

练习一：理解内核启动中的程序入口操作

entry.S文件是引导加载程序运行时跳转到的文件，里面包含内核栈空间的分配和内核初始化。

1. la sp, bootstacktop 指令是load address的伪指令，sp栈指针寄存器指向bootstack的顶部也就是内核栈的高地址起始位置。编译器会把它编译成：

```
auipc sp, %pcrel_hi(bootstacktop)
addi sp, sp, %pcrel_lo(bootstacktop)
```

目的：为内核准备安全的栈空间，存储函数调用的返回地址，局部变量等。

2. tail kern_init指令是跳转到c程序kern_init，执行内核初始化流程。**目的：**启动内核初始化，建立页表；初始化内存管理；设置中断；启动调度；进入主循环。

练习二：使用GDB验证启动流程

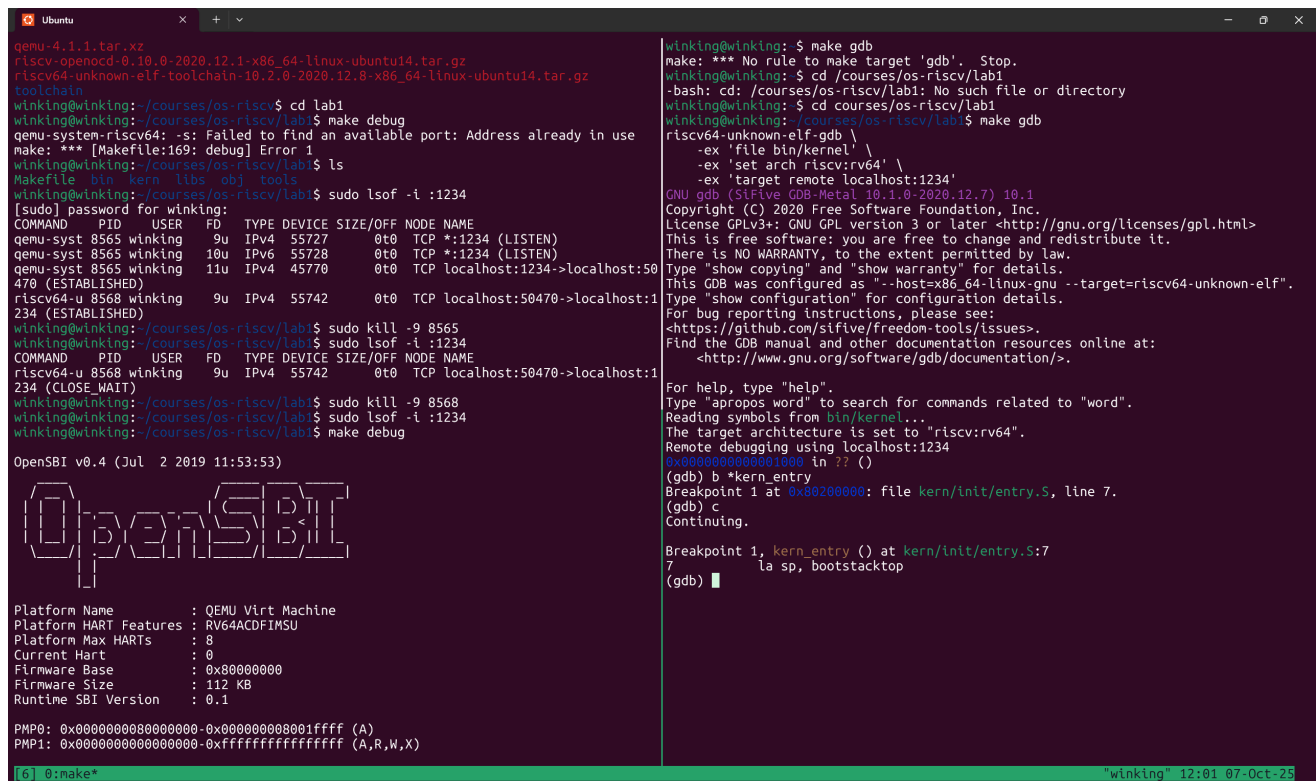
CPU加电复位后从固定复位地址开始执行也就是0x1000

最初几条指令位于 **0x1000**（OpenSBI 固件入口），简要功能如下：

1. 设置堆栈指针（SP）→ 初始化 CPU 栈
2. 设置全局指针（GP）→ 便于访问全局数据
3. 初始化内存和页表 → 为内核加载做好准备
4. 初始化基本设备 → 如 UART 控制台
5. 跳转到内核入口 → 把控制权交给内核（地址 0x80200000）

如图，PC 已经指向内核入口第一条指令。

此时还没执行第一条指令



```
qemu-4.1.1.tar.xz
riscv-openocd-0.10.0-2020.12.1-x86_64-linux-ubuntu14.tar.gz
riscv64-unknown-elf-toolchain-10.2.0-2020.12.8-x86_64-linux-ubuntu14.tar.gz
toolchain
winking@winking:~/courses/os-riscv$ cd lab1
winking@winking:~/courses/os-riscv/lab1$ make debug
qemu-system-riscv64: -s: Failed to find an available port: Address already in use
make: *** [Makefile:160: debug] Error 1
winking@winking:~/courses/os-riscv/lab1$ ls
Makefile bin kern libs obj tools
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
[sudo] password for winking:
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
qemu-syst 8565 winking 9u IPv4 55727 0t0 TCP *:1234 (LISTEN)
qemu-syst 8565 winking 10u IPv6 55728 0t0 TCP *:1234 (LISTEN)
qemu-syst 8565 winking 11u IPv4 45770 0t0 TCP localhost:1234->localhost:50
470 (ESTABLISHED)
riscv64-u 8568 winking 9u IPv4 55742 0t0 TCP localhost:50470->localhost:1
234 (ESTABLISHED)
winking@winking:~/courses/os-riscv/lab1$ sudo kill -9 8565
