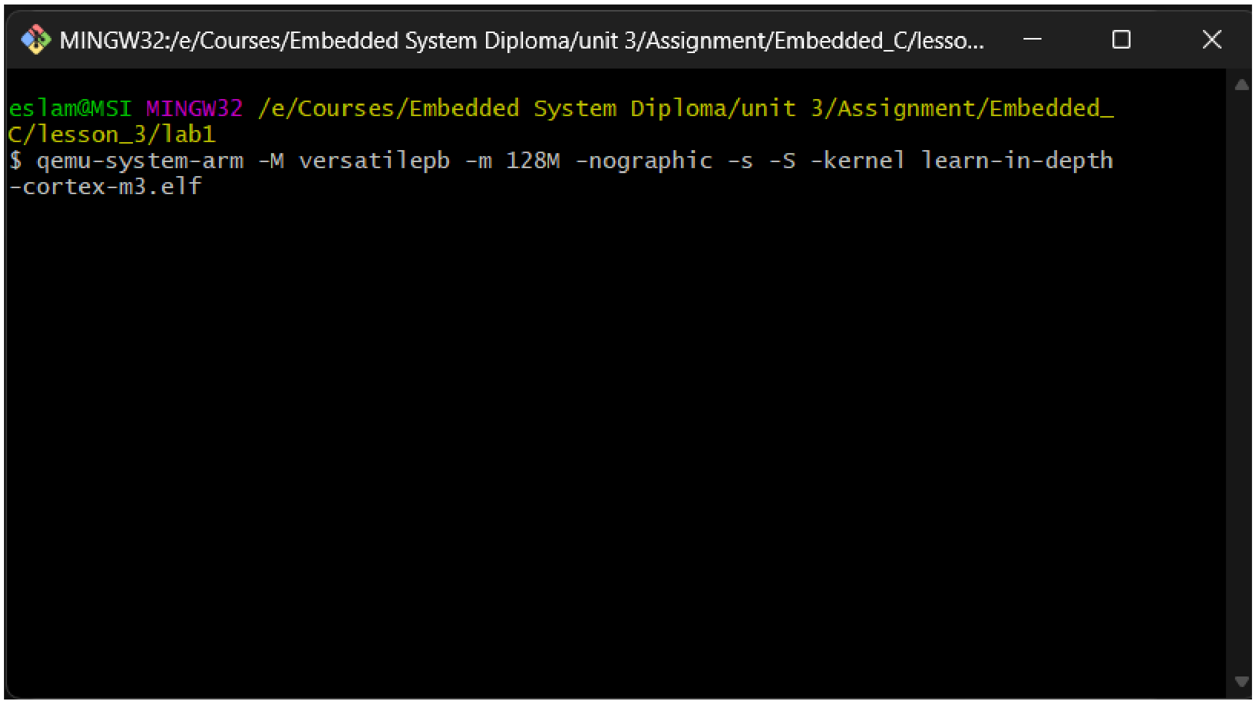
LESSON 3 - LABS

Name : Eslam Mostafa Mohamed

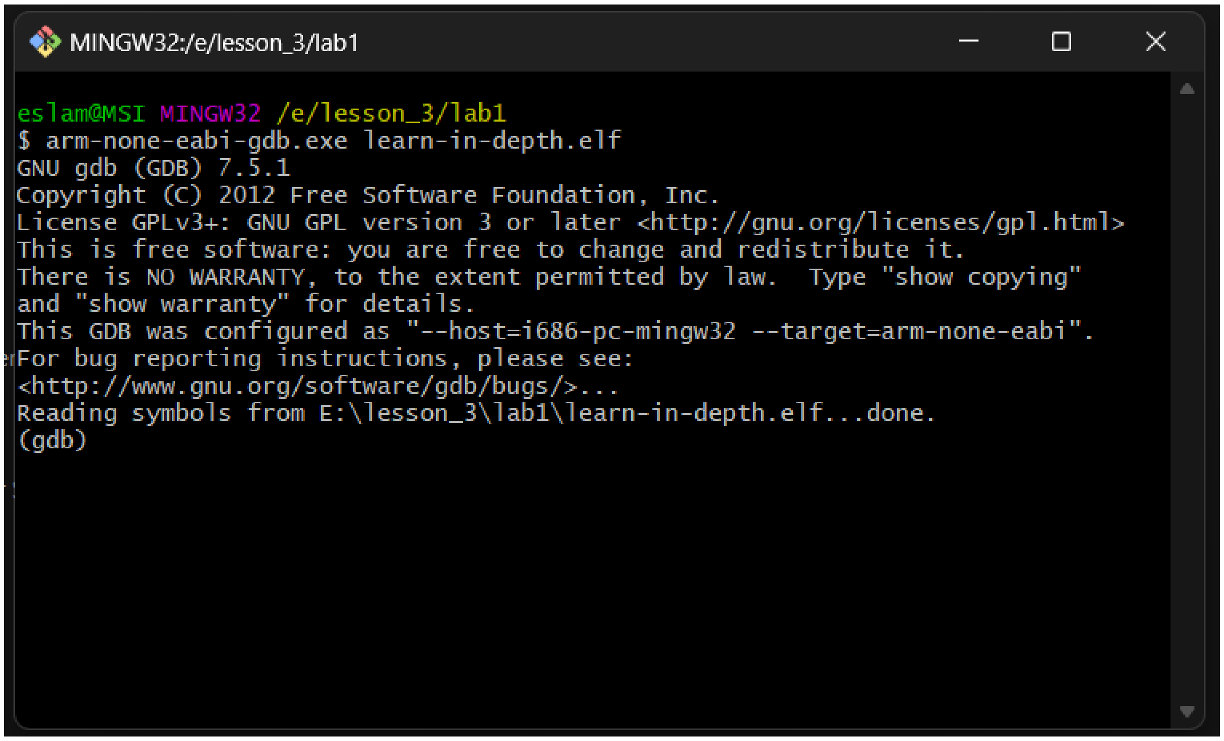
Debugging Steps with Terminal :

• Start Debugging



MINGW32:/e/Courses/Embedded System Diploma/unit 3/Assignment/Embedded_C/lesso...

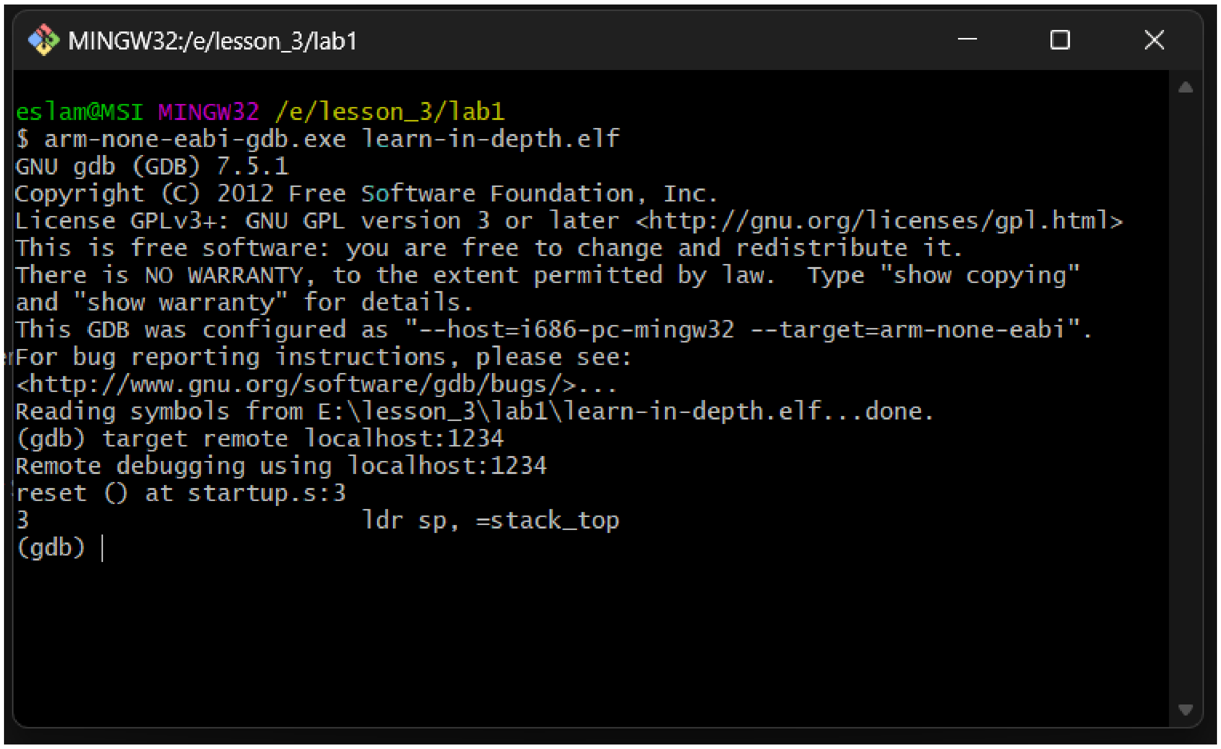
```
eslam@MSI MINGW32 /e/Courses/Embedded System Diploma/unit 3/Assignment/Embedded_
C/lesson_3/lab1
$ qemu-system-arm -M versatilepb -m 128M -nographic -s -S -kernel learn-in-depth
-cortex-m3.elf
```



MINGW32:/e/lesson_3/lab1

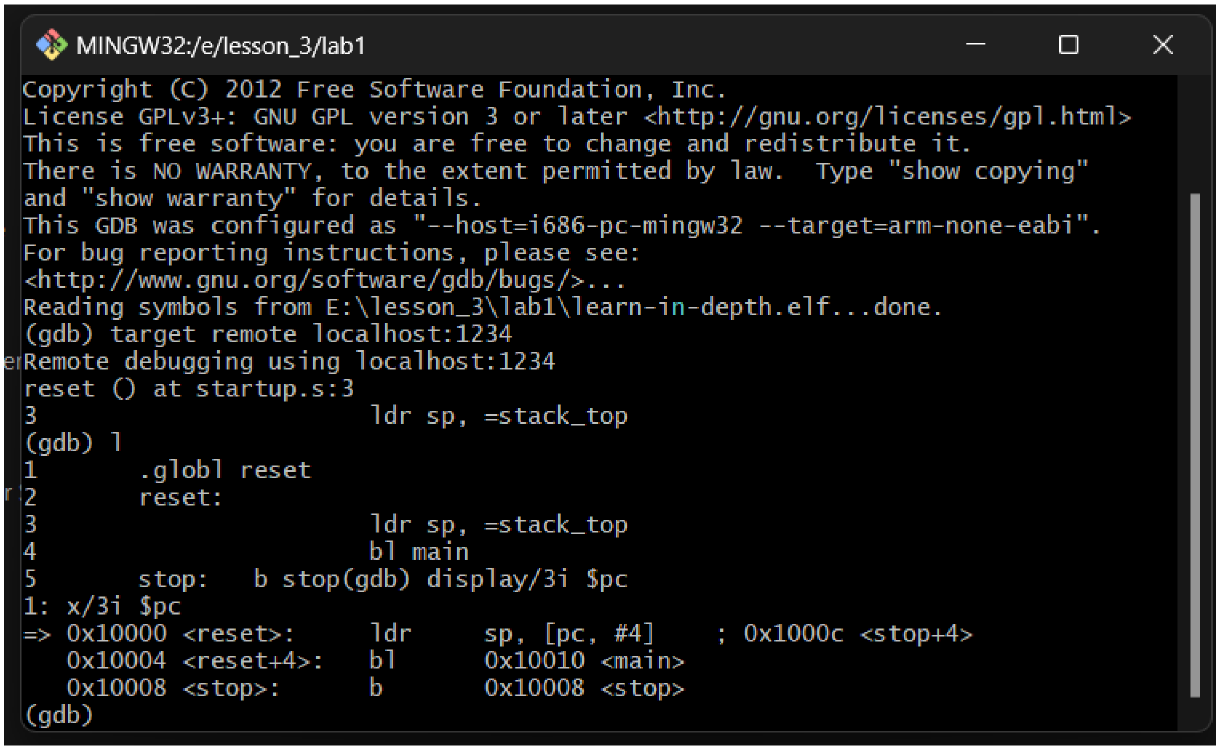
```
eslam@MSI MINGW32 /e/lesson_3/lab1
$ arm-none-eabi-gdb.exe learn-in-depth.elf
GNU gdb (GDB) 7.5.1
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from E:\lesson_3\lab1\learn-in-depth.elf...done.
(gdb)
```

