第4季

搭建RISC-V实验环境

主讲： 笨叔

本节课主要内容

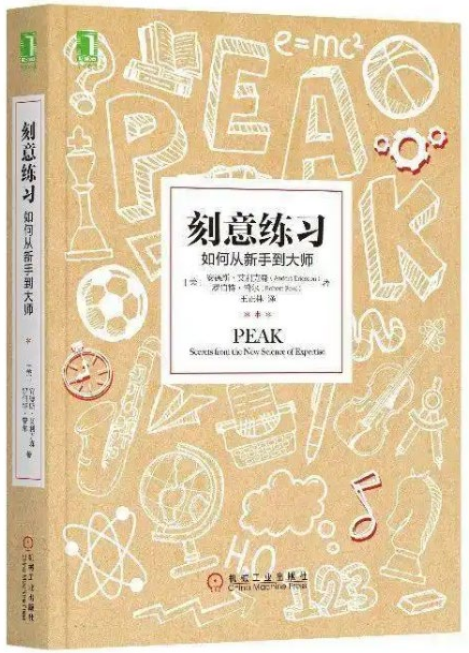
- 本章主要内容
  - 实验环境搭建
  - 调试BenOS
  - BenOS基础代码讲解

扫码订阅RISC-V视频课程

本节课主要讲解书上第2章内容

刻意练习

- 要熟练掌握RISC-V指令集和体系结构，唯一的办法 只能多加练习，其他“n小时精通”的方法都是骗人的



RISC-V开发板

- 目前市面上能买到的RISC-V开发板
  - SiFive HiFive Unmatched
  - 赛昉星光VisionFive
  - D1-H开发板哪吒
  - 东山哪吒
  - K210开发板

- 本季课采用免费的QEMU作为实验平台

QEMU模拟器

- QEMU Virt实验平台模拟的是一款通用的RISC-V开发板
  - ✓ 最多支持8个RV32GC/RV64GC通用处理器核心
  - ✓ 支持CLINT本地中断控制器
  - ✓ 支持PLIC中断控制器
  - ✓ 支持NOR Flash
  - ✓ 支持兼容NS16550的串口
  - ✓ 支持RTC
  - ✓ 支持8个VirtIO-MMIO传输设备
  - ✓ 支持1个PCIe主机桥接设备
  - ✓ 支持fw_cfg，用于从QEMU获取固件配置信息

QEMU基本能满足我们学习RSIC-V体系结构的要求，而且免费，调试方便

NEMU模拟器

- NEMU（NJU Emulator）最早是由南京大学实现的一个用于教学的计算机指令集架构（Instruction Set Architecture, ISA）模拟器
- 香山处理器在开发过程中用于difftest的对比机
- RV64版本NEMU支持如下硬件特性：
  - ✓ 支持1个RV64GC通用处理器核心
  - ✓ 支持CLINT本地中断控制器
  - ✓ 支持NOR Flash
  - ✓ 支持兼容NS16550的串口
  - ✓ 支持Xilinx UartLite串口控制器
  - ✓ 支持RTC
  - ✓ 支持单步调试

从最基本的BenOS开始

- 本季实验从最简单的BenOS开始，逐步添加代码和完善代码，最终形成一个简单的、多任务的小OS，从而达到学习和熟悉RSIC-V体系结构的目的

- BenOS包括：
  - ✓ 最简洁的SBI启动代码 - mysbi
  - ✓ 最简洁的启动汇编
  - ✓ 仅仅打印“Hello RISC-V”

实验环境

建议使用我们提供的vmware镜像。下载方法：
关注“奔跑吧linux社区”微信公众号，输入“vmware”下载。

- 主机：Ubuntu 20.04
- QEMU： qemu-system-riscv64 4.2.1
- GCC： riscv64-linux-gnu-gcc 9.3
- gdb-multiarch：9.2

运行BenOS

本节课实验代码，参考《RISC-V体系结构编程与实践》第2章配套代码

https://github.com/runninglinuxkernel/riscv_programming_practice

```
$ cd benos
$ make
$ make run
```

```
rlk@master:benos$ make run
qemu-system-riscv64 -nographic -machine virt -m 128M  -bios mysbi.bin  -device loader,
file=benos.bin,addr=0x80200000  -kernel benos.elf
Welcome RISC-V!
```

