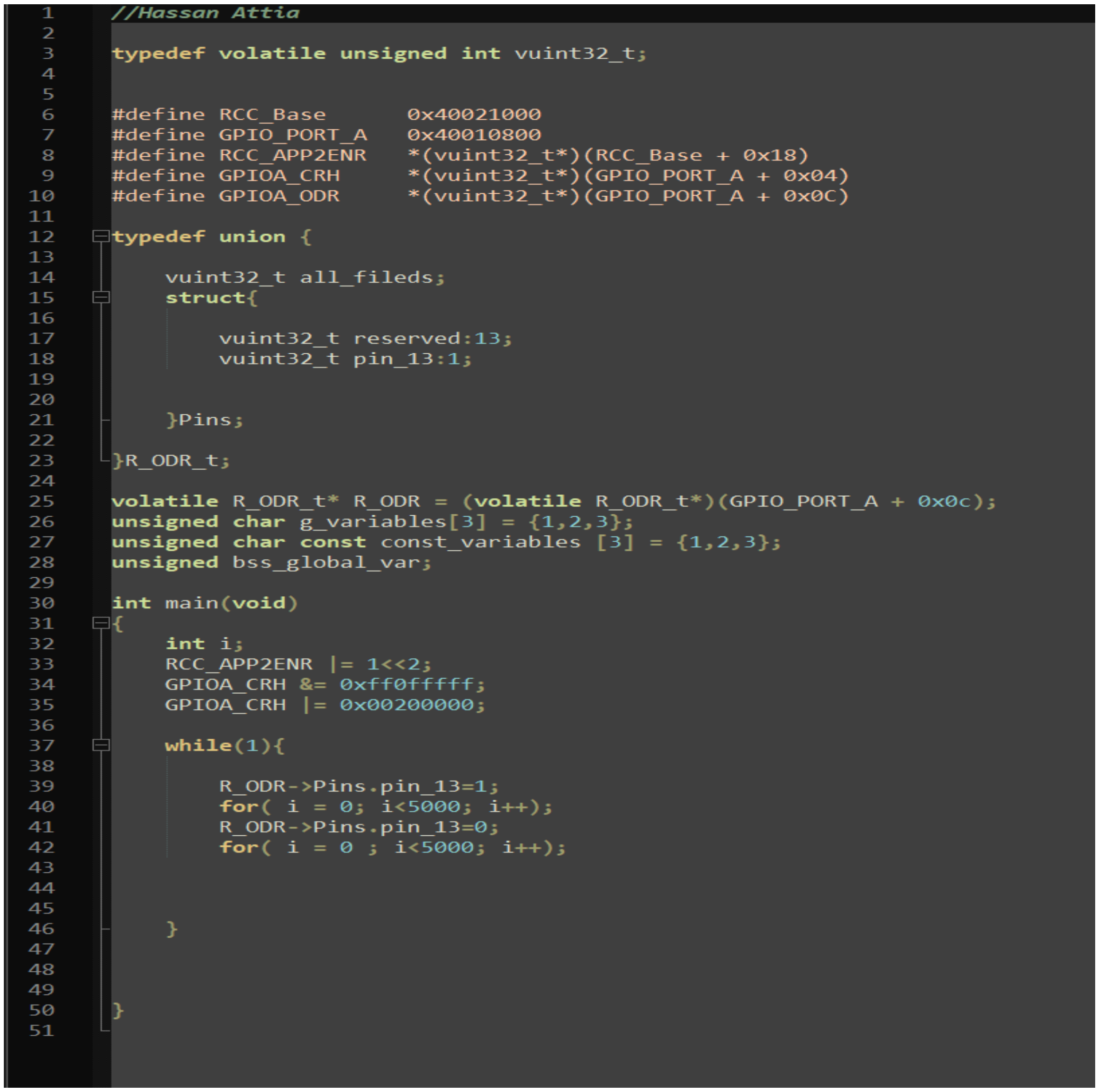
BAREMETAL SOFTWARE TO TOGGLE A LED ON ARM CORTEX M3

Author: Hassan Attia

Brief:

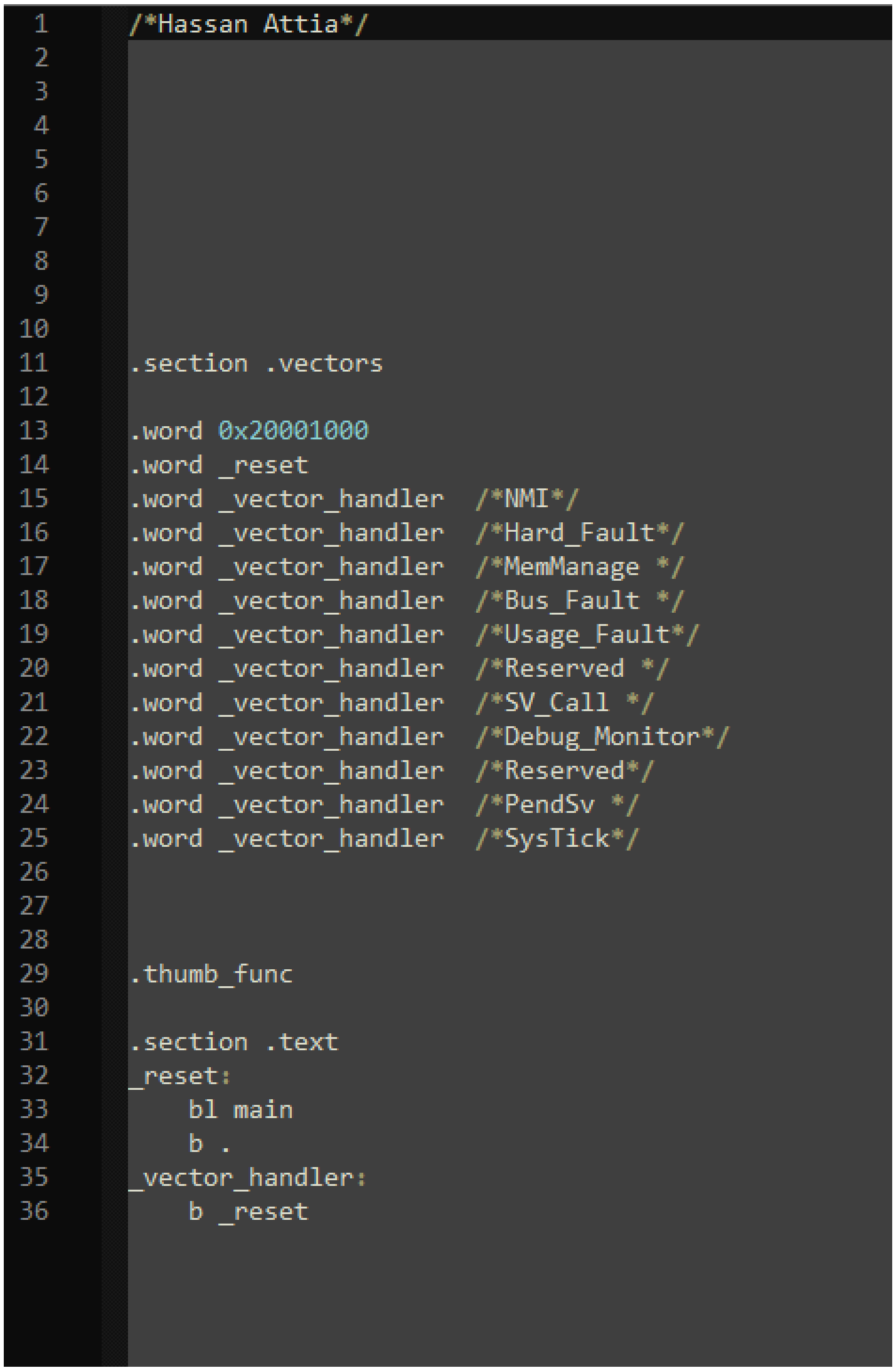
- **I made a bare metal software on cortex M3 processor but with startup.s.**
- **I'll put some screenshots for these processes including(compiling the files using git terminal, LED toggling simulation, map file sections, and debugging via proteus).**

