

Санкт-Петербургский политехнический университет Петра Великого

Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе

Дисциплина: Низкоуровневое программирование

Тема: Раздельная компиляция

Выполнил студент гр. 3530901/00002 _____Сергеева Е.О

(подпись)

Преподаватель _____ Степанов Д.С

(подпись)

“ ____ ” _____ 2021 г.

Санкт-Петербург

2021

Вариант: 15. Слияние отсортированных массивов

Исходное задание:

Написать функцию на языке C.

Произвести отдельную компиляцию. Проанализировать выход препроцессора и компилятора. Проанализировать состав и содержимое секций, таблицы символов, таблицы перемещений и отладочную информацию, содержащуюся в объектных файлах и исполняемом файле.

Выделить разработанную функцию в статическую библиотеку.

Решение задачи:

- 1) В цикле сравниваем элементы из массивов A и B, меньший элемент добавляем в выходной массив.
- 2) Если мы добавили элемент из массива A, то счетчик элемента массива A увеличивается на 1
- 3) Если мы добавили элемент из массива B, то счетчик элемента массива B увеличивается на 1
- 4) При добавлении элемента в выходной массив, счетчик выходного массива увеличивается на 1
- 5) Выход из цикла происходит, когда счетчик выходного массива достигает суммы двух массивов

Часть 1. Программа на C

```
#ifndef MERGE_H
#define MERGE_H
int * merge(int const a[], int const b[], int a_size, int b_size);
#endif
```

Рис. 1. Заголовочный файл merge.h

```

int * merge(int const a[], int const b[], int a_size, int b_size){
    int *c = calloc( count: a_size + b_size, size: sizeof(int));
    int i = 0;
    int j = 0;
    int k = 0;
    while(k < a_size + b_size) {
        if(a[i] < b[j]) {
            c[k] = a[i];
            k++;
            i++;
        } else {
            c[k] = b[j];
            k++;
            j++;
        }
    }
    return c;
}

```

Рис. 2. Функция merge.c

Функция `calloc()` возвращает указатель на выделенную память. Размер выделенной памяти равен величине `num*size`, где `size` задается в байтах. Это означает, что функция `calloc()` выделяет достаточно памяти для массива из `num` объектов каждый размером `size` байт.

```

int main() {
    int a[] = { [0]: 1, [1]: 2, [2]: 3, [3]: 4, [4]: 5};
    int b[] = { [0]: 6, [1]: 7, [2]: 8, [3]: 9, [4]: 10};
    int a_size = sizeof(a) / sizeof (a[0]);
    int b_size = sizeof(b) / sizeof (b[0]);
    int *c = merge(a, b, a_size, b_size);
    for(int i = 0; i < a_size; i++) {
        printf("%i", a[i]);
    }
    printf("\n");
    for(int i = 0; i < b_size; i++) {
        printf("%i", b[i]);
    }
    printf("\n");
    for(int i = 0; i < b_size + a_size; i++) {
        printf("%i", c[i]);
    }
    return 0;
}

```

Рис. 3 Функция main.c

```

12345
678910
12345678910

```

Рис. 4.1. Результаты

```

24689
135710
12345678910
Process finished with exit code 0

```

Рис. 4.2. Результаты

Часть 2. Сборка программы

Для сборки программы выполним команду `riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -v main.c > log 2>&1`

В результате получили следующие файлы:

`main.i` – текстовый файл, содержащий результат препроцессирования файла “`main.c`” – единица трансляции (translation unit) в терминах стандарта языка C;

`main.s` – текстовый файл, содержащий код на языке ассемблера, сгенерированный компилятором в результате обработки файла “`main.i`”.

`main.o` – объектный файл (object file), сгенерированный ассемблером в результате обработки файла “`main.s`”;

`a.out` – исполняемый файл (executable file), сгенерированный компоновщиком в результате обработки файла “`main.o`”, а также других объектных файлов и файлов библиотек, входящих в состав пакета средств разработки.

