**題目一**

將檔案分類放好後，我開始改寫Makefile

因為把這個資料夾獨立拉出來，所以把之前include的上層common內容直接貼過來

CROSS\_COMPILE = riscv64-unknown-elf-

CFLAGS = -nostdlib -fno-builtin -march=rv32ima -mabi=ilp32 -g

QEMU = qemu-system-riscv32

QFLAGS = -nographic -smp 1 -machine virt -bios none

GDB = riscv64-unknown-elf-gdb

CC = ${CROSS\_COMPILE}gcc

OBJCOPY = ${CROSS\_COMPILE}objcopy

OBJDUMP = ${CROSS\_COMPILE}objdump

把最後的編譯目標修改為objs資料夾底下的.o檔

OBJS = \

    ./objs/start.o \

    ./objs/mem.o \

    ./objs/entry.o \

    ./objs/kernel.o \

    ./objs/uart.o \

    ./objs/printf.o \

    ./objs/page.o \

    ./objs/sched.o \

分別設定各個資料夾下的目標與依賴關係

os.elf: ${OBJS}

    ${CC} ${CFLAGS} -T os.ld -o os.elf $^

    ${OBJCOPY} -O binary os.elf os.bin

./objs/%.o : ./startup/%.s

    $(CC) $(CFLAGS) -c -o $@ $<

./objs/%.o : ./asm/%.s

    $(CC) $(CFLAGS) -c -o $@ $<

./objs/%.o : ./kernel/%.c

    $(CC) $(CFLAGS) -c -o $@ $<

./objs/%.o : ./lib/%.c

    $(CC) $(CFLAGS) -c -o $@ $<

將objs/\*.o 加到clean 偽目標中

clean:

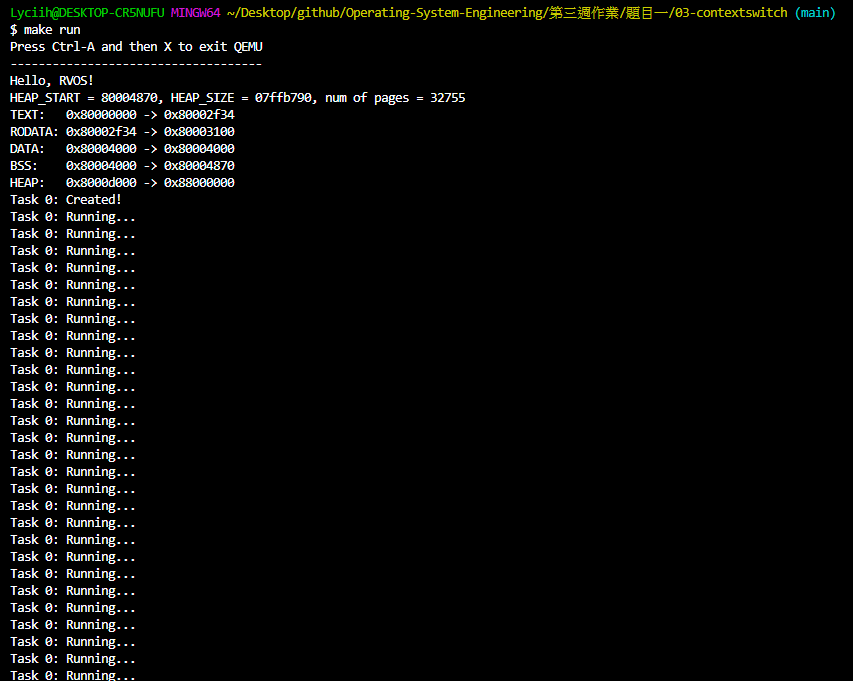
    rm -rf \*.o \*.bin \*.elf objs/\*.o

編譯看看



成功

測試看看是否能運作

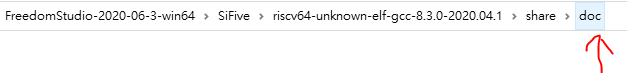
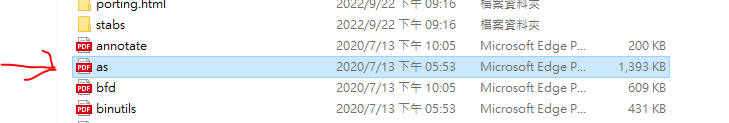


順利運行

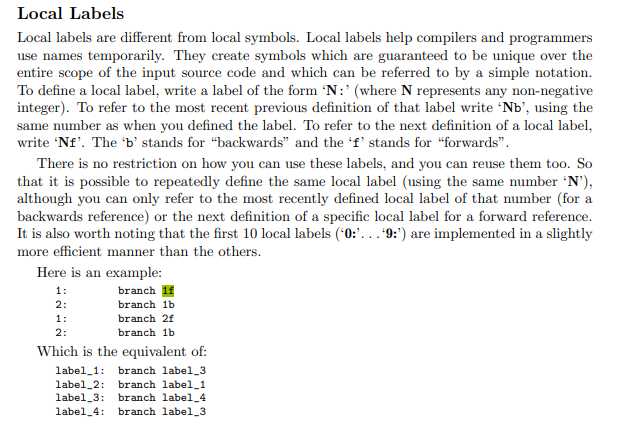
**題目二**

1. **請說明 beqz t6, 1f 指令會做何事?**

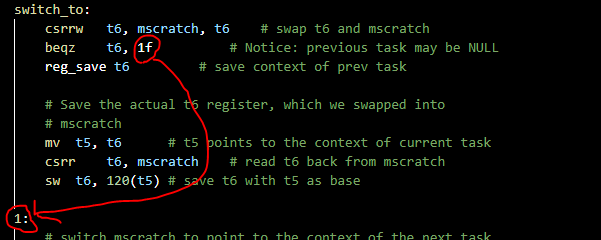
我檢查了教材中的各個檔案，最後找到一份名為 as.pdf 的文件，這是一份介紹組譯器指令的說明書

在裡面我找到了以下敘述



簡單來說，這是一個可以在label之間跳躍的指令，數字代表label名，< f > 代表forward ，< b >代表 back，所以beqz t6, 1f 的意思就是判斷如果t6為0，則往前跳到下一個名為1:的標籤



1. **請修改 entry.S 裡的switch\_to 函數，修改成 kernel start後switch到第一個Task用  
   並在entry.S裡增加一個 sys\_switch函數，主要讓kernel start以後 contextSwitch用**

