Отчет по заданию №2 в рамках вычислительного практикума

Этапы получения исполняемого файла

1. Простая программа на языке Си.

```c
#include <stdio.h>
#include <stdlib.h>

#define RATE 104

int main(void)
{
    double amount_dollars;
    printf("Enter the amount of dollars: ");
    scanf("%lf", &amount_dollars);
    double amount_rubles = amount_dollars * RATE;
    printf("%.2lf dollars in rubles will be %.2lf", amount_dollars,
amount_rubles);

    return 0;
}
```

2. Этапы получения исполняемого файла.

1 Этап: Обработка препроцессором.

Для обработки программы препроцессором используется следующая конструкция

```
cpp main.c
```

В общем случае препроцессор выводит результат своей работы на экран.
Если мы хотим перенаправить вывод результата работы в файл, то используем одну из следущих конструкций :

```
cpp main.c > main.i
```

```
cpp main.c -o main.i
```

Ключ “o” помогает задать имя файла результата
До обработки программы препроцессором , программа занимала 338 байтов, а после размер составил 43.5Кбайт
Результат работы препроцессора:

```c
int main(void)
{
    double amount_dollars;
    printf("Enter the amount of dollars: ");
    scanf("%lf", &amount_dollars);
    double amount_rubles = amount_dollars * 104;
    printf("%.2lf dollars in rubles will be %.2lf", amount_dollars,
amount_rubles);
```

```
    return 0;
}
```

Препроцессор убрал директивы #include и #define , выполнил числовые замены.
2 Этап: Трансляция на язык ассемблера.
Для того чтобы выполнить трансляцию с языка Си на язык ассемблера нужно выполнить следующую команду:

```
c99 -S -fverbose-asm -fmasm main.i
```

c99- То , как согласно POSIX'у должен называться компилятор языка Си для этого стандарта .
-
Основная часть вывода в файл main.s:

```
        .arch armv8-a
        .file   "main.c"
// ...
        .text
        .section        .rodata
        .align  3
.LC0:
        .string "Enter the amount of dollars: "
        .align  3
.LC1:
        .string "%lf"
        .align  3
.LC2:
        .string "%.2lf dollars in rubles will be %.2lf"
        .text
        .align  2
        .global main
        .type   main, %function

main:
.LFB6:
        .cfi_startproc
        stp     x29, x30, [sp, -48]!    //,,,
        .cfi_def_cfa_offset 48
        .cfi_offset 29, -48
        .cfi_offset 30, -40
        mov     x29, sp //,
// main.c:7: {
        adrp    x0, :got:__stack_chk_guard      // tmp95,
        ldr     x0, [x0, #:got_lo12:__stack_chk_guard]  // tmp94, tmp95,
        ldr     x1, [x0]        // tmp106, __stack_chk_guard
        str     x1, [sp, 40]    // tmp106, D.4174
        mov     x1,0    // tmp106
// main.c:9:     printf("Enter the amount of dollars: ");
        adrp    x0, .LC0        // tmp96,
        add     x0, x0, :lo12:.LC0      //, tmp96,
        bl      printf          //
// main.c:10:     scanf("%lf", &amount_dollars);
        add     x0, sp, 24      // tmp97,,
        mov     x1, x0  //, tmp97
        adrp    x0, .LC1        // tmp98,
        add     x0, x0, :lo12:.LC1      //, tmp98,
        bl      __isoc99_scanf          //
// main.c:11:     double amount_rubles = amount_dollars * RATE;
```

```
        ldr     d0, [sp, 24]    // amount_dollars.0_1, amount_dollars
// main.c:11:     double amount_rubles = amount_dollars * RATE;
        mov     x0, 4637018766331346944 // tmp105,
        fmov    d1, x0  // tmp100, tmp105
        fmul    d0, d0, d1      // tmp99, amount_dollars.0_1, tmp100
        str     d0, [sp, 32]    // tmp99, amount_rubles
// main.c:12:     printf("%.2lf dollars in rubles will be %.2lf",
amount_dollars, amount_rubles);
        ldr     d0, [sp, 24]    // amount_dollars.1_2, amount_dollars
        ldr     d1, [sp, 32]    //, amount_rubles
        adrp    x0, .LC2        // tmp101,
        add     x0, x0, :lo12:.LC2      //, tmp101,
        bl      printf          //
// main.c:14:     return 0;
        mov     w0, 0   // _8,
        mov     w1, w0  // <retval>, _8
// main.c:15: }
        adrp    x0, :got:__stack_chk_guard      // tmp104,
        ldr     x0, [x0, #:got_lo12:__stack_chk_guard]  // tmp103, tmp104,
        ldr     x2, [sp, 40]    // tmp107, D.4174
        ldr     x3, [x0]        // tmp108, __stack_chk_guard
        subs    x2, x2, x3      // tmp107, tmp108
        mov     x3, 0   // tmp108
        beq     .L3             //,
        bl      __stack_chk_fail                //
.L3:
        mov     w0, w1  //, <retval>
        ldp     x29, x30, [sp], 48      //,,,
        .cfi_restore 30
        .cfi_restore 29
        .cfi_def_cfa_offset 0
        ret
// ...
```