`log` – текстовый файл, содержащий сообщения компилятора, ассемблера и компоновщика, а также выполняемые команды и дополнительную информацию;

1) Файл препроцессора

```
# 3 "C:\\Users\\Goose\\Desktop\\Kora\\main.c" 2
# 1 "C:\\Users\\Goose\\Desktop\\Kora\\merge.h" 1

# 4 "C:\\Users\\Goose\\Desktop\\Kora\\merge.h"
int * merge(int const a[], int const b[], int a_size, int b_size);
# 4 "C:\\Users\\Goose\\Desktop\\Kora\\main.c" 2

int main() {
    int a[] = { [0]: 1, [1]: 2, [2]: 3, [3]: 4, [4]: 5};
    int b[] = { [0]: 6, [1]: 7, [2]: 8, [3]: 9, [4]: 10};
    int a_size = sizeof(a) / sizeof (a[0]);
    int b_size = sizeof(b) / sizeof (b[0]);
    int *c = merge(a, b, a_size, b_size);
    for(int i = 0; i < a_size; i++) {
        printf("%i", a[i]);
    }
    printf("\n");
    for(int i = 0; i < b_size; i++) {
        printf("%i", b[i]);
    }
    printf("\n");
    for(int i = 0; i < b_size + a_size; i++) {
        printf("%i", c[i]);
    }
    return 0;
}
```

Рис. 4. Файл main.i

Компилируем препроцессорный файл в asm RISC-V

Код asm RISC-V main.c

```
| .file    "main.c"
    .option nopic
    .attribute arch, "rv64i2p0_a2p0_c2p0"
    .attribute unaligned_access, 0
    .attribute stack_align, 16
    .text
    .section    .rodata.str1.8,"aMS",@progbits,1
    .align    3
.LC2:
    .string "%i"
    .text
    .align    1
    .globl    main
    .type     main, @function
main:
    addi      sp,sp,-96
    sd        ra,88(sp)
    sd        s0,80(sp)
    sd        s1,72(sp)
    sd        s2,64(sp)
    sd        s3,56(sp)
    lui       a5,%hi(.ANCHOR0)
    addi      a5,a5,%lo(.ANCHOR0)
    ld        a4,0(a5)
    sd        a4,24(sp)
    ld        a4,8(a5)
    sd        a4,32(sp)
    lw        a4,16(a5)
    sw        a4,40(sp)
    ld        a4,24(a5)
```

```

    ld  a4,24(a5)
    sd  a4,0(sp)
    ld  a4,32(a5)
    sd  a4,8(sp)
    lw  a5,40(a5)
    sw  a5,16(sp)
    li  a3,5
    li  a2,5
    mv  a1,sp
    addi a0,sp,24
    call merge
    mv  s1,a0
    addi s0,sp,24
    addi s3,sp,44
    lui s2,%hi(.LC2)
.L2:
    lw  a1,0(s0)
    addi a0,s2,%lo(.LC2)
    call printf
    addi s0,s0,4
    bne s0,s3,.L2
    li  a0,10
    call putchar
    mv  s0,sp
    addi s3,sp,20
    lui s2,%hi(.LC2)
.L3:
    lw  a1,0(s0)
    addi a0,s2,%lo(.LC2)
    call printf

```



```

    call    printf
    addi    s0,s0,4
    bne s0,s3,.L3
    li      a0,10
    call    putchar
    mv      s0,s1
    addi    s1,s1,40
    lui s2,%hi(.LC2)
.L4:
    lw      a1,0(s0)
    addi    a0,s2,%lo(.LC2)
    call    printf
    addi    s0,s0,4
    bne s0,s1,.L4
    li      a0,0
    ld      ra,88(sp)
    ld      s0,80(sp)
    ld      s1,72(sp)
    ld      s2,64(sp)
    ld      s3,56(sp)
    addi    sp,sp,96
    jr      ra
.size      main, .-main
.section   .rodata
.align    3
.set      .LANCHOR0,. + 0
.LC0:
.word     1
.word     2
.word     3

