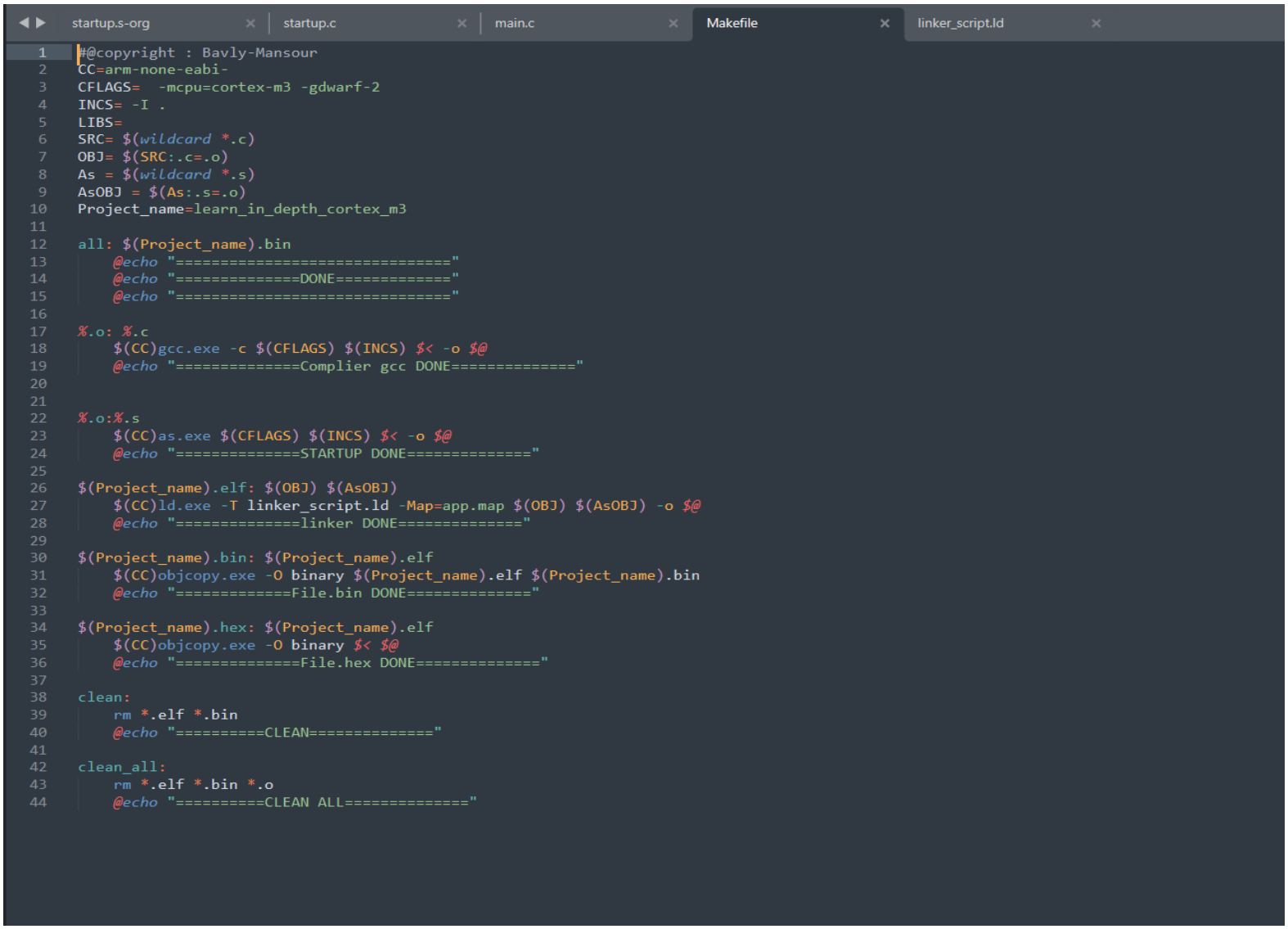
Lesson 3 - LAB 2 – Report

Makefile:

*This Makefile compatible with CPU: Cortex-M3



* -gdwarf-2 instead of -g with Cortex-M3.

* $@ is for target, $< for dependency.

* % is a generalization means all of.

