Assignment 1 Report

1) [20pts] The `make qemu` command refers to a label in the corresponding Makefile, which corresponds to:

```
qemu-system-riscv64 \
    -machine virt \
    -nographic \
    -bios default \
    -device loader,file=bin/ucore.bin,addr=0x80200000
```

Please explain the function of each option in the above command.

Answer:

- `-machine virt` : Use the generic virtual platform `virt` board as the machine.
- `-nographic` : No GUI, which means the user should use command line to interaction. and redirect the virtual machine's console to the current terminal.
- `-bios default` : This option will load the default OpenSBI firmware automatically.
- `-device loader` : Load data from the file `ucore.bin` , and `addr` specifics the address to store the data into the memory.

2) [20pts] Please explain the function of each line in the following snippet from the `tools/kernel.ld` linker script file. (Refer to: https://sourceware.org/binutils/docs/ld/Scripts.html)

Answer:

```
SECTIONS /* This command is used to describe the memory layout of the output file */
{
    /* Load the kernel at this address: "." means the current address */
    . = BASE_ADDRESS; /* Let the location counter be the `BASE_ADDRESS` which is assigned the value 0x80200000 in pervious lines */

    .text : { /* This block list the source of `.text` (code) section for output file */
        *(.text.kern_entry) /* Place the `.text.kern_entry` input section to the output section for every input file */
        *(.text .stub .text.* .gnu.linkonce.t.*) /* Place these input section to the output section for every input file */
    }

    PROVIDE(etext = .); /* Define the 'etext' symbol to this value */
```

```
12
13      .rodata : { /* This block list the source of `.rodata` (read only
    data) section for output file */
14          *(.rodata .rodata.* .gnu.linkonce.r.*) /* Place these input
    section to output file for every input file */
15      }
16
17      /* Adjust the address for the data segment to the next page */
18      . = ALIGN(0x1000); /* Specify the alignment of data segment */
19
20      /* Below is the rest of the code */
21  }
```

3) [10pts] Please explain the parameters and the purpose of the statement `memset(edata, 0, end - edata);` within `kern/init/init.c` . (The relevant code to be read includes `init.c` and `kernel.ld`)

Answer:

- Parameters
  - `edata` : In linker script, `edata` is end address of `.data` section and start address of `.bss` section.
  - `end` : In linker script, `end` is end address of `.bss` section. `end - edata` means the length of `.bss` section.
  - `0` : Initial the section with value 0
- Function
  - Initial the whole `.bss` section with 0. "Block Started by Symbol" also known as "Zero Initialization" section. Corresponds to global variables that are not explicitly initialized in C language.

4) [20pts] Please describe how the `cputs()` instruction prints characters through the SBI.

Answer:

1. `cputs()` calls `cputch()` in a loop, passing a single character and a pointer of a counter each time, and stops until the string terminator `\0` .
2. `cputch()` calls `cons_putc()` and passes the character, then let the counter increment by 1.
3. `cons_putc()` calls `sbi_console_putchar()` , `sbi_console_putchar()` calls `sbi_call()` , and passes the character.
4. `sbi_call()` uses the SBI type code and the character to generate inline assembly RISC-V code. It let type code `1` to `x17` register, and target character to `x10` register, then use `ecall` instruction to make system calls. After that one character will be printed. Finally, `sbi_call()` return the value of `x10` register which is also the value of the character.

5) [30pts] Programming Download the code from GitLab: `git clone ssh://git@mirrors.sustech.edu.cn:13389/operating-systems/project/kernel_assignment_12XXXXXX.git` (Replace `12XXXXXX` with your student ID) According to the description, complete the `sbi_shutdown()` function within the `libs/sbi.c`

and the `double_cputs()` function within the `kern/libs/stdio.c` .

Output:

```
riscv64-unknown-elf-objcopy bin/kernel --strip-all -O binary bin/ucore.bin

OpenSBI v0.6
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name          : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs     : 8
Current Hart           : 0
Firmware Base          : 0x80000000
Firmware Size          : 120 KB
Runtime SBI Version    : 0.2

MIDELEG : 0x0000000000000222
MEDELEG : 0x000000000000b109
PMP0    : 0x0000000080000000-0x000000008001ffff (A)
PMP1    : 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
os is loading ...

ooss  iiss  llooaaddiinngg  ......

```

Answer:

`sbi_shutdown()` function within the `libs/sbi.c`:

```c
void sbi_shutdown()
{
    sbi_call(SBI_SHUTDOWN, 0, 0, 0);
}
```

`double_cputs()` function within the `kern/libs/stdio.c`

```c
int double_cputs(const char *str)
{
    int cnt = 0;
    char c;
    while ((c = *str++) != '\0') {
        cputch(c, &cnt);
        cputch(c, &cnt);
    }
    cputch('\n', &cnt);

    return cnt >> 1;
}
```