```

```
ld    s0,80(sp)
ld    s1,72(sp)
ld    s2,64(sp)
ld    s3,56(sp)
addi   sp,sp,96
jr     ra
.size   main, .-main
.section .rodata
.align  3
.set    .LANCHOR0,. + 0
.LC0:
.word   1
.word   2
.word   3
.word   4
.word   5
.zero   4
.LC1:
.word   6
.word   7
.word   8
.word   9
.word   10
.ident  "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"
```

Код на asm RISC-V для merge.c

```
.file "merge.c"
.option nopic
.attribute arch, "rv64i2p0_a2p0_c2p0"
.attribute unaligned_access, 0
.attribute stack_align, 16
.text
.align 1
globl merge
.type merge, @function
merge:
    addi    sp, sp, -32
    sd     ra, 24(sp)
    sd     s0, 16(sp)
    sd     s1, 8(sp)
    sd     s2, 0(sp)
    mv     s1, a0
    mv     s2, a1
    addw    s0, a2, a3
    li     a1, 4
    mv     a0, s0
    call   calloc
    ble    s0, zero, .L1
    mv     a6, a0
    li     a4, 1
    li     a3, 0
    li     t1, 0
    j      .L5
.L3:
    sw     a5, 0(a6)
    sxt.w   a5, a4
```

```

    sxt.w    a5,a4
    addiw    a3,a3,1
.L4:
    addiw    a4,a4,1
    addi     a6,a6,4
    ble      s0,a5,.L1
.L5:
    slli     a5,t1,2
    add      a5,s1,a5
    lw       a7,0(a5)
    slli     a5,a3,2
    add      a5,s2,a5
    lw       a5,0(a5)
    bge      a7,a5,.L3
    sw       a7,0(a6)
    sxt.w    a5,a4
    addiw    t1,t1,1
    j        .L4
.L1:
    ld       ra,24(sp)
    ld       s0,16(sp)
    ld       s1,8(sp)
    ld       s2,0(sp)
    addi     sp,sp,32
    jr      ra
    .size    merge, .-merge
    .ident   "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"

```

2) Результаты ассемблирования (Файлы main.o merge.o)

Sections:						
Idx	Name	Size	VMA	LMA	File off	Algn
0	.text	000000bc	0000000000000000	0000000000000000	00000040	2**1
	CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE					
1	.data	00000000	0000000000000000	0000000000000000	000000fc	2**0
	CONTENTS, ALLOC, LOAD, DATA					
2	.bss	00000000	0000000000000000	0000000000000000	000000fc	2**0
	ALLOC					
3	.rodata.str1.8	00000003	0000000000000000	0000000000000000	00000100	2**3
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
4	.rodata	0000002c	0000000000000000	0000000000000000	00000108	2**3
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
5	.comment	00000031	0000000000000000	0000000000000000	00000134	2**0
	CONTENTS, READONLY					
6	.riscv.attributes	00000026	0000000000000000	0000000000000000	00000165	2**0
	CONTENTS, READONLY					

Рис. 5. Ассемблирование main.c

Sections:						
Idx	Name	Size	VMA	LMA	File off	Algn
0	.text	0000006c	0000000000000000	0000000000000000	00000040	2**1
	CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE					
1	.data	00000000	0000000000000000	0000000000000000	000000ac	2**0
	CONTENTS, ALLOC, LOAD, DATA					
2	.bss	00000000	0000000000000000	0000000000000000	000000ac	2**0
	ALLOC					
3	.comment	00000031	0000000000000000	0000000000000000	000000ac	2**0
	CONTENTS, READONLY					
4	.riscv.attributes	00000026	0000000000000000	0000000000000000	000000dd	2**0
	CONTENTS, READONLY					

Рис. 6. Ассемблирование merge.c

- .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

3) Результат дизассемблирования

Disassembly of section .text:

0000000000000000 <main>:

0:	711d	c.addi16sp	sp, -96
2:	ec86	c.sdsp	ra, 88(sp)
4:	e8a2	c.sdsp	s0, 80(sp)
6:	e4a6	c.sdsp	s1, 72(sp)
8:	e0ca	c.sdsp	s2, 64(sp)
a:	fc4e	c.sdsp	s3, 56(sp)
c:	000007b7	lui	a5, 0x0
10:	00078793	addi	a5, a5, 0 # 0 <main>
14:	6398	c.ld	a4, 0(a5)
16:	ec3a	c.sdsp	a4, 24(sp)
18:	6798	c.ld	a4, 8(a5)
1a:	f03a	c.sdsp	a4, 32(sp)
1c:	4b98	c.lw	a4, 16(a5)
1e:	d43a	c.swsp	a4, 40(sp)
20:	6f98	c.ld	a4, 24(a5)
22:	e03a	c.sdsp	a4, 0(sp)
24:	7398	c.ld	a4, 32(a5)
26:	e43a	c.sdsp	a4, 8(sp)
28:	579c	c.lw	a5, 40(a5)
2a:	c83e	c.swsp	a5, 16(sp)
2c:	4695	c.li	a3, 5
2e:	4615	c.li	a2, 5
30:	858a	c.mv	a1, sp
32:	0828	c.addi4spn	a0, sp, 24
34:	00000097	auipc	ra, 0x0
38:	000080e7	jalr	ra, 0(ra) # 34 <main+0x34>
3c:	84aa	c.mv	s1, a0
3e:	0820	c.addi4spn	s0, sp, 24
40:	02c10993	addi	s3, sp, 44
44:	00000937	lui	s2, 0x0

```

0000000000000048 <.L2>:
 48: 400c          c.lw      a1,0(s0)
 4a: 00090513     addi      a0,s2,0 # 0 <main>
 4e: 00000097     auipc     ra,0x0
 52: 000080e7     jalr      ra,0(ra) # 4e <.L2+0x6>
 56: 0411         c.addi     s0,4
 58: ff3418e3     bne       s0,s3,48 <.L2>
 5c: 4529         c.li      a0,10
 5e: 00000097     auipc     ra,0x0
 62: 000080e7     jalr      ra,0(ra) # 5e <.L2+0x16>
 66: 840a         c.mv      s0,sp
 68: 01410993     addi      s3,sp,20
 6c: 00000937     lui       s2,0x0

0000000000000070 <.L3>:
 70: 400c          c.lw      a1,0(s0)
 72: 00090513     addi      a0,s2,0 # 0 <main>
 76: 00000097     auipc     ra,0x0
 7a: 000080e7     jalr      ra,0(ra) # 76 <.L3+0x6>
 7e: 0411         c.addi     s0,4
 80: ff3418e3     bne       s0,s3,70 <.L3>
 84: 4529         c.li      a0,10
 86: 00000097     auipc     ra,0x0
 8a: 000080e7     jalr      ra,0(ra) # 86 <.L3+0x16>
 8e: 8426         c.mv      s0,s1
 90: 02848493     addi      s1,s1,40
 94: 00000937     lui       s2,0x0

0000000000000098 <.L4>:
 98: 400c          c.lw      a1,0(s0)
 9a: 00090513     addi      a0,s2,0 # 0 <main>
 9e: 00000097     auipc     ra,0x0
 a2: 000080e7     jalr      ra,0(ra) # 9e <.L4+0x6>
 a6: 0411         c.addi     s0,4
 a8: fe9418e3     bne       s0,s1,98 <.L4>
 ac: 4501         c.li      a0,0
 ae: 60e6         c.ldsp     ra,88(sp)
 b0: 6446         c.ldsp     s0,80(sp)
 b2: 64a6         c.ldsp     s1,72(sp)
 b4: 6906         c.ldsp     s2,64(sp)
 b6: 79e2         c.ldsp     s3,56(sp)
 b8: 6125         c.addi16sp sp,96
 ba: 8082         c.jr      ra

