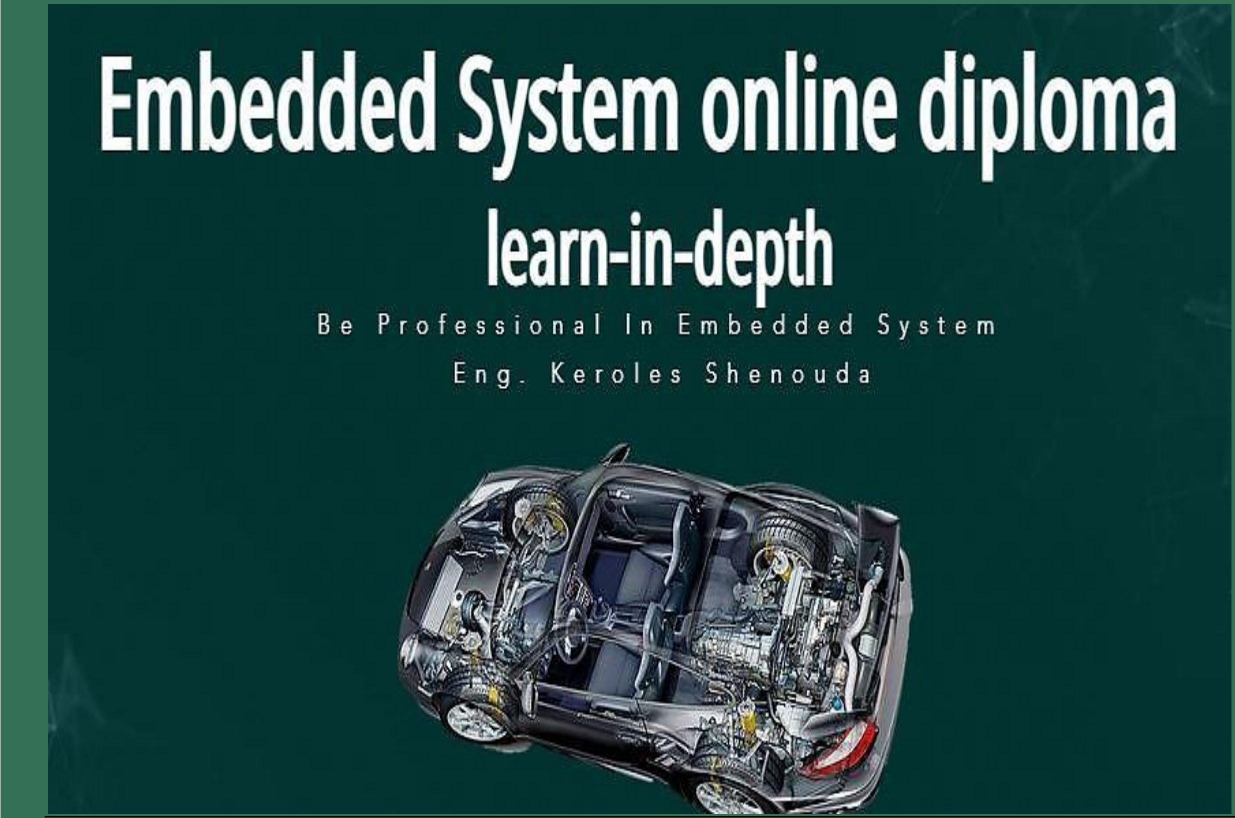

Embedded System online diploma
learn-in-depth
Be Professional In Embedded System
Eng. Keroles Shenouda

MOHAMED MOSTAFA SHABAN MOHAMED

Creating BareMetal SW from scratch

REPORT

Learn in Depth Diploma - Embedded C Unit3 – Lesson 4 Assignment - Lab_4

Eng. Keroles Shenoda

https://github.com/mohamedmostaf21/online_embedded_systems_diploma/tree/main/Unit3_Embedded_C/Embedded_C_Lesson4

Lab4 Report

Contents

Objective:

In this lab we will simulate and debug application code on TivaC kit with

tm4c123gh6pz and arm-cortexM4 processor family.

The application is toggling green Led (pin_3 in PortF).

We will write from scratch: main.c, startup.c, makefile.

Information about TM4C123GH6PZ:

Flash memory occupies addresses from 0x00000000 to 0x20000000.

SRAM memory occupies addresses from 0x20000000 to 0x40000000.

SYSCTL register is register control enabling and disabling clock on each register in system, it has address 0x400FE000

To enable portf we need to assign 0x20 to address away from 0x400FE000 by offset 0x108.

Then we need to define the direction of pin3 as output, we define direction by putting 1 on bit3 on register GPIO_PORTF_DIR_R, which has address 0x40025000 and offset 0x400.

Then enable the pin3 by putting 1 on bit3 of register GPIO_PORTF_DEN_R which has address 0x40025000 and offset 0x51C.

Finally, to turn on and off of led we put 1 and 0 respectively on register GPIO_PORTF_DR_R that has address 0x40025000 and offset 0x3FC.

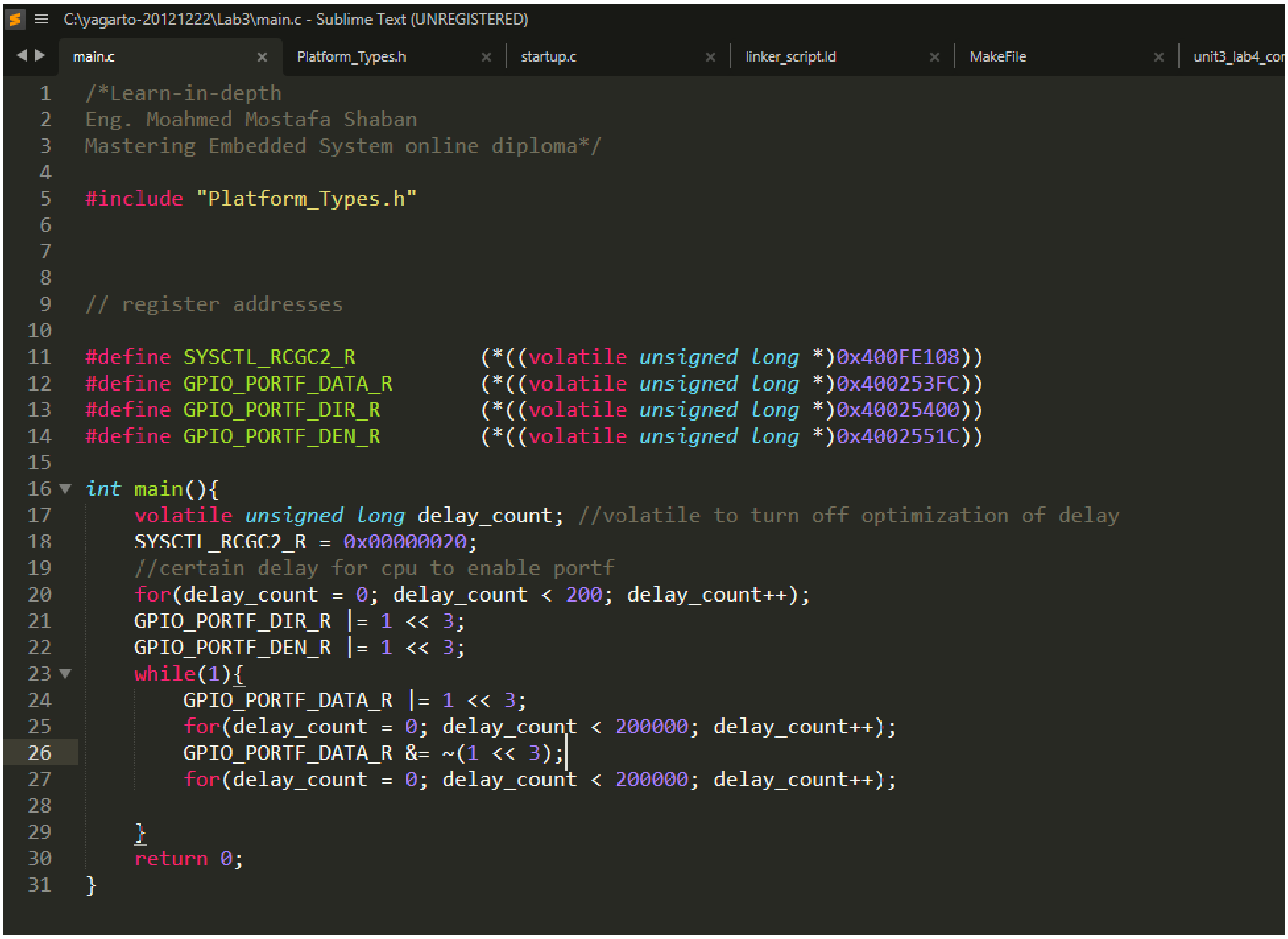
Architecture and Design:

The architecture of the BareMetal software consists of three components: main.c, startup.c or stratup.s, linker_script.ld , makefile, and Platform_Types.h. The Platform_Types.h file contains all datatypes will be used, while main.c implements code of toggle led.

Lab4 Report

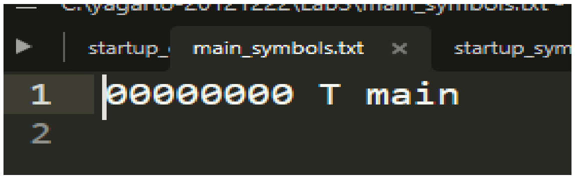
Main file:

Main.c:



```c
/*Learn-in-depth
Eng. Moahmed Mostafa Shaban
Mastering Embedded System online diploma*/

#include "Platform_Types.h"



// register addresses

#define SYSCTL_RCGC2_R          (*((volatile unsigned long *)0x400FE108))
#define GPIO_PORTF_DATA_R       (*((volatile unsigned long *)0x400253FC))
#define GPIO_PORTF_DIR_R        (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_DEN_R        (*((volatile unsigned long *)0x4002551C))

int main(){
    volatile unsigned long delay_count; //volatile to turn off optimization of delay
    SYSCTL_RCGC2_R = 0x00000020;
    //certain delay for cpu to enable portf
    for(delay_count = 0; delay_count < 200; delay_count++);
    GPIO_PORTF_DIR_R |= 1 << 3;
    GPIO_PORTF_DEN_R |= 1 << 3;
    while(1){
        GPIO_PORTF_DATA_R |= 1 << 3;
        for(delay_count = 0; delay_count < 200000; delay_count++);
        GPIO_PORTF_DATA_R &= ~(1 << 3);
        for(delay_count = 0; delay_count < 200000; delay_count++);

    }
    return 0;
}
```

Symbols:

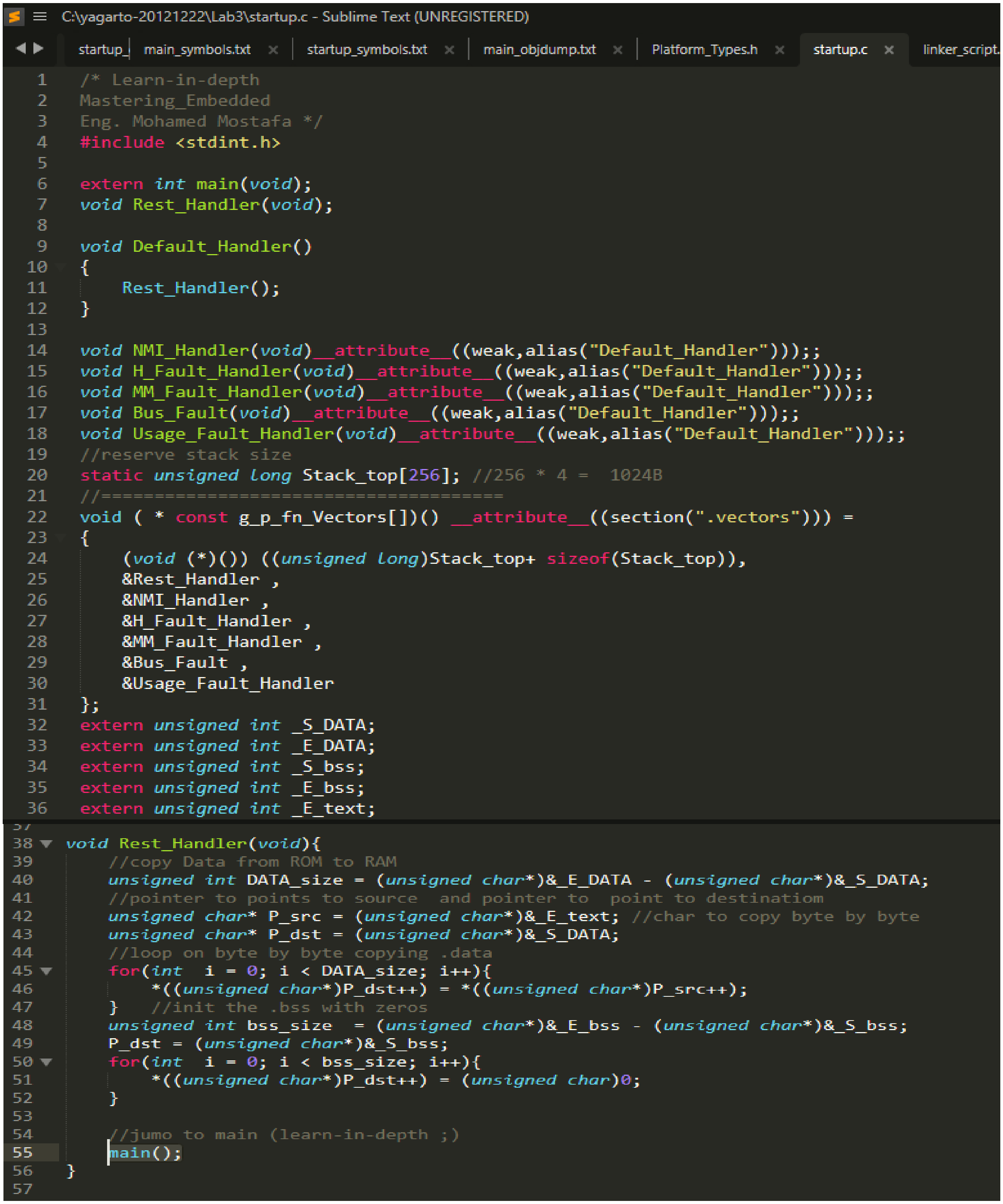


```
00000000 T main
```

Lab4 Report

Startup file:

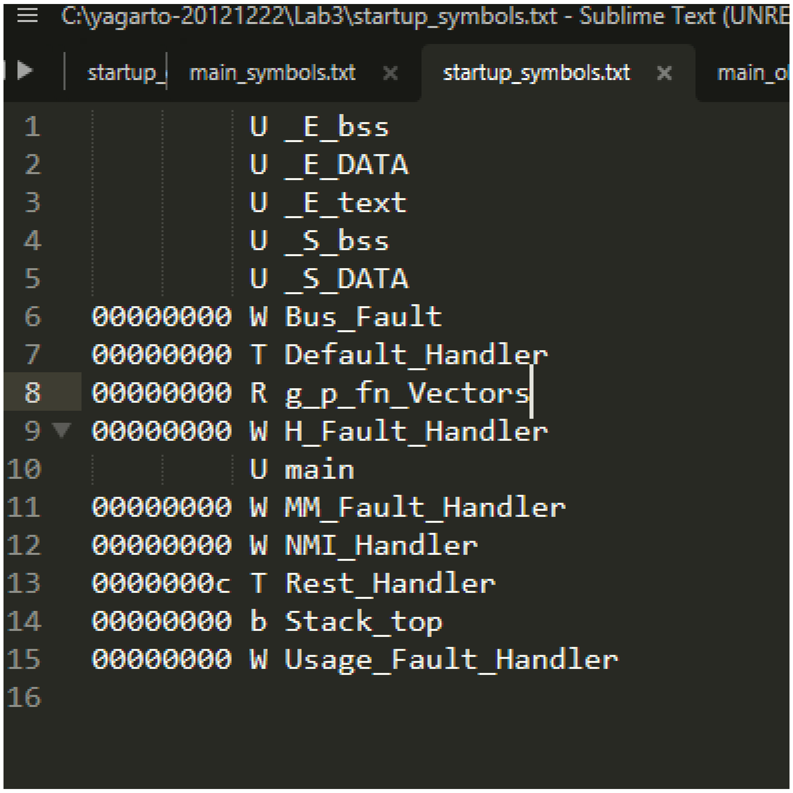
Startup.c:



```c
/* Learn-in-depth
Mastering_Embedded
Eng. Mohamed Mostafa */
#include <stdint.h>

extern int main(void);
void Rest_Handler(void);

void Default_Handler()
{
    Rest_Handler();
}

void NMI_Handler(void)__attribute__((weak,alias("Default_Handler")));;
void H_Fault_Handler(void)__attribute__((weak,alias("Default_Handler")));;
void MM_Fault_Handler(void)__attribute__((weak,alias("Default_Handler")));;
void Bus_Fault(void)__attribute__((weak,alias("Default_Handler")));;
void Usage_Fault_Handler(void)__attribute__((weak,alias("Default_Handler")));;
//reserve stack size
static unsigned long Stack_top[256]; //256 * 4 =  1024B
//======================================
void ( * const g_p_fn_Vectors[])() __attribute__((section(".vectors"))) =
{
    (void (*)()) ((unsigned long)Stack_top+ sizeof(Stack_top)),
    &Rest_Handler ,
    &NMI_Handler ,
    &H_Fault_Handler ,
    &MM_Fault_Handler ,
    &Bus_Fault ,
    &Usage_Fault_Handler
};
extern unsigned int _S_DATA;
extern unsigned int _E_DATA;
extern unsigned int _S_bss;
extern unsigned int _E_bss;
extern unsigned int _E_text;

void Rest_Handler(void){
    //copy Data from ROM to RAM
    unsigned int DATA_size = (unsigned char*)&_E_DATA - (unsigned char*)&_S_DATA;
    //pointer to points to source  and pointer to  point to destinatiom
    unsigned char* P_src = (unsigned char*)&_E_text; //char to copy byte by byte
    unsigned char* P_dst = (unsigned char*)&_S_DATA;
    //loop on byte by byte copying .data
    for(int  i = 0; i < DATA_size; i++){
        *((unsigned char*)P_dst++) = *((unsigned char*)P_src++);
    }   //init the .bss with zeros
    unsigned int bss_size  = (unsigned char*)&_E_bss - (unsigned char*)&_S_bss;
    P_dst = (unsigned char*)&_S_bss;
    for(int  i = 0; i < bss_size; i++){
        *((unsigned char*)P_dst++) = (unsigned char)0;
    }

    //jumo to main (learn-in-depth ;)
    main();
}
```

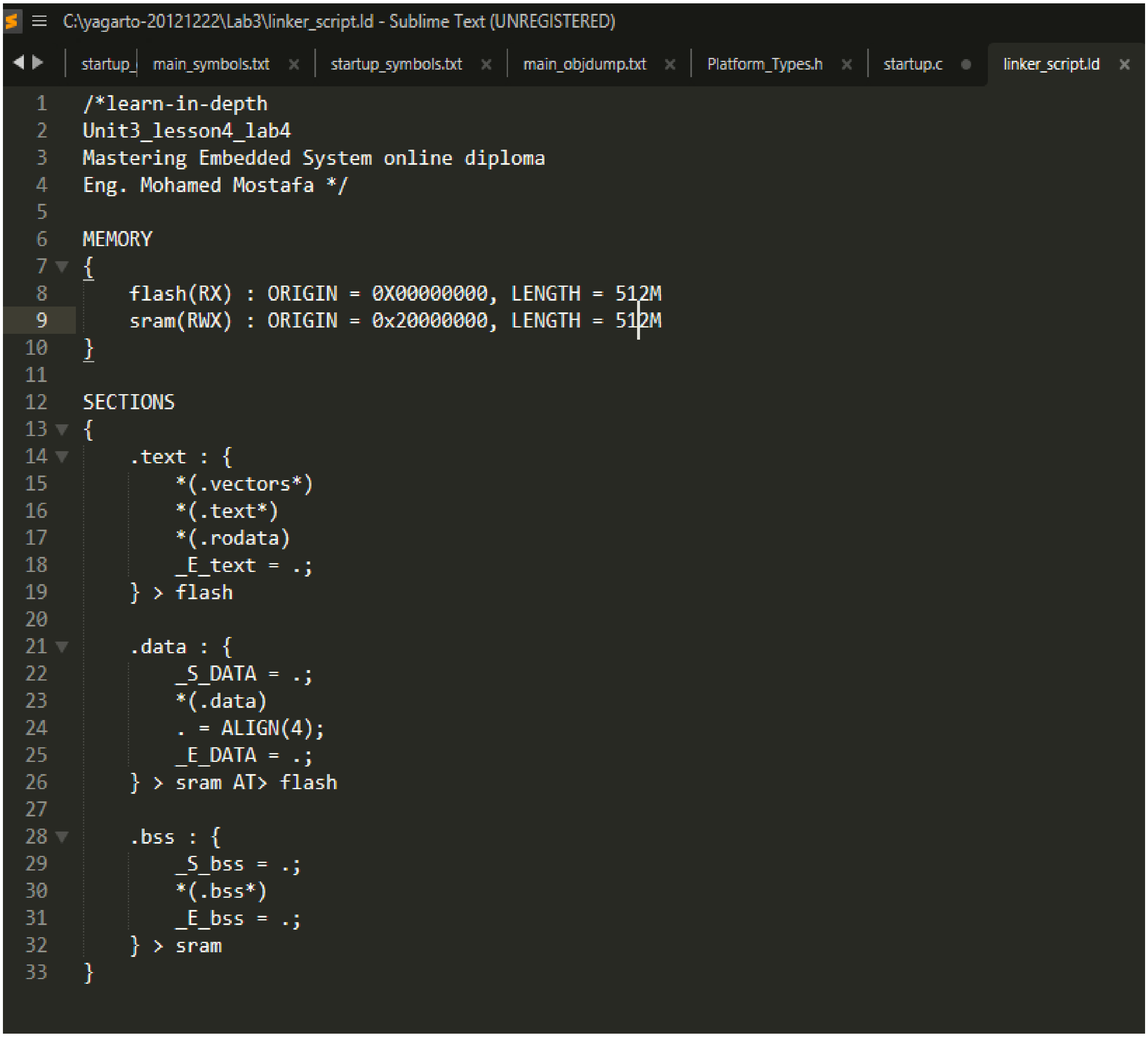
Lab4 Report

Symbols:


C:\yagarto-20121222\Lab3\startup_symbols.txt - Sublime Text (UNRE
startup_
main_symbols.txt
startup_symbols.txt
main_o

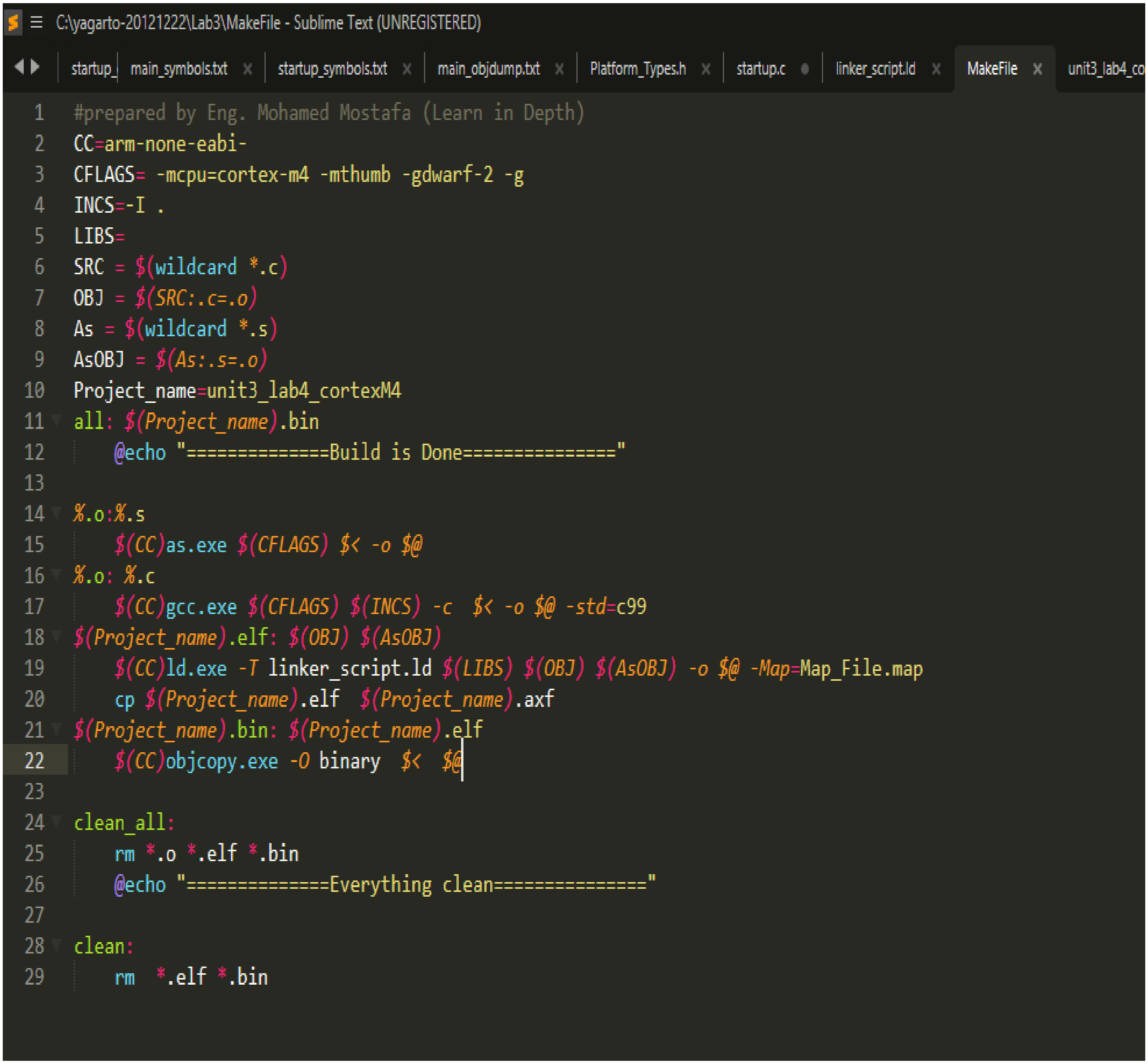
```
         U _E_bss
         U _E_DATA
         U _E_text
         U _S_bss
         U _S_DATA
00000000 W Bus_Fault
00000000 T Default_Handler
00000000 R g_p_fn_Vectors
00000000 W H_Fault_Handler
         U main
00000000 W MM_Fault_Handler
00000000 W NMI_Handler
0000000c T Rest_Handler
00000000 b Stack_top
00000000 W Usage_Fault_Handler
```

Linker_script.ld:



```
/*learn-in-depth
Unit3_lesson4_lab4
Mastering Embedded System online diploma
Eng. Mohamed Mostafa */

MEMORY
{
    flash(RX) : ORIGIN = 0X00000000, LENGTH = 512M
    sram(RWX) : ORIGIN = 0x20000000, LENGTH = 512M
}

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

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

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

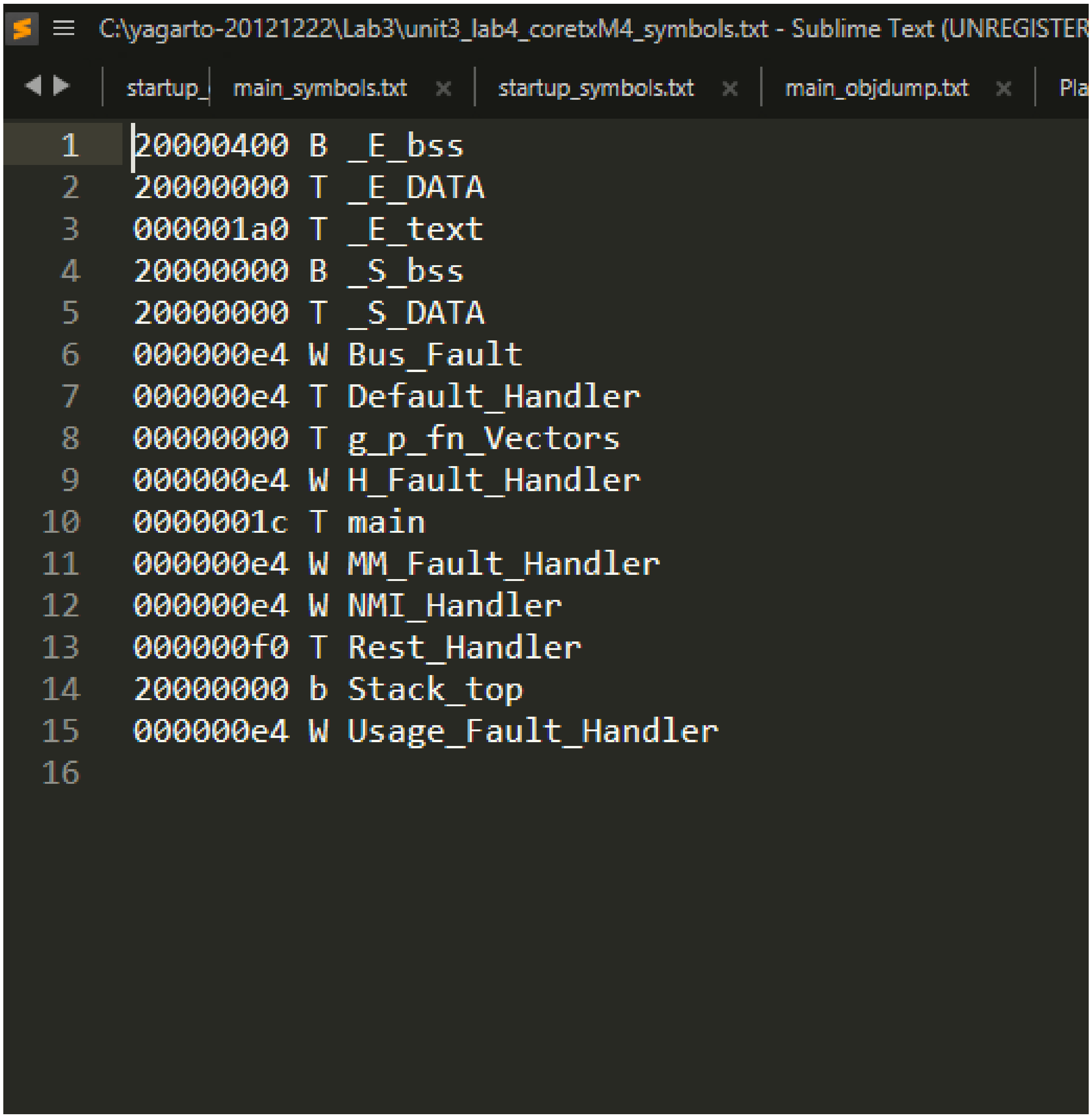
Makefile:



```
#prepared by Eng. Mohamed Mostafa (Learn in Depth)
CC=arm-none-eabi-
CFLAGS= -mcpu=cortex-m4 -mthumb -gdwarf-2 -g
INCS=-I .
LIBS=
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
As = $(wildcard *.s)
AsOBJ = $(As:.s=.o)
Project_name=unit3_lab4_cortexM4
all: $(Project_name).bin
    @echo "=============Build is Done=============="

%.o:%.s
    $(CC)as.exe $(CFLAGS) $< -o $@
%.o: %.c
    $(CC)gcc.exe $(CFLAGS) $(INCS) -c  $< -o $@ -std=c99
$(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
    @echo "=============Everything clean=============="

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

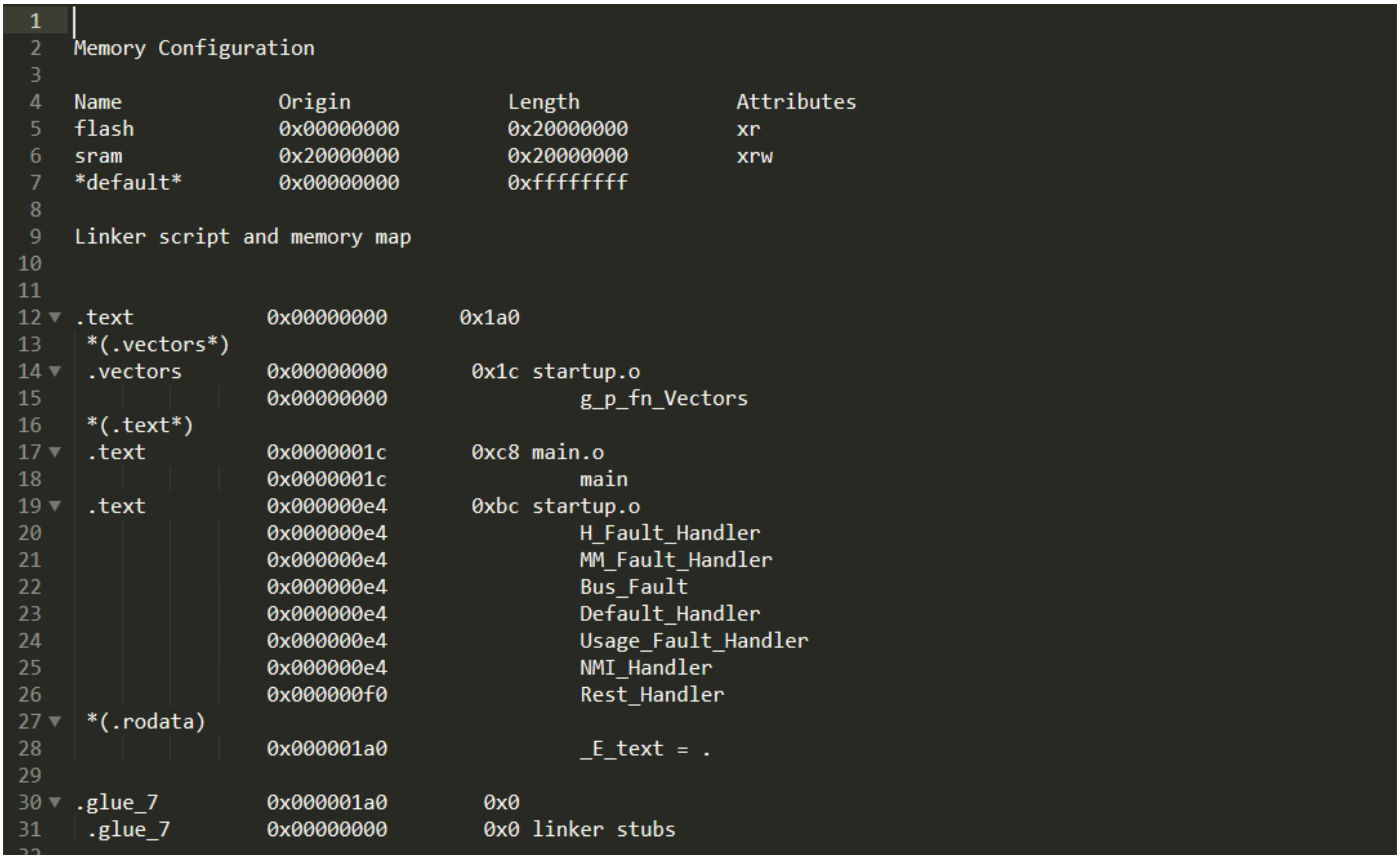
unit3_lab4_cortexM4.elf:

symbols:

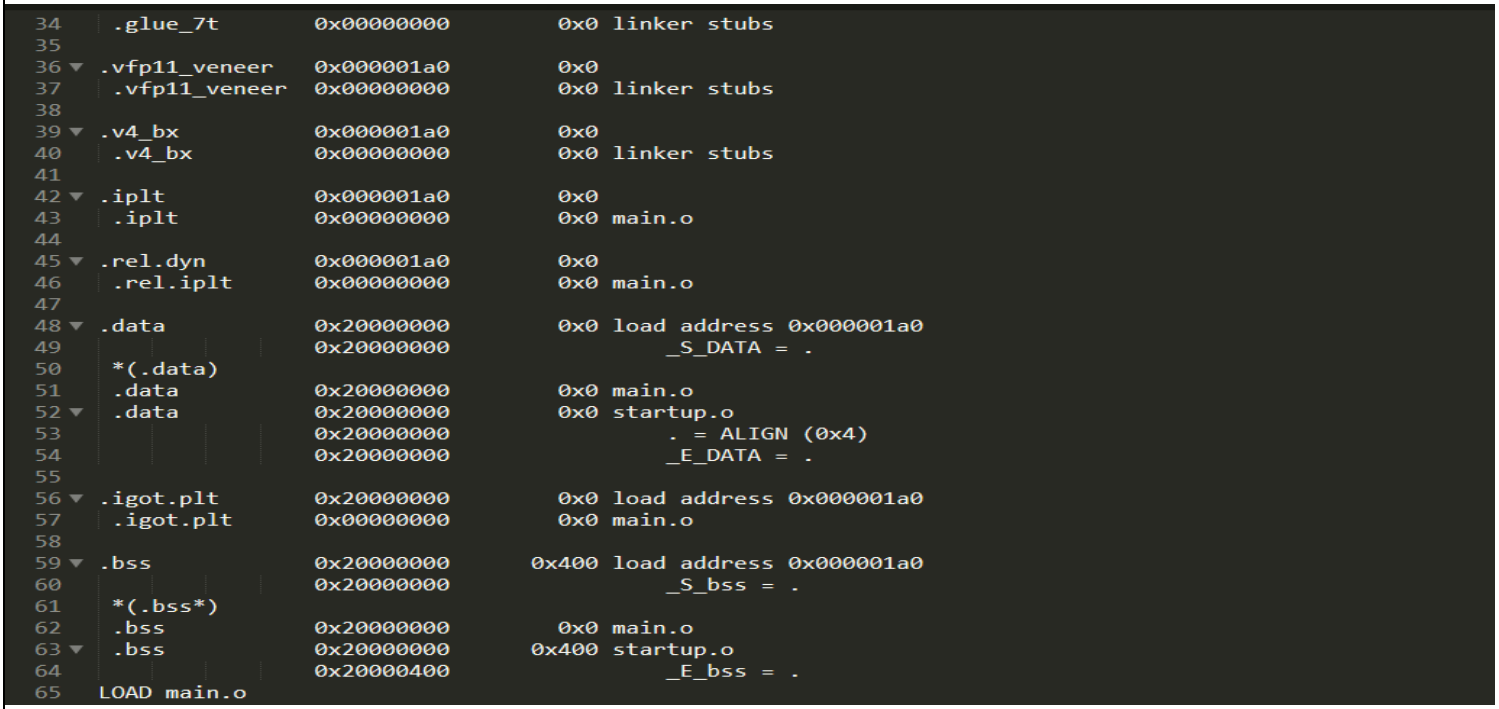

C:\yagarto-20121222\Lab3\unit3_lab4_coretxM4_symbols.txt - Sublime Text (UNREGISTER
startup_
main_symbols.txt
startup_symbols.txt
main_objdump.txt
Pla

```
20000400 B _E_bss
20000000 T _E_DATA
000001a0 T _E_text
20000000 B _S_bss
20000000 T _S_DATA
000000e4 W Bus_Fault
000000e4 T Default_Handler
00000000 T g_p_fn_Vectors
000000e4 W H_Fault_Handler
0000001c T main
000000e4 W MM_Fault_Handler
000000e4 W NMI_Handler
000000f0 T Rest_Handler
20000000 b Stack_top
000000e4 W Usage_Fault_Handler
```

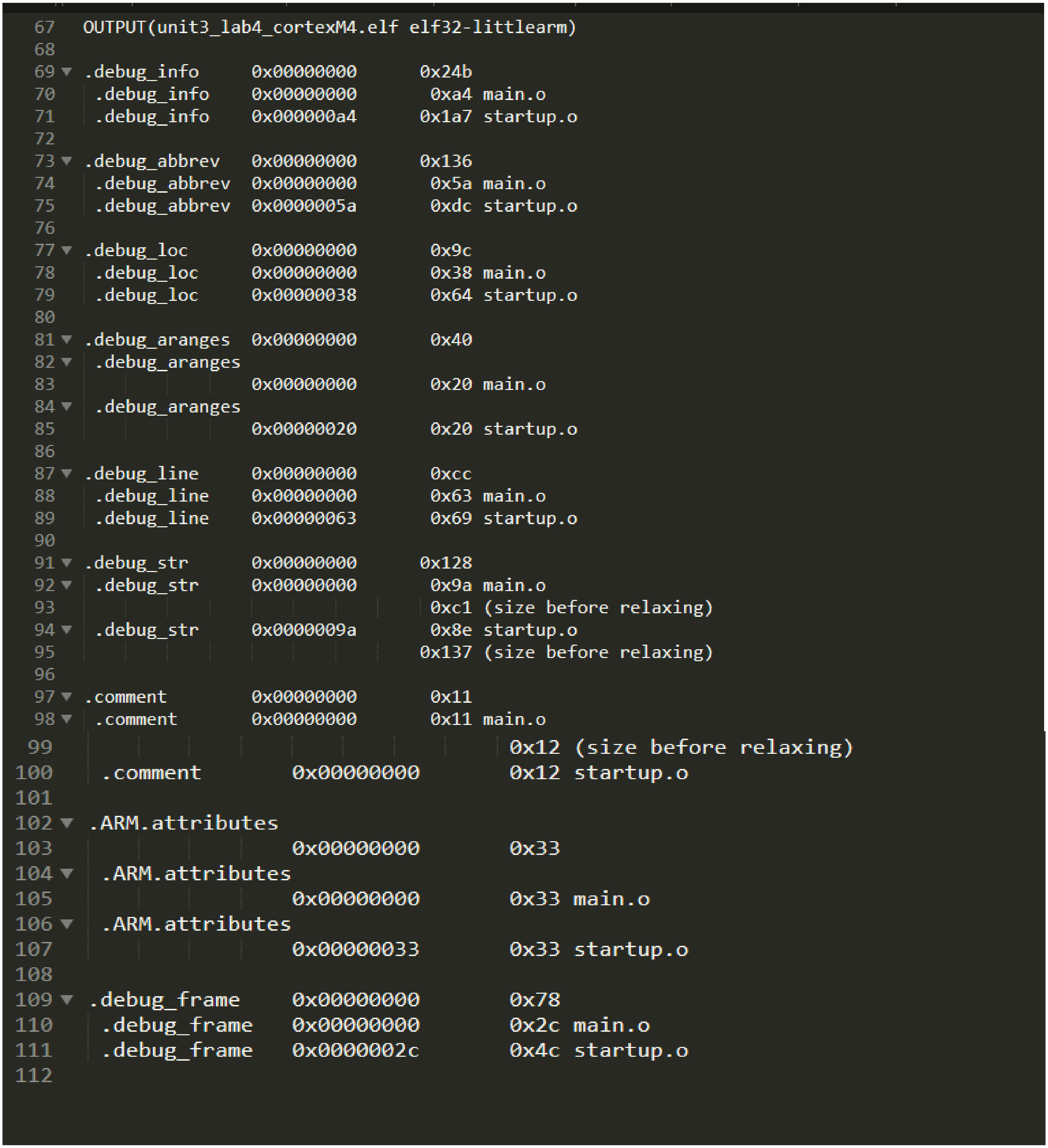
Map_file.map:



```

Memory Configuration

Name             Origin             Length             Attributes
flash            0x00000000         0x20000000         xr
sram             0x20000000         0x20000000         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map


.text           0x00000000      0x1a0
 *(.vectors*)
 .vectors       0x00000000       0x1c startup.o
                0x00000000                g_p_fn_Vectors
 *(.text*)
 .text          0x0000001c       0xc8 main.o
                0x0000001c                main
 .text          0x000000e4       0xbc startup.o
                0x000000e4                H_Fault_Handler
                0x000000e4                MM_Fault_Handler
                0x000000e4                Bus_Fault
                0x000000e4                Default_Handler
                0x000000e4                Usage_Fault_Handler
                0x000000e4                NMI_Handler
                0x000000f0                Rest_Handler
 *(.rodata)
                0x000001a0                _E_text = .

.glue_7         0x000001a0        0x0
 .glue_7        0x00000000        0x0 linker stubs
```



```
 .glue_7t       0x00000000        0x0 linker stubs

.vfp11_veneer   0x000001a0        0x0
 .vfp11_veneer  0x00000000        0x0 linker stubs

.v4_bx          0x000001a0        0x0
 .v4_bx         0x00000000        0x0 linker stubs

.iplt           0x000001a0        0x0
 .iplt          0x00000000        0x0 main.o

.rel.dyn        0x000001a0        0x0
 .rel.iplt      0x00000000        0x0 main.o

.data           0x20000000        0x0 load address 0x000001a0
                0x20000000                _S_DATA = .
 *(.data)
 .data          0x20000000        0x0 main.o
 .data          0x20000000        0x0 startup.o
                0x20000000                . = ALIGN (0x4)
                0x20000000                _E_DATA = .

.igot.plt       0x20000000        0x0 load address 0x000001a0
 .igot.plt      0x00000000        0x0 main.o

.bss            0x20000000      0x400 load address 0x000001a0
                0x20000000                _S_bss = .
 *(.bss*)
 .bss           0x20000000        0x0 main.o
 .bss           0x20000000      0x400 startup.o
                0x20000400                _E_bss = .
LOAD main.o
```



```
OUTPUT(unit3_lab4_cortexM4.elf elf32-littlearm)

.debug_info     0x00000000      0x24b
 .debug_info    0x00000000       0xa4 main.o
 .debug_info    0x000000a4      0x1a7 startup.o

.debug_abbrev   0x00000000      0x136
 .debug_abbrev  0x00000000       0x5a main.o
 .debug_abbrev  0x0000005a       0xdc startup.o

.debug_loc      0x00000000       0x9c
 .debug_loc     0x00000000       0x38 main.o
 .debug_loc     0x00000038       0x64 startup.o

.debug_aranges  0x00000000       0x40
 .debug_aranges
                0x00000000       0x20 main.o
 .debug_aranges
                0x00000020       0x20 startup.o

.debug_line     0x00000000       0xcc
 .debug_line    0x00000000       0x63 main.o
 .debug_line    0x00000063       0x69 startup.o

.debug_str      0x00000000      0x128
 .debug_str     0x00000000       0x9a main.o
                                 0xc1 (size before relaxing)
 .debug_str     0x0000009a       0x8e startup.o
                                0x137 (size before relaxing)

.comment        0x00000000       0x11
 .comment       0x00000000       0x11 main.o
                                 0x12 (size before relaxing)
 .comment       0x00000000       0x12 startup.o

.ARM.attributes
                0x00000000       0x33
 .ARM.attributes
                0x00000000       0x33 main.o
 .ARM.attributes
                0x00000033       0x33 startup.o

.debug_frame    0x00000000       0x78
 .debug_frame   0x00000000       0x2c main.o
 .debug_frame   0x0000002c       0x4c startup.o
```