Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

# Отчёт по лабораторной работе № 4

Дисциплина: Низкоуровневое программирование

Тема: Раздельная компиляция.

Вариант:14

Выполнил студент гр. 3530901/00002 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Чэнь Аосюань

(подпись)

Принял преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д.С.Степанов

(подпись)

“ ” 2021 г.

Санкт-Петербург

2021

остановка задачи

1. Изучить методические материалы, опубликованные на сайте курса.

2. Установить пакет средств разработки “SiFive GNU Embedded Toolchain” для RISC-V.

3. На языке C разработать функцию, реализующую определенную вариантом задания функциональность. Поместить определение функции в отдельный исходный файл, оформить заголовочный файл. Разработать тестовую программу на языке C.

4. Собрать программу «по шагам». Проанализировать выход препроцессора и компилятора. Проанализировать состав и содержимое секций, таблицы символов, таблицы перемещений и отладочную информацию, содержащуюся в объектных файлах и исполняеммом файле.

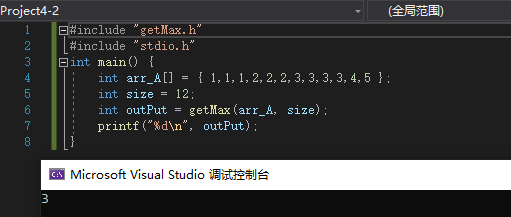
5. Выделить разработанную функцию в статическую библиотеку. Разработать make-файлы для сборки библиотеки и использующей ее тестовой программы. Проанализировать ход сборки библиотеки и программы, созданные файлы зависимостей.

## Задача

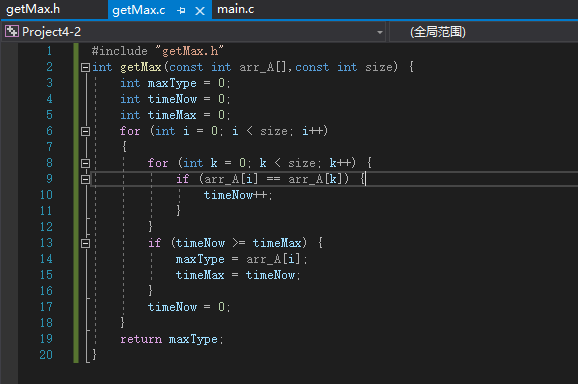
Определение наиболее часто встречающегося в массиве значения

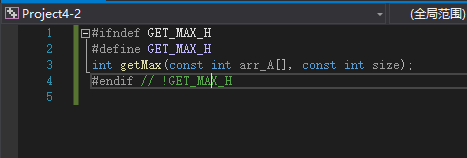
# Программа на язык с

С помошью Visual Studio2019 разработал тествую программу(Рис 1) подпрограмму(Рис 2) и заголовочный файл подпрограммы(Рис 3)



(Рис 1 тествая программа и вывод 3)

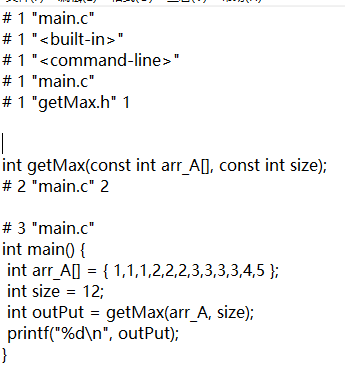
(Рис2 Подпрограмма на язык с)



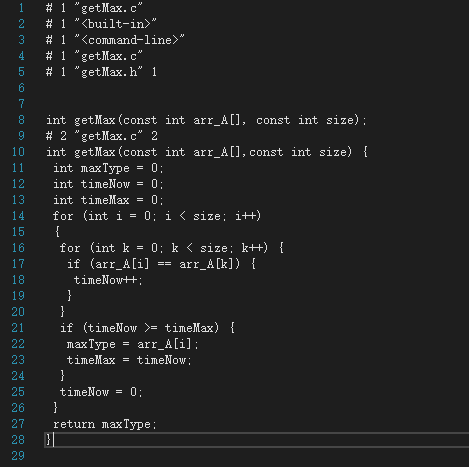
(Рис 3 Заголовочный файл подпрограммы)