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
riscv64-u 8568 winking 9u IPv4 55742 0t0 TCP localhost:50470->localhost:1
234 (CLOSE_WAIT)
winking@winking:~/courses/os-riscv/lab1$ sudo kill -9 8568
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
winking@winking:~/courses/os-riscv/lab1$ make debug

OpenSBI v0.4 (Jul 2 2019 11:53:53)

Platform Name : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs : 8
Current Hart : 0
Firmware Base : 0x80000000
Firmware Size : 112 KB
Runtime SBI Version : 0.1

PMP0: 0x0000000000000000-0x0000000000001fffff (A)
PMP1: 0x0000000000000000-0xffffffffffffffff (A,R,W,X)

winking@winking: $ make gdb
make: *** No rule to make target 'gdb'. Stop.
winking@winking: $ cd /courses/os-riscv/lab1
-bash: cd: /courses/os-riscv/lab1: No such file or directory
winking@winking: $ cd /courses/os-riscv/lab1
winking@winking:~/courses/os-riscv/lab1$ make gdb
riscv64-unknown-elf-gdb \
-ex 'file bin/kernel' \
-ex 'set arch riscv:rv64' \
-ex 'target remote localhost:1234'
GNU gdb (Sifive GDB-Metal 10.1.0-2020.12.7) 10.1
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-linux-gnu --target=riscv64-unknown-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://github.com/sifive/freedom-tools/issues>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
Reading symbols from bin/kernel...
The target architecture is set to "riscv:rv64".
Remote debugging using localhost:1234
0x0000000000001000 in ?? ()
(gdb) b *kern_entry
Breakpoint 1 at 0x80200000: file kern/init/entry.S, line 7.
(gdb) c
Continuing.

Breakpoint 1, kern_entry () at kern/init/entry.S:7
7      la sp, bootstacktop
(gdb) █
```

