Práctica 01

DOCENTE	CARRERA	CURSO
MSc. Vicente Enrique Machaca Arceda	Escuela Profesional de Ingeniería de Software	Compiladores

PRÁCTICA	TEMA	DURACIÓN
01	Introducción	3 horas

1. Datos de los estudiantes

- Grupo: 1
- Git Hub: https://github.com/CrazyDani17/Practica1-Compiladores
- Integrantes:
  - Guillermo Alemán
  - Marvik Del Carpio
  - Daniel Mendiguri
  - Daniela Vilchez

2. Ejercicios

1. Redacta el siguiente código, genera el código ensamblador y explica en qué parte (del código ensamblador) se definen las variables c y m. (2 puntos).

```
int main(){
    char* c = "abcdef";
    int m = 11148;
    return 0;
}
```

Código Assembler:

```
        .file   "code.c"
        .def    ___main;        .scl    2;      .type   32;     .end
        .section .rdata,"dr"
LC0:
        .ascii "abcdef\0"
        .text
        .globl  _main
        .def    _main;  .scl    2;      .type   32;     .endef
_main:
LFB0:
```

```
        .cfi_startproc
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        andl    $-16, %esp
        subl    $16, %esp
        call    ___main
        movl    $LC0, 12(%esp)
        movl    $11148, 8(%esp)
        movl    $0, %eax
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
LFE0:
        .ident  "GCC: (MinGW.org GCC-6.3.0-1) 6.3.0"
```

Solución

La variable 'c' se define en las líneas detalladas abajo correspondinetes al código Assembler:

```
LC0:
.ascii "abcdef\0"
.text
.globl    _main
.def    _main;    .scl    2;    .type    32;    .endef
movl    $LC0, 12(%esp)
```

Mientras que la variable 'm' corresponde a la siguiente:

```
movl    $11148, 8(%esp)
```

2. Redacta el siguiente código, genera el código ensamblador y explica en qué parte (del código ensamblador) se define la división entre 8. (2 puntos).

```
int main(){
    char* c = "abcdef";
    int m = 11148;
    int x = m/8;
    return 0;
    }
```

Código Assembler:

```
        .file   "ej2.cpp"
        .text
        .section        .rodata
        .type   _ZStL19piecewise_construct, @object
        .size   _ZStL19piecewise_construct, 1
_ZStL19piecewise_construct:
```

```
        .zero   1
        .local  _ZStL8__ioinit
        .comm   _ZStL8__ioinit,1,1
.LC0:
        .string "abcdef"
        .text
        .globl  main
        .type   main, @function
main:
.LFB1493:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    .LC0(%rip), %rax
        movq    %rax, -8(%rbp)
        movl    $11148, -16(%rbp)
        movl    -16(%rbp), %eax
        leal    7(%rax), %edx
        testl   %eax, %eax
        cmovs   %edx, %eax
        sarl    $3, %eax
        movl    %eax, -12(%rbp)
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1493:
        .size   main, .-main
        .type   _Z41__static_initialization_and_destruction_0ii, @fu
_Z41__static_initialization_and_destruction_0ii:
.LFB1974:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    %edi, -4(%rbp)
        movl    %esi, -8(%rbp)
        cmpl    $1, -4(%rbp)
        jne     .L5
        cmpl    $65535, -8(%rbp)
        jne     .L5
        leaq    _ZStL8__ioinit(%rip), %rdi
        call    _ZNSt8ios_base4InitC1Ev@PLT
        leaq    __dso_handle(%rip), %rdx
        leaq    _ZStL8__ioinit(%rip), %rsi
```

```
        movq    _ZNSt8ios_base4InitD1Ev@GOTPCREL(%rip), %rax
        movq    %rax, %rdi
        call    __cxa_atexit@PLT
.L5:
        nop
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1974:
        .size   _Z41__static_initialization_and_destruction_0ii, .-_
        .type   _GLOBAL__sub_I_main, @function
_GLOBAL__sub_I_main:
.LFB1975:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $65535, %esi
        movl    $1, %edi
        call    _Z41__static_initialization_and_destruction_0ii
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1975:
        .size   _GLOBAL__sub_I_main, .-_GLOBAL__sub_I_main
        .section        .init_array,"aw"
        .align 8
        .quad   _GLOBAL__sub_I_main
        .hidden __dso_handle
        .ident  "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"
        .section        .note.GNU-stack,"",@progbits
```