• Commands Used to Debug



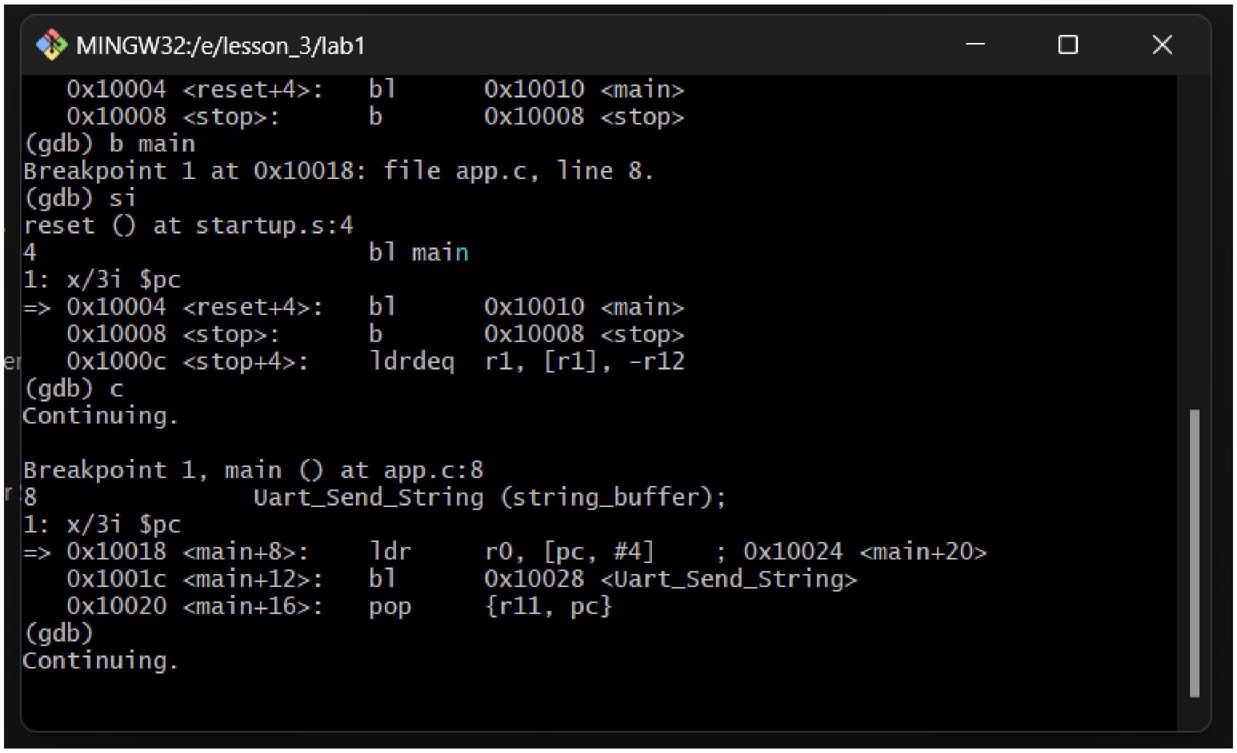
MINGW32:/e/lesson_3/lab1

```
eslam@MSI MINGW32 /e/lesson_3/lab1
$ arm-none-eabi-gdb.exe learn-in-depth.elf
GNU gdb (GDB) 7.5.1
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from E:\lesson_3\lab1\learn-in-depth.elf...done.
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
reset () at startup.s:3
3                       ldr sp, =stack_top
(gdb) |
```

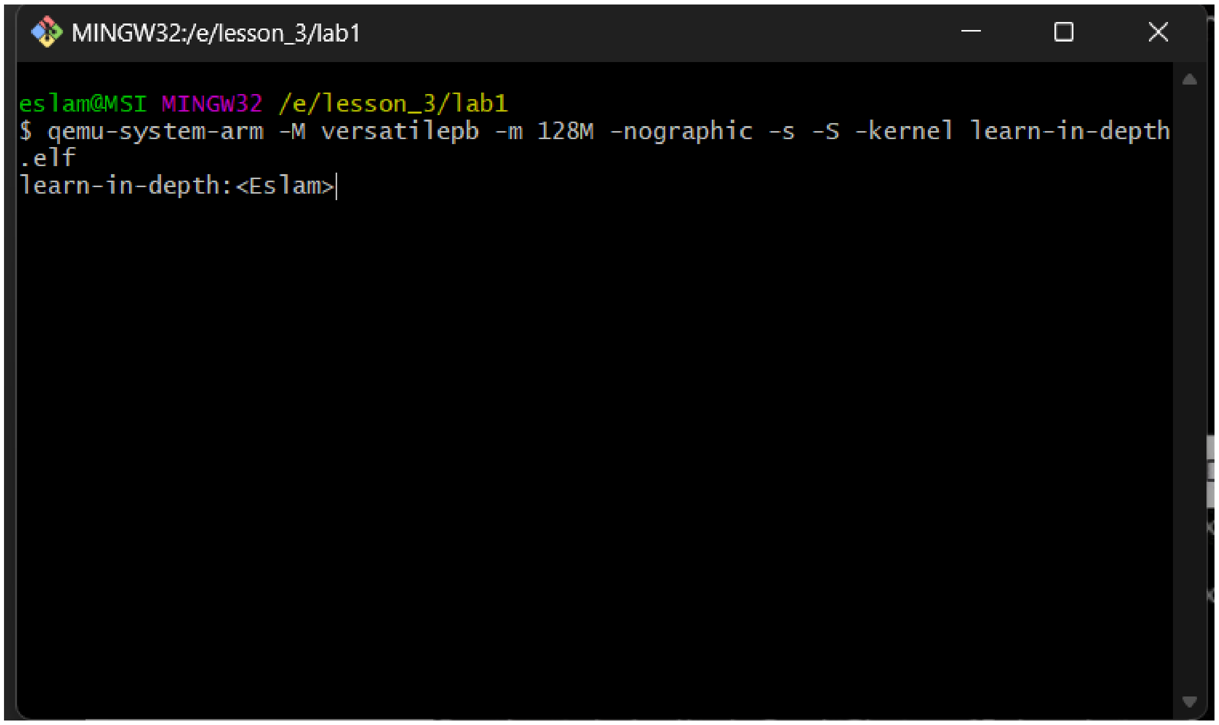


MINGW32:/e/lesson_3/lab1

```
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from E:\lesson_3\lab1\learn-in-depth.elf...done.
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
reset () at startup.s:3
3                       ldr sp, =stack_top
(gdb) l
1       .globl reset
2       reset:
3                       ldr sp, =stack_top
4                       bl main
5       stop:   b stop(gdb) display/3i $pc
1: x/3i $pc
=> 0x10000 <reset>:     ldr     sp, [pc, #4]    ; 0x1000c <stop+4>
   0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      b       0x10008 <stop>
(gdb)
```

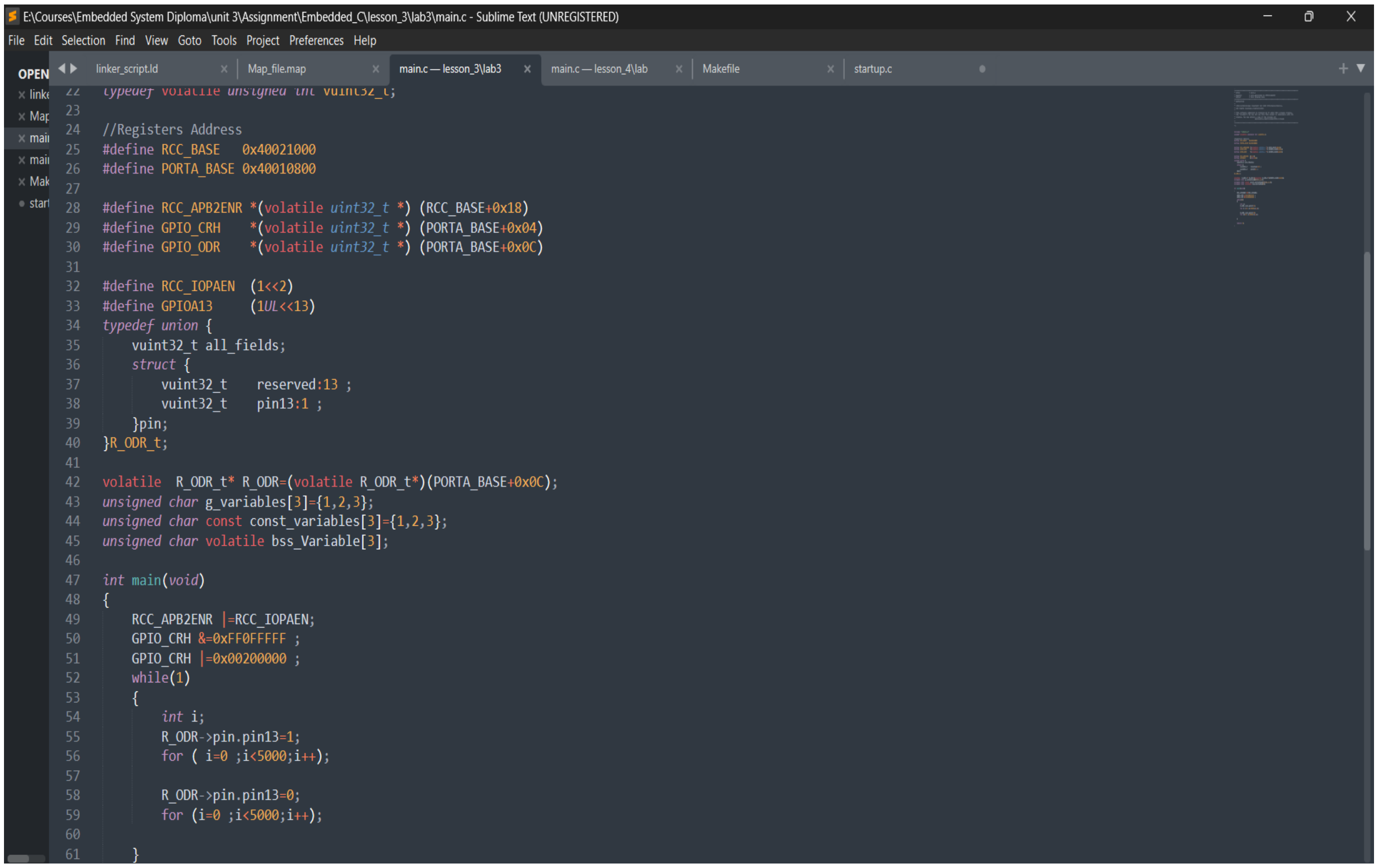

```
MINGW32:/e/lesson_3/lab1
   0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      b       0x10008 <stop>
(gdb) b main
Breakpoint 1 at 0x10018: file app.c, line 8.
(gdb) si
reset () at startup.s:4
4                       bl main
1: x/3i $pc
=> 0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      b       0x10008 <stop>
   0x1000c <stop+4>:    ldrdeq  r1, [r1], -r12
(gdb) c
Continuing.

Breakpoint 1, main () at app.c:8
8               Uart_Send_String (string_buffer);
1: x/3i $pc
=> 0x10018 <main+8>:    ldr     r0, [pc, #4]    ; 0x10024 <main+20>
   0x1001c <main+12>:   bl      0x10028 <Uart_Send_String>
   0x10020 <main+16>:   pop     {r11, pc}
(gdb)
Continuing.
```

• The Output During Debug


```
MINGW32:/e/lesson_3/lab1

eslam@MSI MINGW32 /e/lesson_3/lab1
$ qemu-system-arm -M versatilepb -m 128M -nographic -s -S -kernel learn-in-depth
.elf
learn-in-depth:<Eslam>
```

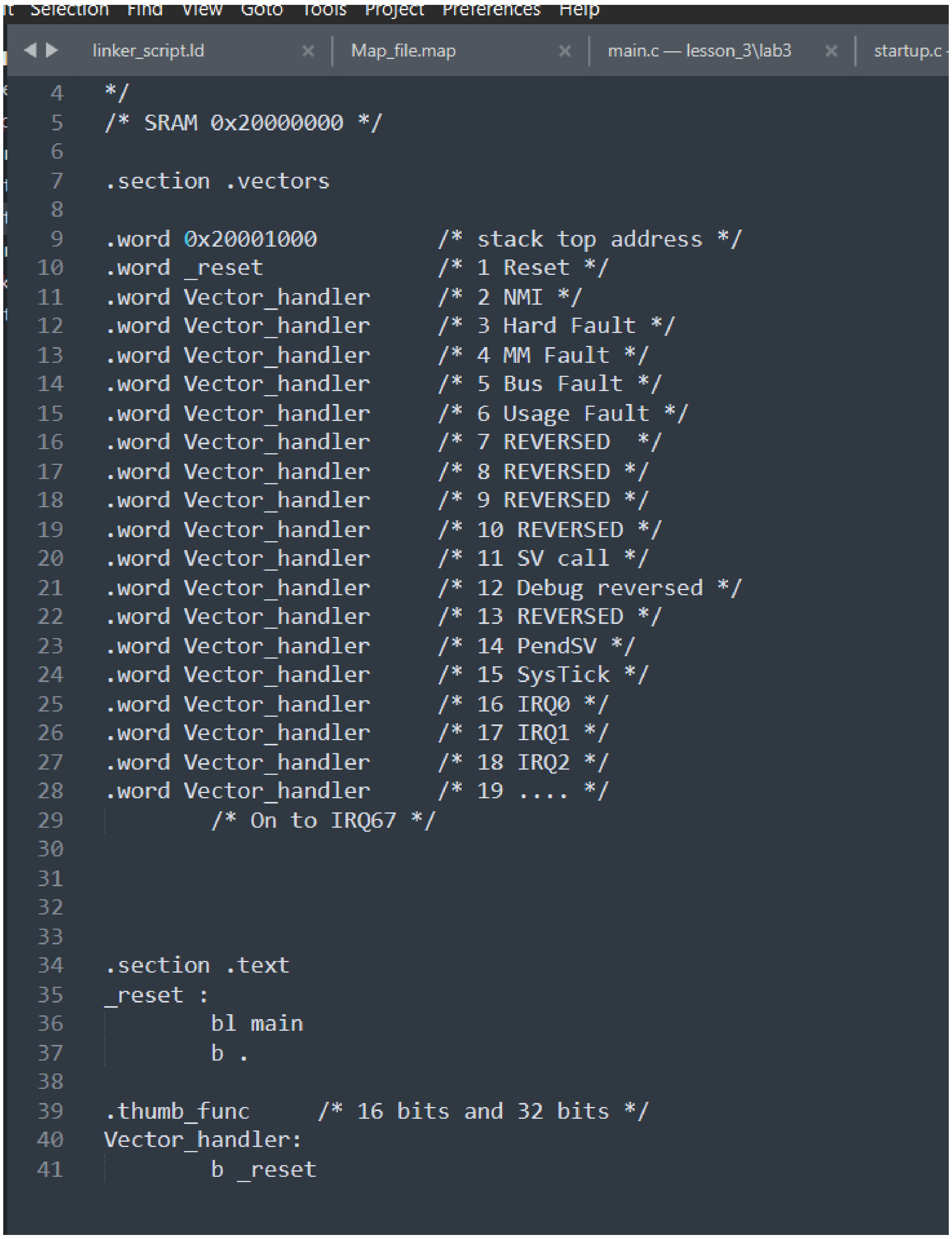
Lesson 3 Lab 2 with Startup.s file :