3 Этап: Ассемблирование в объектный файл

С языка ассемблера программа переводится в машинный код с помощью транслятора as.

```
as hello.s -o hello.o
```

На выходе этого транслятора получается не текстовый (как на двух предыдущих этапах), а двоичный файл. Этот файл называется объектным файлом.

Для просмотра объектного файла используется команда “hexdump -C main.o”. Реализуется просмотр данных двоичного файла в виде ASCII символов.

Для преобразования машинного кода в ассемблер

Для работы с объектными файлами есть ряд утилит, одним из таких является “objdump”. Эта утилита позволяет выполнять разные действия с объектными файлами. С помощью нее можно сделать обратное преобразование (из машинного кода в ассемблер).

Результат работы ассемблера

```
0000000000000000 <main>:
   0:   a9bd7bfd        stp     x29, x30, [sp, #-48]!
   4:   910003fd        mov     x29, sp
   8:   90000000        adrp    x0, 0 <__stack_chk_guard>      8:
R_AARCH64_ADR_GOT_PAGE __stack_chk_guard
   c:   f9400000        ldr     x0, [x0]        c:
R_AARCH64_LD64_GOT_LO12_NC      __stack_chk_guard
```

```
  10:   f9400001        ldr     x1, [x0]
  14:   f90017e1        str     x1, [sp, #40]
  18:   d2800001        mov     x1, #0x0                        // #0
  1c:   90000000        adrp    x0, 0 <main>    1c:
R_AARCH64_ADR_PREL_PG_HI21      .rodata
  20:   91000000        add     x0, x0, #0x0    20:
R_AARCH64_ADD_ABS_LO12_NC       .rodata
  24:   94000000        bl      0 <printf>      24: R_AARCH64_CALL26    printf
  28:   910063e0        add     x0, sp, #0x18
  2c:   aa0003e1        mov     x1, x0
  30:   90000000        adrp    x0, 0 <main>    30:
R_AARCH64_ADR_PREL_PG_HI21      .rodata+0x20
  34:   91000000        add     x0, x0, #0x0    34:
R_AARCH64_ADD_ABS_LO12_NC       .rodata+0x20
  38:   94000000        bl      0 <__isoc99_scanf>      38:
R_AARCH64_CALL26        __isoc99_scanf
  3c:   fd400fe0        ldr     d0, [sp, #24]
  40:   d2e80b40        mov     x0, #0x405a000000000000         //
#4637018766331346944
  44:   9e670001        fmov    d1, x0
  48:   1e610800        fmul    d0, d0, d1
  4c:   fd0013e0        str     d0, [sp, #32]
  50:   fd400fe0        ldr     d0, [sp, #24]
  54:   fd4013e1        ldr     d1, [sp, #32]
  58:   90000000        adrp    x0, 0 <main>    58:
R_AARCH64_ADR_PREL_PG_HI21      .rodata+0x28
  5c:   91000000        add     x0, x0, #0x0    5c:
R_AARCH64_ADD_ABS_LO12_NC       .rodata+0x28
  60:   94000000        bl      0 <printf>      60: R_AARCH64_CALL26    printf
  64:   52800000        mov     w0, #0x0                        // #0
  68:   2a0003e1        mov     w1, w0
  6c:   90000000        adrp    x0, 0 <__stack_chk_guard>       6c:
R_AARCH64_ADR_GOT_PAGE __stack_chk_guard
  70:   f9400000        ldr     x0, [x0]        70:
R_AARCH64_LD64_GOT_LO12_NC      __stack_chk_guard
  74:   f94017e2        ldr     x2, [sp, #40]
  78:   f9400003        ldr     x3, [x0]
  7c:   eb030042        subs    x2, x2, x3
  80:   d2800003        mov     x3, #0x0                        // #0
  84:   54000040        b.eq    8c <main+0x8c>  // b.none
  88:   94000000        bl      0 <__stack_chk_fail>    88:
R_AARCH64_CALL26        __stack_chk_fail
  8c:   2a0103e0        mov     w0, w1
  90:   a8c37bfd        ldp     x29, x30, [sp], #48
  94:   d65f03c0        ret
```

