LAB 4:

- In this lab the code is going to run on TivaC board of the following properties:
  - Processor ARM CortexM4.
  - SRAM 32KB.
  - Flash Memory 256KB.
- First editing (main.c and startup.c):

  main.c: defined GPIO Port F registers' addresses and initialized GPIO Port F.

```c
#include "Platform_Types.h"

#define SYSCTL_RCG2_R     *((volatile uint32*)0x400FE108)
#define GPIO_PORTF_DIR_R  0x40025400
#define GPIO_PORTF_DEN_R  0x4002551C
#define GPIO_PORTF_DATA_R 0x400253FC


typedef union{
    volatile uint32 ALL_FIELDS;
    struct {
        volatile uint32 :3;
        volatile uint32 pin3:1;
    }pins;
}GPIO_PF_Rs;



volatile GPIO_PF_Rs * dir_R= (volatile GPIO_PF_Rs *)(GPIO_PORTF_DIR_R);
volatile GPIO_PF_Rs * den_R= (volatile GPIO_PF_Rs *)(GPIO_PORTF_DEN_R);
volatile GPIO_PF_Rs * data_R= (volatile GPIO_PF_Rs *)(GPIO_PORTF_DATA_R);

static uint32 i;

int main(void)
{
    SYSCTL_RCG2_R=0x20;

    for(i=0;i<200;i++);
    dir_R->pins.pin3=1;
    den_R->pins.pin3=1;

    while(1){
        data_R->pins.pin3=1;
        for(i=0;i<200000;i++);
        data_R->pins.pin3=0;
        for( i=0;i<200000;i++);
    }
}
```

startup.c: defined stack_top without using the linker script and edited the vectors to be array of pointers to functions instead of array of integers.

```c
/*startup.c Mostafa Mahmoud*/
#include "Platform_Types.h"

extern int main(void);
void Reset_Handler();
void Default_Handler(){
    Reset_Handler();
}
void NMI_Handler() __attribute__((weak,alias("Default_Handler")));;
void H_fault_Handler()__attribute__((weak,alias("Default_Handler")));;

extern uint32 _E_text;
extern uint32 _S_data;
extern uint32 _E_data;
extern uint32 _S_bss;
extern uint32 _E_bss;

static uint32 stack_top[256];

void (* const vectors[])() __attribute__((section(".vectors"))) ={

   (void(*)())((uint32) stack_top+sizeof(stack_top)),
    Reset_Handler,
    NMI_Handler,
    H_fault_Handler,
  };
void Reset_Handler(){
/*copy data section*/
 uint32 i;
 uint32 DATA_size= (uint8*)&_E_data-(uint8*)&_S_data;
 uint8 *P_src =(uint8*)&_E_text;
 uint8 *P_dst =(uint8*)&_S_data;
 for(i=0;i<DATA_size;i++){
     *((uint8*)P_dst++)=*((uint8*)P_src++);
    }
/*init bss section*/
uint32 BSS_size=(uint8*)&_E_bss-(uint8*)_S_bss;
P_dst=(uint8*)&_S_bss;
for(i=0;i<BSS_size;i++){
    *((uint8*)P_dst++)=(uint8)0;
}
/*call main()*/
    main();
}
```

- # Editing linker_script.ld:

linker_script.ld: just removed the stack_top symbol and edited the addresses of SRAM and Flash.

```
/*linker script CortexM4
Mostafa Mahmoud */

MEMORY
{
flash (rx) : ORIGIN = 0x00000000, LENGTH = 256k
sram (rwx) : ORIGIN = 0x20000000, LENGTH = 32k
}

SECTIONS{

   .text :
   {
    *(.vectors*)
    *(.text*)
    *(.rodata*)
    _E_text = . ;
   }>flash

   .data :
   {
    _S_data = . ;
     *(.data)
    . = ALIGN(4) ;
     _E_data = . ;
   }> sram AT> flash

   .bss :
   {
    _S_bss = . ;
    *(.bss*)
    . = ALIGN(4);
    _E_bss = . ;
   }>sram

}
```

- Now editing Makefile:

```makefile
#@copyright : Mostafa Mahmoud
CC=arm-none-eabi-
CFLAGS=-mthumb -mcpu=cortex-m4 -gdwarf-2 -g
INCS=-I.
LIBS=
SRC= $(wildcard *.c)
OBJ= $(SRC:.c=.o)
As= $(wildcard *.s)
AsOBJ= $(As:.s=.o)
Project_name=unit3_lab4_coretxM4



all: $(Project_name).bin
    @echo "======Build is Done=========="


%.o: %.c
    $(CC)gcc.exe -c $(INCS) $(CFLAGS) $< -o $@

$(Project_name).elf: $(OBJ) $(AsOBJ)
    $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $@ -Map=Map_file.map
    cp $(Project_name).elf $(Project_name).axf

$(Project_name).bin: $(Project_name).elf
    $(CC)objcopy.exe -O binary $< $@


clean_all:
    rm *.o *.elf *.bin

clean:
    rm *.elf *.bin
```

- Building files:

```
MINGW32:/d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_diploma/Unit3_EmbeddedC/Assignment4

mosta@Mostafa-PC MINGW32 /d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_dipl
oma/Unit3_EmbeddedC/Assignment4 (main)
$ make
arm-none-eabi-gcc.exe -c -I. -mthumb -mcpu=cortex-m4 -gdwarf-2 -g main.c -o main.o
arm-none-eabi-gcc.exe -c -I. -mthumb -mcpu=cortex-m4 -gdwarf-2 -g startup.c -o startup.o
arm-none-eabi-ld.exe -T linker_script.ld  main.o startup.o  -o unit3_lab4_coretxM4.elf -Map=Map_file.map
cp unit3_lab4_coretxM4.elf unit3_lab4_coretxM4.axf
arm-none-eabi-objcopy.exe -O binary unit3_lab4_coretxM4.elf unit3_lab4_coretxM4.bin
======Build is Done=========

mosta@Mostafa-PC MINGW32 /d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_dipl
oma/Unit3_EmbeddedC/Assignment4 (main)
$
```

- Checking symbols in main.o, startup.o and final executable .elf:

```
MINGW32:/d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_diploma/Unit3_EmbeddedC/Assignment4
mosta@Mostafa-PC MINGW32 /d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_dipl
oma/Unit3_EmbeddedC/Assignment4 (main)
$ arm-none-eabi-nm.exe main.o
00000008 D data_R
00000004 D den_R
00000000 D dir_R
00000000 b i
00000000 T main

mosta@Mostafa-PC MINGW32 /d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_dipl
oma/Unit3_EmbeddedC/Assignment4 (main)
$ arm-none-eabi-nm.exe startup.o
         U _E_bss
         U _E_data
         U _E_text
         U _S_bss
         U _S_data
00000000 T Default_Handler
00000000 W H_fault_Handler
         U main
00000000 W NMI_Handler
0000000c T Reset_Handler
00000000 b stack_top
00000000 R vectors

mosta@Mostafa-PC MINGW32 /d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_dipl
oma/Unit3_EmbeddedC/Assignment4 (main)
$ arm-none-eabi-nm.exe unit3_lab4_coretxM4.elf
20000410 B _E_bss
2000000c D _E_data
000001e0 T _E_text
2000000c B _S_bss
20000000 D _S_data
20000008 D data_R
00000120 T Default_Handler
20000004 D den_R
20000000 D dir_R
00000120 W H_fault_Handler
2000000c b i
00000010 T main
00000120 W NMI_Handler
0000012c T Reset_Handler
20000010 b stack_top
00000000 T vectors
```

- Simulating and debugging on KEIL: