School of Mathematical and Computational Sciences

Indian Association for the Cultivation of Science

Compiler Construction: COM 5202

Tutorial I (07 January, 2026)

M. Sc Semester IV: 2025-2026 *Instructor:* Goutam Biswas

1. Explanation of the assembly code of `function.c`

```
#/* function.c contains a function */
#void function(int d[], int n){
#     int i;
#
#     for(i=0; i<n; ++i) d[i] = 6*d[i];
#}

   .file   "function.c"       # Function name
   .globl   function          # 'function' is
                              # a global name
   .type   function, @function # function is a
                              # function
function:                     # Label function
.LFB0:
   .cfi_startproc
   pushq   %rbp               # push rbp
   .cfi_def_cfa_offset 16
   .cfi_offset 6, -16
   movq   %rsp, %rbp          # rbp <-- rsp
   .cfi_def_cfa_register 6
   movq   %rdi, -24(%rbp)     # M[rbp-24] <-- rdi
                              # 1st parameter, d[],
                              # starting address of
                              #  the array
   movl   %esi, -28(%rbp)     # M[ebp-28] <-- esi
                              # 2nd parameter, n
   movl   $0, -4(%rbp)        # M[rbp-4] (i) <-- 0
   jmp   .L2                  # goto .L2
                              #  loop condition test
.L3:                          # .L3, loop starts
   movl   -4(%rbp), %eax      # eax <-- i
   cltq                       # rax <-- eax, 32-bit to
                              #    64-bit
   leaq   0(,%rax,4), %rdx    # rdx <-- 4*rax (4*i)
   movq   -24(%rbp), %rax     # rax <-- d[]
   addq   %rdx, %rax          # rax <-- d+4*i (&d[i])
   movl   (%rax), %edx        # edx <-- M[rax], d[i]
   movl   -4(%rbp), %eax      # eax <-- i
   cltq                       # rax <-- eax, 32-bit to
                              #    64-bit
```

```
   leaq   0(,%rax,4), %rcx    # rcx <-- 4*rax (4*i)
   movq   -24(%rbp), %rax     # rax <-- d[]
   addq   %rax, %rcx          # rcx <-- d+4*i (&d[i])
   movl   %edx, %eax          # eax <-- edx (d[i])
   addl   %eax, %eax          # eax <-- eax+eax (2*d[i])
   addl   %edx, %eax          # eax <-- eax+ edx (d[i]+
                              #          2*d[i]=3*d[i])
   addl   %eax, %eax          # eax <-- eax+eax (2*3*d[i]
                              #         = 6*d[i])
   movl   %eax, (%rcx)        # M[rcx] <-- eax
                              #   (d[i] = 6*d[i])
   addl   $1, -4(%rbp)        # i <-- i+1
.L2:                          # Loop condition test
   movl   -4(%rbp), %eax      # eax <-- i
   cmpl   -28(%rbp), %eax     # if i (eax) < n (M[rbp-4])
   jl   .L3                   #   goto .L3
   nop                        # no operation
   nop                        # no operation
   popq   %rbp                # pop rbp
   .cfi_def_cfa 7, 8
   ret                        # return
   .cfi_endproc
.LFE0:
   .size   function, .-function
   .ident   "GCC: (Ubuntu 9.4.0-1ubuntu1~18.04) 9.4.0"
   .section   .note.GNU-stack,"",@progbits
```