Main.c file :


E:\Courses\Embedded System Diploma\unit 3\Assignment\Embedded_C\lesson_3\lab3\main.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN
linker_script.ld
Map_file.map
main.c — lesson_3\lab3
main.c — lesson_4\lab
Makefile
startup.c

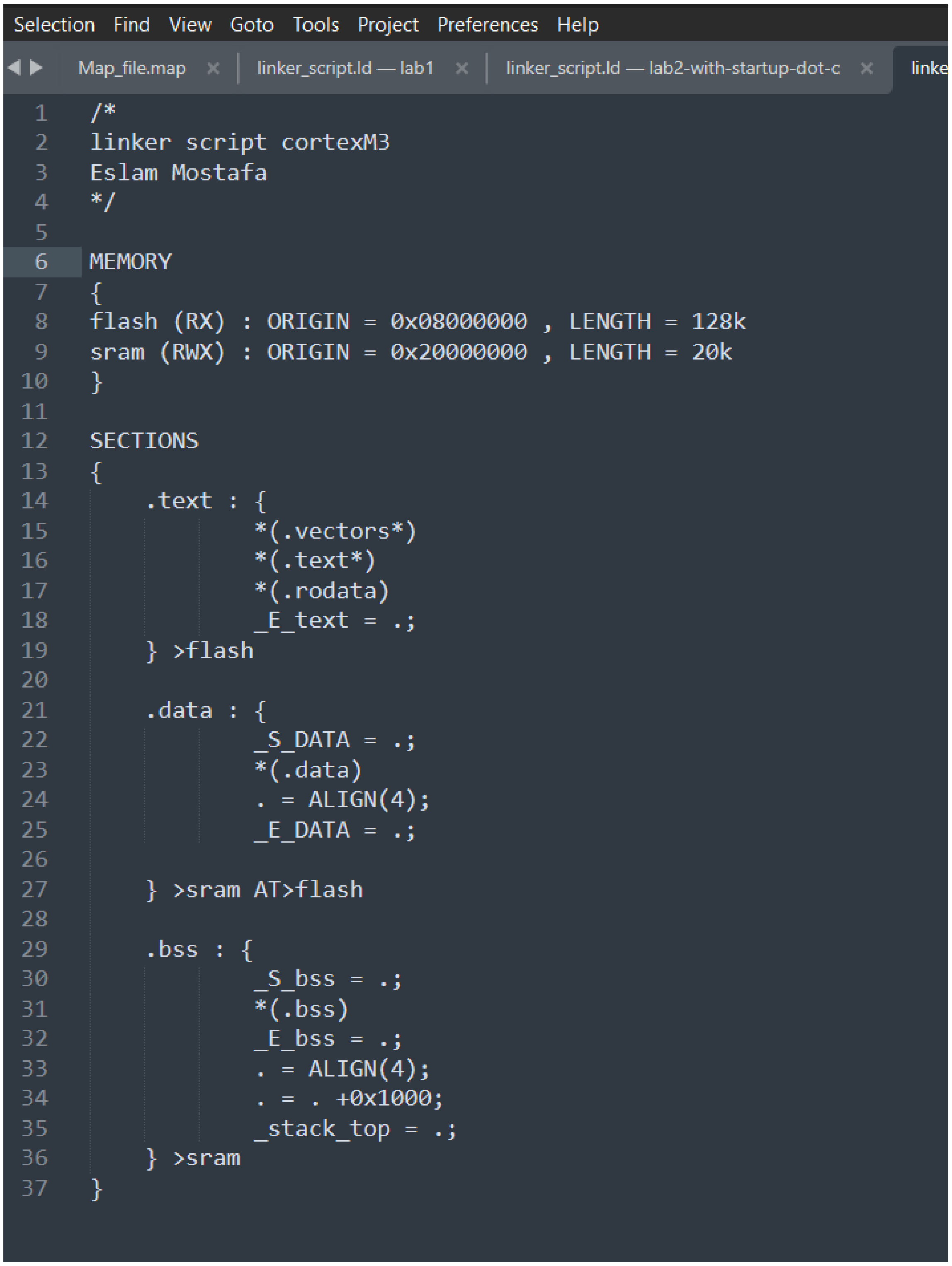
```c
typedef volatile unsigned int vuint32_t;

//Registers Address
#define RCC_BASE   0x40021000
#define PORTA_BASE 0x40010800

#define RCC_APB2ENR *(volatile uint32_t *) (RCC_BASE+0x18)
#define GPIO_CRH    *(volatile uint32_t *) (PORTA_BASE+0x04)
#define GPIO_ODR    *(volatile uint32_t *) (PORTA_BASE+0x0C)

#define RCC_IOPAEN  (1<<2)
#define GPIOA13     (1UL<<13)
typedef union {
    vuint32_t all_fields;
    struct {
        vuint32_t    reserved:13 ;
        vuint32_t    pin13:1 ;
    }pin;
}R_ODR_t;

volatile  R_ODR_t* R_ODR=(volatile R_ODR_t*)(PORTA_BASE+0x0C);
unsigned char g_variables[3]={1,2,3};
unsigned char const const_variables[3]={1,2,3};
unsigned char volatile bss_Variable[3];

int main(void)
{
    RCC_APB2ENR |=RCC_IOPAEN;
    GPIO_CRH &=0xFF0FFFFF ;
    GPIO_CRH |=0x00200000 ;
    while(1)
    {
        int i;
        R_ODR->pin.pin13=1;
        for ( i=0 ;i<5000;i++);

        R_ODR->pin.pin13=0;
        for (i=0 ;i<5000;i++);

    }
```

• Startup.s file



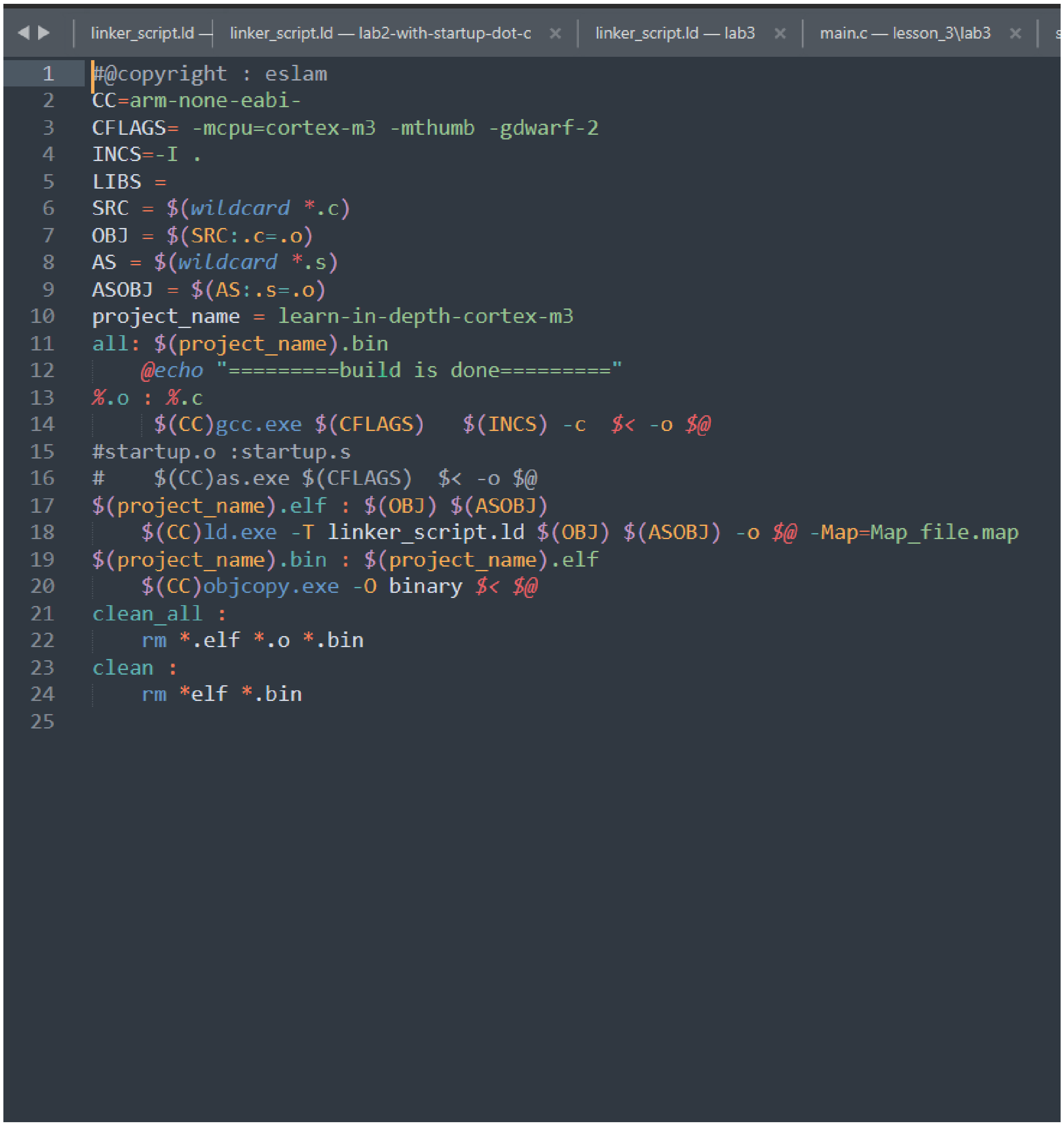
```
*/
/* SRAM 0x20000000 */

.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 REVERSED  */
.word Vector_handler      /* 8 REVERSED */
.word Vector_handler      /* 9 REVERSED */
.word Vector_handler      /* 10 REVERSED */
.word Vector_handler      /* 11 SV call */
.word Vector_handler      /* 12 Debug reversed */
.word Vector_handler      /* 13 REVERSED */
.word Vector_handler      /* 14 PendSV */
.word Vector_handler      /* 15 SysTick */
.word Vector_handler      /* 16 IRQ0 */
.word Vector_handler      /* 17 IRQ1 */
.word Vector_handler      /* 18 IRQ2 */
.word Vector_handler      /* 19 .... */
        /* On to IRQ67 */




.section .text
_reset :
        bl main
        b .

.thumb_func     /* 16 bits and 32 bits */
Vector_handler:
        b _reset
```

• Linker_Script file


Selection Find View Goto Tools Project Preferences Help
Map_file.map
linker_script.ld — lab1
linker_script.ld — lab2-with-startup-dot-c
linke

