3_1)
```
        .file   "lab5_prog3_1.c"
        .section        .rodata         \\ This is preparing the read only data in the program.
.LC0:
        .string "Hello, world"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc                  \\This is used at the beginning of each function that should
```
have an entry in `.eh_frame`. It initializes some internal data structures.

```
        pushq   %rbp                    \\ This is pushing the register rbp into the stack
        .cfi_def_cfa_offset 16          \\The change of stack pointer is declared in the debugging
```
information using the `.cfi_def_cfa_offset` directive, and you can see that the CFA is now at an offset of 16 bytes from the current stack pointer.

```
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6         \\modifies a rule for computing CFA. From now on register
```
will be used instead of the old one. Offset remains the same.

```
        subq    $16, %rsp               \\ An Immediate operand, this one is marked with a long
```
value 16
```
        movl    %edi, -4(%rbp)          \\Indexing or indirection is done by enclosing the index
```
register or indirection memory cell address in parentheses. This moves the contents from offset -4 to the cell pointed at by rbp to register edi

```
        movq    %rsi, -16(%rbp)
        movl    $.LC0, %edi
        call    puts
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8               \\Defines a rule for computing CFA as: take address from
```
register and add offset to it.

```
        ret
        .cfi_endproc                    \\ Used at the end of a function where it closes its unwind
```
entry previously opened by `.cfi_startproc`, and emits it to `.eh_frame`.
```
.LFE0:
        .size   main, .-main
        .ident  "GCC: (SUSE Linux) 4.7.3"
        .section        .note.GNU-stack,"",@progbits
```

3_2)

```
        .file   "lab5_prog3_2.c"
        .section        .rodata
.LC0:
        .string  "The value of i is %d\n"
        .text
        .globl   main
        .type    main, @function
main:
.LFB0:
        .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    %edi, -20(%rbp)
        movq    %rsi, -32(%rbp)
        movl    $1, -4(%rbp)
        addl    $1, -4(%rbp)                    \\the values are added together and the result is
```

stored in `rbp`. However, memory indirect addressing is used (this is denoted by parentheses). This means that `rbp` is treated as a pointer, so the right operand is taken from the address pointed to by `rbp`, and the result is stored to the same address.

```
        movl    -4(%rbp), %eax
        movl    %eax, %esi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size    main, .-main
        .ident   "GCC: (SUSE Linux) 4.7.3"
        .section        .note.GNU-stack,"",@progbits
```

```
4)      .file   "lab5_prog4.c"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .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)
        movq   %rsi, -16(%rbp)
        movl   $0, %eax
        call   print_hello
        movl   $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .section        .rodata
.LC0:
        .string  "Hello, world"
        .text
        .globl  print_hello
        .type   print_hello, @function
print_hello:
.LFB1:
        .cfi_startproc
        pushq  %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq   %rsp, %rbp
        .cfi_def_cfa_register 6
        movl   $.LC0, %edi
        call   puts
        popq   %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1:
        .size   print_hello, .-print_hello
        .ident  "GCC: (SUSE Linux) 4.7.3"
        .section        .note.GNU-stack,"",@progbits
```

```
5)      .file   "lab5_prog5_main.c"
        .section        .rodata
.LC0:
        .string  "Hello, world"
        .text
        .globl   print_hello
        .type    print_hello, @function
print_hello:
.LFB0:
        .cfi_startproc
        pushq  %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq   %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $.LC0, %edi
        call    puts
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size    print_hello, .-print_hello
        .globl   main
        .type    main, @function
main:                                           \\Notice that main comes after print_hello() in prob5
.LFB1:
        .cfi_startproc
        pushq  %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq   %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp               \\This line and command was done in print_hello in prob4
        movl    %edi, -4(%rbp)
        movq   %rsi, -16(%rbp)
        movl    $0, %eax
        call    print_hello
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1:
        .size    main, .-main
        .ident   "GCC: (SUSE Linux) 4.7.3"
        .section        .note.GNU-stack,"",@progbits
```