Solución
La división entre 8 se da en las siguientes líneas:

```
    movl        -16(%rbp), %eax
leal    7(%rax), %edx
testl   %eax, %eax
cmovs   %edx, %eax
sarl    $3, %eax
movl    %eax, -12(%rbp)
```

3. Redacta el siguiente código, genera el código ensamblador y explica en qué parte (del código ensamblador) se define la división entre 4. (2 puntos).

```
int main(){
    char* c = "abcdef";
```

```
int m = 11148;
int x = m/8;
int y = m/4;
int z = m/2;
return 0;
}
```

Código Assembler:

```
        .file   "ej3.cpp"
        .text
        .section        .rodata
        .type   _ZStL19piecewise_construct, @object
        .size   _ZStL19piecewise_construct, 1
_ZStL19piecewise_construct:
        .zero   1
        .local  _ZStL8__ioinit
        .comm   _ZStL8__ioinit,1,1
        .text
        .globl  main
        .type   main, @function
main:
.LFB1493:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $11148, -16(%rbp)
        movl    -16(%rbp), %eax
        leal    7(%rax), %edx
        testl   %eax, %eax
        cmovs   %edx, %eax
        sarl    $3, %eax
        movl    %eax, -12(%rbp)
        movl    -16(%rbp), %eax
        leal    3(%rax), %edx
        testl   %eax, %eax
        cmovs   %edx, %eax
        sarl    $2, %eax
        movl    %eax, -8(%rbp)
        movl    -16(%rbp), %eax
        movl    %eax, %edx
        shrl    $31, %edx
        addl    %edx, %eax
        sarl    %eax
        movl    %eax, -4(%rbp)
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
```

```
        .cfi_endproc
.LFE1493:
        .size   main, .-main
        .type   _Z41__static_initialization_and_destruction_0ii, @fu
_Z41__static_initialization_and_destruction_0ii:
.LFB1974:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    %edi, -4(%rbp)
        movl    %esi, -8(%rbp)
        cmpl    $1, -4(%rbp)
        jne     .L5
        cmpl    $65535, -8(%rbp)
        jne     .L5
        leaq    _ZStL8__ioinit(%rip), %rdi
        call    _ZNSt8ios_base4InitC1Ev@PLT
        leaq    __dso_handle(%rip), %rdx
        leaq    _ZStL8__ioinit(%rip), %rsi
        movq    _ZNSt8ios_base4InitD1Ev@GOTPCREL(%rip), %rax
        movq    %rax, %rdi
        call    __cxa_atexit@PLT
.L5:
        nop
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1974:
        .size   _Z41__static_initialization_and_destruction_0ii, .-_
        .type   _GLOBAL__sub_I_main, @function
_GLOBAL__sub_I_main:
.LFB1975:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $65535, %esi
        movl    $1, %edi
        call    _Z41__static_initialization_and_destruction_0ii
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1975:
        .size   _GLOBAL__sub_I_main, .-_GLOBAL__sub_I_main
```

```
        .section        .init_array,"aw"
        .align 8
        .quad   _GLOBAL__sub_I_main
        .hidden __dso_handle
        .ident  "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"
        .section        .note.GNU-stack,"",@progbits
```

Solución:
La división entre 4 se da en las siguientes líneas:

```
    movl        -16(%rbp), %eax
leal    3(%rax), %edx
testl   %eax, %eax
cmovs   %edx, %eax
sarl    $2, %eax
movl    %eax, -8(%rbp)
```

4. Redacta el siguiente cóodigo, genera código ensamblador y explica en qué parte (del código ensamblador) se define la división entre 2. (2 puntos).

```
int main(){
    char* c = "abcdef";
    int m = 11148;
    int x = m/8;
    int y = m/4;
    int z = m/2;
    return 0;
    }
```