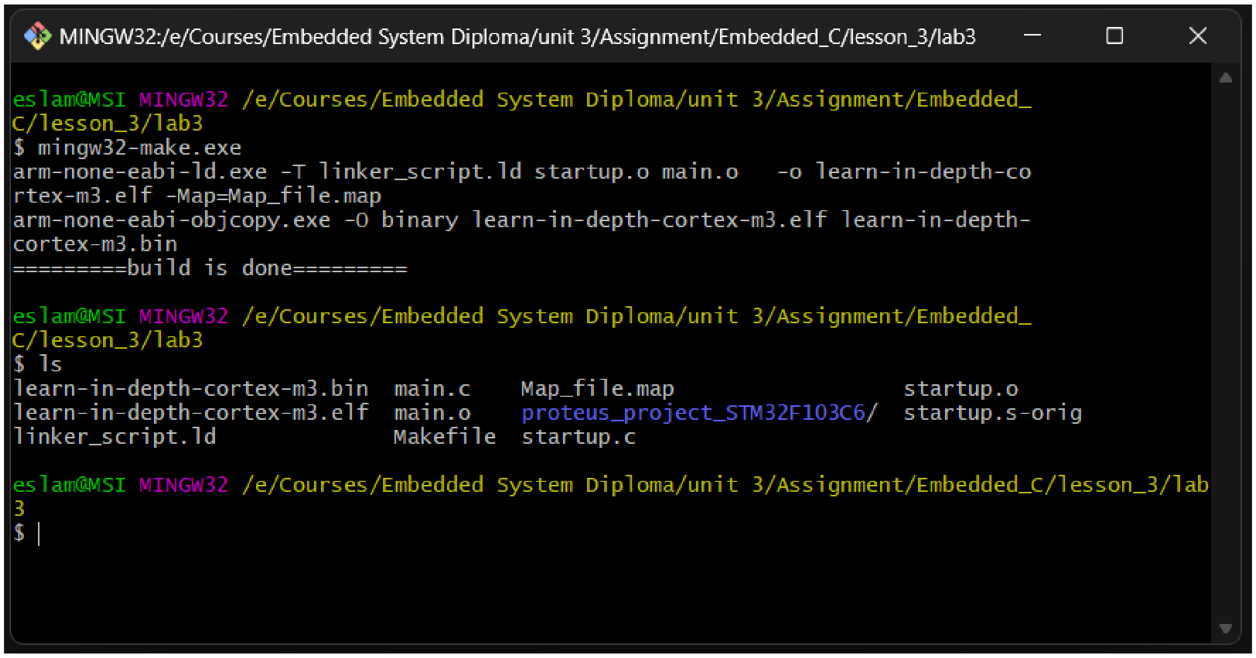
```
/*
linker script cortexM3
Eslam Mostafa
*/

MEMORY
{
flash (RX) : ORIGIN = 0x08000000 , LENGTH = 128k
sram (RWX) : ORIGIN = 0x20000000 , LENGTH = 20k
}

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 = .;
            . = ALIGN(4);
            . = . +0x1000;
            _stack_top = .;
    } >sram
}
```

• Make file



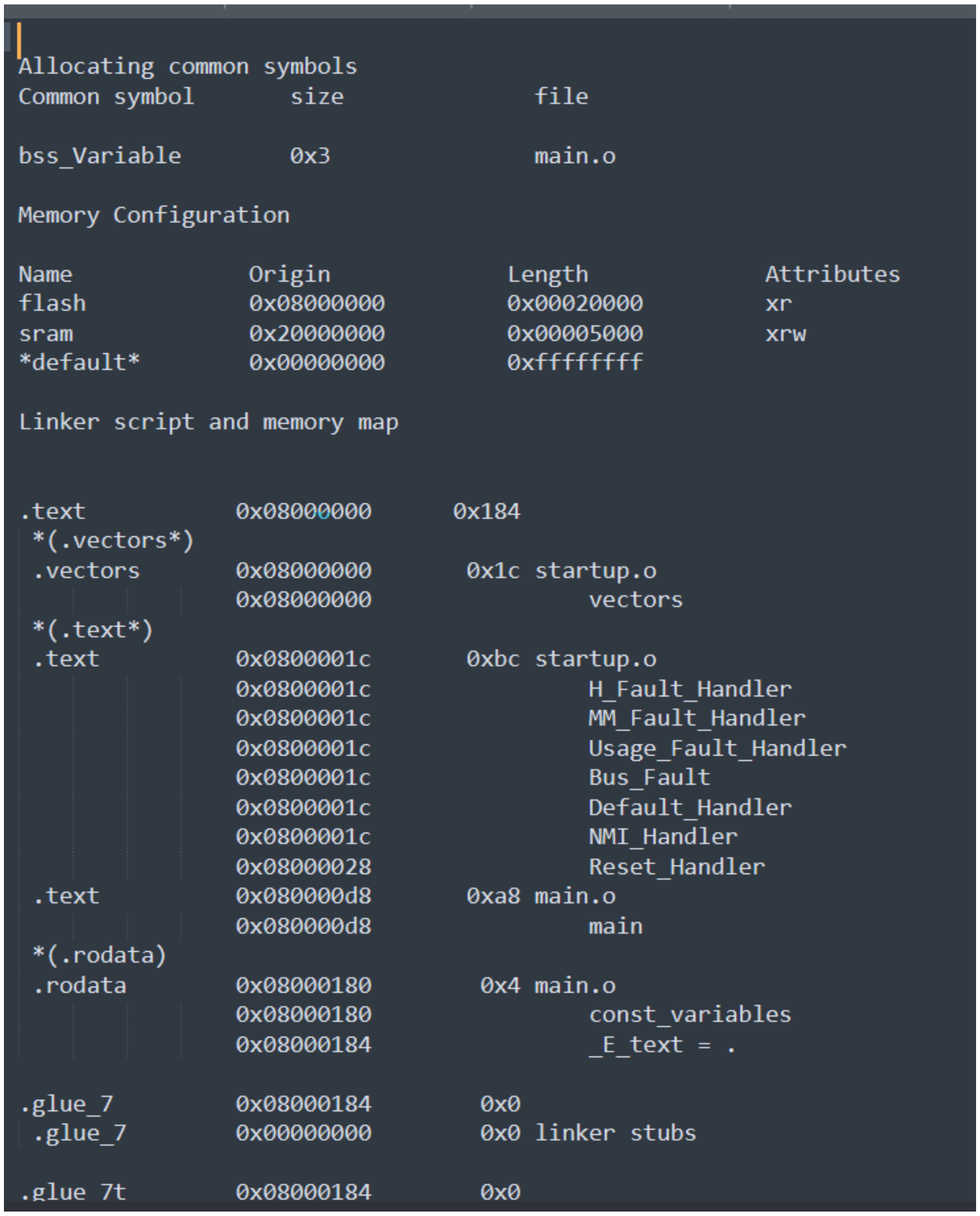
```makefile
#@copyright : eslam
CC=arm-none-eabi-
CFLAGS= -mcpu=cortex-m3 -mthumb -gdwarf-2
INCS=-I .
LIBS =
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
AS = $(wildcard *.s)
ASOBJ = $(AS:.s=.o)
project_name = learn-in-depth-cortex-m3
all: $(project_name).bin
	@echo "=========build is done========"
%.o : %.c
	 $(CC)gcc.exe $(CFLAGS)   $(INCS) -c  $< -o $@
#startup.o :startup.s
#	 $(CC)as.exe $(CFLAGS)  $< -o $@
$(project_name).elf : $(OBJ) $(ASOBJ)
	$(CC)ld.exe -T linker_script.ld $(OBJ) $(ASOBJ) -o $@ -Map=Map_file.map
$(project_name).bin : $(project_name).elf
	$(CC)objcopy.exe -O binary $< $@
clean_all :
	rm *.elf *.o *.bin
clean :
	rm *elf *.bin

```

- Building is Done

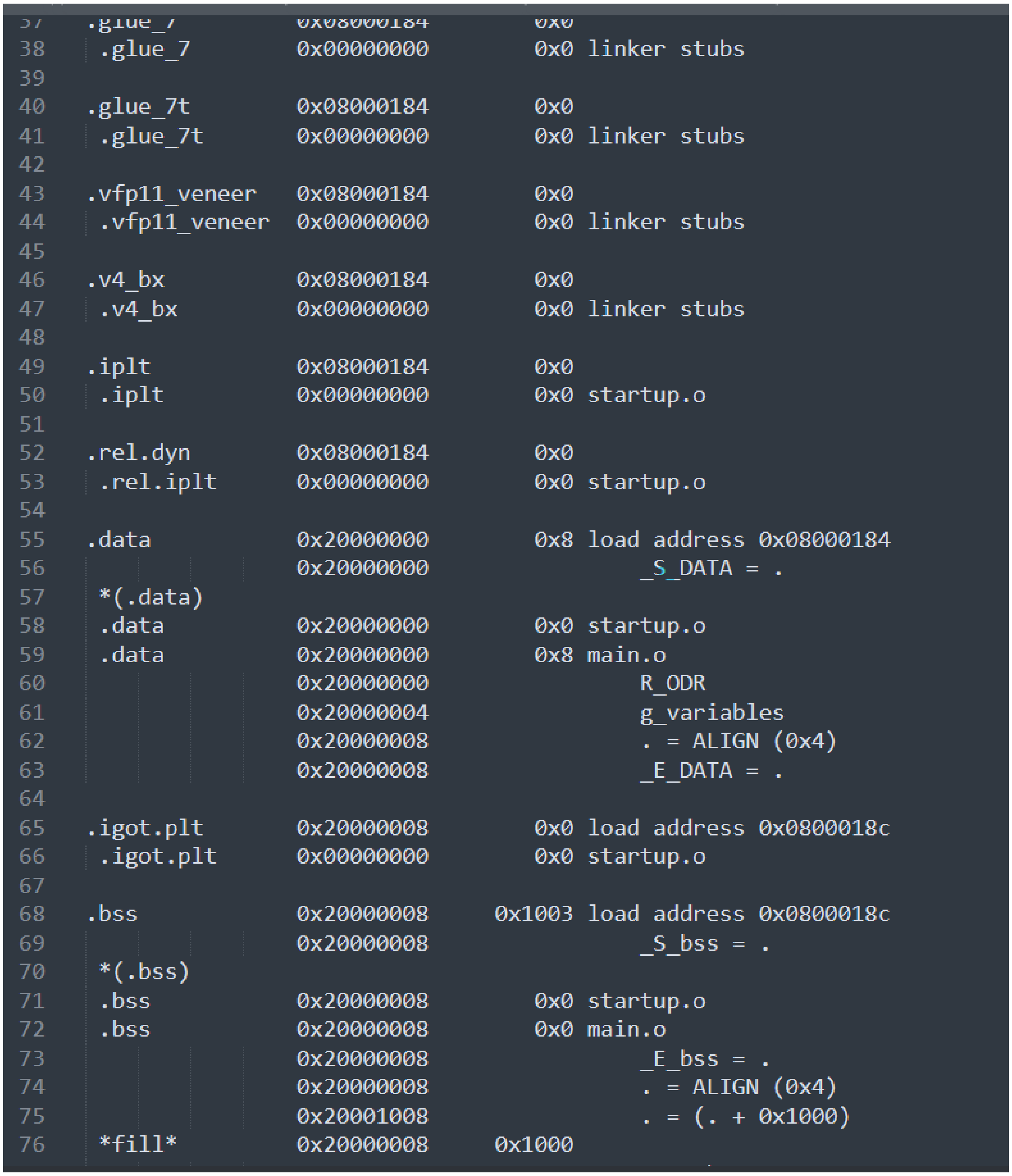

MINGW32:/e/Courses/Embedded System Diploma/unit 3/Assignment/Embedded_C/lesson_3/lab3

```
eslam@MSI MINGW32 /e/Courses/Embedded System Diploma/unit 3/Assignment/Embedded_
C/lesson_3/lab3
$ mingw32-make.exe
arm-none-eabi-ld.exe -T linker_script.ld startup.o main.o   -o learn-in-depth-co
rtex-m3.elf -Map=Map_file.map
arm-none-eabi-objcopy.exe -O binary learn-in-depth-cortex-m3.elf learn-in-depth-
cortex-m3.bin
=========build is done=========

eslam@MSI MINGW32 /e/Courses/Embedded System Diploma/unit 3/Assignment/Embedded_
C/lesson_3/lab3
$ ls
learn-in-depth-cortex-m3.bin  main.c    Map_file.map                 startup.o
learn-in-depth-cortex-m3.elf  main.o    proteus_project_STM32F103C6/  startup.s-orig
linker_script.ld              Makefile  startup.c

eslam@MSI MINGW32 /e/Courses/Embedded System Diploma/unit 3/Assignment/Embedded_C/lesson_3/lab
3
$ |
```