4 Этап: Компоновка
Чтобы получить исполняемый файл необходимо вызвать компоновщик.

```
ld другие_параметры -o hello.exe hello.o
```

Обычно “ld” используется для линковки стандартных объектных файлов Юникса на стандартной Юникс системе.

Параметры для ld:
-A<архитектура>

В текущей версии ld эта опция используется только для процессоров семейства Intel 960. В конфигурации LD для этого процессора аргумент <архитектура> идентифицирует используемую в системе модификацию процессора и, руководствуясь полученной информацией, включает дополнительные диагностические процедуры и изменяет некоторые значения по умолчанию

-Bstatic

Не использовать при линковке разделяемые библиотеки. Эта опция имеет смысл только на платформах, поддерживающих разделяемые библиотеки.

-Bdynamic

Использовать динамические библиотеки. Эта опция имеет смысл только на платформах, поддерживающих разделяемые библиотеки. Эта опция на большинстве платформ обычно установлена по умолчанию.

-Bsymbolic

При создании разделяемой библиотеки связать ссылки на глобальные имена с описаниями в разделяемой библиотеке. Эта опция имеет смысл только на ELF платформах, которые поддерживают разделяемые библиотеки.

-G значение

Устанавливает максимальный размер объектов для оптимизации с использованием регистра GP под форматом объектного файла MIPS ECOFF. Игнорируется для остальных форматов объектного файла.

-help

Выводит список опций командной строки и завершает выполнение линкера.

-i

Выполняет инкрементальную линковку аналогично опции '-r'.

-l имя

Добавляет архивный файл с указанным именем в список файлов для линковки. Эта опция может быть использована неограниченное количество раз. LD будет искать по всем указанным путям архивный файл (библиотеку) с именем «lib<имя>.a»

В процессе получения исполняемого файла компоновщик решает несколько задач

- объединяет несколько объектных файлов в единый исполняемый файл;
- выполняет связывание переменных и функций, которые требуются очередному объектному файлу, но находятся где-то в другом месте;
- добавляет специальный код, который подготавливает окружение для вызова функции main, а после ее завершения выполняет обратные действия.

На вход компоновщику подается объектный файл main.o который состоит из секций. Также на вход приходит системный код, который также может содержать как код, так и разного рода данные. текстовые данные . Компоновщик на основе нескольких объектных файлов строит исполняемый файл. Первое что делает компоновщик – это объединяет однотипные секции в одну. Второе- разрешает ссылки на те функции которые появились только в исполняемом файле.

3. Драйвер- низкоуровневая программа, содержащая специфический код для работы с устройством, которая позволяет пользовательским программам (и самой ОС) управлять им стандартным образом.
   gcc - это программа-драйвер. Он выполняет свою работу, вызывая последовательность других программ для выполнения работы по компиляции, сборке и компоновке. GCC интерпретирует свои параметры командной строки и

использует их, чтобы определить, какие программы следует вызывать и какие параметры командной строки следует разместить в своих командных строках. clang является драйвером, который анализирует входные аргументы и определяет, какие компиляторы / ассемблеры / компоновщики вызывают на каких файлах какие аргументы командной строки.

