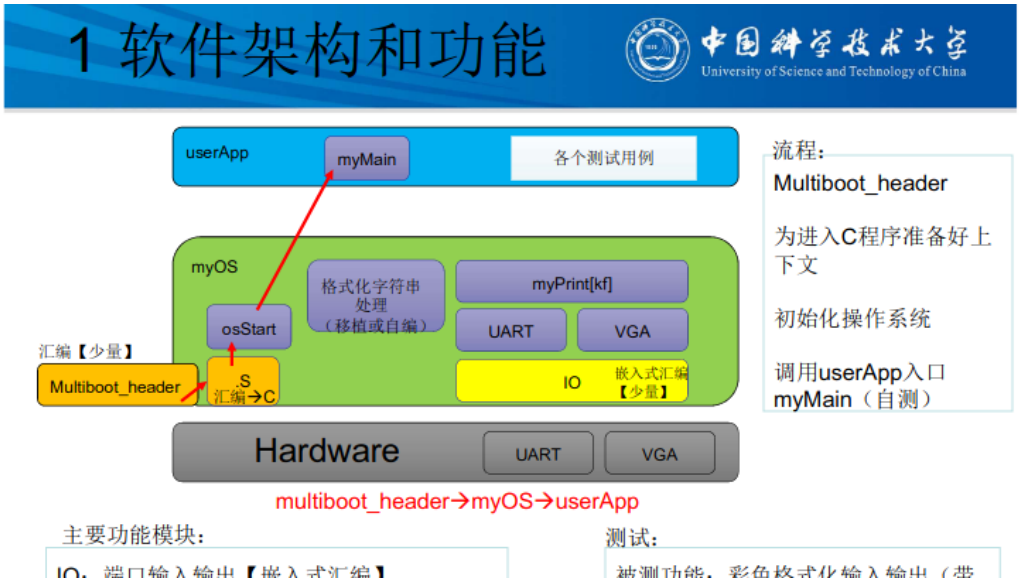
实验报告

李卓 pb19000064

一， 给出软件主流程实现及其框图



qemu 用 multibootheader 启动协议，引导进入 myos，start.s 为 c 语言程序初始化，然后 call osStart 执行 osstart 程序调用用户程序，用户程序中使用 myOs 中完成的 myprintk myprintf，这两个函数完成串口输出和屏幕输出的任务

IO：端口输入输出(嵌入式汇编) UART：串口输入 VGA：清屏、屏幕彩色输出（带滚屏）

二， 说明主要功能模块及其实现

1．vga.c 提供 clear_screen 和 append2screen 接口，完成清屏和 vga 输出

append2screen：循环调用 put_char2pos

clear_screen(): 输出 w*l 个黑色空格然后光标定位 0

put_char2pos():换行符光标下移一行，其他字符则修改显存光标右移，自动判断滚屏并重定位光标

scroll_screen(): 直接修改显存，逐行上移覆盖前一行的内容，最后一行全部修改为黑色空格。此函数不会修改光标位置。

set_cursor_pos() 和 get_cursor_pos()很简单 见代码

2．uart.c 向串口输出字符串

uart_put_chars()循环调用 uart_put_char()，使用 outb 写入

3．myprintk.c 提供 myOs 中的 myprintk() myprintf()

vsprintf 从 C 语言库函数中移植，通过 vsprintf 将 args 以指定格式 format 输出到句柄数组，然后调用 append2screen 和 uart_put_chars 输出

三， 源代码组织说明

四， 代码布局说明（地址空间）

Section	Offset (Base = 1M)	align
.multiboot_header	0	8
.text(代码段)	16	8
.data(数据段)	16+.text section	16
.bss	当前	16
_end	当前	16

在 1M 位置处放置 multiboot_header 启动代码 然后放置代码段 然后放置数据段 接着放置 bss 段

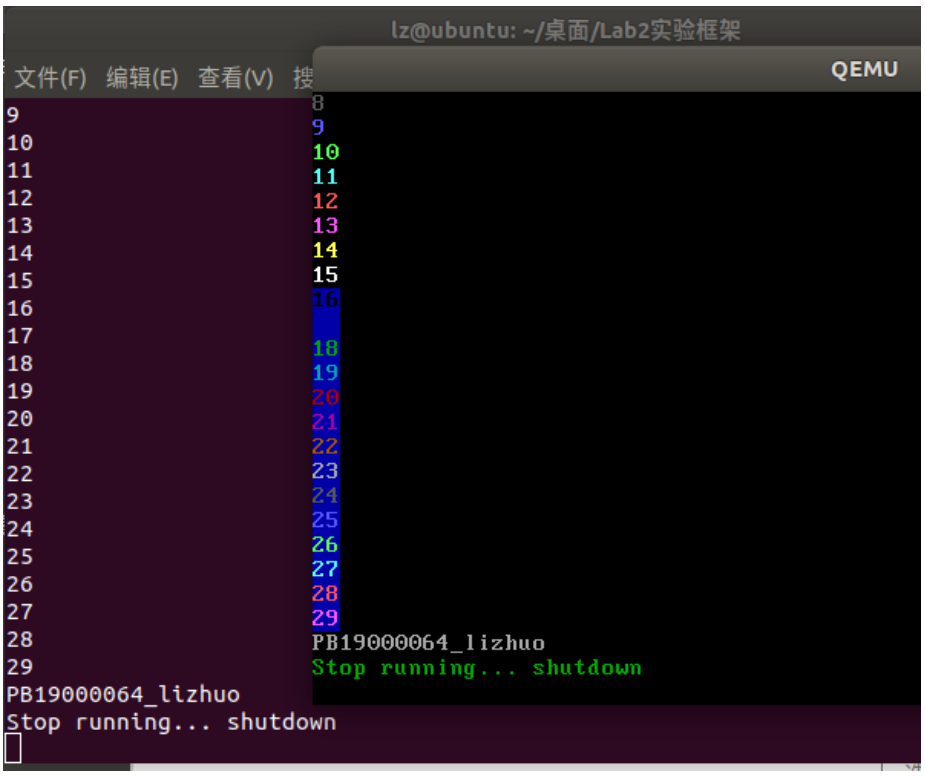
五， 编译过程说明

直接运行 source2img.sh

gcc 编译各个文件 把.s文件生成生成相应的.o文件 用链接器把.o文件按照 ld 部署要求把他链接成.bin文件 生成 myOS.elf 文件

六， 运行和运行结果说明

直接运行 source2img.sh


lz@ubuntu: ~/桌面/Lab2实验框架
文件(F) 编辑(E) 查看(V) 搜
QEMU
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
PB19000064_lizhuo
Stop running... shutdown
8
9
10
11
12
13
14
15
16
18
19
20
21
22
23
24
25
26
27
28
29
PB19000064_lizhuo
Stop running... shutdown