接下来单步执行指令si

```
qemu-4.1.1.tar.xz
riscv-openocd-0.10.0-2020.12.1-x86_64-linux-ubuntu14.tar.gz
riscv64-unknown-elf-toolchain-10.2.0-2020.12.8-x86_64-linux-ubuntu14.tar.gz
toolchain
winking@winking:~/courses/os-riscv$ cd lab1
winking@winking:~/courses/os-riscv/lab1$ make debug
qemu-system-riscv64: -s: Failed to find an available port: Address already in use
make: *** [Makefile:169: debug] Error 1
winking@winking:~/courses/os-riscv/lab1$ ls
Makefile bin kern libs obj tools
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
[sudo] password for winking:
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
qemu-syst 8565 winking 9u IPv4 55727 0t0 TCP *:1234 (LISTEN)
qemu-syst 8565 winking 10u IPv6 55728 0t0 TCP *:1234 (LISTEN)
qemu-syst 8565 winking 11u IPv4 45770 0t0 TCP localhost:1234->localhost:50470 (ESTABLISHED)
riscv64-u 8568 winking 9u IPv4 55742 0t0 TCP localhost:50470->localhost:1234 (ESTABLISHED)
winking@winking:~/courses/os-riscv/lab1$ sudo kill -9 8565
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
riscv64-u 8568 winking 9u IPv4 55742 0t0 TCP localhost:50470->localhost:1234 (CLOSE_WAIT)
winking@winking:~/courses/os-riscv/lab1$ sudo kill -9 8568
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
winking@winking:~/courses/os-riscv/lab1$ make debug

OpenSBI v0.4 (Jul 2 2019 11:53:53)

Platform Name : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs : 8
Current Hart : 0
Firmware Base : 0x80000000
Firmware Size : 112 KB
Runtime SBI Version : 0.1

PMP0: 0x0000000000000000-0x0000000000000000 (A)
PMP1: 0x0000000000000000-0xffffffffffffffff (A,R,W,X)

[6] 0:make*

0x0000000000001000 in ?? ()
(gdb) b *kern_entry
Breakpoint 1 at 0x80200000: file kern/init/entry.S, line 7.
(gdb) c
Continuing.

Breakpoint 1, kern_entry () at kern/init/entry.S:7
7 la sp, bootstacktop
(gdb) si
0x0000000000020004 in kern_entry () at kern/init/entry.S:7
7 la sp, bootstacktop
(gdb)
9 tail kern_init
(gdb) i r
ra 0x80000a02 0x80000a02
sp 0x80203000 0x80203000 <SBI_CONSOLE_PUTCHAR>
gp 0x0 0x0
tp 0x8001be00 0x8001be00
t0 0x80200000 2149580800
t1 0x1 1
t2 0x1 1
fp 0x8001bd90 0x8001bd90
s1 0x8001be00 2147597824
a0 0x0 0
a1 0x82200000 2183135232
a2 0x80200000 2149580800
a3 0x1 1
a4 0x800 2048
a5 0x1 1
a6 0x82200000 2183135232
a7 0x80200000 2149580800
s2 0x800095c0 2147521984
s3 0x0 0
s4 0x0 0
s5 0x0 0
s6 0x0 0
s7 0x8 8
s8 0x2000 8192
s9 0x0 0
s10 0x0 0
s11 0x0 0
t3 0x0 0
t4 0x0 0
t5 0x0 0
t6 0x82200000 2183135232
pc 0x80200000 0x80200000 <kern_entry+8>
dscratch Could not fetch register "dscratch"; remote failure reply 'E14'
mucounteren Could not fetch register "mucounteren"; remote failure reply 'E14'
(gdb)
```