4. Ключ -v (--version) выводит информацию о версии ld.

```
gcc -v main.c
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/aarch64-linux-gnu/9/lto-wrapper
Target: aarch64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu
9.4.0-1ubuntu1~20.04' --with-bugurl=file:///usr/share/doc/gcc-
9/README.Bugs --enable-languages=c,ada,c++,go,d,fortran,objc,obj-
c++,gm2 --prefix=/usr --with-gcc-major-version-only --program-
suffix=-9 --program-prefix=aarch64-linux-gnu- --enable-shared --
enable-linker-build-id --libexecdir=/usr/lib --without-included-
gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --
enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-
time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-
object --disable-libquadmath --disable-libquadmath-support --
enable-plugin --enable-default-pie --with-system-zlib --with-
target-system-zlib=auto --enable-objc-gc=auto --enable-multiarch --
enable-fix-cortex-a53-843419 --disable-werror --enable-
checking=release --build=aarch64-linux-gnu --host=aarch64-linux-gnu
--target=aarch64-linux-gnu
Thread model: posix
gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04)
COLLECT_GCC_OPTIONS='-v' '-mlittle-endian' '-mabi=lp64'
 /usr/lib/gcc/aarch64-linux-gnu/9/cc1 -quiet -v -imultiarch
aarch64-linux-gnu main.c -quiet -dumpbase main.c -mlittle-endian -
mabi=lp64 -auxbase main -version -fasynchronous-unwind-tables -
fstack-protector-strong -Wformat -Wformat-security -fstack-clash-
protection -o /tmp/ccra4Ebv.s
GNU C17 (Ubuntu 9.4.0-1ubuntu1~20.04) version 9.4.0 (aarch64-linux-
gnu)
     compiled by GNU C version 9.4.0, GMP version 6.2.0, MPFR
version 4.0.2, MPC version 1.1.0, isl version isl-0.22.1-GMP

GGC heuristics: --param ggc-min-expand=100 --param ggc-min-
heapsize=131072
ignoring nonexistent directory "/usr/local/include/aarch64-linux-
gnu"
ignoring nonexistent directory "/usr/lib/gcc/aarch64-linux-
gnu/9/include-fixed"
ignoring nonexistent directory "/usr/lib/gcc/aarch64-linux-
gnu/9/../../../../aarch64-linux-gnu/include"
#include "..." search starts here:
#include <...> search starts here:
 /usr/lib/gcc/aarch64-linux-gnu/9/include
 /usr/local/include
 /usr/include/aarch64-linux-gnu
 /usr/include
End of search list.
```

```
GNU C17 (Ubuntu 9.4.0-1ubuntu1~20.04) version 9.4.0 (aarch64-linux-
gnu)
    compiled by GNU C version 9.4.0, GMP version 6.2.0, MPFR
version 4.0.2, MPC version 1.1.0, isl version isl-0.22.1-GMP

GGC heuristics: --param ggc-min-expand=100 --param ggc-min-
heapsize=131072
Compiler executable checksum: 540b4a8f86e6693ced694adf18321a51
COLLECT_GCC_OPTIONS='-v' '-mlittle-endian' '-mabi=lp64'
 as -v -EL -mabi=lp64 -o /tmp/ccHLjlfx.o /tmp/ccra4Ebv.s
GNU ассемблер, версия 2.34 (aarch64-linux-gnu); используется BFD
версии (GNU Binutils for Ubuntu) 2.34
COMPILER_PATH=/usr/lib/gcc/aarch64-linux-
gnu/9/:/usr/lib/gcc/aarch64-linux-gnu/9/:/usr/lib/gcc/aarch64-
linux-gnu/:/usr/lib/gcc/aarch64-linux-gnu/9/:/usr/lib/gcc/aarch64-
linux-gnu/
LIBRARY_PATH=/usr/lib/gcc/aarch64-linux-
gnu/9/:/usr/lib/gcc/aarch64-linux-gnu/9/../../../aarch64-linux-
gnu/:/usr/lib/gcc/aarch64-linux-
gnu/9/../../../../lib/:/lib/aarch64-linux-
gnu/:/lib/../lib/:/usr/lib/aarch64-linux-
gnu/:/usr/lib/../lib/:/usr/lib/gcc/aarch64-linux-
gnu/9/../../../:/lib/:/usr/lib/
COLLECT_GCC_OPTIONS='-v' '-mlittle-endian' '-mabi=lp64'
 /usr/lib/gcc/aarch64-linux-gnu/9/collect2 -plugin
/usr/lib/gcc/aarch64-linux-gnu/9/liblto_plugin.so -plugin-
opt=/usr/lib/gcc/aarch64-linux-gnu/9/lto-wrapper -plugin-opt=-
fresolution=/tmp/cc0mF5fv.res -plugin-opt=-pass-through=-lgcc -
plugin-opt=-pass-through=-lgcc_s -plugin-opt=-pass-through=-lc -
plugin-opt=-pass-through=-lgcc -plugin-opt=-pass-through=-lgcc_s --
build-id --eh-frame-hdr --hash-style=gnu --as-needed -dynamic-
linker /lib/ld-linux-aarch64.so.1 -X -EL -maarch64linux --fix-
cortex-a53-843419 -pie -z now -z relro /usr/lib/gcc/aarch64-linux-
gnu/9/../../../aarch64-linux-gnu/Scrt1.o /usr/lib/gcc/aarch64-
linux-gnu/9/../../../aarch64-linux-gnu/crti.o /usr/lib/gcc/aarch64-
linux-gnu/9/crtbeginS.o -L/usr/lib/gcc/aarch64-linux-gnu/9 -
L/usr/lib/gcc/aarch64-linux-gnu/9/../../../aarch64-linux-gnu -
L/usr/lib/gcc/aarch64-linux-gnu/9/../../../../lib -L/lib/aarch64-
linux-gnu -L/lib/../lib -L/usr/lib/aarch64-linux-gnu -
L/usr/lib/../lib -L/usr/lib/gcc/aarch64-linux-gnu/9/../../..
/tmp/ccHLjlfx.o -lgcc --push-state --as-needed -lgcc_s --pop-state
-lc -lgcc --push-state --as-needed -lgcc_s --pop-state
/usr/lib/gcc/aarch64-linux-gnu/9/crtendS.o /usr/lib/gcc/aarch64-
linux-gnu/9/../../../aarch64-linux-gnu/crtn.o
COLLECT_GCC_OPTIONS='-v' '-mlittle-endian' '-mabi=lp64'
```