单步调试BenOS

1. 在终端中输入：
$ cd benos
$ make
$ make debug

2. 在另一个终端输入如下命令来启动GDB。
$ gdb-multiarch --tui benos.elf

3. 在GDB命令行中输入如下命令。
(gdb) target remote localhost:1234
(gdb) b _start
Breakpoint 1 at 0x80200000: file src/boot.S, line 6.
(gdb) c

```
src/boot.S
   1            .section ".text.boot"
   2
   3            .globl _start
   4            _start:
   5                    /* ^%^  )^  $^  &^  */
B+>6                    csrw sie, zero
   7
   8                    /* ^(^ [redacted]^!^  &^   ^&^  '^  %^  %^  $^  KB */
   9                    la sp, stacks_start
   10                   li t0, 4096
   11                   add sp, sp, t0
   12
   13                   /* ^(^  (^  %^  ^(^  (^ [redacted] */
   14                   tail kernel_main
   15
   16           .section .data
   17           .align  12
   18           .global stacks_start
   19           stacks_start:
   20                   .skip 4096
remote Thread 1.1 In:  _start
Reading symbols from benos.elf...
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
0x0000000000001000 in ?? ()
(gdb) b _start
Breakpoint 1 at 0x80200000: file src/boot.S, line 6.
(gdb) c
Continuing.

Breakpoint 1, _start () at src/boot.S:6
(gdb)
```

layout regs 打开寄存器窗口，动态查看寄存器

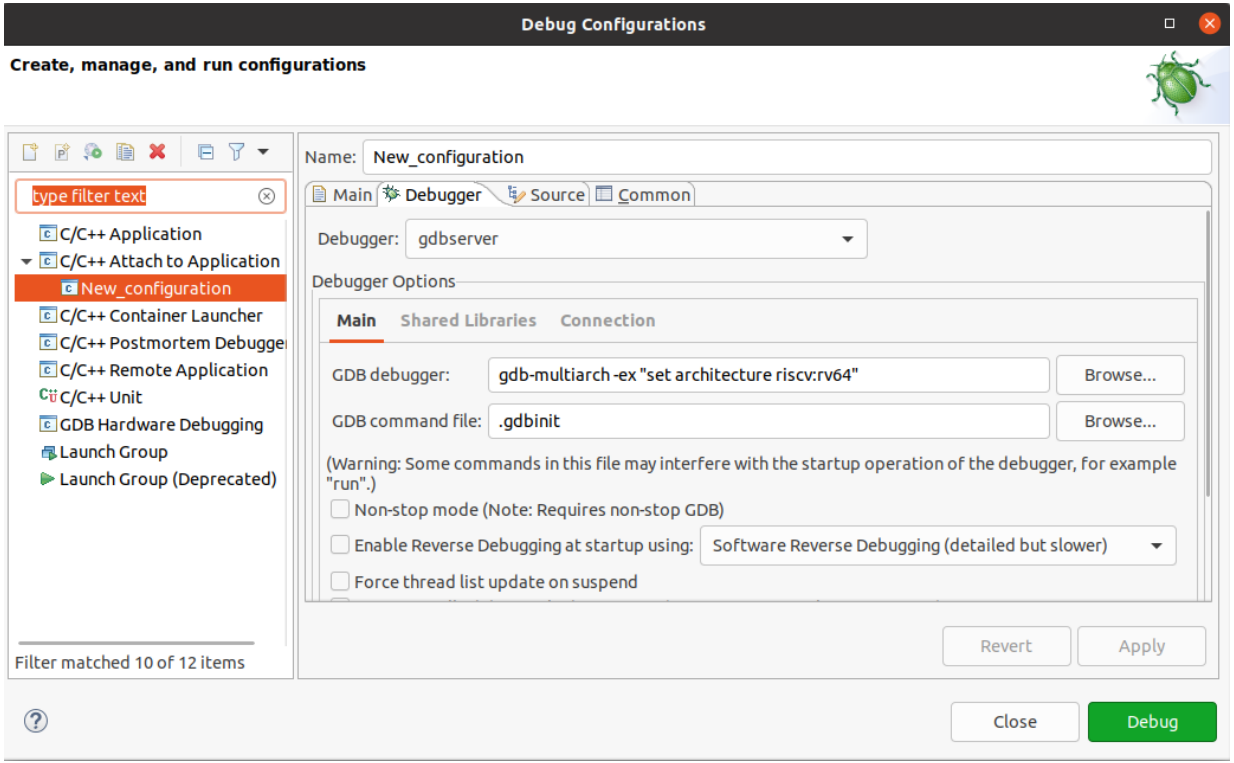
```
─Register group: general──────────────────────────────────────────────────────
zero           0x0      0                                    ra             0x0      0x0
sp             0x80003fc0       0x80003fc0                   gp             0x0      0x0
tp             0x0      0x0                                  t0             0x1000   4096
t1             0x0      0                                    t2             0x0      0
fp             0x0      0x0                                  s1             0x0      0
a0             0x0      0                                    a1             0x1020   4128
a2             0x0      0                                    a3             0x0      0
a4             0x0      0                                    a5             0x0      0
a6             0x0      0                                    a7             0x0      0
s2             0x0      0                                    s3             0x0      0
```

```
B+>6                    csrw sie, zero
   7
   8                    /* ^(^  ^'^  &^   ^&^  '^  %^  %^  $^  KB */
   9                    la sp, stacks_start
   10                   li t0, 4096
   11                   add sp, sp, t0
   12
   13                   /* ^(^  (^  %^  ^(^  (^  */
   14                   tail kernel_main
   15
```

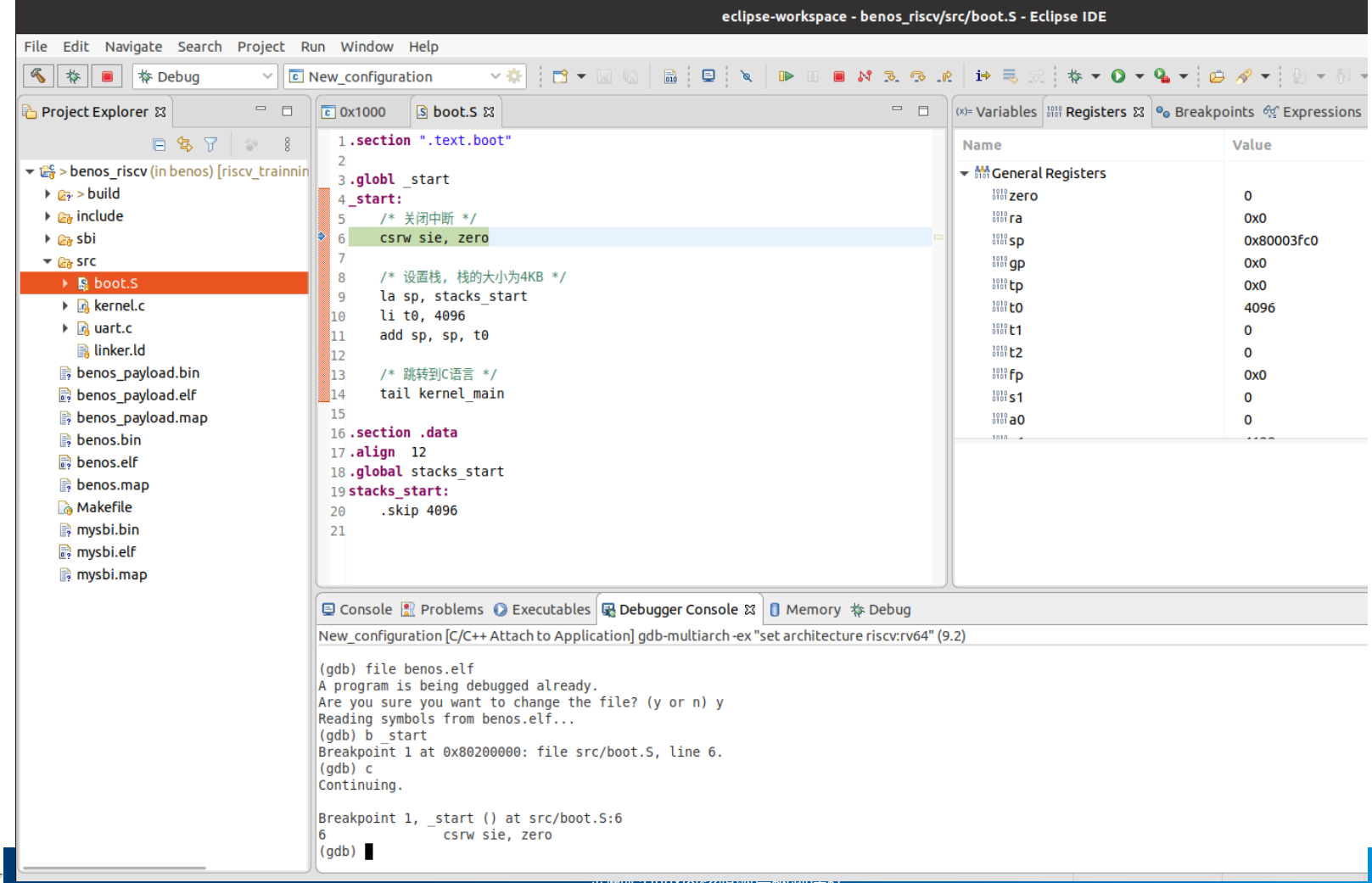
```
remote Thread 1.1 In: _start
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
0x0000000000001000 in ?? ()
(gdb) b _start
Breakpoint 1 at 0x80200000: file src/boot.S, line 6.
(gdb) c
Continuing.

Breakpoint 1, _start () at src/boot.S:6
(gdb) layout regs
(gdb)
```

