

# Lab1

## 1. 课程 > 内核、Boot 和 printf > 操作系统的启动 > 内核在哪里？

### 习题一（多项选择）

1/1 point (ungraded)

1. 以下哪些是非易失性存储？

☒ ROM

☐ RAM

☒ 磁盘

☒ FLASH



[保存](#) | [Show answer](#)

### 习题二（多项选择）

1/1 point (ungraded)

2. 下列有关bootloader的说法正确的是？

☒ bootloader是在操作系统内核运行之前运行的

☒ 使用bootloader有助于操作系统的移植

☐ bootloader只需要进行硬件初始化

☒ 有了bootloader，一台电脑上就有可能运行多个操作系统了



[保存](#) | [Show answer](#)

## 2. 课程 > 内核、Boot 和 printf > 操作系统的启动 > Bootloader

### 问题一（多项选择）

1/1 point (ungraded)

1.以下说法错误的是

☒ stage1阶段会初始化stage1与stage2所需所有硬件设备

☒ stage1会准备RAM空间，并运行在RAM上

☐ stage2将内核镜像文件读入RAM中

☐ stage2最终将控制权交给操作系统内核



[保存](#) | [Show answer](#)

## 3. 课程 内核、Boot 和 printf Let's hack the kernel! Makefile—内核代码的地图

### 问题一（判断正误）

1/1 point (ungraded)

执行make clean会在目录下生成可执行文件clean。

☐ 正确

☒ 错误



[Show answer](#)

## 4. 课程 > 内核、Boot 和 printf > Let's hack the kernel! > ELF—深入探究编译与链接

### 问题一（多项选择）

1/1 point (ungraded)

ELF文件包含哪几种文件类型？

- ☒ 可重定位文件
- ☐ .h头文件
- ☒ 可执行文件
- ☒ 共享对象文件



[保存](#) | [Show answer](#)

### 问题二（单项选择）

1/1 point (ungraded)

我们可以使用Elf32\_Ehdr结构中的哪部分判断是否是ELF文件？

- ☒ e\_ident
- ☐ e\_type
- ☐ e\_machine
- ☐ e\_version



[保存](#) | [Show answer](#)

### 问题三（单项选择）

1/1 point (ungraded)

请阅读readelf/kerelf.h, 回答：

Elf32\_Section的数据类型实际上是？

- ☐ int32\_t
- ☐ u\_int32\_t
- ☒ u\_int16\_t
- ☐ u\_int64\_t



[保存](#) | [Show answer](#)

### 问题四（单项选择）

1/1 point (ungraded)

对于可重定位(relocatable)文件，查看其e\_type值为多少？

- ☒ 1
- ☐ 2
- ☐ 3
- ☐ 4



[保存](#) | [Show answer](#)

### 问题五（单项选择题）

1/1 point (ungraded)

printf的实现是在以下哪个阶段被插入到最终的可执行文件中的？

- ☐ 预处理
- ☐ 编译
- ☒ 链接
- ☐ 执行



[保存](#) | [Show answer](#)

### 问题六（单项选择题）

1/1 point (ungraded)

可执行文件中，filesize()memsize

- ☐ 大于等于
- ☐ 一定等于
- ☒ 小于等于
- ☐ 没有关系



[保存](#) | [Show answer](#)

## 5. 课程 > 内核、Boot 和 printf > Let's hack the kernel! > MIPS内存布局—寻找内核的正确位置

### 习题一（单项选择题）

1/1 point (ungraded)

操作系统Mips内存中可以不经过cache直接映射的是[]

- ☐ kuseg
- ☐ kseg0
- ☒ kseg1
- ☐ kseg2



[保存](#) | [Show answer](#)

## 6. 课程 > 内核、Boot 和 printf > Let's hack the kernel! > Linker Script—控制加载地址

### 问题一（单项选择题）

1/1 point (ungraded)

1.保存已初始化的全局变量和静态变量的段是？

- ☐ .text
- ☒ .data
- ☐ .bss
- ☐ .static



[保存](#) | [Show answer](#)

### 问题二（单项选择题）

1/1 point (ungraded)

2.观察代码，在我们的实验中，是如何指定程序入口的？

- ☐ 使用ld命令时，通过参数“-e”设置
- ☒ 在linker script中使用ENTRY()指令指定了程序入口
- ☐ 定义start为程序入口
- ☐ .text段的第一个字节
- ☐ 地址0处



保存 | [Show answer](#)

### 问题三（填空题）

1/1 point (ungraded)

3.浏览我们的代码库，回答以下问题。

在我们的实验中，内核的启动代码在哪里？请寻找并回答包含启动函数的文件路径。  
回答格式为相对xxx(你的学号)的相对路径。例如：readelf.c的路径为：./readelf/readelf.c

[Show answer](#)

## 7. 课程 > 内核、Boot 和 printf > MIPS汇编与C语言 > 循环与判断

### 问题一（填空题）

1/1 point (ungraded)

阅读下列汇编代码，回答问题：

```

1  ...
2  ...
3  func:
4  sw $fp, 0($sp)
5  addi $fp, $sp, 4
6  sw $ra, -8($fp)
7  add $t1, $zero, $a0
8  add $t2, $zero, $a1
9  bgez $t1, if_else_0
10 li $t8, 0
11 sub $t0, $t8, $t1
12 add $t0, $t0, 0
13 add $v0, $t2, $t0
14 move $sp, $fp
15 lw $fp, -4($sp)
16 lw $ra, -8($sp)
17 jr $ra
18 if_else_0:
19 if_end_0:
20 add $t0, $t1, -0
21 add $v0, $t2, $t0
22 move $sp, $fp
23 lw $fp, -4($sp)

```

```
24 lw $ra, -8($sp)
25 jr $ra
26 ...
27 ...
```

当调用func函数并传入参数依次为 -2020,-2018 时，函数返回值是多少？  
直接回答数字结果，例如：2333




[Show answer](#)

## 8. 课程 > 内核、Boot 和 printf > MIPS汇编与C语言 > 函数调用

### 问题一（单项选择）

1/1 point (ungraded)

观察mmu.h的内存地图，内核栈指针应该设到什么位置？

☐ USTACKTOP

☐ UXSTACKTOP

☐ KERNBASE

☒ KSTACKTOP


[保存](#) | [Show answer](#)

### 问题二（单项选择）

1/1 point (ungraded)

mips中函数跳转，一般使用下列哪条指令？(它在跳转的同时把返回地址存储在\$ra中)

☐ j

☒ jal

☐ jalr

☐ jr


[保存](#) | [Show answer](#)

## 9. 课程 > 内核、Boot 和 printf > MIPS汇编与C语言 > 通用寄存器使用约定

### 问题一（多项选择）

1/1 point (ungraded)

1.以下哪条指令会用到\$at寄存器？（可以在mars中编译试试看）

☐ addi \$t1, \$t1, 0x100

☒ beq \$t1, 1, label

☐ lui \$t1, 0x100

☒ ori \$t1, \$t1, 0x10000


[保存](#) | [Show answer](#)

问题二（多项选择）

1/1 point (ungraded)

2.按照表格中对寄存器的使用说明，一般在调用函数时，以下哪些寄存器需要保存在栈中？

☐ \$a0--\$a3

☐ \$t0--\$t7

☒ \$s0--\$s7

☒ \$ra



保存 | [Show answer](#)

0. 课程 > 内核、Boot 和 printf > 实战printf > 实战printf

问题一（填空题）

1/1 point (ungraded)

外部设备通常是通过特定地址写入不同的值来控制的。请阅读printf相关代码，回答以下问题。

在我们的实验中，当我们使用printf输出字符时，实际上是通过写入哪个内存地址实现单个字符的输出？  
使用16进制数字回答问题，A-F使用大写字母。例如：0x2333333A

0xB0000000



[Show answer](#)