观察堆栈初始化并跟踪 OpenSBI 到内核加载过程：

```
Platform Name : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs : 8
Current Hart : 0
Firmware Base : 0x80000000
Firmware Size : 112 KB
Runtime SBI Version : 0.1

PMP0: 0x0000000000000000-0x0000000000000000 (A)
PMP1: 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
(THU.CST) os is loading ...

QEMU: Terminated
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
[sudo] password for winking:
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
riscv64-u 8682 winking 9u IPv4 65201 0t0 TCP localhost:58378->localhost:1234 (CLOSE_WAIT)
winking@winking:~/courses/os-riscv/lab1$ sudo kill -9 8682
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
winking@winking:~/courses/os-riscv/lab1$ make debug

OpenSBI v0.4 (Jul 2 2019 11:53:53)

Platform Name : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs : 8
Current Hart : 0
Firmware Base : 0x80000000
Firmware Size : 112 KB
Runtime SBI Version : 0.1

PMP0: 0x0000000000000000-0x0000000000000000 (A)
PMP1: 0x0000000000000000-0xffffffffffffffff (A,R,W,X)

[6] 0:make*

<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
Reading symbols from bin/kernel...
The target architecture is set to "riscv:rv64".
Remote debugging using localhost:1234
Ignoring packet error, continuing...
warning: unrecognized item "timeout" in "qSupported" response
Ignoring packet error, continuing...
Remote replied unexpectedly to 'vMustReplyEmpty': timeout
(gdb)
[6]+ Stopped make gdb
winking@winking:~/courses/os-riscv/lab1$ make gdb
riscv64-unknown-elf-gdb \
  -ex 'file bin/kernel' \
  -ex 'set arch riscv:rv64' \
  -ex 'target remote localhost:1234'
GNU gdb (Sifive GDB-Metal 10.1.0-2020.12.7) 10.1
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-linux-gnu --target=riscv64-unknown-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://github.com/sifive/freedom-tools/issues>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
Reading symbols from bin/kernel...
The target architecture is set to "riscv:rv64".
Remote debugging using localhost:1234
0x0000000000001000 in ?? ()
(gdb) b *0x80200000
Breakpoint 1 at 0x80200000: file kern/init/entry.S, line 7.
(gdb) watch *0x80200000
Hardware watchpoint 2: *0x80200000
(gdb) c
Continuing.

Breakpoint 1, kern_entry () at kern/init/entry.S:7
7 la sp, bootstacktop
(gdb) info registers pc
pc 0x80200000 0x80200000 <kern_entry>
(gdb)
```

watch: 监控内核写入内存的瞬间，CPU 会在内核写入 0x80200000 时暂停，用 info registers pc 查看 PC，确认内核已经被加载。