# Сборка простейшей программы «по шагам»

## Первый шаг – Препроцессирование

riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -v -E main.c -o main.i

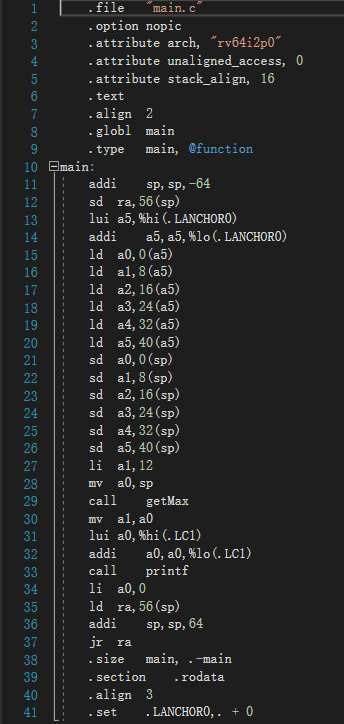
(Рис 3 График main.i)

riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -v -E getMax.c -o getMax.i

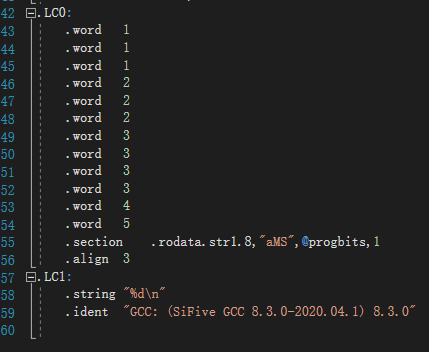
(Рис 4 График getMax.i)

Видно что в результате препроцессирования . Код мало отличается от исходного кода.Кроме некоторый симпол как #1 #2 который показал используются для передачи информации об исходном тексте из препроцессора в компилятор.

## Второй шаг – Компиляция

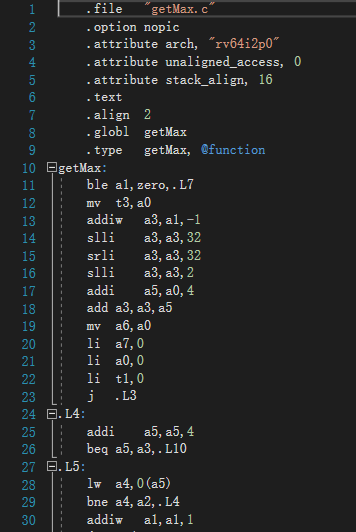


(Рис5 график main.s 1-41)

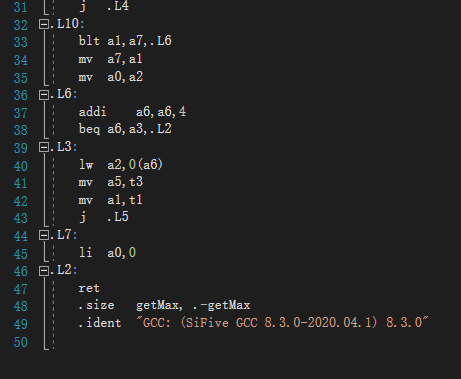


(Рис6 График main.s 42-60)

Комант используемый: riscv64-unknown-elf-gcc.exe -march=rv64i -mabi=lp64 -O1 -S main.i -o main.s



(Рис7 график getMax.s 1-30)



(Рис8 график getMax.s 31-50)

Комант используемый: riscv64-unknown-elf-gcc.exe -march=rv64i -mabi=lp64 -O1 -S getMax.i -o getMax.s

Видно что переходил на код для Risc-V и тоже одиноговые функции как в исходных кодах.

