语言，很少部分采用汇编；
- 演示了简单的内存分配管理实现；
- 演示了可抢占多线程调度实现，线程调度采用轮转调度法；
- 演示了简单的任务互斥实现；
- 演示了软件定时器实现；
- 演示了系统调用实现（M 和 U 模式）；
- 支持 RV32；
- 支持 QEMU-virt 平台。

改正后正确的讲稿页截图：

课程项目简介

ISCAS NIST

RVOS

RVOS (<https://github.com/plctlab/riscv-operating-system-mooc>) 是一个用于教学演示的操作系统内核。诞生于 2021 年。采用 BSD 2-Clause 许可证发布。

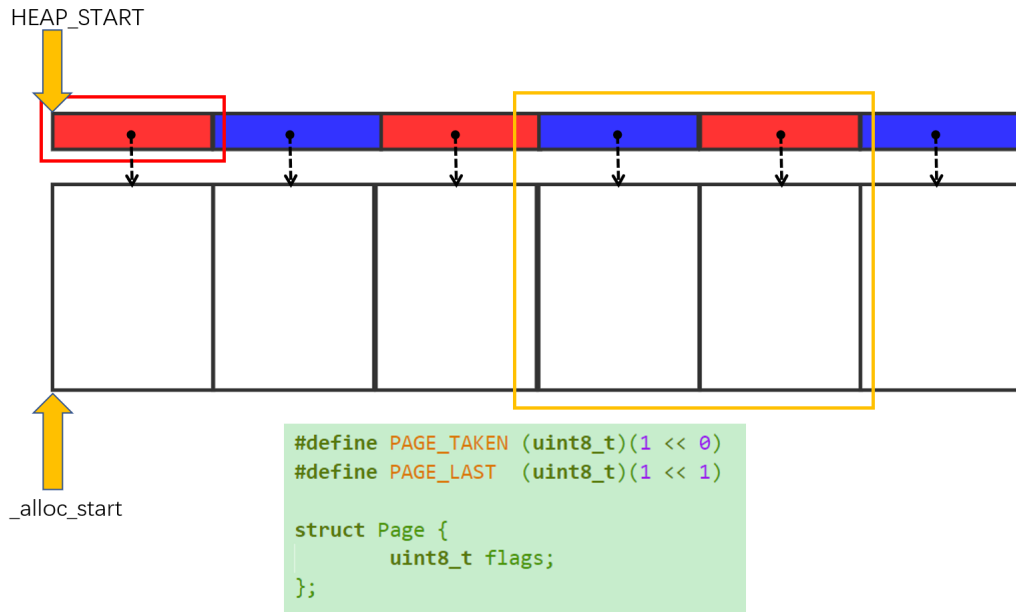
- 设计小巧，整个核心有效代码 ~ 1000 行；
- 可读性强，易维护，绝大部分代码为 C 语言，很少部分采用汇编；
- 演示了简单的内存分配管理实现；
- 演示了可抢占多线程调度实现，线程调度采用轮转调度法；
- 演示了简单的任务互斥实现；
- 演示了软件定时器实现；
- 演示了系统调用实现（M + U 模式）；
- 支持 RV32；
- 支持 QEMU-virt 平台。

- **第7章(上)-Hello RVOS:** 视频 37:39 左右处的代码有错误，详细介绍参考：
 - [issue1 描述](#)。相关代码已经在 v0.8 版本中改正。
 - [issue2 描述](#)。相关代码已经在 v0.9 版本中改正。
- **第 8 章 内存管理:** "Page 描述符数据结构设计" 那一页的图片有误。

错误的讲稿页视频截图（视频 P18 播放第 49 分钟左右处）：

Page 描述符数据结构设计

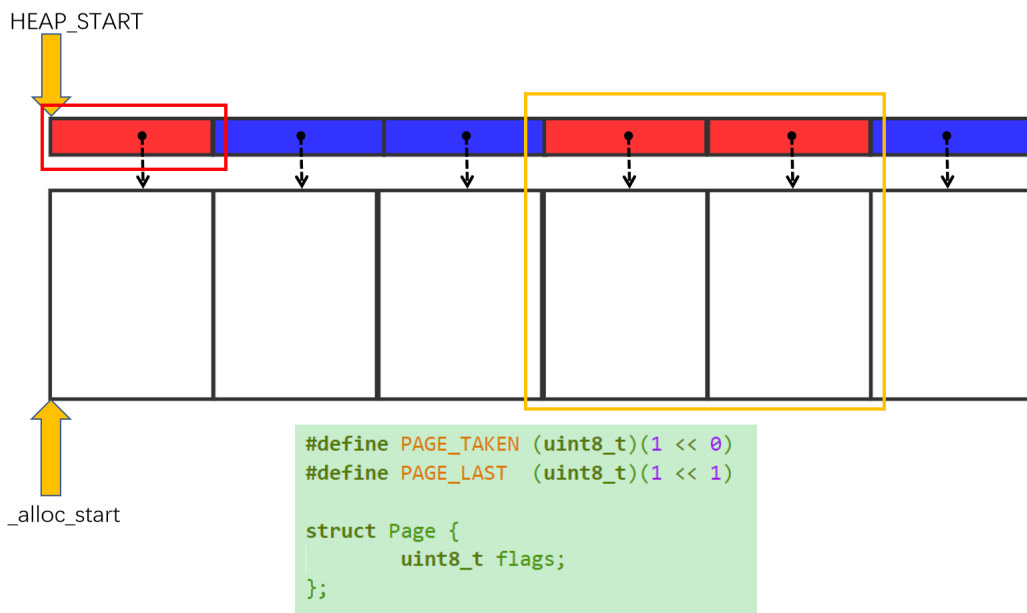
ISCAS NIST



改正后正确的讲稿页截图：

Page 描述符数据结构设计

ISCAS NIST



- 第 11 章 外部设备中断 “PLIC 编程接口 - 寄存器” 介绍 Pending 寄存器那一页的内存映射地址的公式有错误，详细描述参考 [issue 描述](#)。

错误的讲稿页视频截图（[视频 P21](#) 播放第 20 分钟左右处）：

PLIC 编程接口 - 寄存器



可编程寄存器	功能描述	内存映射地址
Pending	用于指示某一路中断源是否发生。	$\text{BASE} + 0x1000 + ((\text{interrupt-id}) / 32)$

- 每个 PLIC 包含 2 个 32 位的 Pending 寄存器，每一个 bit 对应一个中断源，如果为 1 表示该中断源上发生了中断（进入 Pending 状态），有待 hart 处理，否则表示该中断源上当前无中断发生。
- Pending 寄存器中断的 Pending 状态可以通过 claim 方式清除。
- 第一个 Pending 寄存器的第 0 位对应不存在的 0 号中断源，其值永远为 0。



改正后正确的讲稿页截图：

PLIC 编程接口 - 寄存器



可编程寄存器	功能描述	内存映射地址
Pending	用于指示某一路中断源是否发生。	$\text{BASE} + 0x1000 + ((\text{interrupt-id}) / 32) * 4$

- 每个 PLIC 包含 2 个 32 位的 Pending 寄存器，每一个 bit 对应一个中断源，如果为 1 表示该中断源上发生了中断（进入 Pending 状态），有待 hart 处理，否则表示该中断源上当前无中断发生。
- Pending 寄存器中断的 Pending 状态可以通过 claim 方式清除。
- 第一个 Pending 寄存器的第 0 位对应不存在的 0 号中断源，其值永远为 0。

相关代码涉及 `PLIC_PENDING` 这个宏的定义，已经在 v0.9 版本中改正。