1. 如果使用gcc编译汇编文件的话, 使用main符号作为入口(gcc -o a.out a.S)
2. 如果使用as和ld命令编译汇编的话, 使用_start符号作为入口(as test.S -o test.o; ld test.o -o a.out)

以上生成两个文件都是ELF格式的可执行文件, 只不过第一个会自动添加一些环境的初始化代码. 对于ELF文件可以通过objcopy命令来得到干净的二进制文件. 命令如下:

objcopy -S -O binary -j .text a.out a.bin (-S : 去掉调试符号, -O binary 生成原始二进制文件, -j .text 只保留.text段)

下面是测试的Makefile和test.S文件

```
#========================== Makefile Start  ==========================
.PHONY:all clean obj_clean

target := test
main_source := test.S
start_source := test.S

main_target := $(target).main
start_target := $(target).start

main_objs := $(patsubst %.S, %.o, $(main_source))
start_objs := $(patsubst %.S, %.o, $(start_source))

main: obj_clean $(main_target)
start: obj_clean $(start_target)

$(main_target) : CFLAGS += -DGNUGCC
$(main_target) : $(main_objs)
    gcc -o $@ $<

$(start_target) : $(start_objs)
    ld -o $@ $<

obj_clean:
```

```
    rm -rf *.o

clean: obj_clean
    rm -rf $(main_target) $(start_target)

%.o : %.S
    gcc -c $(CFLAGS) $< -o $@

#========================== Makefile End  ==========================



#========================== test.S Start  ==========================
.text
.globl main

#ifndef GNUGCC
    .globl _start
    _start:
#endif

main:
    .ascii  "hello wolrd"
    movl $2, %eax
    int $0x80
#========================== test.S Start  ==========================
```