Output of Makefile:

```
Bavly@DESKTOP-CNE79H4 MINGW64 /d/Embedded system diploma/Embedded System Diploma
/Diploma Assignmets/First_Term/Unit_3_Embedded_C/Lesson 3/Lab 2 (master)
$ make clean_all
rm *.elf *.bin *.o
==========CLEAN ALL=============

Bavly@DESKTOP-CNE79H4 MINGW64 /d/Embedded system diploma/Embedded System Diploma
/Diploma Assignmets/First_Term/Unit_3_Embedded_C/Lesson 3/Lab 2 (master)
$ make
arm-none-eabi-gcc.exe -c -mcpu=cortex-m3 -gdwarf-2 -I . main.c -o main.o
==============Complier gcc DONE==============
arm-none-eabi-gcc.exe -c -mcpu=cortex-m3 -gdwarf-2 -I . startup.c -o startup.o
==============Complier gcc DONE==============
arm-none-eabi-ld.exe -T linker_script.ld -Map=app.map main.o startup.o  -o learn
_in_depth_cortex_m3.elf
==============linker DONE==============
arm-none-eabi-objcopy.exe -O binary learn_in_depth_cortex_m3.elf learn_in_depth_
cortex_m3.bin
=============File.bin DONE=============
================================
==============DONE=============
================================
```

Main.c file :

startup.s-org | startup.c | main.c

```c
/**
 ******************************************************************************
 * @file           : main.c
 * @author         : Eng Bavly Mansour
 * @brief          : Main program body
 ******************************************************************************
 */

#include <stdint.h>
//#include <platform_Types.h>


#define RCC_BASE        0x40021000
#define GPIOA_BASE      0x40010800
#define RCC_APB2ENR     *(volatile uint32_t*) (RCC_BASE + 0x18)
// 0x18 is the offset of base memory address
#define GPIOA_CHR       *(volatile uint32_t*) (GPIOA_BASE +0x04)
// general purpose input output port A
#define GPIOA_ODR       *(volatile uint32_t*) (RCC_BASE + 0x0c)
// output data register

//#if !defined(__SOFT_FP__) && defined(__ARM_FP)
//  #warning "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU before use."
//#endif

unsigned char g_variable[3] = {1,2,3};
unsigned char const const_var[3] = {1,2,3};
unsigned char bss_var[3] ;

/*g_variable : global variable
const_var    : constant variable
bss_var      : uninitialized variable*/

int main(void)
{
    /* Loop forever */
    int i;
    RCC_APB2ENR |=(1<<2);
    GPIOA_CHR &= 0xff0ffff;
    GPIOA_CHR |= 0x00200000;
    while (1){

        GPIOA_ODR |=1<<13;
        // set 1 to a bit |= 1<<n
        for (i =0 ; i < 5000 ;  i++)
            GPIOA_ODR &=~ (1<<13);
        // clear a bit &=~(1<<n)
        for (i=0; i< 5000 ; i++);

    }
    return 0;
}
```

Startup.s file for Cortex-M3:

startup.s-org | startup.c | main.c | Makefile

```
/* startup_cortexM3.s By: Bavly Mansour*/

/*SRAM 0x20000000 */

.section .vectors

.word 0x20001000            /*stack top address*/
.word _reset                /*1 reset */
.word Vector_handler        /*2 NMI*/
.word Vector_handler        /*3 hard fault*/
.word Vector_handler        /*4 MM fault*/
.word Vector_handler        /*5 Bus fault*/
.word Vector_handler        /*6 Usage fault*/
.word Vector_handler        /*7 Reserved*/
.word Vector_handler        /*8 Reserved*/
.word Vector_handler        /*9 Reserved*/
.word Vector_handler        /*10 Reserved*/
.word Vector_handler        /*11 SV call*/
.word Vector_handler        /*12 Debug reserve*/
.word Vector_handler        /*13 Reserved*/
.word Vector_handler        /*14 PendSV*/
.word Vector_handler        /*15 SysTick*/
.word Vector_handler        /*16 TRQ0*/
.word Vector_handler        /*17 TRQ1*/
.word Vector_handler        /*18 TRQ2*/
.word Vector_handler        /*19 ...  */
                /* on to IRQ67*/

.section .text

_reset:
    bl main
    b .

.thumb_func     /* dealing with 16 and 32 bits*/

Vector_handler :
    b _reset

```

- .word is for enabling memory alignment.
- .thumb_func dealing with 16 and 32 bits instructions.

Startup.c file for Cortex-M3:

startup.s-org | startup.c | main.c | Makefile | linker_script.ld

```c
/*
**************************************
**************************************
learn-in-depth
By:Bavly Mansour
**************************************
**************************************
*/

#include <stdint.h>
#define STACK_Start_SP 0x20001000  //txt replacement of STACK_Start_SP "stack pointer" to address 0x20001000
extern int main (void);            // define external int main (void)
void Rest_Handler (void);          //define rest_handler function


/*Copying data from flash to SRAM and allocate space
 for bss and stack according to the memory bourders */


void NMI_Handler() __attribute__ ((weak,alias("Default_Handler")));;
void H_Fault_Handler() __attribute__ ((weak,alias("Default_Handler")));;
void MM_Fault_Handler() __attribute__ ((weak,alias("Default_Handler")));;
void Bus_Fault() __attribute__ ((weak,alias("Default_Handler")));;
void Usage_Fault_Handler() __attribute__ ((weak,alias("Default_Handler")));;


extern unsigned int _E_text;
extern unsigned int _S_DATA;
extern unsigned int _E_DATA;
extern unsigned int _S_bss;
extern unsigned int _E_bss;
extern unsigned int _stack_top;

void Default_Handler ()
{
    Rest_Handler();
}

void Rest_Handler (void)
{
    // copying data from ROM to RAM
    unsigned int DATA_size = (unsigned char*)&_E_DATA - (unsigned char*)&_S_DATA;
    unsigned char* P_src = (unsigned char*)&_E_text ;
    unsigned char* P_dst = (unsigned char*)&_S_DATA ;

    for (int i =0 ; i < DATA_size ; i++){
        *((unsigned char *)P_dst++) = *((unsigned char*)P_src++) ;
    }