下一步继续单步跟踪，观察堆栈指针、全局指针、函数参数等寄存器的初始化

```
Platform Name       : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs    : 8
Current Hart         : 0
Firmware Base        : 0x80000000
Firmware Size        : 112 KB
Runtime SBI Version   : 0.1

PMP0: 0x00000000-0x00000000-0x000000001fffff (A)
PMP1: 0x00000000-0x00000000-0xffffffffffff (A,R,W,X)
(THU.CST) os is loading ...

QEMU: Terminated
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
[sudo] password for winking:
COMMAND  PID  USER  FD  TYPE  DEVICE  SIZE/OFF  NODE NAME
riscv64-u 8682 winking 9u  IPv4  65201    0t0  TCP localhost:58378->localhost:1234 (CLOSE_WAIT)
winking@winking:~/courses/os-riscv/lab1$ sudo kill -9 8682
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
winking@winking:~/courses/os-riscv/lab1$ make debug

OpenSBI v0.4 (Jul  2 2019 11:53:53)

Platform Name       : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs    : 8
Current Hart         : 0
Firmware Base        : 0x80000000
Firmware Size        : 112 KB
Runtime SBI Version   : 0.1

PMP0: 0x00000000-0x00000000-0x000000001fffff (A)
PMP1: 0x00000000-0x00000000-0xffffffffffff (A,R,W,X)

0x000000000200004 in kern_entry () at kern/init/entry.S:7
7      la sp, bootstacktop
(gdb) info registers
ra      0x80000a02      0x80000a02
sp      0x80203000      0x80203000 <SBI_CONSOLE_PUTCHAR>
gp      0x0             0x0
tp      0x8001be00      0x8001be00
t0      0x80200000      2149580800
t1      0x1             1
t2      0x1             1
fp      0x8001bd90      0x8001bd90
s1      0x8001be00      2147597824
a0      0x0             0
a1      0x82200000      2183135232
a2      0x80200000      2149580800
a3      0x1             1
a4      0x800           2048
a5      0x1             1
a6      0x82200000      2183135232
a7      0x80200000      2149580800
s2      0x800095c0      2147521984
s3      0x0             0
s4      0x0             0
s5      0x0             0
s6      0x0             0
s7      0x8             8
s8      0x2000          8192
s9      0x0             0
s10     0x0             0
s11     0x0             0
t3      0x0             0
t4      0x0             0
t5      0x0             0
t6      0x82200000      2183135232
pc      0x80200004      0x80200004 <kern_entry+4>
dscratch Could not fetch register "dscratch"; remote failure reply 'E14'
mucounteren Could not fetch register "mucounteren"; remote failure reply 'E14'
(gdb) x/10i $pc
=> 0x80200004 <kern_entry+4>: mv     sp,sp
0x80200008 <kern_entry+8>: j      0x8020000a <kern_init>
0x8020000a <kern_init>: auipc  a0,0x3
0x8020000e <kern_init+4>: addi   a0,a0,-2
0x80200012 <kern_init+8>: auipc  a2,0x3
0x80200016 <kern_init+12>: addi   a2,a2,-10
0x8020001a <kern_init+16>: addi   sp,sp,-16
0x8020001c <kern_init+18>: li     a1,0
0x8020001e <kern_init+20>: sub    a2,a2,a0
0x80200020 <kern_init+22>: sd     ra,8(sp)
(gdb)
```

以上是完整调试过程，回答问题：**RISC-V 硬件加电后最初执行的几条指令位于什么地址？它们主要完成了哪些功能？**

RISC-V 硬件加电后的最初执行地址

- **复位地址**：RISC-V CPU 加电后最先执行的指令位于 0x1000。
- 这些地址对应的是 **SBI 固件 (OpenSBI) 的汇编代码段**。

最初几条指令的主要功能：

1. 0x1000: auipc t0,0x0

生成 PC 相对地址，初始化 t0，为后续内存访问或跳转做准备。

初始化堆栈指针为固件本身的运行设置临时堆栈

2. 0x1004: addi a1,t0,32 #0x1020

t0 + 32 → a1，用作下一步跳转或参数传递，准备进入 OpenSBI 主入口。

3. 0x1008 csrr a0,mhartid 读取当前 hart (CPU 核心) ID，存入 a0，多核初始化需要。

4. 0x100c ld t0,24(t0) 从 t0+24 的内存地址加载值到 t0，通常是读取跳转地址或固件参数。

0x1010-0x80000000

进入 OpenSBI 的初始化函数，主要完成最基础的硬件初始化：

- 配置 CPU 寄存器 (CSR)
- 设置中断控制器
- 初始化内存布局 (如清零 BSS 段)

- 为内核加载做准备

```
Platform Max HARTs : 8
Current Hart : 0
Firmware Base : 0x80000000
Firmware Size : 112 KB
Runtime SBI Version : 0.1

PMP0: 0x0000000000000000-0x0000000000001fffff (A)
PMP1: 0x0000000000000000-0xfffffffffffff (A,R,W,X)
(THU.CST) os is loading ...

QEMU: Terminated
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
[sudo] password for winking:
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
riscv64-u 8682 winking 9u IPv4 65201 0t0 TCP localhost:58378->localhost:1234 (CLOSE_WAIT)
winking@winking:~/courses/os-riscv/lab1$ sudo kill -9 8682
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
winking@winking:~/courses/os-riscv/lab1$ make debug

OpenSBI v0.4 (Jul 2 2019 11:53:53)

Platform Name : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs : 8
Current Hart : 0
Firmware Base : 0x80000000
Firmware Size : 112 KB
Runtime SBI Version : 0.1

PMP0: 0x0000000000000000-0x0000000000001fffff (A)
PMP1: 0x0000000000000000-0xfffffffffffff (A,R,W,X)
QEMU: Terminated
winking@winking:~/courses/os-riscv/lab1$ sudo lsof -i :1234
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
riscv64-u 8845 winking 9u IPv4 931 0t0 TCP localhost:33498->localhost:1234 (CLOSE_WAIT)
winking@winking:~/courses/os-riscv/lab1$ sudo kill -9 8845
winking@winking:~/courses/os-riscv/lab1$ make debug

[6] 0:make*

t3 0x0 0
t4 0x0 0
t5 0x0 0
t6 0x0 0
pc 0x100c 0x100c
dscratch Could not fetch register "dscratch"; remote failure reply 'E14'
mucounteren Could not fetch register "mucounteren"; remote failure reply 'E14'
(gdb) si
0x0000000000000010 in ?? ()
(gdb) si
0x0000000000000000 in ?? ()
(gdb) info register
ra 0x0 0x0
sp 0x0 0x0
gp 0x0 0x0
tp 0x0 0x0
t0 0x80000000 2147483648
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
s4 0x0 0
s5 0x0 0
s6 0x0 0
s7 0x0 0
s8 0x0 0
s9 0x0 0
s10 0x0 0
s11 0x0 0
t3 0x0 0
t4 0x0 0
t5 0x0 0
t6 0x0 0
pc 0x80000000 0x80000000
dscratch Could not fetch register "dscratch"; remote failure reply 'E14'
mucounteren Could not fetch register "mucounteren"; remote failure reply 'E14'
(gdb) si
0x0000000000000004 in ?? ()
(gdb)
```

总结：本实验中重要的知识点以及对对应得os原理中的知识点

1.RISCV加电复位和启动流程：

实验通过qemu+GDb的模拟与调试方式，模拟了cpu从加电到将进入entry的初始化的过程中的一系列变化，包括sp指针的指向以及从汇编转为c语言。**差异：**实验中更低层，涉及具体硬件地址和指令；OS原理课程中更抽象，只讲 CPU 启动流程和 OS 初始化概念。

2.PC、SP、寄存器状态观察：

通过info register指令观察各个寄存器状态以及栈的使用情况，偏于理解函数调用与寄存器使用。**差异：**实验可以看到真实地址和寄存器变化；原理课程更抽象，注重概念。

3.内核映像加载与地址空间布局：

通过 watch *0x80200000 观察内核被写入；这对应 OS 原理中“从磁盘加载内核到内存后执行”的阶段。差异在于实验中由 OpenSBI 完成内存拷贝，而原理课中通常用引导加载器描述这个过程。

4.控制权转移：

实验通过 b *0x80200000 验证了控制权的移交，这是理论中的“从引导程序进入操作系统”的关键一步。差异在于理论上只讲“加载并跳转”，实验中可以在 GDB 中精确看到这条跳转指令的执行。

5.操作系统启动的全过程

QEMU启动 → OpenSBI固件 → 内核加载 → kern_entry → kern_init

从QEMU模拟器加载OpenSBI固件开始，到内核被加载到指定的内存地址0x80200000，最终通过kern_entry汇编入口跳转到kern_init的C语言环境。这一完整的启动链条具体展现了操作系统原理中系统引导过程的理论概念。

6.特权分级

M-mode、S-mode、U-mode的三级划分以及ecall指令的模式切换机制，对应着操作系统原理中处理器保护环和特权级分离的核心概念。这种硬件支持的权限分离是现代操作系统安全性的基石，不同特权级对应不同的权限和能力集合，通过标准化的切换机制确保系统的稳定运行。控制寄存器的使用进一步加深了对硬件资源管理的理解。

7.内存管理

在内存管理方面，实验通过链接脚本具体实现了文本段、数据段、BSS段的布局规划，展现了地址相关代码的特性和栈空间的手动管理。对应着操作系统原理中程序内存模型和虚拟内存基础的理论知识。不同内存段的划分特性和管理策略直接关系到系统的可靠性和性能表现。

8.I/O系统

I/O系统的构建采用了层次化架构，从底层的SBI控制台驱动到上层的cprintf函数，形成了完整的设备抽象栈。这种自底向上的构建过程对应着操作系统原理中设备驱动模型和I/O子系统的设计理念。

9.EFL和BIN

ELF格式: 包含符号表、重定位信息、调试信息的结构化格式

BIN格式: 纯粹的二进制镜像，线性内存映射

ELF格式与BIN格式的对比转换展现了不同文件格式在调试支持、文件大小、加载复杂度等方面的权衡取舍。ELF格式的结构化信息为符号调试和动态链接提供了支持，而BIN格式的简洁性则更适合嵌入式和无OS环境。

10.调试

QEMU与GDB的交叉调试环境配合远程调试协议的使用，提供了系统级开发的强大工具支持。这对应着操作系统原理中系统调试技术和开发方法学的理论体系，

11.构建系统

Makefile自动化构建: 编译、链接、格式转换的流水线，这对应着操作系统将源代码转换为可执行程序的过程，自动化构建显著提高了开发效率和软件质量，完整的工具链支撑是系统开发成功的重要保障。

本实验未涉及(课堂见过):

1. 进程调度与上下文切换，OS原理课程重点讲进程状态、调度算法、寄存器保存与恢复。
2. 虚拟内存与分页机制，内核管理虚拟地址空间，页表转换机制。
3. 文件系统与 I/O 子系统：内核管理磁盘数据和目录结构