Código Assembler:

```
        .file   "ej4.cpp"
        .text
        .section        .rodata
        .type   _ZStL19piecewise_construct, @object
        .size   _ZStL19piecewise_construct, 1
_ZStL19piecewise_construct:
        .zero   1
        .local  _ZStL8__ioinit
        .comm   _ZStL8__ioinit,1,1
        .text
        .globl  main
        .type   main, @function
main:
.LFB1493:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $11148, -16(%rbp)
```

```
        movl    -16(%rbp), %eax
        leal    7(%rax), %edx
        testl   %eax, %eax
        cmovs   %edx, %eax
        sarl    $3, %eax
        movl    %eax, -12(%rbp)
        movl    -16(%rbp), %eax
        leal    3(%rax), %edx
        testl   %eax, %eax
        cmovs   %edx, %eax
        sarl    $2, %eax
        movl    %eax, -8(%rbp)
        movl    -16(%rbp), %eax
        movl    %eax, %edx
        shrl    $31, %edx
        addl    %edx, %eax
        sarl    %eax
        movl    %eax, -4(%rbp)
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1493:
        .size   main, .-main
        .type   _Z41__static_initialization_and_destruction_0ii, @fu
_Z41__static_initialization_and_destruction_0ii:
.LFB1974:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    %edi, -4(%rbp)
        movl    %esi, -8(%rbp)
        cmpl    $1, -4(%rbp)
        jne     .L5
        cmpl    $65535, -8(%rbp)
        jne     .L5
        leaq    _ZStL8__ioinit(%rip), %rdi
        call    _ZNSt8ios_base4InitC1Ev@PLT
        leaq    __dso_handle(%rip), %rdx
        leaq    _ZStL8__ioinit(%rip), %rsi
        movq    _ZNSt8ios_base4InitD1Ev@GOTPCREL(%rip), %rax
        movq    %rax, %rdi
        call    __cxa_atexit@PLT
.L5:
        nop
        leave
        .cfi_def_cfa 7, 8
```

```
        ret
        .cfi_endproc
.LFE1974:
        .size   _Z41__static_initialization_and_destruction_0ii , .-_
        .type   _GLOBAL__sub_I_main , @function
_GLOBAL__sub_I_main:
.LFB1975:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp , %rbp
        .cfi_def_cfa_register 6
        movl    $65535 , %esi
        movl    $1 , %edi
        call    _Z41__static_initialization_and_destruction_0ii
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1975:
        .size   _GLOBAL__sub_I_main , .-_GLOBAL__sub_I_main
        .section        .init_array ,"aw"
        .align 8
        .quad   _GLOBAL__sub_I_main
        .hidden __dso_handle
        .ident  "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"
        .section        .note.GNU-stack ,"",@progbits
```

Solución:
La división entre 2 se da en las siguientes líneas:

```
movl    -16(%rbp), %eax
movl    %eax, %edx
shrl    $31, %edx
addl    %edx, %eax
sarl    %eax
movl    %eax, -4(%rbp)
```

5. Redacta el siguiente código, genera el código ensamblador y explica: (4 puntos):

En qué parte del código ensamblador se define la función div4.
En qué parte del código ensamblador se invoca a la función div4.
En qué parte del código ensamblador dentro de la función div4 se procesa la división.

```
int div4(int x){
    return x/4;
    }
int main(){
    char* c = "abcdef";
    int m = 11148;
```

```
int x = m/8;
int y = m/4;
int z = m/2;
int rpt = div4(5);
return 0;
}
```