使用qemu+eclipse调试

- 用gdb不爽？可以使用eclipse来调试



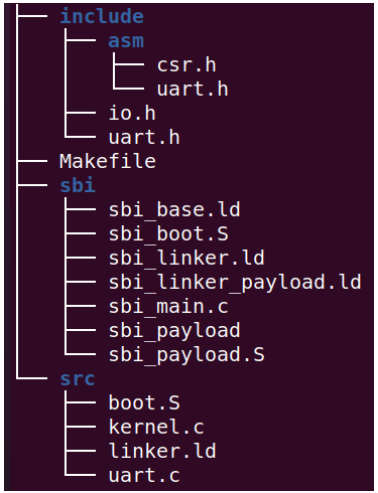
GDB debugger: gdb-multiarch -ex "set architecture riscv:rv64"


eclipse-workspace - benos_riscv/src/boot.S - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Debug
New_configuration
Project Explorer
benos_riscv (in benos) [riscv_trainnin
build
include
sbi
src
boot.S
kernel.c
uart.c
linker.ld
benos_payload.bin
benos_payload.elf
benos_payload.map
benos.bin
benos.elf
benos.map
Makefile
mysbi.bin
mysbi.elf
mysbi.map
0x1000
boot.S
1 .section ".text.boot"
2
3 .globl _start
4 _start:
5 /* 关闭中断 */
6 csrw sie, zero
7
8 /* 设置栈，栈的大小为4KB */
9 la sp, stacks_start
10 li t0, 4096
11 add sp, sp, t0
12
13 /* 跳转到C语言 */
14 tail kernel_main
15
16 .section .data
17 .align 12
18 .global stacks_start
19 stacks_start:
20 .skip 4096
21
Variables
Registers
Breakpoints
Expressions
Name
Value
General Registers
zero 0
ra 0x0
sp 0x80003fc0
gp 0x0
tp 0x0
t0 4096
t1 0
t2 0
fp 0x0
s1 0
a0 0
Console
Problems
Executables
Debugger Console
Memory
Debug
New_configuration [C/C++ Attach to Application] gdb-multiarch -ex "set architecture riscv:rv64" (9.2)
(gdb) file benos.elf
A program is being debugged already.
Are you sure you want to change the file? (y or n) y
Reading symbols from benos.elf...
(gdb) b _start
Breakpoint 1 at 0x80200000: file src/boot.S, line 6.
(gdb) c
Continuing.
Breakpoint 1, _start () at src/boot.S:6
6 csrw sie, zero
(gdb)

BenOS代码结构

BenOS基础实验代码包含MySBI和BenOS两部分代码

- ✓ MySBI是运行在M模式下的固件，为运行在S模式下的操作系统提供引导和统一的接口服务。
- ✓ BenOS为S模式的OS



sbi目录：包含MySBI的源文件。
src目录：包含BenOS的源文件。
include目录：包含BenOS和MySBI共用的头文件。

MySBI基础代码分析

- 没有采用业界流行的OpenSBI固件，而是从零开始编写一个小型可用的SBI固件，以便我们从底层深入学习RISC-V架构。
- MySBI运行在M模式，本质上是一个裸机程序

<benos/sbi/sbi_linker.ld>

```
1    OUTPUT_ARCH(riscv)
2    ENTRY(_start)
3
4    SECTIONS
5    {
6        INCLUDE "sbi/sbi_base.ld"
7    }
```

<benos/sbi/sbi_base.ld>

```
1     /*
2      * 设置sbi的加载入口地址为0x80000000
3      */
4
5     . = 0x80000000,
6
7     .text.boot : { *(.text.boot) }
8     .text : { *(.text) }
9     .rodata : { *(.rodata) }
10    .data : { *(.data) }
11    . = ALIGN(0x8);
12    bss_begin = .;
13    .bss : { *(.bss*) }
14    bss_end = .;
```