*以下是运行到断点处再往后逐步运行几条指令，观察后继变化

现在已经：

1. 用 QEMU 模拟了 RISC-V CPU
2. 用 GDB 在内核入口停下
3. 看到了内核 **第一条汇编指令**在做栈初始化
4. 下一步就是单步跟踪 kern_init 的执行

```
oftmmu
ls qemu-system-riscv64
qemu-system-riscv64
winking@winking:~/courses/os-riscv/qemu-4.1.1/riscv64-softmmu$ cd ~/courses/os-riscv/qemu-4.1.1/riscv64-softmmu
winking@winking:~/courses/os-riscv/qemu-4.1.1/riscv64-softmmu$ ls qemu-system-riscv64
qemu-system-riscv64
winking@winking:~/courses/os-riscv/qemu-4.1.1/riscv64-softmmu$ cd ~/courses/os-riscv/lab1
winking@winking:~/courses/os-riscv/lab1$ ../../qemu-4.1.1/riscv64-softmmu/qemu-system-riscv64 -machine virt -nographic -bios default -kernel bin/ucore.img -s -S
-bash: ../../qemu-4.1.1/riscv64-softmmu/qemu-system-riscv64: No such file or directory
winking@winking:~/courses/os-riscv/lab1$ ../../qemu-4.1.1/riscv64-softmmu/qemu-system-riscv64 -machine virt -nographic -bios default -kernel bin/ucore.img -s -S
-bash: ../../qemu-4.1.1/riscv64-softmmu/qemu-system-riscv64: No such file or directory
winking@winking:~/courses/os-riscv/lab1$ file ../../qemu-4.1.1/riscv64-softmmu/qemu-system-riscv64
../../qemu-4.1.1/riscv64-softmmu/qemu-system-riscv64: cannot open ' ../../qemu-4.1.1/riscv64-softmmu/qemu-system-riscv64' (No such file or directory)
winking@winking:~/courses/os-riscv/lab1$ cd ~/courses/os-riscv/qemu-4.1.1/riscv64-softmmu
winking@winking:~/courses/os-riscv/qemu-4.1.1/riscv64-softmmu$ ./qemu-system-riscv64 -machine virt -nographic -bios default -kernel ~/courses/os-riscv/lab1/bin/ucore.img -s -S

OpenSBI v0.4 (Jul  2 2019 11:53:53)

Platform Name       : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs   : 8
Current Hart        : 0
Firmware Base        : 0x80000000
Firmware Size        : 112 KB
Runtime SBI Version   : 0.1

PMP0: 0x0000000000000000-0x0000000000000000 (A)
PMP1: 0x0000000000000000-0xffffffffffffffff (A,R,W,X)

winking@winking:~/courses/os-riscv/qemu-4.1.1/riscv64-softmmu$ cd ~/courses/os-riscv/lab1/bin
winking@winking:~/courses/os-riscv/lab1/bin$ ./riscv64-unknown-elf-gdb ~/courses/os-riscv/lab1/bin/kernel
GNU gdb (Sifive GDB-Metal 10.1.0-2020.12.7) 10.1
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-linux-gnu --target=riscv64-unknown-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://github.com/sifive/freedom-tools/issues>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from /home/winking/courses/os-riscv/lab1/bin/kernel...
(gdb) (gdb) target remote :1234
Undefined command: "... Try "help".
(gdb) (gdb) b *0x80200000
Undefined command: "... Try "help".
(gdb) (gdb) c
Undefined command: "... Try "help".
(gdb) (gdb) target remote :1234
Undefined command: "... Try "help".
(gdb) target remote :1234
Remote debugging using :1234
0x0000000000000000 in ?? ()
(gdb) b *0x80200000
Breakpoint 1 at 0x80200000: file kern/init/entry.S, line 7.
(gdb) c
Continuing.

Breakpoint 1, kern_entry () at kern/init/entry.S:7
7      la sp, bootstacktop
(gdb) █
```

在向下运行一条指令查看寄存器状态，发现bootstacktop的地址传给sp指针。


```

0x00000000080200004 in kern_entry () at kern/init/entry.S:7
7      la sp, bootstacktop
(gdb) info register
ra             0x80000a02      0x80000a02
sp             0x80203000      0x80203000 <SBI_CONSOLE_PUTCHAR>
gp             0x0            0x0
tp             0x8001be00      0x8001be00
t0             0x80200000      2149580800
t1             0x1            1
t2             0x1            1
fp             0x8001bd90      0x8001bd90
s1             0x8001be00      2147597824
a0             0x0            0
a1             0x82200000      2183135232
a2             0x80200000      2149580800
a3             0x1            1
a4             0x800          2048
a5             0x1            1
a6             0x82200000      2183135232
a7             0x80200000      2149580800
s2             0x800095c0      2147521984
s3             0x0            0
s4             0x0            0
s5             0x0            0
s6             0x0            0
s7             0x8            8
s8             0x2000          8192
s9             0x0            0
s10            0x0            0
s11            0x0            0
t3             0x0            0
t4             0x0            0
t5             0x0            0
t6             0x82200000      2183135232
pc             0x80200004      0x80200004 <kern_entry+4>
dscratch      Could not fetch register "dscratch"; remote failure reply 'E14'
mucounteren   Could not fetch register "mucounteren"; remote failure reply 'E14'
(gdb) p &bootstacktop
$1 = (<data variable, no debug info> *) 0x80203000 <SBI_CONSOLE_PUTCHAR>

```

接下来执行kern_inti, pc寄存器地址发生改变, 说明内核启动流程已经从 **汇编入口** (kern_entry) 转到 **C 语言初始化** (kern_init)