Código Assembler:

```
        .file   "ej5.cpp"
        .text
        .section        .rodata
        .type   _ZStL19piecewise_construct, @object
        .size   _ZStL19piecewise_construct, 1
_ZStL19piecewise_construct:
        .zero   1
        .local  _ZStL8__ioinit
        .comm   _ZStL8__ioinit,1,1
        .text
        .globl  _Z4div4i
        .type   _Z4div4i, @function
_Z4div4i:
.LFB1493:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    %edi, -4(%rbp)
        movl    -4(%rbp), %eax
        leal    3(%rax), %edx
        testl   %eax, %eax
        cmovs   %edx, %eax
        sarl    $2, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1493:
        .size   _Z4div4i, .-_Z4div4i
        .globl  main
        .type   main, @function
main:
.LFB1494:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $32, %rsp
```

```
        movl    $11148, -20(%rbp)
        movl    -20(%rbp), %eax
        leal    7(%rax), %edx
        testl   %eax, %eax
        cmovs   %edx, %eax
        sarl    $3, %eax
        movl    %eax, -16(%rbp)
        movl    -20(%rbp), %eax
        leal    3(%rax), %edx
        testl   %eax, %eax
        cmovs   %edx, %eax
        sarl    $2, %eax
        movl    %eax, -12(%rbp)
        movl    -20(%rbp), %eax
        movl    %eax, %edx
        shrl    $31, %edx
        addl    %edx, %eax
        sarl    %eax
        movl    %eax, -8(%rbp)
        movl    $5, %edi
        call    _Z4div4i
        movl    %eax, -4(%rbp)
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1494:
        .size   main, .-main
        .type   _Z41__static_initialization_and_destruction_0ii, @fu
_Z41__static_initialization_and_destruction_0ii:
.LFB1975:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    %edi, -4(%rbp)
        movl    %esi, -8(%rbp)
        cmpl    $1, -4(%rbp)
        jne     .L7
        cmpl    $65535, -8(%rbp)
        jne     .L7
        leaq    _ZStL8__ioinit(%rip), %rdi
        call    _ZNSt8ios_base4InitC1Ev@PLT
        leaq    __dso_handle(%rip), %rdx
        leaq    _ZStL8__ioinit(%rip), %rsi
        movq    _ZNSt8ios_base4InitD1Ev@GOTPCREL(%rip), %rax
        movq    %rax, %rdi
        call    __cxa_atexit@PLT
```

```
.L7:
        nop
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1975:
        .size   _Z41__static_initialization_and_destruction_0ii, .-_
        .type   _GLOBAL__sub_I__Z4div4i, @function
_GLOBAL__sub_I__Z4div4i:
.LFB1976:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $65535, %esi
        movl    $1, %edi
        call    _Z41__static_initialization_and_destruction_0ii
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1976:
        .size   _GLOBAL__sub_I__Z4div4i, .-_GLOBAL__sub_I__Z4div4i
        .section        .init_array,"aw"
        .align 8
        .quad   _GLOBAL__sub_I__Z4div4i
        .hidden __dso_handle
        .ident  "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"
        .section        .note.GNU-stack,"",@progbits
```

Soluciones:
La función div4 se define en las siguientes líneas:

```
_ZStL19piecewise_construct:
.zero   1
.local  _ZStL8__ioinit
.comm   _ZStL8__ioinit,1,1
.text
.globl  _Z4div4i
.type   _Z4div4i, @function
_Z4div4i:
.LFB1493:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    %edi, -4(%rbp)
```

```
movl    -4(%rbp), %eax
leal    3(%rax), %edx
testl   %eax, %eax
cmovs   %edx, %eax
sarl    $2, %eax
popq    %rbp
.cfi_def_cfa 7, 8
ret
.cfi_endproc
```

La función div4 se invoca en las siguientes líneas:

```
movl    $5, %edi
call    _Z4div4i
```

La división se proecesa en las siguientes líneas

```
movl    %edi, -4(%rbp)
movl    -4(%rbp), %eax
leal    3(%rax), %edx
testl   %eax, %eax
cmovs   %edx, %eax
sarl    $2, %eax
popq    %rbp
```

6. Redacta el siguiente código, genera el código ensamblador y explica: (4 puntos):

En qué parte del código ensamblador se define la función div.
En qué parte del código ensamblador se invoca a la función div.
En qué parte del código ensamblador dentro de la función div se procesa la división.

```
int div(int x, int y){
    return x/y;
    }
int div4(int x){
    return x/4;
    }
int main(){
    char* c = "abcdef";
    int m = 11148;
    int x = m/8;
    int y = m/4;
    int z = m/2;
    int rpt = div(5,4);
    int rpt2 = div4(5);
    return 0;
    }
```