Ключ -save-temps указывает gcc сохранить промежуточные файлы. Файлы будут сохранены в текущей директории с именами, зависящими от имени исходного файла. Параметр -S нужен не прекращения компиляции main.o. При компиляции без -S создаются файлы : main.i, main.o, main.s

```
skiper22@skiper22:~/Рабочий стол/PTP2$ gcc -save-temps main.c -lm
skiper22@skiper22:~/Рабочий стол/PTP2$ ls
a.out  main.c  main.i  main.o  main.s
skiper22@skiper22:~/Рабочий стол/PTP2$ gcc -save-temps -S main.c -lm
skiper22@skiper22:~/Рабочий стол/PTP2$ ls
a.out  main.c  main.i  main.o  main.s
```

a) Сохраняются временные файлы с расширениями *.o и *.i , которые были созданы на этапе компиляции.
b) *.s – код ассемблера
   *.i – содержит программный код, добавляя #include, #define, расширяет макросы.
c) При выполнении второго пункта этих файлов не было.
d) Компоновка программы осуществляется с использованием файлов с расширениями : *.i, *.o, *.s

5.

```
skiper22@skiper22:~/Рабочий стол/PTP$ clang -v -save-temps -o main.c
main.exe
clang version 10.0.0-4ubuntu1
Target: aarch64-unknown-linux-gnu
Thread model: posix
InstalledDir: /usr/bin
Found candidate GCC installation: /usr/bin/../lib/gcc/aarch64-linux-gnu/9
Found candidate GCC installation: /usr/lib/gcc/aarch64-linux-gnu/9
Selected GCC installation: /usr/bin/../lib/gcc/aarch64-linux-gnu/9
Candidate multilib: .;@m64
Selected multilib: .;@m64
 "/usr/bin/ld" -EL -z relro --hash-style=gnu --build-id --eh-frame-hdr -m
aarch64linux -dynamic-linker /lib/ld-linux-aarch64.so.1 -o main.c
/usr/bin/../lib/gcc/aarch64-linux-gnu/9/../../../aarch64-linux-gnu/crt1.o
/usr/bin/../lib/gcc/aarch64-linux-gnu/9/../../../aarch64-linux-gnu/crti.o
/usr/bin/../lib/gcc/aarch64-linux-gnu/9/crtbegin.o -
L/usr/bin/../lib/gcc/aarch64-linux-gnu/9 -L/usr/bin/../lib/gcc/aarch64-
linux-gnu/9/../../../aarch64-linux-gnu -L/lib/aarch64-linux-gnu -
L/usr/lib/aarch64-linux-gnu -L/usr/bin/../lib/gcc/aarch64-linux-
gnu/9/../../.. -L/usr/lib/llvm-10/bin/../lib -L/lib -L/usr/lib main.exe -
lgcc --as-needed -lgcc_s --no-as-needed -lc -lgcc --as-needed -lgcc_s --no-
as-needed /usr/bin/../lib/gcc/aarch64-linux-gnu/9/crtend.o
/usr/bin/../lib/gcc/aarch64-linux-gnu/9/../../../aarch64-linux-gnu/crtn.o
/usr/bin/ld: /usr/bin/../lib/gcc/aarch64-linux-gnu/9/../../../aarch64-linux-
gnu/crt1.o: в функции «__wrap_main»:
```

Clang выполняет то же самое что и gcc , но создавая на один файл больше (bc – бинарный файл).