```

Рис. 7. Дизассемблирование

4) Таблица символов

```
SYMBOL TABLE:
0000000000000000 1    df *ABS* 0000000000000000 main.c
0000000000000000 1    d  .text 0000000000000000 .text
0000000000000000 1    d  .data 0000000000000000 .data
0000000000000000 1    d  .bss 0000000000000000 .bss
0000000000000000 1    d  .rodata.str1.8 0000000000000000 .rodata.str1.8
0000000000000000 1    d  .rodata 0000000000000000 .rodata
0000000000000000 1    .rodata 0000000000000000 .LANCHOR0
0000000000000000 1    .rodata.str1.8 0000000000000000 .LC2
0000000000000048 1    .text 0000000000000000 .L2
0000000000000070 1    .text 0000000000000000 .L3
0000000000000098 1    .text 0000000000000000 .L4
0000000000000000 1    d  .comment 0000000000000000 .comment
0000000000000000 1    d  .riscv.attributes 0000000000000000 .riscv.attributes
0000000000000000 g    F  .text 00000000000000bc main
0000000000000000      *UND* 0000000000000000 merge
0000000000000000      *UND* 0000000000000000 printf
0000000000000000      *UND* 0000000000000000 putchar
```

Рис. 8. Таблица символов main. о

5) Таблица перемещений

```
RELOCATION RECORDS FOR [.text]:
OFFSET          TYPE          VALUE
000000000000000c R_RISCV_HI20    .LANCHOR0
000000000000000c R_RISCV_RELAX   *ABS*
0000000000000010 R_RISCV_LO12_I  .LANCHOR0
0000000000000010 R_RISCV_RELAX   *ABS*
0000000000000034 R_RISCV_CALL    merge
0000000000000034 R_RISCV_RELAX   *ABS*
0000000000000044 R_RISCV_HI20    .LC2
0000000000000044 R_RISCV_RELAX   *ABS*
000000000000004a R_RISCV_LO12_I  .LC2
000000000000004a R_RISCV_RELAX   *ABS*
000000000000004e R_RISCV_CALL    printf
000000000000004e R_RISCV_RELAX   *ABS*
000000000000005e R_RISCV_CALL    putchar
000000000000005e R_RISCV_RELAX   *ABS*
000000000000006c R_RISCV_HI20    .LC2
000000000000006c R_RISCV_RELAX   *ABS*
0000000000000072 R_RISCV_LO12_I  .LC2
0000000000000072 R_RISCV_RELAX   *ABS*
0000000000000076 R_RISCV_CALL    printf
0000000000000076 R_RISCV_RELAX   *ABS*
0000000000000086 R_RISCV_CALL    putchar
0000000000000086 R_RISCV_RELAX   *ABS*
0000000000000094 R_RISCV_HI20    .LC2
0000000000000094 R_RISCV_RELAX   *ABS*
000000000000009a R_RISCV_LO12_I  .LC2
000000000000009a R_RISCV_RELAX   *ABS*
000000000000009e R_RISCV_CALL    printf
000000000000009e R_RISCV_RELAX   *ABS*
0000000000000058 R_RISCV_BRANCH  .L2
0000000000000080 R_RISCV_BRANCH  .L3
00000000000000a8 R_RISCV_BRANCH  .L4
```