Código Assembler:

```
    .file   "ej6.cpp"
    .text
    .section        .rodata
    .type   _ZStL19piecewise_construct, @object
    .size   _ZStL19piecewise_construct, 1
_ZStL19piecewise_construct:
    .zero   1
    .local  _ZStL8__ioinit
    .comm   _ZStL8__ioinit,1,1
    .text
    .globl  _Z8divii
    .type   _Z8divii, @function
_Z8divii:
.LFB1493:
    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    movl    %edi, -4(%rbp)
    movl    %esi, -8(%rbp)
    movl    -4(%rbp), %eax
    cltd
    idivl   -8(%rbp)
    popq    %rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE1493:
    .size   _Z8divii, .-_Z8divii
    .globl  _Z4div4i
    .type   _Z4div4i, @function
_Z4div4i:
.LFB1494:
    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    movl    %edi, -4(%rbp)
    movl    -4(%rbp), %eax
    leal    3(%rax), %edx
    testl   %eax, %eax
    cmovs   %edx, %eax
    sarl    $2, %eax
    popq    %rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE1494:
```

```
        .size   _Z4div4i, .-_Z4div4i
        .globl  main
        .type   main, @function
main:
.LFB1495:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $32, %rsp
        movl    $11148, -20(%rbp)
        movl    -20(%rbp), %eax
        leal    7(%rax), %edx
        testl   %eax, %eax
        cmovs   %edx, %eax
        sarl    $3, %eax
        movl    %eax, -16(%rbp)
        movl    -20(%rbp), %eax
        leal    3(%rax), %edx
        testl   %eax, %eax
        cmovs   %edx, %eax
        sarl    $2, %eax
        movl    %eax, -12(%rbp)
        movl    -20(%rbp), %eax
        movl    %eax, %edx
        shrl    $31, %edx
        addl    %edx, %eax
        sarl    %eax
        movl    %eax, -8(%rbp)
        movl    $4, %esi
        movl    $5, %edi
        call    _Z8divii
        movl    %eax, -4(%rbp)
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1495:
        .size   main, .-main
        .type   _Z41__static_initialization_and_destruction_0ii, @fu
_Z41__static_initialization_and_destruction_0ii:
.LFB1976:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
```

```
        movl    %edi, -4(%rbp)
        movl    %esi, -8(%rbp)
        cmpl    $1, -4(%rbp)
        jne     .L9
        cmpl    $65535, -8(%rbp)
        jne     .L9
        leaq    _ZStL8__ioinit(%rip), %rdi
        call    _ZNSt8ios_base4InitC1Ev@PLT
        leaq    __dso_handle(%rip), %rdx
        leaq    _ZStL8__ioinit(%rip), %rsi
        movq    _ZNSt8ios_base4InitD1Ev@GOTPCREL(%rip), %rax
        movq    %rax, %rdi
        call    __cxa_atexit@PLT
.L9:
        nop
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1976:
        .size   _Z41__static_initialization_and_destruction_0ii, .-_
        .type   _GLOBAL__sub_I__Z8divii, @function
_GLOBAL__sub_I__Z8divii:
.LFB1977:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $65535, %esi
        movl    $1, %edi
        call    _Z41__static_initialization_and_destruction_0ii
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1977:
        .size   _GLOBAL__sub_I__Z8divii, .-_GLOBAL__sub_I__Z8divii
        .section        .init_array,"aw"
        .align 8
        .quad   _GLOBAL__sub_I__Z8divii
        .hidden __dso_handle
        .ident  "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"
        .section        .note.GNU-stack,"",@progbits
```

Soluciones:

La función div se define en las siguientes líneas:

```
_ZStL19piecewise_construct:
```

```
.zero   1
.local  _ZStL8__ioinit
.comm   _ZStL8__ioinit,1,1
.text
.globl  _Z8divii
.type   _Z8divii, @function
_Z8divii:
.LFB1493:
.cfi_startproc
pushq   %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq    %rsp, %rbp
.cfi_def_cfa_register 6
movl    %edi, -4(%rbp)
movl    %esi, -8(%rbp)
movl    -4(%rbp), %eax
cltd
idivl   -8(%rbp)
popq    %rbp
.cfi_def_cfa 7, 8
ret
.cfi_endproc
```

La función div se invoca en las siguientes líneas:

```
movl    $4, %esi
movl    $5, %edi
call    _Z8divii
```

La división se procesa en las siguientes líneas:

```
movl    %edi, -4(%rbp)
movl    %esi, -8(%rbp)
movl    -4(%rbp), %eax
cltd
idivl   -8(%rbp)
popq    %rbp
```

7. De las preguntas anteriores, se ha generado código por cada función, ambas dividen entre 4, pero difieren un poco en su implementación. Investigue a qué se debe dicha diferencia y comente cuáles podrían ser las consecuencias. (4 puntos)

   La función Div recibe dos parámetros los cuales representan el divisor y el dividendo y estas son variables o sea que podríamos dividir entre cualquier número, mientras que en la función Div4 sólo recibe un parámetro, el cual representa el divisor y en la función se divide entre una constante 4.