• Map File Details

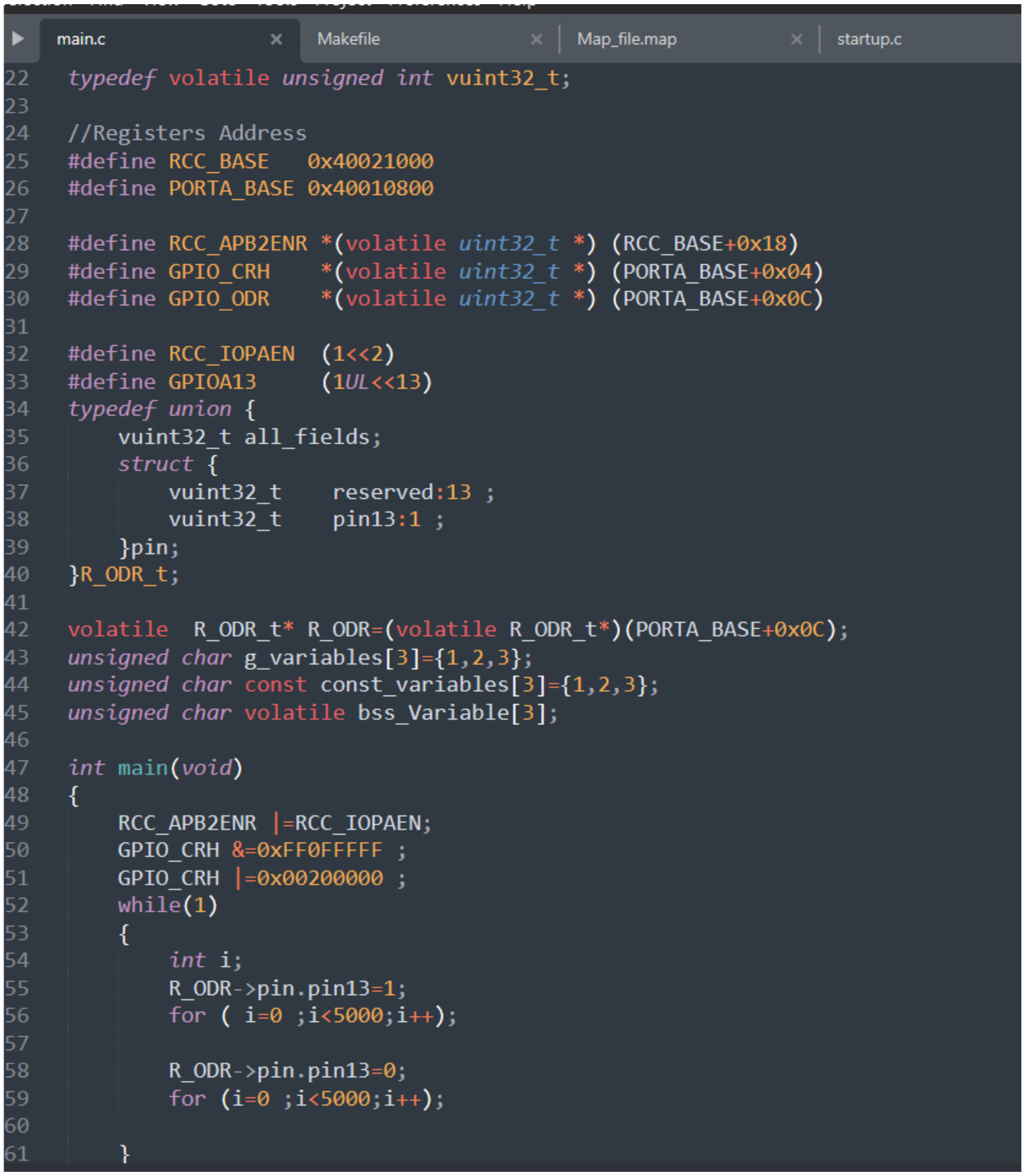


```
Allocating common symbols
Common symbol       size              file

bss_Variable        0x3               main.o

Memory Configuration

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

Linker script and memory map


.text           0x08000000      0x184
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000               vectors
 *(.text*)
 .text          0x0800001c       0xbc startup.o
                0x0800001c               H_Fault_Handler
                0x0800001c               MM_Fault_Handler
                0x0800001c               Usage_Fault_Handler
                0x0800001c               Bus_Fault
                0x0800001c               Default_Handler
                0x0800001c               NMI_Handler
                0x08000028               Reset_Handler
 .text          0x080000d8       0xa8 main.o
                0x080000d8               main
 *(.rodata)
 .rodata        0x08000180        0x4 main.o
                0x08000180               const_variables
                0x08000184               _E_text = .

.glue_7         0x08000184        0x0
 .glue_7        0x00000000        0x0 linker stubs

.glue_7t        0x08000184        0x0
```



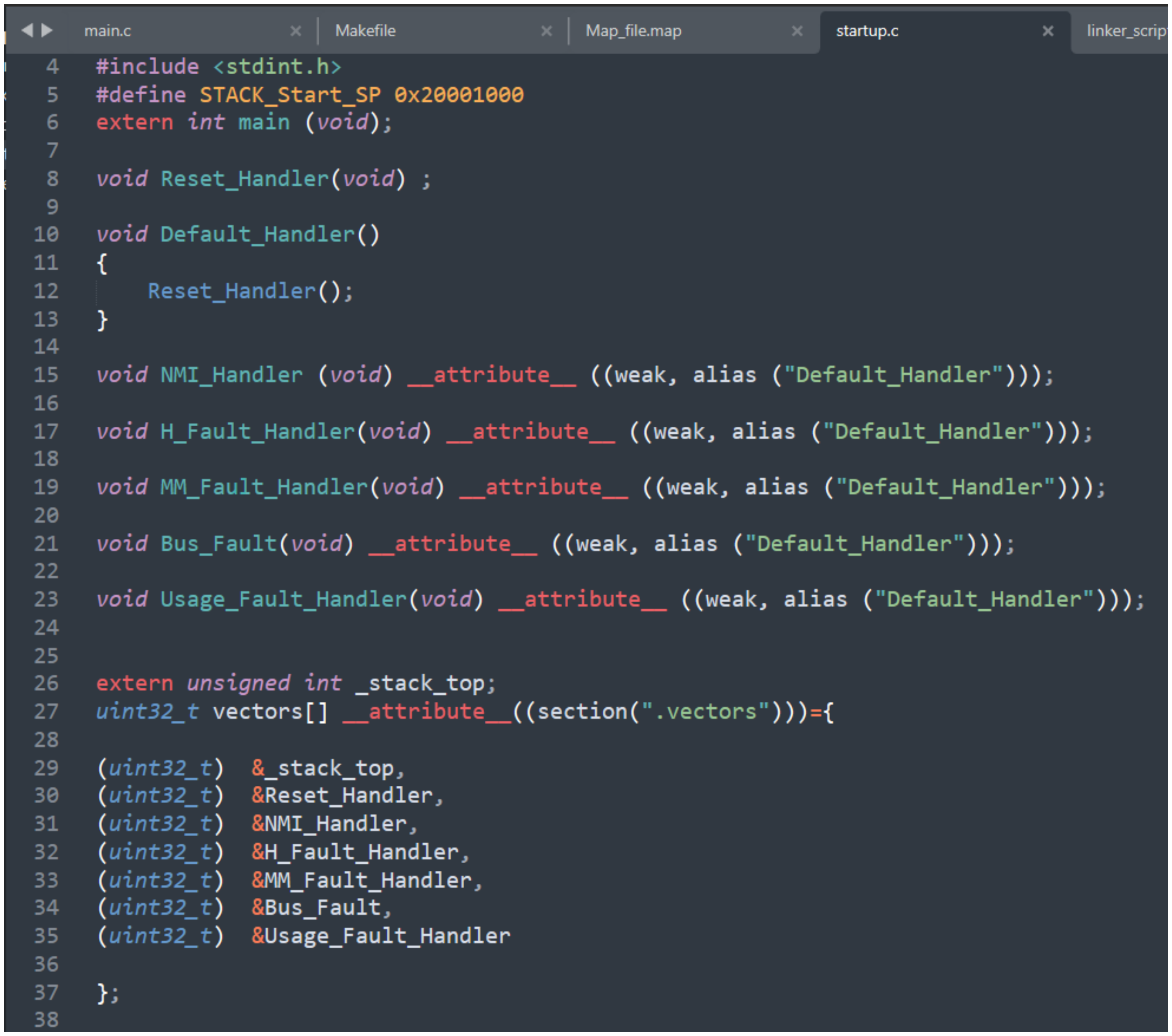
```
.glue_7         0x08000184         0x0
 .glue_7        0x00000000         0x0 linker stubs

.glue_7t        0x08000184         0x0
 .glue_7t       0x00000000         0x0 linker stubs

.vfp11_veneer   0x08000184         0x0
 .vfp11_veneer  0x00000000         0x0 linker stubs

.v4_bx          0x08000184         0x0
 .v4_bx         0x00000000         0x0 linker stubs

.iplt           0x08000184         0x0
 .iplt          0x00000000         0x0 startup.o

.rel.dyn        0x08000184         0x0
 .rel.iplt      0x00000000         0x0 startup.o

.data           0x20000000         0x8 load address 0x08000184
                0x20000000                 _S_DATA = .
 *(.data)
 .data          0x20000000         0x0 startup.o
 .data          0x20000000         0x8 main.o
                0x20000000                 R_ODR
                0x20000004                 g_variables
                0x20000008                 . = ALIGN (0x4)
                0x20000008                 _E_DATA = .

.igot.plt       0x20000008         0x0 load address 0x0800018c
 .igot.plt      0x00000000         0x0 startup.o

.bss            0x20000008      0x1003 load address 0x0800018c
                0x20000008                 _S_bss = .
 *(.bss)
 .bss           0x20000008         0x0 startup.o
 .bss           0x20000008         0x0 main.o
                0x20000008                 _E_bss = .
                0x20000008                 . = ALIGN (0x4)
                0x20001008                 . = (. + 0x1000)
 *fill*         0x20000008      0x1000
```

Lesson 3 Lab 3 with Startup.c file :

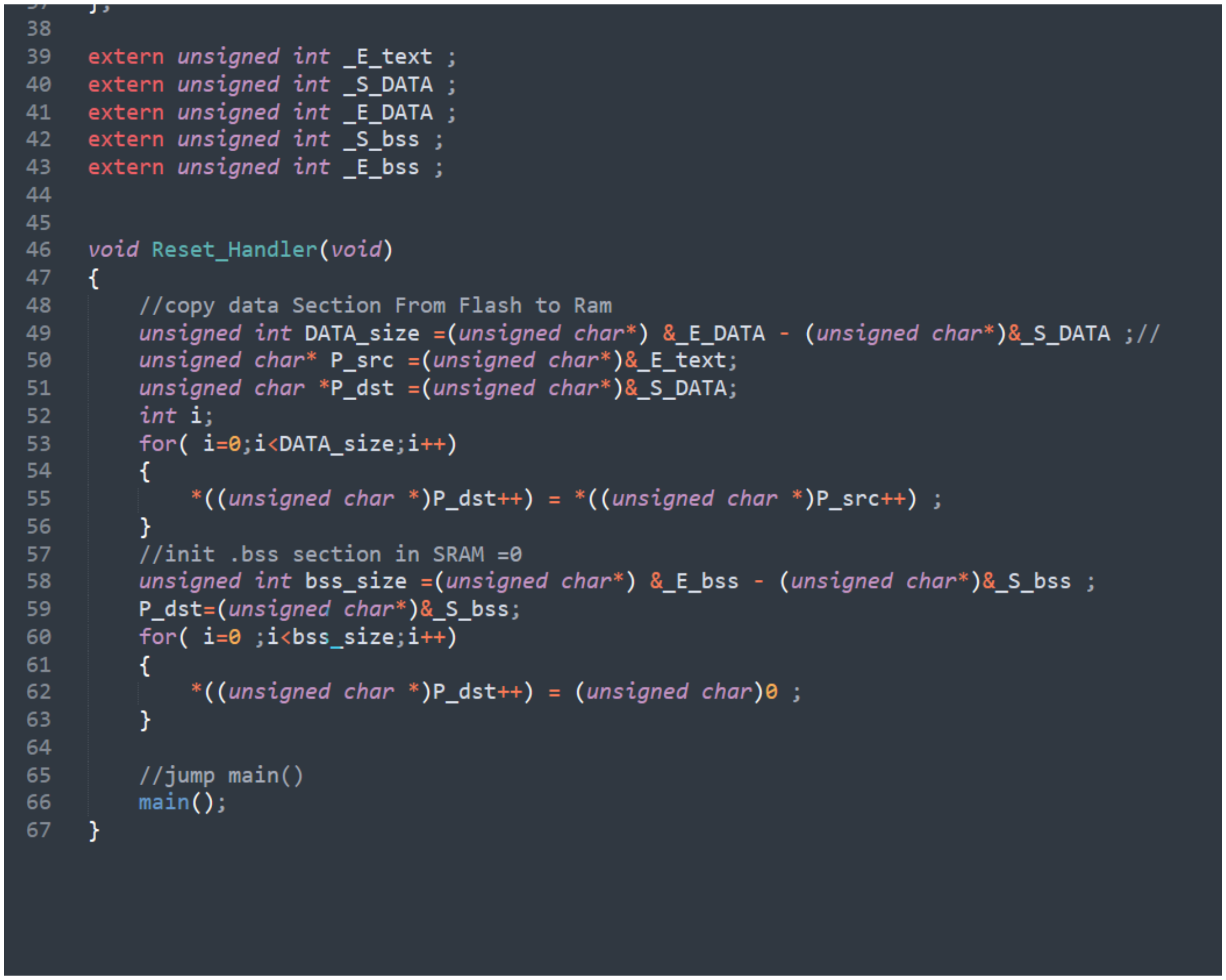


```c
typedef volatile unsigned int vuint32_t;

//Registers Address
#define RCC_BASE    0x40021000
#define PORTA_BASE 0x40010800

#define RCC_APB2ENR *(volatile uint32_t *) (RCC_BASE+0x18)
#define GPIO_CRH    *(volatile uint32_t *) (PORTA_BASE+0x04)
#define GPIO_ODR    *(volatile uint32_t *) (PORTA_BASE+0x0C)

#define RCC_IOPAEN  (1<<2)
#define GPIOA13     (1UL<<13)
typedef union {
    vuint32_t all_fields;
    struct {
        vuint32_t     reserved:13 ;
        vuint32_t     pin13:1 ;
    }pin;
}R_ODR_t;

volatile  R_ODR_t* R_ODR=(volatile R_ODR_t*)(PORTA_BASE+0x0C);
unsigned char g_variables[3]={1,2,3};
unsigned char const const_variables[3]={1,2,3};
unsigned char volatile bss_Variable[3];

int main(void)
{
    RCC_APB2ENR |=RCC_IOPAEN;
    GPIO_CRH &=0xFF0FFFFF ;
    GPIO_CRH |=0x00200000 ;
    while(1)
    {
        int i;
        R_ODR->pin.pin13=1;
        for ( i=0 ;i<5000;i++);

        R_ODR->pin.pin13=0;
        for (i=0 ;i<5000;i++);

    }
```