## Тритий шаг – Ассемблирование

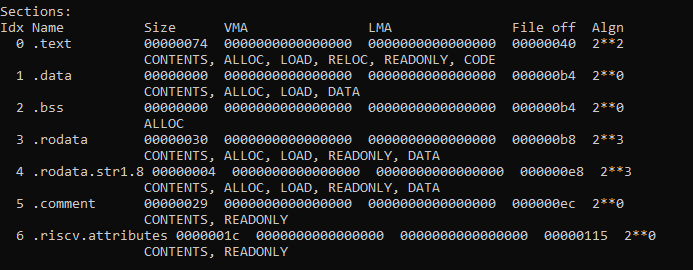
Для ассемблировании файла main.s и getMax.s

Комант используемый: riscv64-unknown-elf-gcc.exe -march=rv64i -mabi=lp64 -c main.s -o main.o

riscv64-unknown-elf-gcc.exe -march=rv64i -mabi=lp64 -c getMax.s -o getMax.o

Для получении заголовки секциий

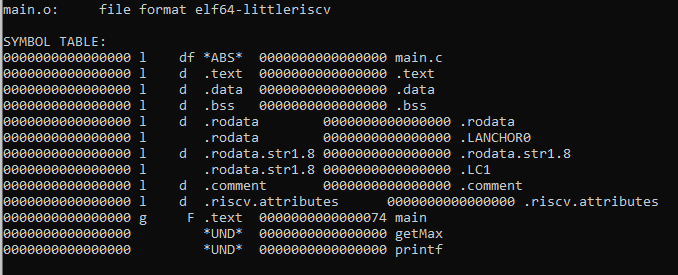
используем: riscv64-unknown-elf-objdump -h main.o



(Рис 9 заголовки секции main)

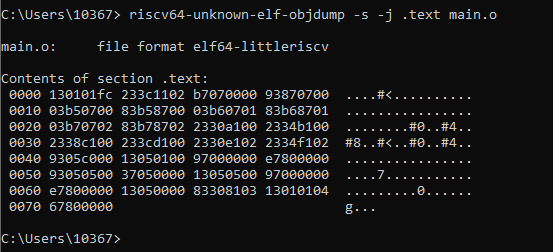
Для симполов файла

используем: riscv64-unknown-elf-objdump -t main.o



(Рис 10 Симполов файла main)

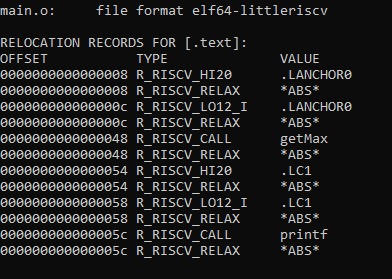
Для содержимоя секции “.text”

используем: riscv64-unknown-elf-objdump –s –j .text main.o

(Рис11 Содержание секции main)

Для таблице переменных

используем: riscv64-unknown-elf-objdump -r main.o



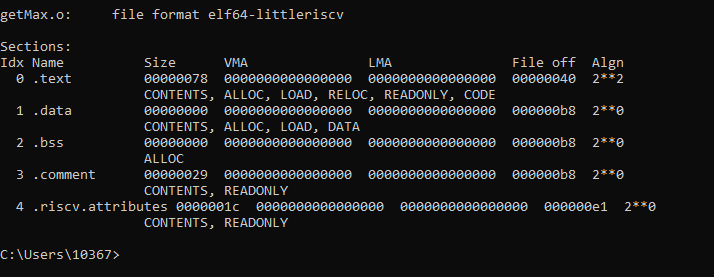
(Рис 12 Таблица переменных main)

Для getMax.o

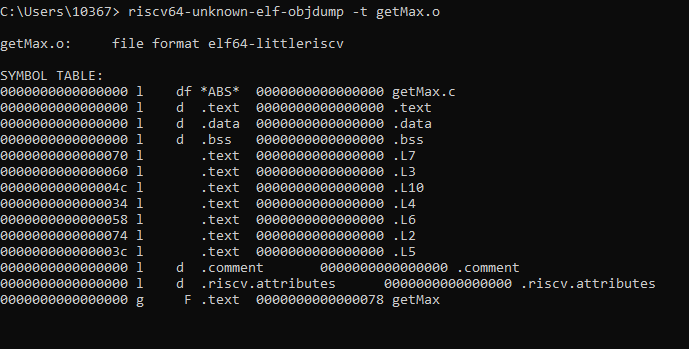
Используем: riscv64-unknown-elf-objdump -h getMax.o

riscv64-unknown-elf-objdump -t getMax.o

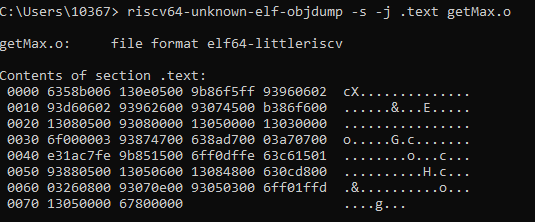
riscv64-unknown-elf-objdump –s –j .text getMax.o

