Assignment 1

zhiyuan zhong, 12110517

Q1

-`machine virt`: specifies the type of machine to emulate (specifying a virtual machine).

-`nographic`: This option tells QEMU not to open a graphical window for the virtual machine's output. Instead, it redirects the output to the console or terminal where QEMU was launched.

-`bios default`: This option specifies the BIOS image to use. In this case, it's using the default BIOS image provided by QEMU.

-`device loader, file=bin/ucore.bin, addr=0x80200000`: This option configures a loader device for QEMU, specifying the file to load (ucore.bin) and the address (0x80200000) where it should be loaded in the emulated system's memory.

> 这条指令相当于给我们的模拟计算机插电，然后qemu会调用内置的OpenSBI初始化硬件环境并启动我们的操作系统。OpenSBI 所做的事情就是把CPU 从 M Mode 切换到 S Mode，接着跳转到一个设置好的地址 `0x80200000`，开始执行内核代码 (from lab slides) --sy

Q2

1. `. = BASE_ADDRESS;`: Sets the current memory address (denoted by `.`) to the base address specified by `BASE_ADDRESS`. This indicates where the kernel will be loaded into memory. It ensures that subsequent sections are loaded starting from this address.

2. `.text : {`: Begins the section for executable code (text segment).

3. `*(.text.kern_entry)`: Includes all code sections labeled `.text.kern_entry`. This typically includes the entry point of the kernel code.

4. `*(.text .stub .text.* .gnu.linkonce.t.*)`: Includes other code sections like `.text`, `.stub`, and any sections matching the pattern `.text.*` or `.gnu.linkonce.t.*`. These sections may contain various kernel code.

5. `PROVIDE(etext = .);`: Defines a symbol named `etext` and assigns it the current memory address. This symbol marks the end of the text segment. It can be used by programs to determine the end of the code section.

6. `.rodata : {`: Begins the read-only data section.

7. `*(.rodata .rodata.* .gnu.linkonce.r.*)`: Includes all read-only data sections, such as `.rodata`, `.rodata.*`, and `.gnu.linkonce.r.*`. This typically contains constants or read-only data used by the kernel.

8. `. = ALIGN(0x1000);`: Aligns the address to the next page boundary (0x1000 bytes).

Q3

- `extern char edata[], end[];`: This line declares two external symbols `edata` and `end`. These symbols are defined in the `kernel.ld`, `edata` marks the end of initialized data sections (.data and .sdata), while end marks the end of the entire data segment (including zero-initialized data)

```
memset(edata, 0, end - edata) /* void *memset(void *b, int c, size_t len) */
```

parameters:

- `edata`: The starting address of the memory region to be set to zero. This is typically the address where initialized data ends.
- `0`: The value to which each byte in the memory region will be set. In this case, the memory will be cleared to 0.
- `end - edata`: The length of the memory region to be set. This is calculated as the difference between the end of initialized data (`end`) and the start of initialized data (`edata`). It represents the size of the initialized data section.

This line uses the `memset` function to set the memory region starting from `edata` and ending at `end` to zero.

purpose

The purpose is to ensure that the BSS segment of the kernel is set to zero, ensuring that any global or static variables declared without an explicit initializer are set to zero before the kernel starts executing. It ensures predictable behavior and prevents potential bugs that may arise from using uninitialized variables.

Q4

> 就像月饼包装，把它封了一层又一层 -- sy

1. `cputs()` iterates through each character in the provided string.
2. For each character, it calls `cputch()` with the character and a counter.
3. `cputch()` function calls `cons_putc()` to print the character to the console (stdout).
4. `cons_putc()` ultimately calls `sbi_console_putchar()` to print a single character to console devices.
5. The `sbi_console_putchar()` function communicates with the console hardware (I/O device) using the `sbi_call` instruction (OpenSBI提供的接口).