6.

```
skiper22@skiper22:~/Рабочий стол/PTP2$ gcc -Xassembler -a main.c >
main_asm.s
```

Данные выведенные в файл main_asm.s

```
AARCH64 GAS  /tmp/ccvaacRK.s                   page 1


   1                            .arch armv8-a
   2                            .file   "main.c"
   3                            .text
   4                            .section       .rodata
   5                            .align  3
   6                    .LC0:
   7 0000 456E7465              .string "Enter the amount of dollars: "
   7      72207468
   7      6520616D
```

```
   7      6F756E74
   7      206F6620
   8 001e 0000                .align  3
   9                    .LC1:
  10 0020 256C6600            .string "%lf"
  11 0024 00000000            .align  3
  12                    .LC2:
  13 0028 252E326C            .string "%.2lf dollars in rubles will
be %.2lf"
..//

..//
  68                          .size   main, .-main
  69                          .ident  "GCC: (Ubuntu 9.4.0-
1ubuntu1~20.04) 9.4.0"
  70                          .section        .note.GNU-
stack,"",@progbits
AARCH64 GAS  /tmp/ccvaacRK.s                 page 3


DEFINED SYMBOLS
                           *ABS*:0000000000000000 main.c
     /tmp/ccvaacRK.s:5      .rodata:0000000000000000 $d
     /tmp/ccvaacRK.s:15     .text:0000000000000000 $x
     /tmp/ccvaacRK.s:18     .text:0000000000000000 main
                         .eh_frame:0000000000000014 $d

UNDEFINED SYMBOLS
__stack_chk_guard
printf
__isoc99_scanf
__stack_chk_fail
```

7.

```
skiper22@skiper22:~/Рабочий стол/PTP$ gcc -Xlinker -Map=main.map
```

Часть данных выведенных в файл main.map

```
Для удовлетворения ссылок на файл (символ) включён член архива

/usr/lib/aarch64-linux-gnu/libc_nonshared.a(elf-init.oS)
                              /usr/lib/gcc/aarch64-linux-
gnu/9/../../../aarch64-linux-gnu/Scrt1.o (__libc_csu_init)

Для удовлетворения ссылок на файл (символ) включена библиотека по
необходимости

libc.so.6                     /usr/lib/gcc/aarch64-linux-
gnu/9/../../../aarch64-linux-gnu/Scrt1.o (abort@@GLIBC_2.17)

Отброшенные входные разделы

 .note.GNU-stack
                0x0000000000000000        0x0 /usr/lib/gcc/aarch64-
linux-gnu/9/../../../aarch64-linux-gnu/Scrt1.o
 .note.GNU-stack
..//
..//
.ARM.attributes
 *(.ARM.attributes)
```

```
 *(.gnu.attributes)

.note.gnu.arm.ident
 *(.note.gnu.arm.ident)

/DISCARD/
 *(.note.GNU-stack)
 *(.gnu_debuglink)
 *(.gnu.lto_*)
OUTPUT(a.out elf64-littleaarch64)
LOAD linker stubs
```

8.

```
skiper22@skiper22:~/Рабочий стол/PTP$ objdump -D main.o

main.o:     формат файла elf64-littleaarch64


Дизассемблирование раздела .text:

0000000000000000 <main>:
   0:   a9bd7bfd        stp     x29, x30, [sp, #-48]!
   4:   910003fd        mov     x29, sp
   8:   90000000        adrp    x0, 0 <__stack_chk_guard>
   c:   f9400000        ldr     x0, [x0]
  10:   f9400001        ldr     x1, [x0]
  14:   f90017e1        str     x1, [sp, #40]
  18:   d2800001        mov     x1, #0x0                        // #0
  1c:   90000000        adrp    x0, 0 <main>
  20:   91000000        add     x0, x0, #0x0
  24:   94000000        bl      0 <printf>
  28:   910063e0        add     x0, sp, #0x18
  2c:   aa0003e1        mov     x1, x0
  30:   90000000        adrp    x0, 0 <main>
  34:   91000000        add     x0, x0, #0x0
  38:   94000000        bl      0 <__isoc99_scanf>
  3c:   fd400fe0        ldr     d0, [sp, #24]
  40:   d2e80b40        mov     x0, #0x405a000000000000         //
#4637018766331346944
  44:   9e670001        fmov    d1, x0
  48:   1e610800        fmul    d0, d0, d1
  4c:   fd0013e0        str     d0, [sp, #32]
  50:   fd400fe0        ldr     d0, [sp, #24]
  54:   fd4013e1        ldr     d1, [sp, #32]
  58:   90000000        adrp    x0, 0 <main>
  5c:   91000000        add     x0, x0, #0x0
  60:   94000000        bl      0 <printf>
  64:   52800000        mov     w0, #0x0                        // #0
  68:   2a0003e1        mov     w1, w0
  6c:   90000000        adrp    x0, 0 <__stack_chk_guard>
  70:   f9400000        ldr     x0, [x0]
  74:   f94017e2        ldr     x2, [sp, #40]
  78:   f9400003        ldr     x3, [x0]
  7c:   eb030042        subs    x2, x2, x3
  80:   d2800003        mov     x3, #0x0                        // #0
  84:   54000040        b.eq    8c <main+0x8c>  // b.none
  88:   94000000        bl      0 <__stack_chk_fail>
  8c:   2a0103e0        mov     w0, w1
  90:   a8c37bfd        ldp     x29, x30, [sp], #48
```

```
  94:   d65f03c0        ret

Дизассемблирование раздела .rodata:

0000000000000000 <.rodata>:
   0:   65746e45        fnmls   z5.h, p3/m, z18.h, z20.h
   4:   68742072        .inst   0x68742072 ; undefined
   8:   6d612065        ldp     d5, d8, [x3, #-496]
   c:   746e756f        .inst   0x746e756f ; undefined
  10:   20666f20        .inst   0x20666f20 ; undefined
  14:   6c6c6f64        ldnp    d4, d27, [x27, #-320]
  18:   3a737261        .inst   0x3a737261 ; undefined
  1c:   00000020        .inst   0x00000020 ; undefined
  20:   00666c25        .inst   0x00666c25 ; undefined
  24:   00000000        .inst   0x00000000 ; undefined
  28:   6c322e25        stnp    d5, d11, [x17, #-224]
  2c:   6f642066        umlal2  v6.4s, v3.8h, v4.h[2]
  30:   72616c6c        .inst   0x72616c6c ; undefined
  34:   6e692073        usubl2  v19.4s, v3.8h, v9.8h
  38:   62757220        .inst   0x62757220 ; undefined
  3c:   2073656c        .inst   0x2073656c ; undefined
  40:   6c6c6977        ldnp    d23, d26, [x11, #-320]
  44:   20656220        .inst   0x20656220 ; undefined
  48:   6c322e25        stnp    d5, d11, [x17, #-224]
  4c:   Address 0x000000000000004c is out of bounds.


Дизассемблирование раздела .comment:

0000000000000000 <.comment>:
   0:   43434700        .inst   0x43434700 ; undefined
   4:   5528203a        .inst   0x5528203a ; undefined
   8:   746e7562        .inst   0x746e7562 ; undefined
   c:   2e392075        usubl   v21.8h, v3.8b, v25.8b
  10:   2d302e34        stp     s20, s11, [x17, #-128]
  14:   75627531        .inst   0x75627531 ; undefined
  18:   3175746e        adds    w14, w3, #0xd5d, lsl #12
  1c:   2e30327e        usubw   v30.8h, v19.8h, v16.8b
  20:   20293430        .inst   0x20293430 ; undefined
  24:   2e342e39        uqsub   v25.8b, v17.8b, v20.8b
  28:   Address 0x0000000000000028 is out of bounds.


Дизассемблирование раздела .eh_frame:

0000000000000000 <.eh_frame>:
   0:   00000010        .inst   0x00000010 ; undefined
   4:   00000000        .inst   0x00000000 ; undefined
   8:   00527a01        .inst   0x00527a01 ; undefined
   c:   011e7804        .inst   0x011e7804 ; undefined
  10:   001f0c1b        .inst   0x001f0c1b ; undefined
  14:   00000020        .inst   0x00000020 ; undefined
  18:   00000018        .inst   0x00000018 ; undefined
  1c:   00000000        .inst   0x00000000 ; undefined
  20:   00000098        .inst   0x00000098 ; undefined
  24:   300e4100        adr     x0, 1c845 <main+0x1c845>
  28:   059e069d        mov     z29.s, p14/z, #52
  2c:   0eddde64        .inst   0x0eddde64 ; undefined
        ...
```

Отличие результатов дизассемблирования от полученной программы на языке ассемблера отличается тем, что ассемблерный файл показывает свой код и еще машинный код, а дизассемблерный показывает только машинный код .