• Startup.c File

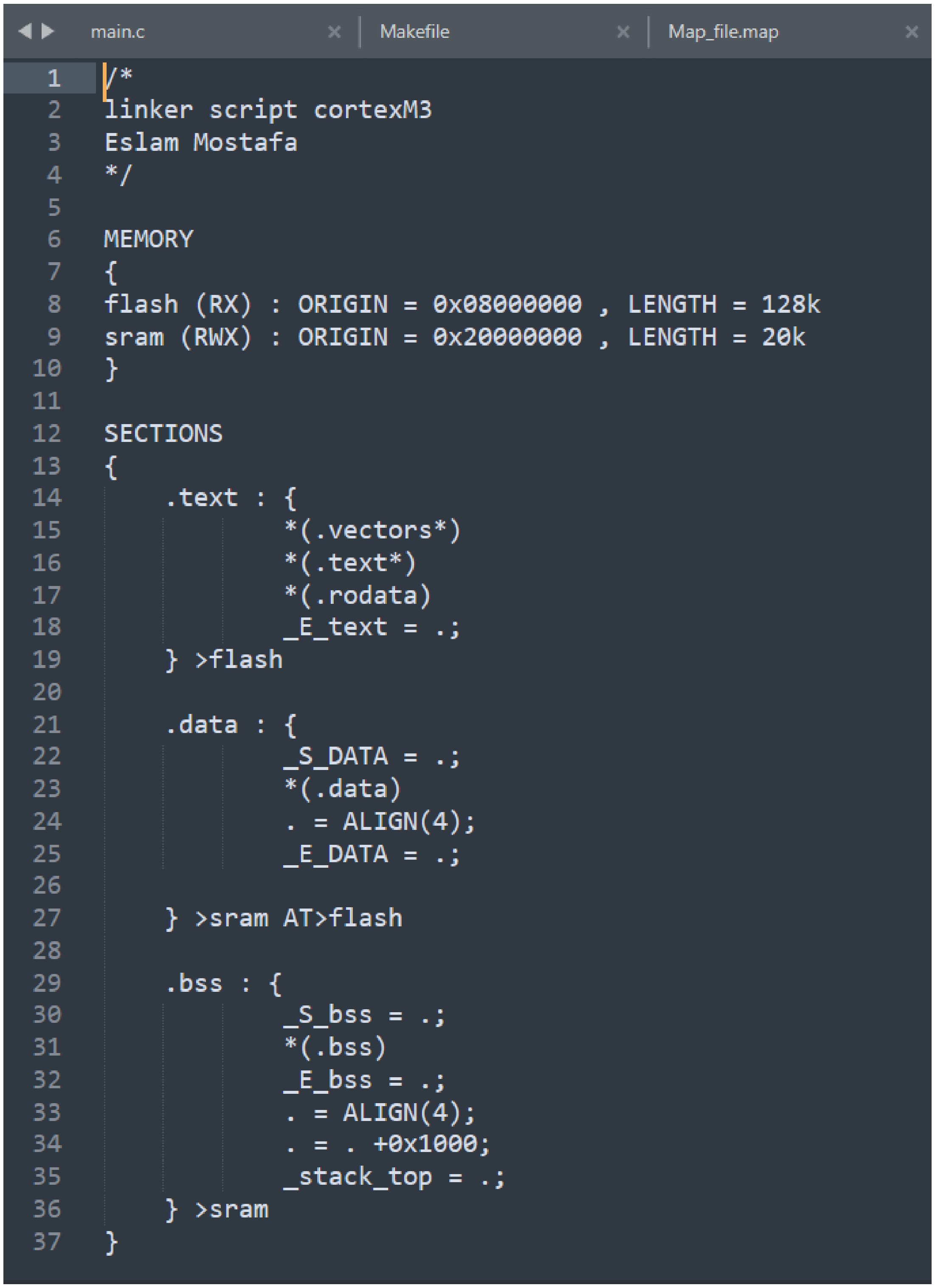


```c
#include <stdint.h>
#define STACK_Start_SP 0x20001000
extern int main (void);

void Reset_Handler(void) ;

void Default_Handler()
{
	Reset_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")));


extern unsigned int _stack_top;
uint32_t vectors[] __attribute__((section(".vectors")))={

(uint32_t)  &_stack_top,
(uint32_t)  &Reset_Handler,
(uint32_t)  &NMI_Handler,
(uint32_t)  &H_Fault_Handler,
(uint32_t)  &MM_Fault_Handler,
(uint32_t)  &Bus_Fault,
(uint32_t)  &Usage_Fault_Handler

};

```



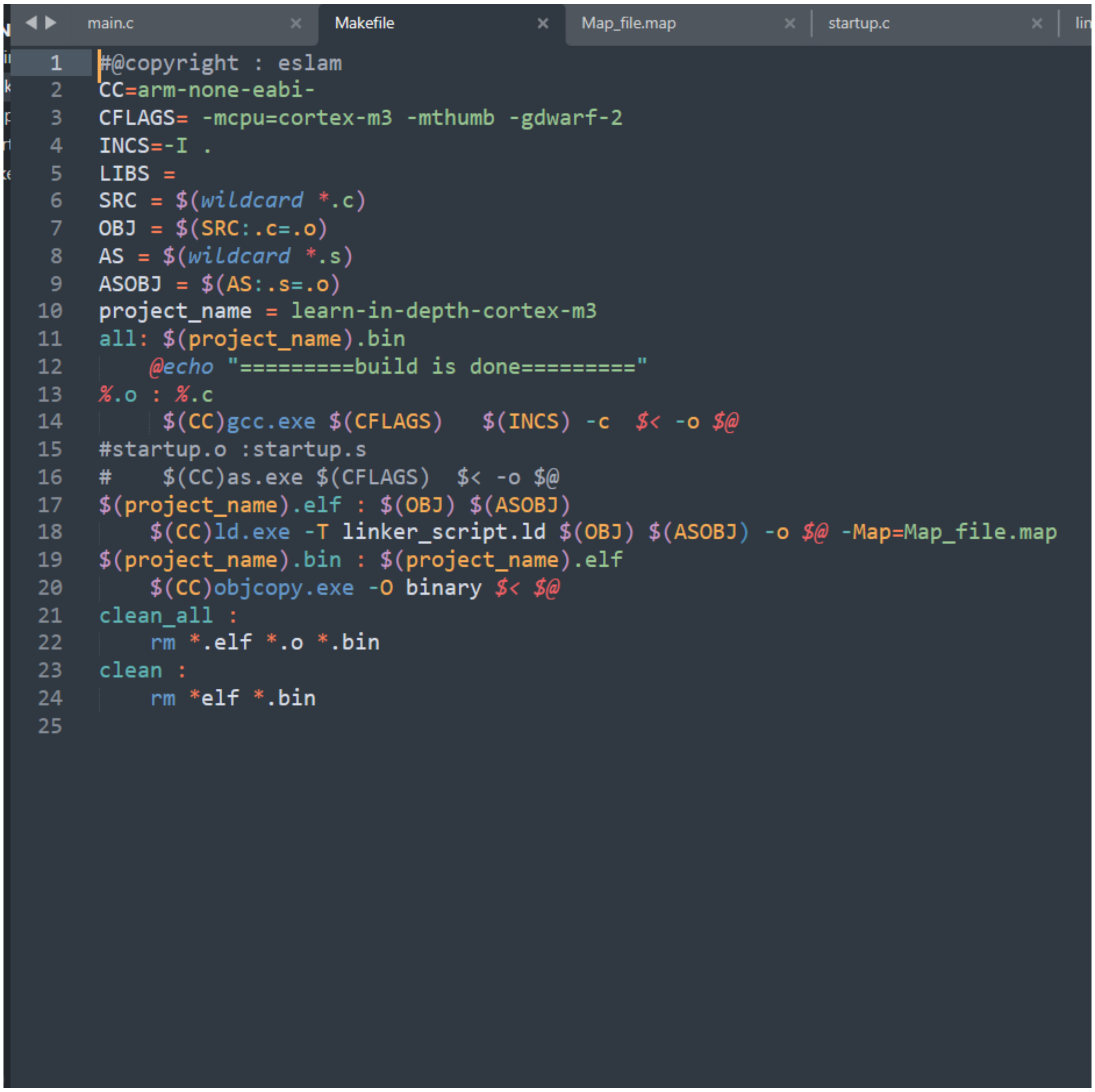
```c
};

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 ;


void Reset_Handler(void)
{
    //copy data Section From Flash 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;
    int i;
    for( i=0;i<DATA_size;i++)
    {
        *((unsigned char *)P_dst++) = *((unsigned char *)P_src++) ;
    }
    //init .bss section in SRAM =0
    unsigned int bss_size =(unsigned char*) &_E_bss - (unsigned char*)&_S_bss ;
    P_dst=(unsigned char*)&_S_bss;
    for( i=0 ;i<bss_size;i++)
    {
        *((unsigned char *)P_dst++) = (unsigned char)0 ;
    }

    //jump main()
    main();
}
```

• LinkerScript File



```
/*
linker script cortexM3
Eslam Mostafa
*/

MEMORY
{
flash (RX) : ORIGIN = 0x08000000 , LENGTH = 128k
sram (RWX) : ORIGIN = 0x20000000 , LENGTH = 20k
}

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 = .;
            . = ALIGN(4);
            . = . +0x1000;
            _stack_top = .;
    } >sram
}
```

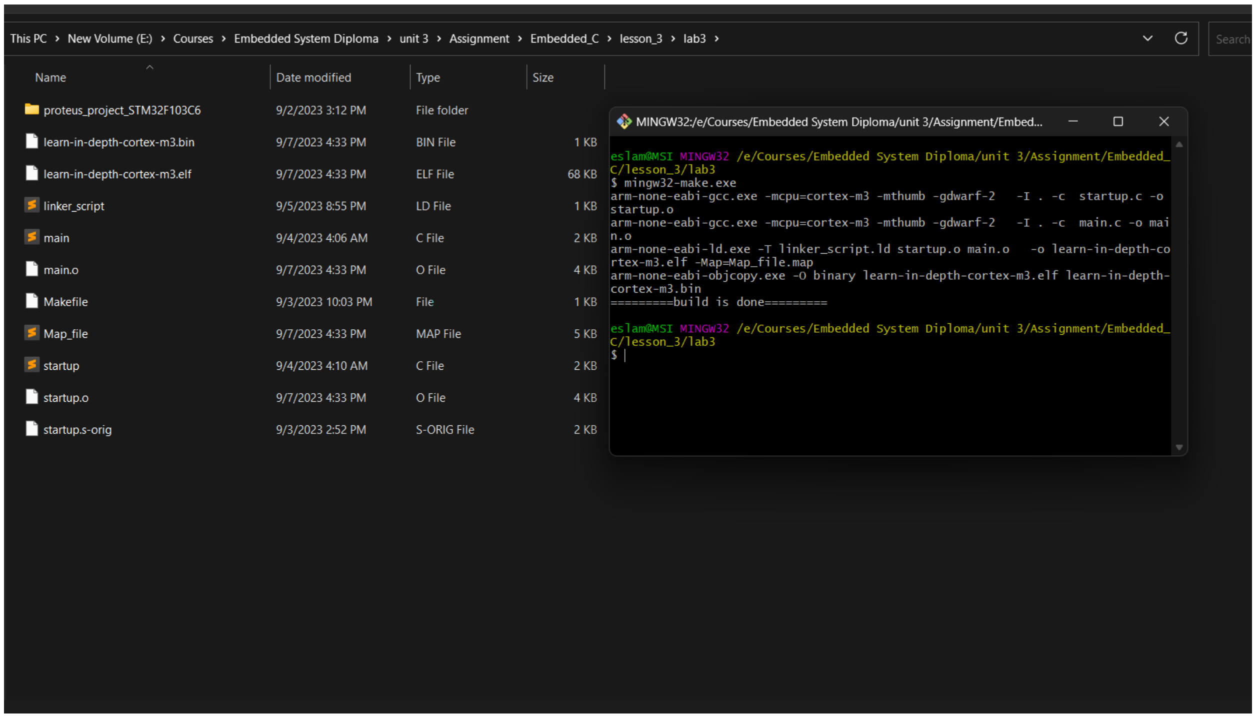
• Make File



main.c | Makefile | Map_file.map | startup.c

```
#@copyright : eslam
CC=arm-none-eabi-
CFLAGS= -mcpu=cortex-m3 -mthumb -gdwarf-2
INCS=-I .
LIBS =
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
AS = $(wildcard *.s)
ASOBJ = $(AS:.s=.o)
project_name = learn-in-depth-cortex-m3
all: $(project_name).bin
	@echo "=========build is done========="
%.o : %.c
	 $(CC)gcc.exe $(CFLAGS)   $(INCS) -c  $< -o $@
#startup.o :startup.s
#	 $(CC)as.exe $(CFLAGS)  $< -o $@
$(project_name).elf : $(OBJ) $(ASOBJ)
	$(CC)ld.exe -T linker_script.ld $(OBJ) $(ASOBJ) -o $@ -Map=Map_file.map
$(project_name).bin : $(project_name).elf
	$(CC)objcopy.exe -O binary $< $@
clean_all :
	rm *.elf *.o *.bin
clean :
	rm *elf *.bin

```

- Build is Done



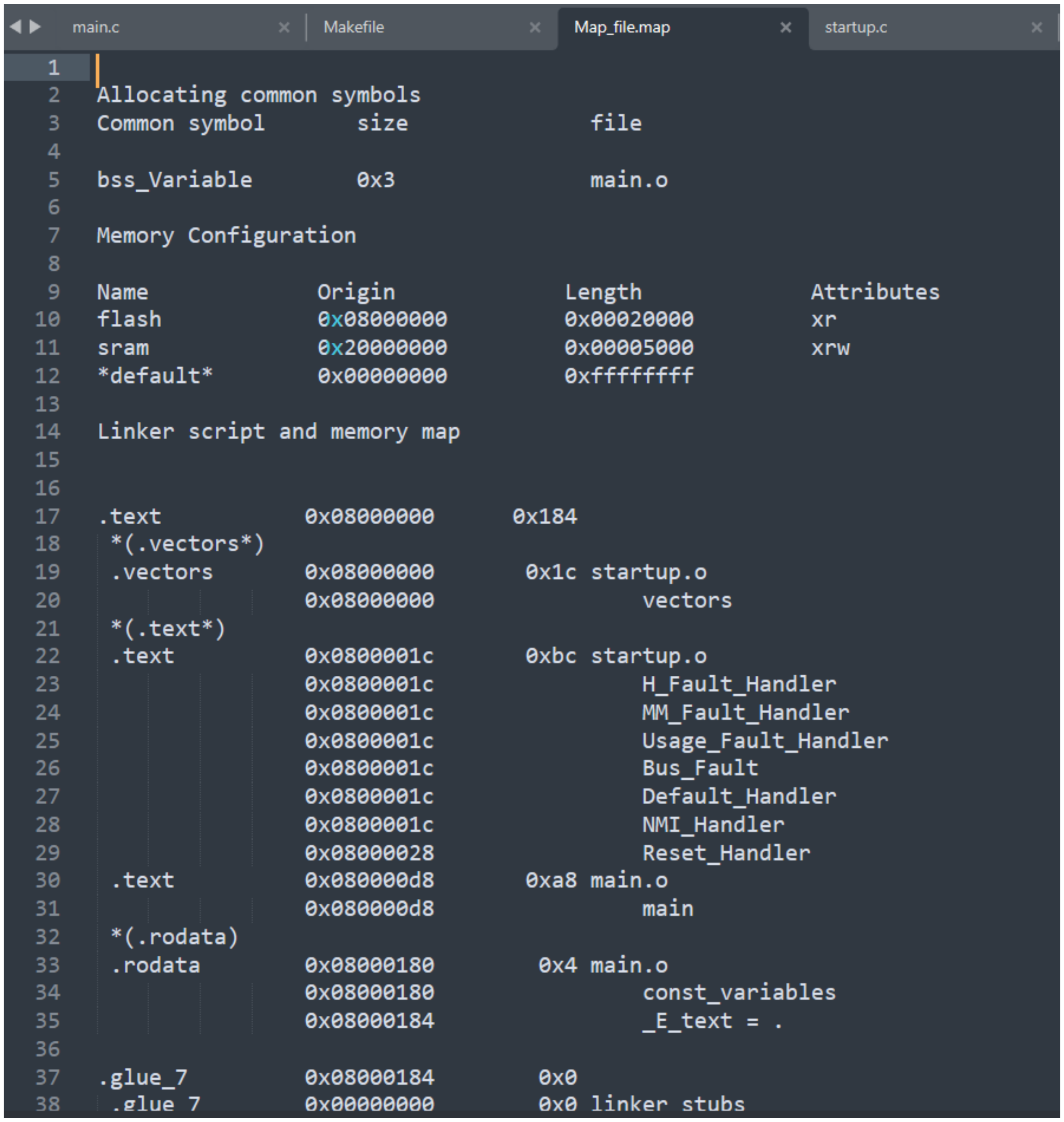
This PC › New Volume (E:) › Courses › Embedded System Diploma › unit 3 › Assignment › Embedded_C › lesson_3 › lab3 ›

Name	Date modified	Type	Size
proteus_project_STM32F103C6	9/2/2023 3:12 PM	File folder	
learn-in-depth-cortex-m3.bin	9/7/2023 4:33 PM	BIN File	1 KB
learn-in-depth-cortex-m3.elf	9/7/2023 4:33 PM	ELF File	68 KB
linker_script	9/5/2023 8:55 PM	LD File	1 KB
main	9/4/2023 4:06 AM	C File	2 KB
main.o	9/7/2023 4:33 PM	O File	4 KB
Makefile	9/3/2023 10:03 PM	File	1 KB
Map_file	9/7/2023 4:33 PM	MAP File	5 KB
startup	9/4/2023 4:10 AM	C File	2 KB
startup.o	9/7/2023 4:33 PM	O File	4 KB
startup.s-orig	9/3/2023 2:52 PM	S-ORIG File	2 KB

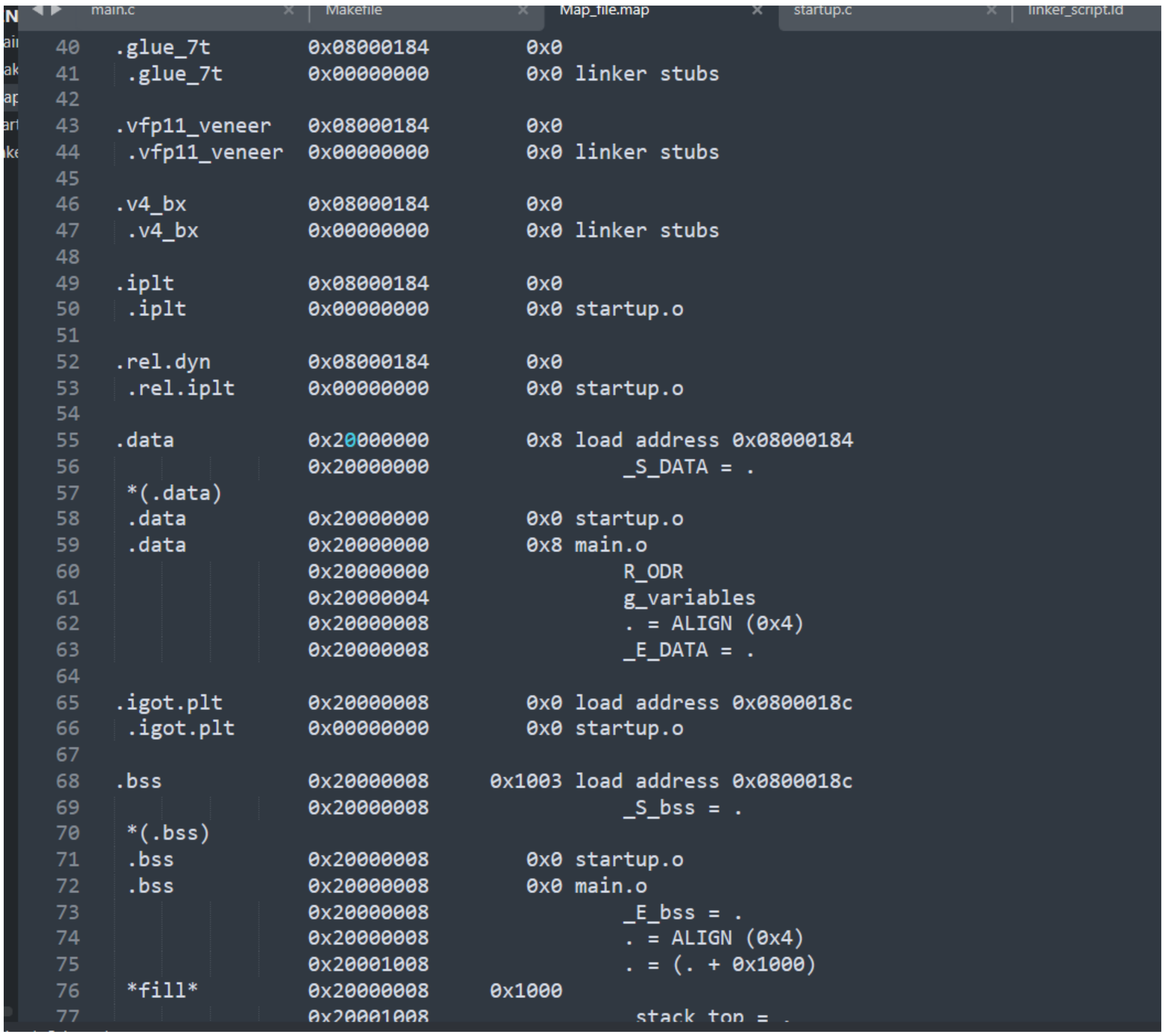
MINGW32:/e/Courses/Embedded System Diploma/unit 3/Assignment/Embed...

```
eslam@MSI MINGW32 /e/Courses/Embedded System Diploma/unit 3/Assignment/Embedded_
C/lesson_3/lab3
$ mingw32-make.exe
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -mthumb -gdwarf-2   -I . -c  startup.c -o
startup.o
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -mthumb -gdwarf-2   -I . -c  main.c -o mai
n.o
arm-none-eabi-ld.exe -T linker_script.ld startup.o main.o   -o learn-in-depth-co
rtex-m3.elf -Map=Map_file.map
arm-none-eabi-objcopy.exe -O binary learn-in-depth-cortex-m3.elf learn-in-depth-
cortex-m3.bin
=========build is done=========

eslam@MSI MINGW32 /e/Courses/Embedded System Diploma/unit 3/Assignment/Embedded_
C/lesson_3/lab3
$ |
```

• Map file Details

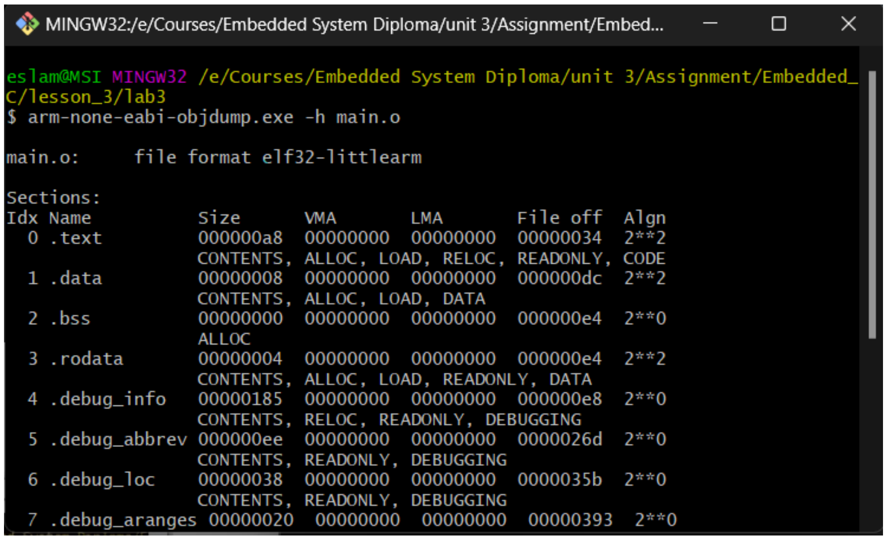


```

Allocating common symbols
Common symbol       size              file

bss_Variable        0x3               main.o

Memory Configuration

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

Linker script and memory map


.text           0x08000000      0x184
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                vectors
 *(.text*)
 .text          0x0800001c       0xbc startup.o
                0x0800001c                H_Fault_Handler
                0x0800001c                MM_Fault_Handler
                0x0800001c                Usage_Fault_Handler
                0x0800001c                Bus_Fault
                0x0800001c                Default_Handler
                0x0800001c                NMI_Handler
                0x08000028                Reset_Handler
 .text          0x080000d8       0xa8 main.o
                0x080000d8                main
 *(.rodata)
 .rodata        0x08000180        0x4 main.o
                0x08000180                const_variables
                0x08000184                _E_text = .

.glue_7         0x08000184        0x0
 .glue_7        0x00000000        0x0 linker stubs
```