Рис. 9. Таблица перемещений merge. о

Компоновка

Main.out

```
0000000000010156 <main>:
10156: 711d          c.addi16sp sp,-96
10158: ec86          c.sdsp ra,88(sp)
1015a: e8a2          c.sdsp s0,80(sp)
1015c: e4a6          c.sdsp s1,72(sp)
1015e: e0ca          c.sdsp s2,64(sp)
10160: fc4e          c.sdsp s3,56(sp)
10162: 67f5          c.lui a5,0x1d
10164: c8878793      addi a5,a5,-888 # 1cc88 <__clzdi2+0x3a>
10168: 6398          c.ld a4,0(a5)
1016a: ec3a          c.sdsp a4,24(sp)
1016c: 6798          c.ld a4,8(a5)
1016e: f03a          c.sdsp a4,32(sp)
10170: 4b98          c.lw a4,16(a5)
10172: d43a          c.swsp a4,40(sp)
10174: 6f98          c.ld a4,24(a5)
10176: e03a          c.sdsp a4,0(sp)
10178: 7398          c.ld a4,32(a5)
1017a: e43a          c.sdsp a4,8(sp)
1017c: 579c          c.lw a5,40(a5)
1017e: c83e          c.swsp a5,16(sp)
10180: 4695          c.li a3,5
10182: 4615          c.li a2,5
```

```
10182: 4615          c.li a2,5
10184: 858a          c.mv a1,sp
10186: 0828          c.addi4spn a0,sp,24
10188: 06a000ef      jal ra,101f2 <merge>
1018c: 84aa          c.mv s1,a0
1018e: 0820          c.addi4spn s0,sp,24
10190: 02c10993      addi s3,sp,44
10194: 6975          c.lui s2,0x1d
10196: 400c          c.lw a1,0(s0)
10198: c8090513      addi a0,s2,-896 # 1cc80 <__clzdi2+0x32>
1019c: 163000ef      jal ra,10afe <printf>
101a0: 0411          c.addi s0,4
101a2: ff341ae3      bne s0,s3,10196 <main+0x40>
101a6: 4529          c.li a0,10
101a8: 187000ef      jal ra,10b2e <putchar>
101ac: 840a          c.mv s0,sp
101ae: 01410993      addi s3,sp,20
101b2: 6975          c.lui s2,0x1d
101b4: 400c          c.lw a1,0(s0)
101b6: c8090513      addi a0,s2,-896 # 1cc80 <__clzdi2+0x32>
101ba: 145000ef      jal ra,10afe <printf>
101be: 0411          c.addi s0,4
101c0: ff341ae3      bne s0,s3,101b4 <main+0x5e>
101c4: 4529          c.li a0,10
```

```

101c4: 4529          c.li    a0,10
101c6: 169000ef     jal ra,10b2e <putchar>
101ca: 8426          c.mv    s0,s1
101cc: 02848493     addi    s1,s1,40
101d0: 6975          c.lui   s2,0x1d
101d2: 400c          c.lw    a1,0(s0)
101d4: c8090513     addi    a0,s2,-896 # 1cc80 <__clzdi2+0x32>
101d8: 127000ef     jal ra,10afe <printf>
101dc: 0411          c.addi   s0,4
101de: fe941ae3     bne     s0,s1,101d2 <main+0x7c>
101e2: 4501          c.li    a0,0
101e4: 60e6          c.ldsp   ra,88(sp)
101e6: 6446          c.ldsp   s0,80(sp)
101e8: 64a6          c.ldsp   s1,72(sp)
101ea: 6906          c.ldsp   s2,64(sp)
101ec: 79e2          c.ldsp   s3,56(sp)
101ee: 6125          c.addi16sp sp,96
101f0: 8082          c.jr     ra