9.

```
skiper22@skiper22:~/Рабочий стол/PTP$ nm -fsysv main.o


Символы из main.o:

Имя                  Знач.           Класс        Тип           Размер
Строка Раздел

gl                  |0000000000000000|   D  |
OBJECT|0000000000000004|     |.data
__isoc99_scanf      |                |   U  |            NOTYPE|
|     |*UND*
main                |0000000000000000|   T  |
FUNC|0000000000000098|     |.text
np                  |0000000000000004|   C  |
OBJECT|0000000000000004|     |*COM*
printf              |                |   U  |            NOTYPE|
|     |*UND*
__stack_chk_fail    |                |   U  |            NOTYPE|
|     |*UND*
__stack_chk_guard   |                |   U  |            NOTYPE|
|     |*UND*
skiper22@skiper22:~/Рабочий стол/PTP$ objdump main.o -h

main.o:     формат файла elf64-littleaarch64

Разделы:
Idx Name          Разм      VMA               LMA               Фа  смещ.
Выр.
  0 .text         00000098  0000000000000000  0000000000000000  00000040
2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000004  0000000000000000  0000000000000000  000000d8
2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  000000dc
2**0
                  ALLOC
  3 .rodata       0000004e  0000000000000000  0000000000000000  000000e0
2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      0000002a  0000000000000000  0000000000000000  0000012e
2**0
                  CONTENTS, READONLY
  5 .note.GNU-stack 00000000  0000000000000000  0000000000000000  00000158
2**0
                  CONTENTS, READONLY
  6 .eh_frame     00000038  0000000000000000  0000000000000000  00000158
2**3
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, DATA
```

Глобальная проинициализированная переменная попала в раздел .data
Глобальная непроинициализированная переменная попала в раздел *COM*

Функция main попала в раздел .text.
Функция printf попала в раздел *UND*
Локальные переменные не обнаружены.

10.

```
skiper22@skiper22:~/Рабочий стол/PTP$ gcc -c -g3 main.c
skiper22@skiper22:~/Рабочий стол/PTP$ vim main.o
skiper22@skiper22:~/Рабочий стол/PTP$ ls -la
итого 136
drwxrwxr-x 2 skiper22 skiper22  4096 мар 20 21:42 .
drwxr-xr-x 9 skiper22 skiper22  4096 мар 20 15:20 ..
-rw-rw-r-- 1 skiper22 skiper22   357 мар 20 21:24 main.c
-rwxrwxr-x 1 skiper22 skiper22  9472 мар 20 21:24 main.exe
-rw-rw-r-- 1 skiper22 skiper22     0 мар 20 21:18 main.i
-rw-rw-r-- 1 skiper22 skiper22 17233 мар 20 20:28 main.map
-rw-rw-r-- 1 skiper22 skiper22 76344 мар 20 21:41 main.o
-rw-rw-r-- 1 skiper22 skiper22  1257 мар 20 13:22 main.s
-rwxrwxr-x 1 skiper22 skiper22  9392 мар 20 12:57 -save-temps
```

По сравнению с предыдущим, объектный файл стал весить в несколько раз больше.

11. Для получения исполняемого файла без отладочной информации надо использовать комманду:

```
gcc -g0 main.c
```

12.

a) Отличие состоит в том, что файлы с отладочной информацией имеют больший размер, нежели файл без отладочной информации.
b) Файл с отладочной информацией содержит больше секций, в отличие от файла без отладочной информации.
c) Функция и глобальная проинициализированная не поменяли свое расположение . Глобальная непроинициализированная переменная поменяла свое расположение.

13.

```
skiper22@skiper22:~/Рабочий стол/PTP$ ldd main.exe
        linux-vdso.so.1 (0x0000ffff8dc6d000)
        libc.so.6 => /lib/aarch64-linux-gnu/libc.so.6 (0x0000ffff8daa4000)
        /lib/ld-linux-aarch64.so.1 (0x0000ffff8dc3d000)
```