```

- Alias attribute causes the declaration to be emitted as an alias for another symbol.
- Weak attribute causes the declaration to be emitted as a weak symbol rather than a global.

- **Handler functions in MapFile :**

```
.text           0x08000000      0x137
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                vectors
 *(.text*)
 .text          0x0800001c       0x80 main.o
                0x0800001c                main
 .text          0x0800009c       0x98 startup.o
                0x0800009c                H_Fault_Handler
                0x0800009c                MM_Fault_Handler
                0x0800009c                Bus_Fault
                0x0800009c                Default_Handler
                0x0800009c                Usage_Fault_Handler
                0x0800009c                NMI_Handler
                0x080000a8                Rest_Handler
 *(.rodata)
 .rodata        0x08000134        0x3 main.o
                0x08000134                const_var
                0x08000137                _E_text = .
```

- Main at 0x0800001c
- And all Handler functions at the same address of the default address due to the alias. Default_Handler at 0x0800009c, while the Rest_handler has a different address at 0x080000a8.

Linker script for Cortex-M3:

startup.s-org | startup.c | main.c | Makefile | linker_script.ld

```
/*This is linker script for LAB2 CPU=CORTEX-M3 ,by : ENG_BAVLY*/

MEMORY
{
    flash(RX) : ORIGIN = 0X08000000 , LENGTH = 128K
    /*flash memory origin and length , flash: read and execute*/
    sram(RWX) : ORIGIN = 0X20000000 , LENGTH = 20K
    /*sRAM origin and length , sram : read write execute*/

}
/*sections*/
SECTIONS {
    .text :{
        *(.vectors*)
            *(.text*)
            *(.rodata)
            _E_text = .;           /*end of text section in flash @ 0x08000b3*/

        }> flash
                                /*everything inside .text put it in flash */

    .data :{
            _S_DATA = .;          /*start of data section @ 0x08000b3 the same  address of end text section */
            *(.data)
            . = ALIGN(4) ;
            _E_DATA = .;         /*end of data section */
        }>sram AT > flash        /*virtual memory :sram */

    .bss :{
            _S_bss = . ;         /*start of bss section */
            *(.bss*)
            . = ALIGN(4);
            _E_bss = . ;
            . = ALIGN(4);       /*end of bss section */
            . = . + 0x1000 ;
            _stack_top = . ;
        }>sram
/* bss are the uninitialized variables = zeros so ignore them from the flash */


```

- According to memory borders:
  _E_text is the end of .text section
  _S_DATA is the start of .data section
  _S_bss is the start of .bss section
- .text section includes .rodata section.
- ALIGN(4) is for memory alignment.

- Memory configuration according to linker script memory borders:

```
Memory Configuration

Name             Origin             Length             Attributes
flash            0x08000000         0x00020000         xr
sram             0x20000000         0x00005000         xrw
*default*        0x00000000         0xffffffff
```

- Linker script memory map:

```
.data           0x20000000        0x4 load address 0x08000137
                0x20000000                _S_DATA = .
 *(.data)
 .data          0x20000000        0x3 main.o
                0x20000000                g_variable
 .data          0x20000003        0x0 startup.o
                0x20000004                . = ALIGN (0x4)
 *fill*         0x20000003        0x1
                0x20000004                _E_DATA = .

.igot.plt       0x20000004        0x0 load address 0x0800013b
 .igot.plt      0x20000004        0x0 main.o

.bss            0x20000004     0x1003 load address 0x0800013b
                0x20000004                _S_bss = .
 *(.bss*)
 .bss           0x20000004        0x0 main.o
 .bss           0x20000004        0x0 startup.o
                0x20000004                . = ALIGN (0x4)
                0x20000004                _E_bss = .
                0x20000004                . = ALIGN (0x4)
                0x20001004                . = (. + 0x1000)
 *fill*         0x20000004     0x1000
                0x20001004                _stack_top = .
 COMMON         0x20001004        0x3 main.o
                0x20001004                bss_var
```

- The start of .bss section is the same address of the end of .data section &_E_DATA = &_S_bss = 0x20000004
- _stack_top is at 0x20001004 = _E_bss + 0x1000