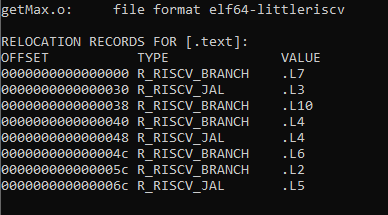
riscv64-unknown-elf-objdump -r getMax.o

(Рис 13 Заголовки секции getMax)



(Рис 14 Симполов файла getMax)





(Рис15 Содержание секции getMax)

(Рис 17Таблица переменных getMax)

В графике

.text – секция кода, в которой содержатся коды инструкций (название секции

обусловлено историческими причинами);

.data – секция инициализированных данных;

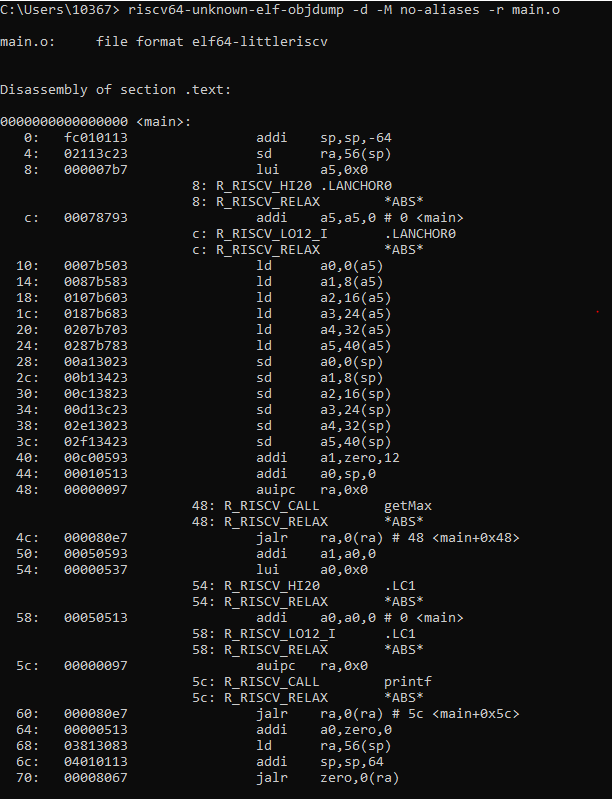
.bss – секция данных, инициализированных нулями (название секции также обусловлено

историческими причинами);

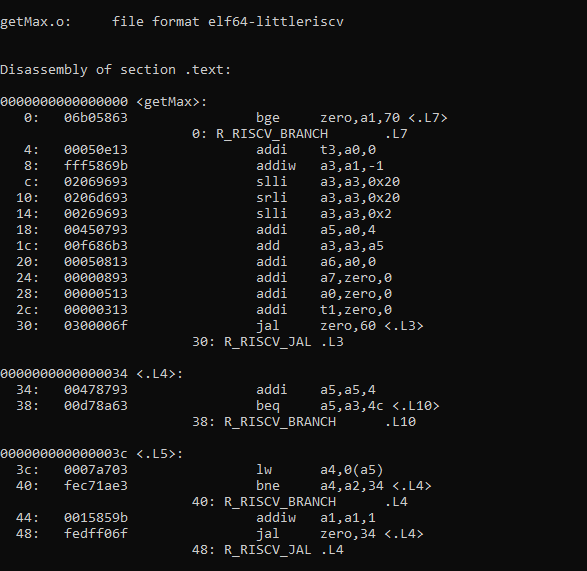
.comment – секция данных о версиях размером 12 байт.