2. Could not complete the explanation of the assembly code of `main.c`

```
#/*  main.c */
##include <stdio.h>
##define MAXNO 100
#void function(int [], int);
#int main() // main.c
#{
#    int no = 0, i ;
#    int data[MAXNO] ;
#
#    printf("Enter the data, terminate with Ctrl+D\n") ;
#    while(scanf("%d", &data[no]) != EOF) ++no;
#    function(data, no) ;
#    printf("Data after processing: ") ;
#    for(i = 0; i < no; ++i) printf("%d ", data[i]);
#    putchar('\n') ;
#    return 0 ;
#}

   .file   "main.c"           # source file name
   .section   .rodata         # read-only data section starts
   .align 8                   # align with 8-byte boundary
```

```
.LC0:                          # Label of string - 1st printf
   .string   "Enter the data, terminate with Ctrl+D"
.LC1:                          # Label of string scanf
   .string   "%d"
.LC2:                          # Label of string - 2nd printf
   .string   "Data after processing: "
.LC3:                          # Label of string - 3rd printf
   .string   "%d "
   .text                       # Code or text starts
   .globl   main               # main is a global name
   .type   main, @function     # main is function
main:                          # Label main:
.LFB0:
   .cfi_startproc
   pushq   %rbp                # Save old base pointer
   .cfi_def_cfa_offset 16
   .cfi_offset 6, -16
   movq   %rsp, %rbp           # rbp <-- rsp set new
   .cfi_def_cfa_register 6     #     base pointer
   subq   $432, %rsp           # Create space for local
                               # array and variables
   movq   %fs:40, %rax         # Save segment info
   movq   %rax, -8(%rbp)       # at M[rbp-8]
   xorl   %eax, %eax           # eax <-- 0
   movl   $0, -424(%rbp)       # no <-- 0
   leaq   .LC0(%rip), %rdi     # rdi <-- 1st parameter
                               #             of printf
   call   puts@PLT             # Calls puts for printf
   jmp   .L2                   # Goto the beginning of the
                               # while loop
.L3:
   addl   $1, -424(%rbp)       # no <-- no+1
.L2:                           # Body of the loop
   leaq   -416(%rbp), %rax     # rax <-- data
   movl   -424(%rbp), %edx     # edx <-- no
   movslq   %edx, %rdx         # rdx <-- edx (no) (32-bits
                               #           to 64-bits)
   salq   $2, %rdx             # rdx <-- 4*rdx (no)
   addq   %rdx, %rax           # rax <-- data + 4*no
                               #           (&data[no])
   movq   %rax, %rsi           # rsi <-- rax (&data[no])
                               #   2nd parameter
   leaq   .LC1(%rip), %rdi     # rdi <-- (addr. of format str)
                               #   1st parameter
   movl   $0, %eax             # eax <-- 0
   call   __isoc99_scanf@PLT   # call to scanf
                               # The return value is in eax
   cmpl   $-1, %eax            # if return value
                               #          != -1 (EOF)
   jne   .L3                   # goto .L3 (loop)
```

```
   movl   -424(%rbp), %edx    # edx <-- no
   leaq   -416(%rbp), %rax    # rax <-- data
   movl   %edx, %esi          # esi <-- edx (no) 2nd
                              #         parameter
   movq   %rax, %rdi          # rdi <-- rax (data), 1st
                              #         parameter
   call   function@PLT        # call to selection sort
   leaq   .LC2(%rip), %rdi    # rdi <-- format str address
                              #    1st param
   movl   $0, %eax            # eax <-- 0
   call   printf@PLT          # call to printf
   movl   $0, -420(%rbp)      # i <-- 0
   jmp   .L4                  # go to .L4
.L5:
   movl   -420(%rbp), %eax    # eax <-- i
   cltq                       # rax <-- eax (sign ext.)
   movl   -416(%rbp,%rax,4), %eax # eax <--
                              # Mem[(rbp - 416) + 4*rax]
                              # eax <-- data[i]
   movl   %eax, %esi          # esi <-- eax (data[i]),
                              #    2nd parameter
   leaq   .LC3(%rip), %rdi    # rdi <-- addr. format str,
                              #     1st parameter
   movl   $0, %eax            # eax <-- 0
   call   printf@PLT          # call to printf
   addl   $1, -420(%rbp)      # i <-- i + 1
.L4:                          # Label to test loop cond
   movl   -420(%rbp), %eax    # eax <-- i
   cmpl   -424(%rbp), %eax    # if i < no (jl is jump <)
   jl   .L5                   #    goto .L5
   movl   $10, %edi           # edi <-- 10 ('\n')
   call   putchar@PLT         # call putchar
   movl   $0, %eax            # eax <-- 0
   movq   -8(%rbp), %rcx      # Restore return info
   xorq   %fs:40, %rcx
   je   .L7                   # return
   call   __stack_chk_fail@PLT
.L7:
   leave                      # return
   .cfi_def_cfa 7, 8
   ret
   .cfi_endproc
.LFE0:
   .size   main, .-main
   .ident   "GCC: (Ubuntu 9.4.0-1ubuntu1~18.04) 9.4.0"
   .section   .note.GNU-stack,"",@progbits
```