```

0000000000101f2 <merge>:

```

101f2: 1101          c.addi   sp,-32
101f4: ec06          c.sdsp   ra,24(sp)
101f6: e822          c.sdsp   s0,16(sp)
101f8: e426          c.sdsp   s1,8(sp)
101fa: e04a          c.sdsp   s2,0(sp)
101fc: 84aa          c.mv     s1,a0
101fe: 892e          c.mv     s2,a1
10200: 00d6043b     addw     s0,a2,a3
10204: 4591          c.li     a1,4
10206: 8522          c.mv     a0,s0
10208: 052000ef     jal ra,1025a <calloc>
1020c: 04805163     bge     zero,s0,1024e <merge+0x5c>
10210: 882a          c.mv     a6,a0
10212: 4705          c.li     a4,1
10214: 4681          c.li     a3,0
10216: 4301          c.li     t1,0
10218: a811          c.j      1022c <merge+0x3a>
1021a: 00f82023     sw      a5,0(a6)
1021e: 0007079b     addiw    a5,a4,0
10222: 2685          c.addiw  a3,1
10224: 2705          c.addiw  a4,1
10226: 0811          c.addi   a6,4
10228: 0287d363     bge     a5,s0,1024e <merge+0x5c>

```

```

10228: 0287d363      bge a5,s0,1024e <merge+0x5c>
1022c: 00231793      slli    a5,t1,0x2
10230: 97a6          c.add    a5,s1
10232: 0007a883      lw      a7,0(a5)
10236: 00269793      slli    a5,a3,0x2
1023a: 97ca          c.add    a5,s2
1023c: 439c          c.lw     a5,0(a5)
1023e: fcf8dee3      bge a7,a5,1021a <merge+0x28>
10242: 01182023      sw      a7,0(a6)
10246: 0007079b      addiw   a5,a4,0
1024a: 2305          c.addiw t1,1
1024c: bfe1          c.j 10224 <merge+0x32>
1024e: 60e2          c.ldsp   ra,24(sp)
10250: 6442          c.ldsp   s0,16(sp)
10252: 64a2          c.ldsp   s1,8(sp)
10254: 6902          c.ldsp   s2,0(sp)
10256: 6105          c.addi16sp sp,32
10258: 8082          c.jr     ra

```

Часть 3. Создание статической библиотеки.

```

merge.o:
0000000000000060 t .L1
000000000000002c t .L3
0000000000000036 t .L4
000000000000003e t .L5
                U calloc
0000000000000000 T merge

```

Рис. 10. Содержимое

```

000000000000100b0 1      F .text  0000000000000014 register_fini
0000000000000000 1      df *ABS* 0000000000000000 crtstuff.c
0000000000001ebf8 1      0 .eh_frame 0000000000000000
00000000000010106 1      F .text  0000000000000000 __do_global_dtors_aux
0000000000001fda8 1      0 .bss    0000000000000001 completed.1
0000000000001ec10 1      0 .fini_array 0000000000000000 __do_global_dtors_aux_fini_array_entry
0000000000001013a 1      F .text  0000000000000000 frame_dummy
0000000000001fdb0 1      0 .bss    0000000000000030 object.0
0000000000001ec08 1      0 .init_array 0000000000000000 __frame_dummy_init_array_entry
0000000000000000 1      df *ABS* 0000000000000000 main.c
0000000000000000 1      df *ABS* 0000000000000000 merge.c
0000000000000000 1      df *ABS* 0000000000000000 calloc.c
0000000000000000 1      df *ABS* 0000000000000000 mallocr.c
0000000000000000 1      df *ABS* 0000000000000000 errno.c
0000000000000000 1      df *ABS* 0000000000000000 exit.c
0000000000000000 1      df *ABS* 0000000000000000 impure.c
0000000000001ec20 1      0 .data  0000000000000748 impure_data
0000000000000000 1      df *ABS* 0000000000000000 init.c
0000000000000000 1      df *ABS* 0000000000000000 mallocr.c
0000000000000000 1      df *ABS* 0000000000000000 mlock.c
0000000000000000 1      df *ABS* 0000000000000000 printf.c
0000000000000000 1      df *ABS* 0000000000000000 putchar.c
0000000000000000 1      df *ABS* 0000000000000000 putc.c
0000000000000000 1      df *ABS* 0000000000000000 sbrkr.c

```

Рис. 11. Таблица символов.

Часть 5. Вывод.

Была разработана функция и заголовочный файл на языке C, выполнена отдельная компиляция, также были изучены принципы создания статической библиотеки