<benos/sbi/sbi_main.c>

```
1    #include "asm/csr.h"
2
3    #define FW_JUMP_ADDR 0x80200000
4
5    /*
6     * 运行在M模式，并且切换到S模式
7     */
8    void sbi_main(void)
9    {
10       unsigned long val;
11
12       /* 设置跳转模式为S模式 */
13       val = read_csr(mstatus);
14       val = INSERT_FIELD(val, MSTATUS_MPP, PRV_S);
15       val = INSERT_FIELD(val, MSTATUS_MPIE, 0);
16       write_csr(mstatus, val);
17
18       /* 设置M模式的Exception Program Counter，用于mret跳转 */
19       write_csr(mepc, FW_JUMP_ADDR);
20       /* 设置S模式异常向量表入口*/
21       write_csr(stvec, FW_JUMP_ADDR);
22       /* 关闭S模式的中断*/
23       write_csr(sie, 0);
24       /* 关闭S模式的页表转换 */
25       write_csr(satp, 0);
26
27       /* 切换到S模式 */
28       asm volatile("mret");
29   }
```

从M模式跳转到S模式

BenOS基础代码分析

- BenOS基础代码进实现了串口打印功能，后续我们在实验中不断添加功能

<benos/src/linker.ld>

```
1    SECTIONS
2    {
3        . = 0x80200000,
4
5        .text.boot : { *(.text.boot) }
6        .text : { *(.text) }
7        .rodata : { *(.rodata) }
8        .data : { *(.data) }
9        . = ALIGN(0x8);
10       bss_begin = .;
11       .bss : { *(.bss*) }
12       bss_end = .;
13   }
```

<benos/src/boot.S>

```
1    .section ".text.boot"
2
3    .globl _start
4    _start:
5        /* 关闭中断 */
6        csrw sie, zero
7
8        /* 设置栈 */
9        la sp, stacks_start
10       li t0, 4096
11       add sp, sp, t0
12
13       call    kernel_main
14
15   hang:
16       wfi
17       j hang
18
19   .section .data
20   .align  12
21   .global stacks_start
22   stacks_start:
23           .skip 4096
```

<benos/src/kernel.c>

```
1    #include "uart.h"
2
3    void kernel_main(void)
4    {
5        uart_init();
6        uart_send_string("Welcome RISC-V!\r\n");
7
8        while (1) {
9            ;
10       }
11   }
```

表 2.5 16550 串口控制器寄存器

DLAB 字段	寄存器偏移地址	寄存器说明
0	0b000	数据寄存器，接收数据或者发送数据
0	0b001	中断使能寄存器
1	0b000	预分频寄存器中的低8位
1	0b0001	预分频寄存器中的高8位
-	0b010	中断标识寄存器或者FIFO控制寄存器
-	0b011	线路控制寄存器
-	0b100	MODEN控制寄存器
-	0b101	线路状态寄存器
-	0b110	MODEN状态寄存器

文字不如声音，声音不如视频

扫描订阅RISC-V视频课程

第4季 奔跑吧Linux社区 视频课程

RISC-V体系结构编程与实践

主讲：笨叔

课程名称	进度	时长（分钟）
第1课：课程介绍（免费）	完成	20
第2课：RISC-V体系结构介绍（免费）	完成	47
第3课：RISC-V处理器微架构（免费）	完成	48
第4课：搭建RISC-V开发环境（免费）	完成	30
第5课：RISC-V指令集（免费）	完成	128
第6课：RISC-V函数调用规范	完成	40
第7课：RISC-V GNU AS汇编器	完成	42
第8课：RISC-V GNU 链接脚本	完成	90
第9课：RISC-V GNU 内嵌汇编	完成	52
第10课：RISC-V异常处理	完成	80
第11课：RISC-V中断处理	完成	52
第12课：RISC-V内存管理	完成	116
第13课：内存管理实验讲解	完成	36
第14课：cache基础知识	完成	78
第15课：缓存一致性	完成	96
第16课：RISC-V TLB管理	完成	54
第17课：RISC-V原子操作	未录制	
第18课：RISC-V内存屏障	未录制	
第19课：BenOS操作系统相关知识	未录制	
第20课：RVV可伸缩矢量计算	未录制	
第21课：RISC-V压缩指令	未录制	
第22课：RISC-V虚拟化	未录制	
		总计17小时

更多精彩内容马上献上....
微信公众号：奔跑吧Linux社区

视频课程持续更新中...