UND = getMax

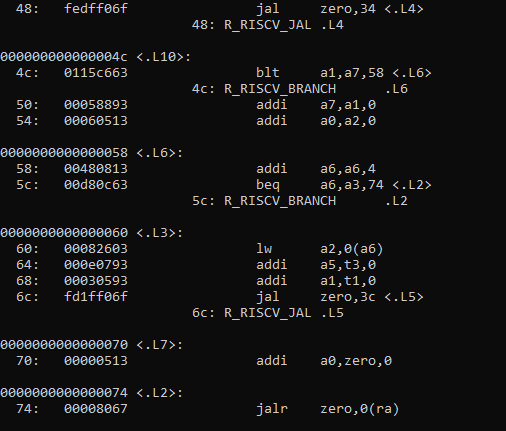
Сейчас используем riscv64-unknown-elf-objdump -d -M no-aliases -r main.o и riscv64-unknown-elf-objdump -d -M no-aliases -r getMax.o для Дизассемблирования



(Рис18 Дизассемблирование main)



(Рис19 Дизассемблирование getMax1-48)

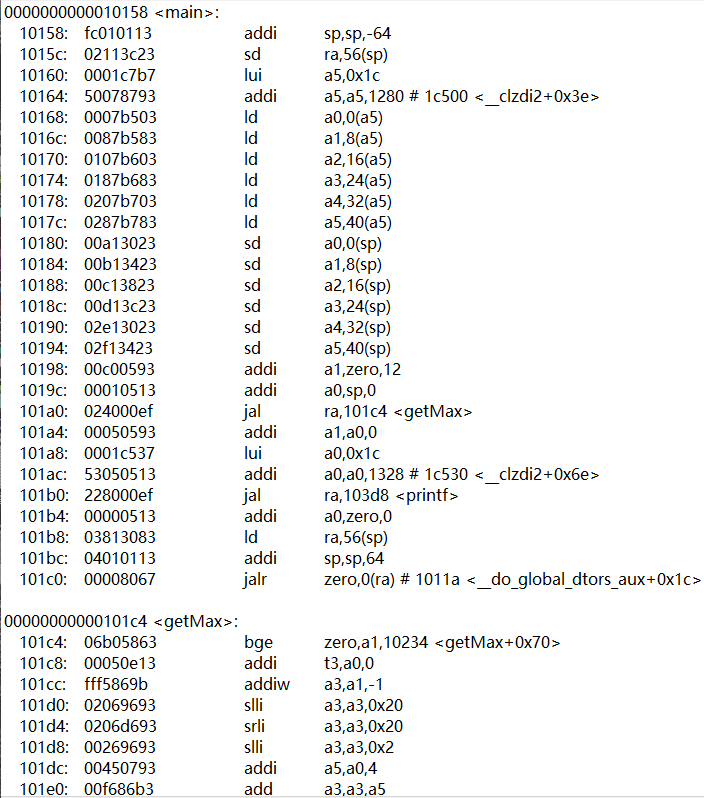


(Рис20 Дизассемблирование getMax48-74)

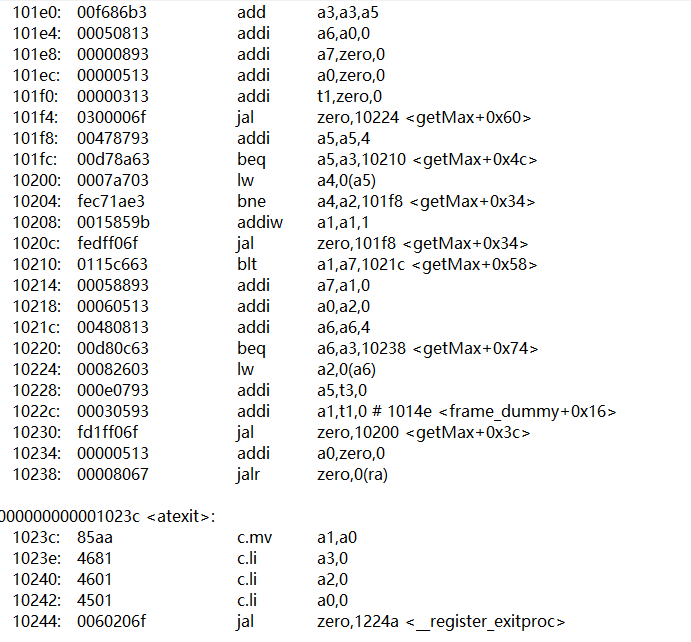
Похоже с кодом который получил в прошлом шаге как код для Risc-v