1-Main.c



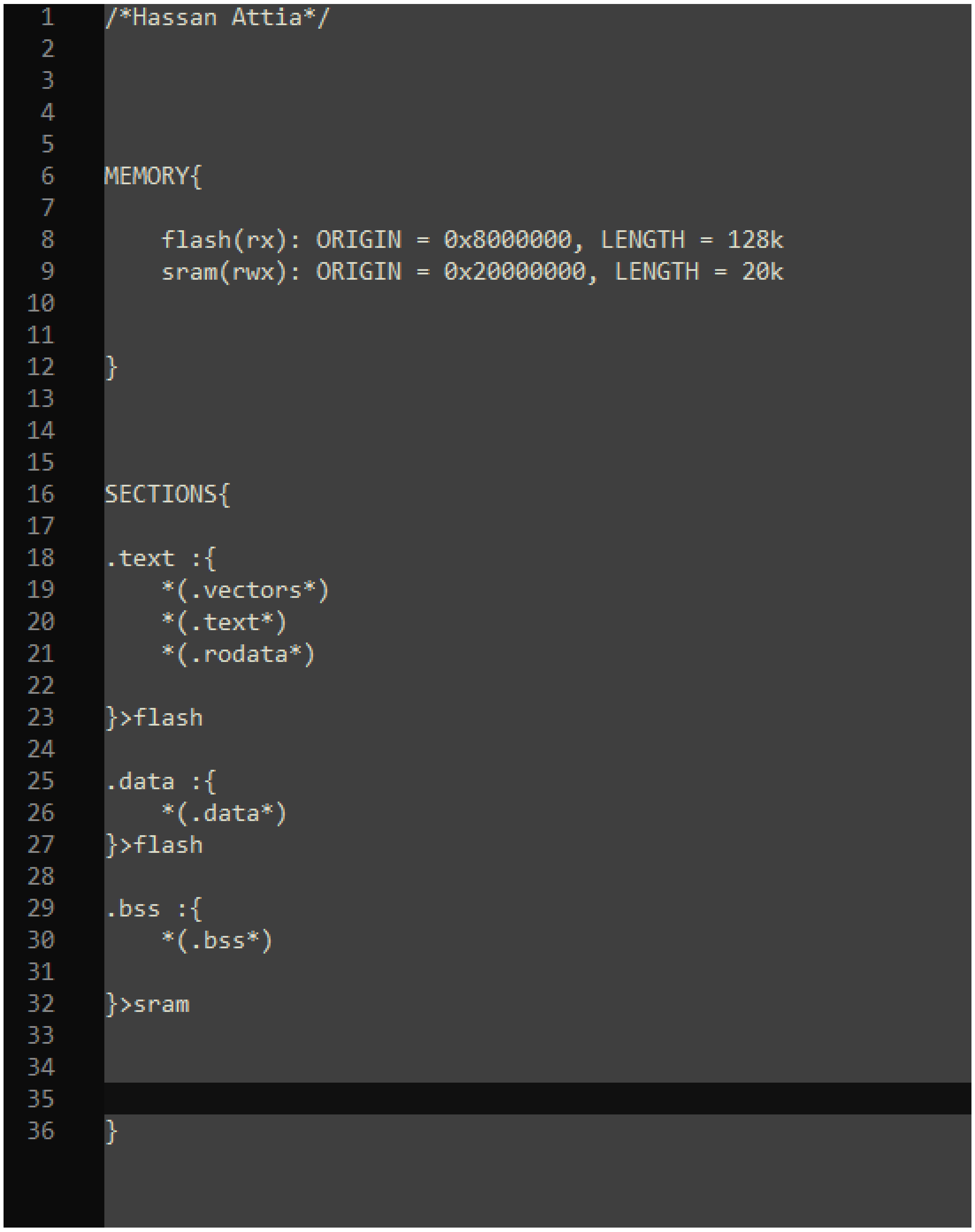
```c
//Hassan Attia

typedef volatile unsigned int vuint32_t;


#define RCC_Base       0x40021000
#define GPIO_PORT_A    0x40010800
#define RCC_APP2ENR    *(vuint32_t*)(RCC_Base + 0x18)
#define GPIOA_CRH      *(vuint32_t*)(GPIO_PORT_A + 0x04)
#define GPIOA_ODR      *(vuint32_t*)(GPIO_PORT_A + 0x0C)

typedef union {

    vuint32_t all_fileds;
    struct{

        vuint32_t reserved:13;
        vuint32_t pin_13:1;


    }Pins;

}R_ODR_t;

volatile R_ODR_t* R_ODR = (volatile R_ODR_t*)(GPIO_PORT_A + 0x0c);
unsigned char g_variables[3] = {1,2,3};
unsigned char const const_variables [3] = {1,2,3};
unsigned bss_global_var;

int main(void)
{
    int i;
    RCC_APP2ENR |= 1<<2;
    GPIOA_CRH &= 0xff0fffff;
    GPIOA_CRH |= 0x00200000;

    while(1){

        R_ODR->Pins.pin_13=1;
        for( i = 0; i<5000; i++);
        R_ODR->Pins.pin_13=0;
        for( i = 0 ; i<5000; i++);



    }



}
```