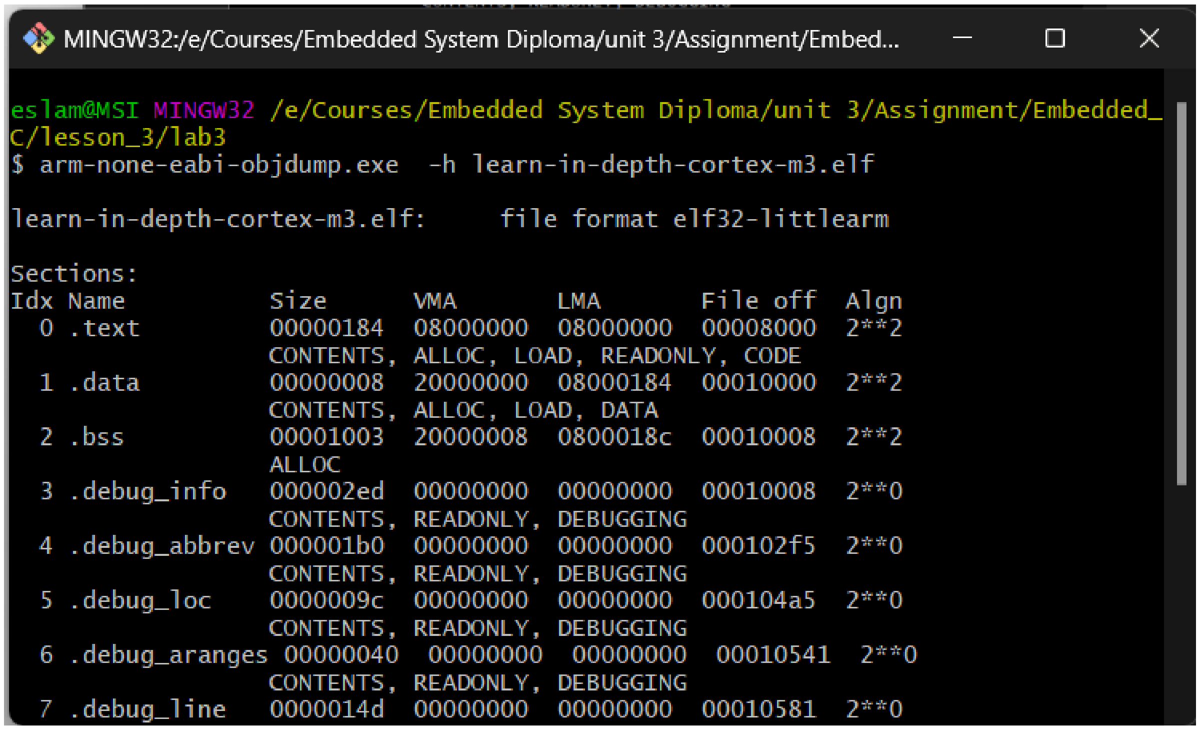
```
.glue_7t        0x08000184        0x0
 .glue_7t       0x00000000        0x0 linker stubs

.vfp11_veneer   0x08000184        0x0
 .vfp11_veneer  0x00000000        0x0 linker stubs

.v4_bx          0x08000184        0x0
 .v4_bx         0x00000000        0x0 linker stubs

.iplt           0x08000184        0x0
 .iplt          0x00000000        0x0 startup.o

.rel.dyn        0x08000184        0x0
 .rel.iplt      0x00000000        0x0 startup.o

.data           0x20000000        0x8 load address 0x08000184
                0x20000000                _S_DATA = .
 *(.data)
 .data          0x20000000        0x0 startup.o
 .data          0x20000000        0x8 main.o
                0x20000000                R_ODR
                0x20000004                g_variables
                0x20000008                . = ALIGN (0x4)
                0x20000008                _E_DATA = .

.igot.plt       0x20000008        0x0 load address 0x0800018c
 .igot.plt      0x00000000        0x0 startup.o

.bss            0x20000008     0x1003 load address 0x0800018c
                0x20000008                _S_bss = .
 *(.bss)
 .bss           0x20000008        0x0 startup.o
 .bss           0x20000008        0x0 main.o
                0x20000008                _E_bss = .
                0x20000008                . = ALIGN (0x4)
                0x20001008                . = (. + 0x1000)
 *fill*         0x20000008     0x1000
                0x20001008                 _stack_top = .
```

• Use Command Objdump To For main.o File



```
eslam@MSI MINGW32 /e/Courses/Embedded System Diploma/unit 3/Assignment/Embedded_
C/lesson_3/lab3
$ arm-none-eabi-objdump.exe -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000a8  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000008  00000000  00000000  000000dc  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000e4  2**0
                  ALLOC
  3 .rodata       00000004  00000000  00000000  000000e4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .debug_info   00000185  00000000  00000000  000000e8  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_abbrev 000000ee  00000000  00000000  0000026d  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_loc    00000038  00000000  00000000  0000035b  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  00000393  2**0
```

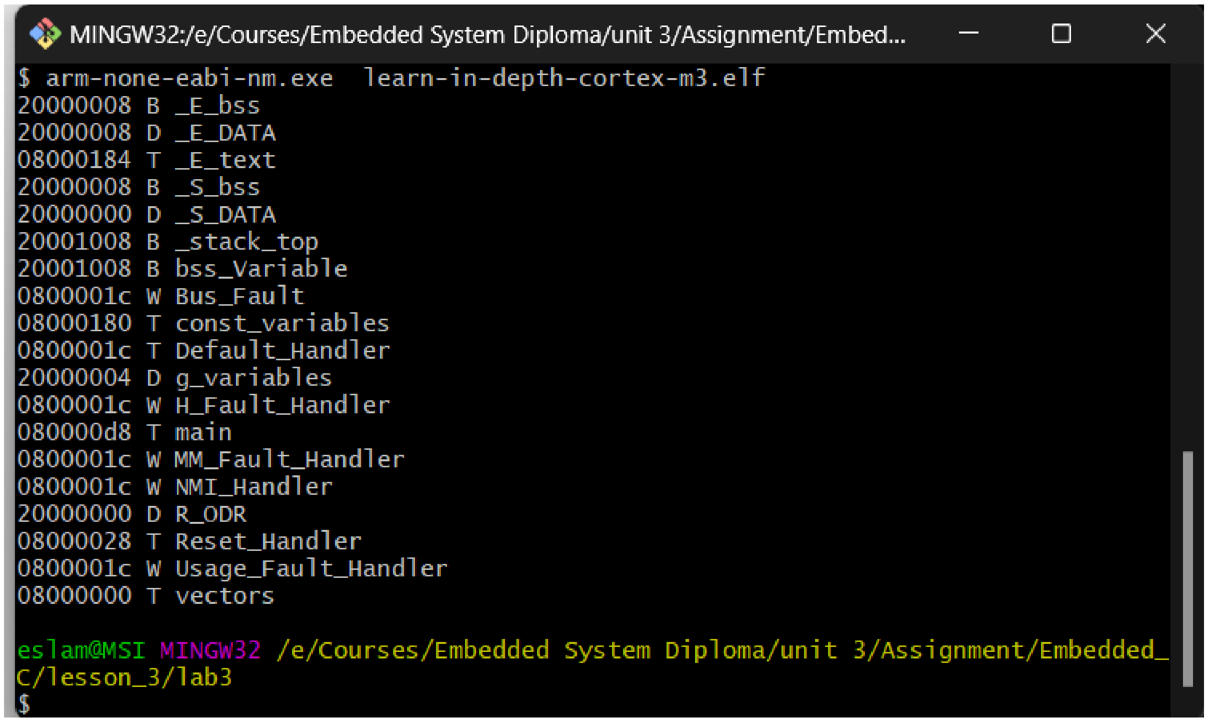
• Use Command Objdump To For The output Elf file



MINGW32:/e/Courses/Embedded System Diploma/unit 3/Assignment/Embed...

```
eslam@MSI MINGW32 /e/Courses/Embedded System Diploma/unit 3/Assignment/Embedded_
C/lesson_3/lab3
$ arm-none-eabi-objdump.exe  -h learn-in-depth-cortex-m3.elf

learn-in-depth-cortex-m3.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000184  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000008  20000000  08000184  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001003  20000008  0800018c  00010008  2**2
                  ALLOC
  3 .debug_info   000002ed  00000000  00000000  00010008  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 000001b0  00000000  00000000  000102f5  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000009c  00000000  00000000  000104a5  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000040  00000000  00000000  00010541  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   0000014d  00000000  00000000  00010581  2**0
```

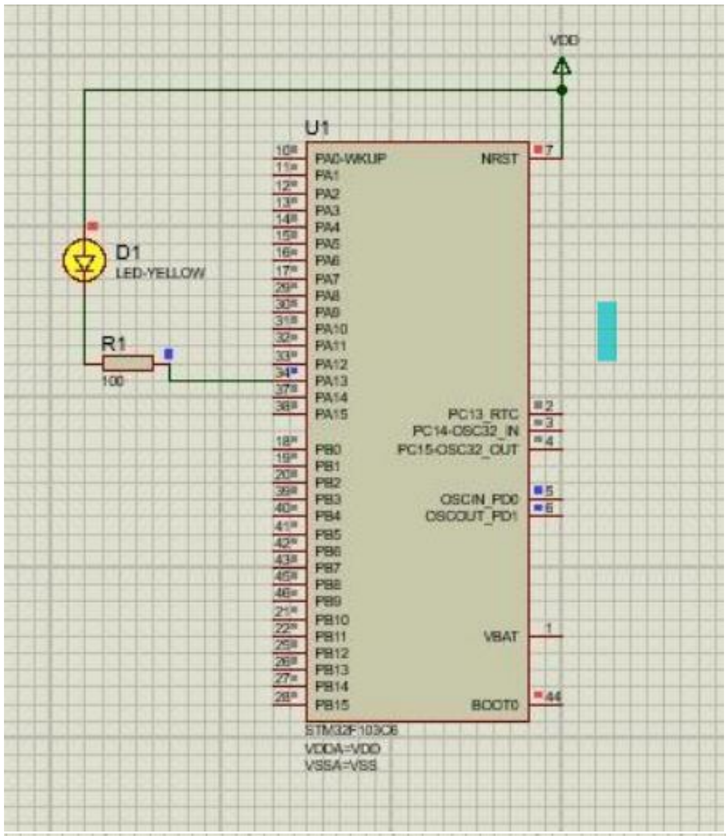
• Use Command nm to See The Symbols of Project elf

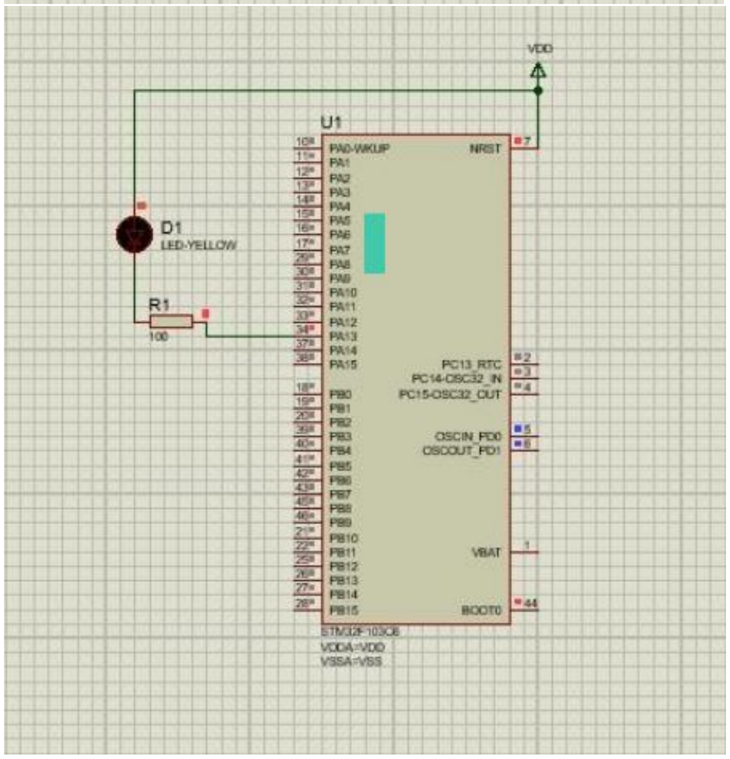


MINGW32:/e/Courses/Embedded System Diploma/unit 3/Assignment/Embed...

```
$ arm-none-eabi-nm.exe  learn-in-depth-cortex-m3.elf
20000008 B _E_bss
20000008 D _E_DATA
08000184 T _E_text
20000008 B _S_bss
20000000 D _S_DATA
20001008 B _stack_top
20001008 B bss_Variable
0800001c W Bus_Fault
08000180 T const_variables
0800001c T Default_Handler
20000004 D g_variables
0800001c W H_Fault_Handler
080000d8 T main
0800001c W MM_Fault_Handler
0800001c W NMI_Handler
20000000 D R_ODR
08000028 T Reset_Handler
0800001c W Usage_Fault_Handler
08000000 T vectors

eslam@MSI MINGW32 /e/Courses/Embedded System Diploma/unit 3/Assignment/Embedded_
C/lesson_3/lab3
$
```

• Output In Proteus


VDD
U1
D1
LED-YELLOW
R1
100
PA0-WKUP
NRST
PC13_RTC
PC14-OSC32_IN
PC15-OSC32_OUT
OSCIN_PD0
OSCOUT_PD1
VBAT
BOOT0
STM32F103C6
VDDA=VDD
VSSA=VSS


VDD
U1
D1
LED-YELLOW
R1
100
PA0-WKUP
NRST
PC13_RTC
PC14-OSC32_IN
PC15-OSC32_OUT
OSCIN_PD0
OSCOUT_PD1
VBAT
BOOT0
STM32F103C6
VDDA=VDD
VSSA=VSS