## Чтвертый шаг – Компоновка

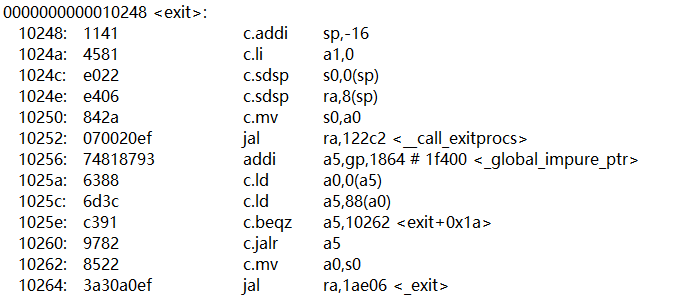
По команте riscv64-unknown-elf-gcc.exe -march=rv64iac -mabi=lp64 -v main.o getMax.o и riscv64-unknown-elf-objdump –j .text –d –M no-aliases a.out >a.ds получаем a.out и a.ds



(Рис21 Фрагмент код a.out)



(Рис22 Фрагмент код a.out)



(Рис23 Фрагмент код a.out)

# Создать библиотеки

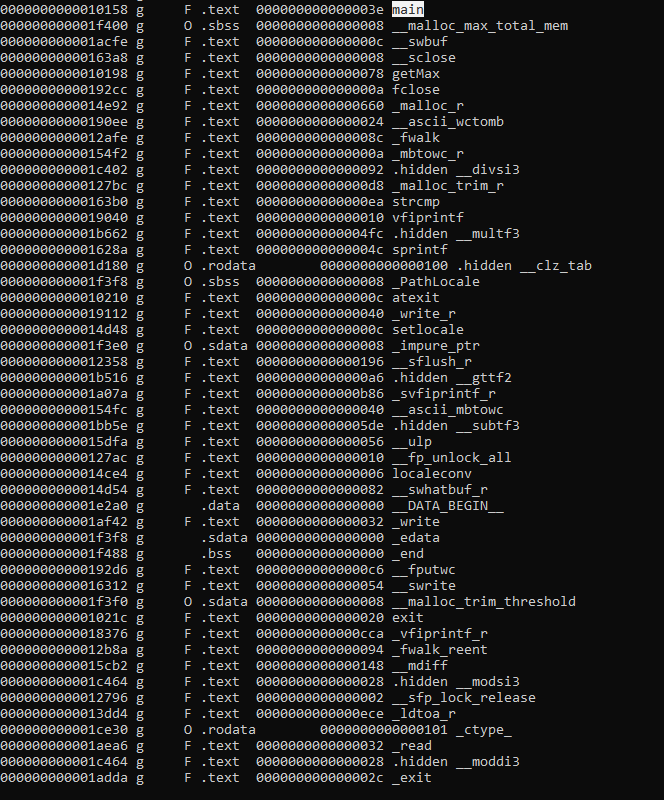
Добовляеь getMax.o к lib

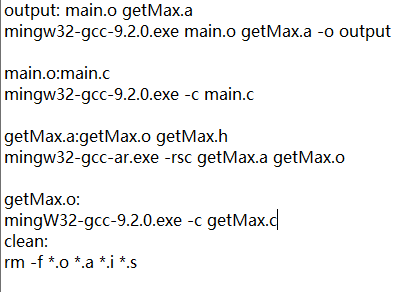
Используем riscv64-unknown-ef-ar -rsc lib.a getMax.o

И main тоже

Используем riscv64-unknown-elf-gcc.exe -march=rv64iac -mabi=lp64 -O1 --save-temps main.c lib.a

Получим таблицу riscv64-unknown-elf-objdump.exe -t a.out





( Рис24 статиескиий makefile)

# Выход лаба

В данном работе ислледовали riscv64-unknown-elf-gc для получение препроцессирования компиляции асемблирования компоновки и makefile с кодом с