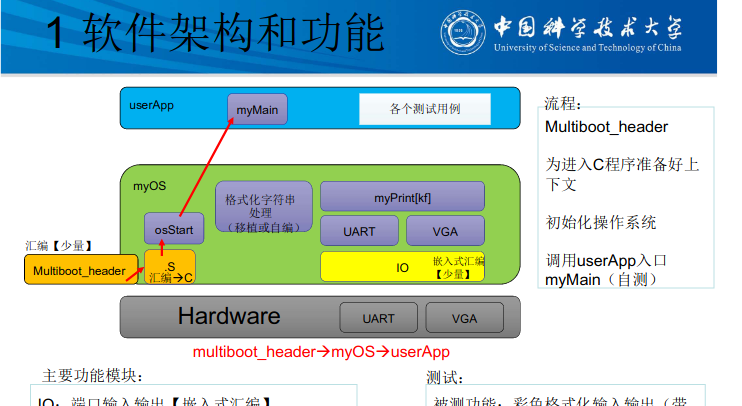
**实验报告**

李卓pb19000064

1. **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）

1. **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

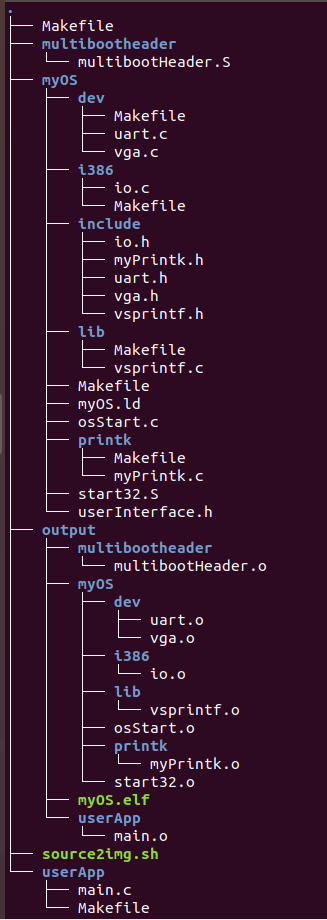
2．uart.c向串口输出字符串

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

1. **源代码组织说明**



1. **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

1. **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

1. **运行和运行结果说明**

直接运行source2img.sh