2-Startup.s



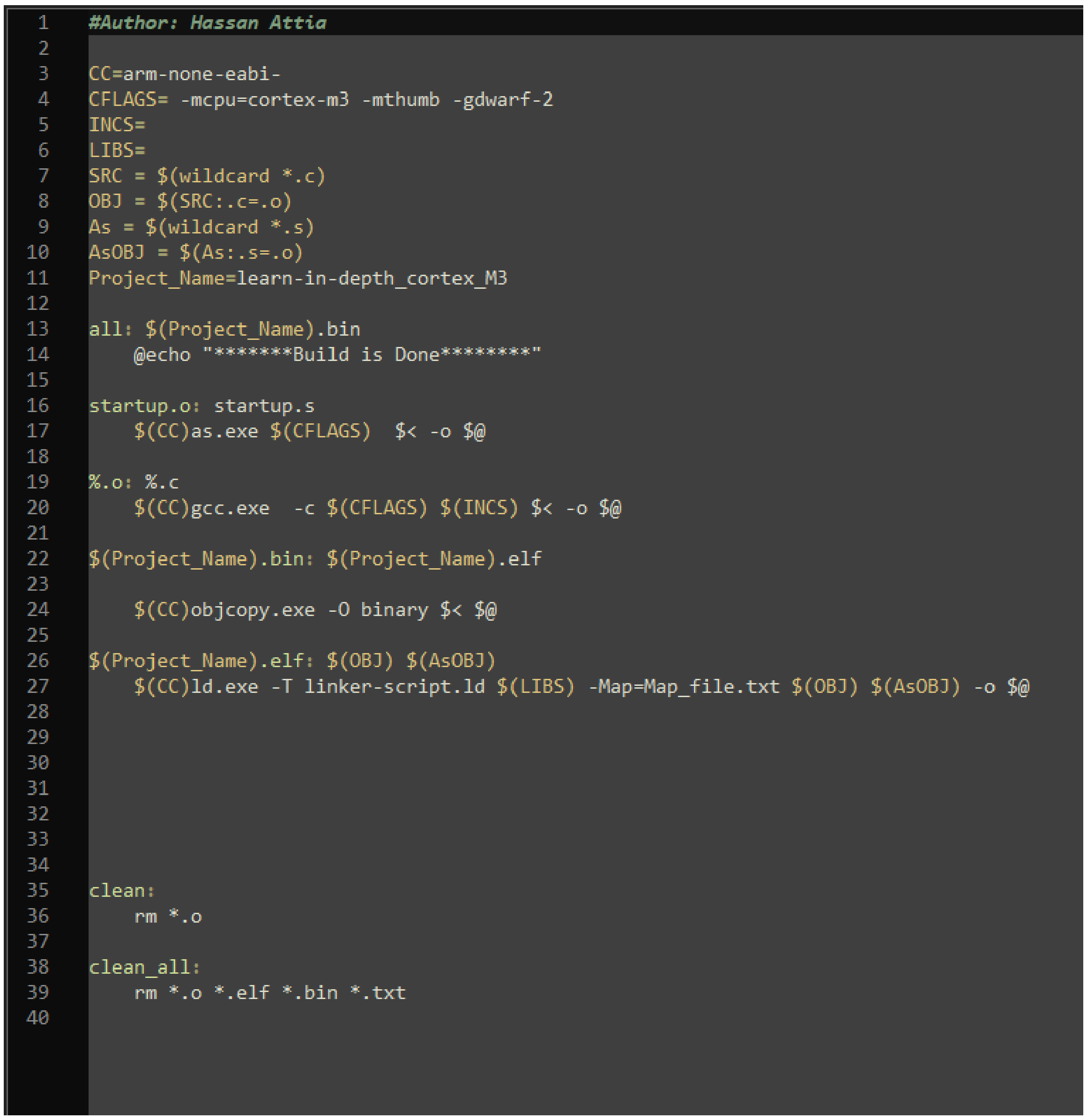
```
/*Hassan Attia*/









.section .vectors

.word 0x20001000
.word _reset
.word _vector_handler   /*NMI*/
.word _vector_handler   /*Hard_Fault*/
.word _vector_handler   /*MemManage */
.word _vector_handler   /*Bus_Fault */
.word _vector_handler   /*Usage_Fault*/
.word _vector_handler   /*Reserved */
.word _vector_handler   /*SV_Call */
.word _vector_handler   /*Debug_Monitor*/
.word _vector_handler   /*Reserved*/
.word _vector_handler   /*PendSv */
.word _vector_handler   /*SysTick*/



.thumb_func

.section .text
_reset:
    bl main
    b .
_vector_handler:
    b _reset
```

3-Linker Script:



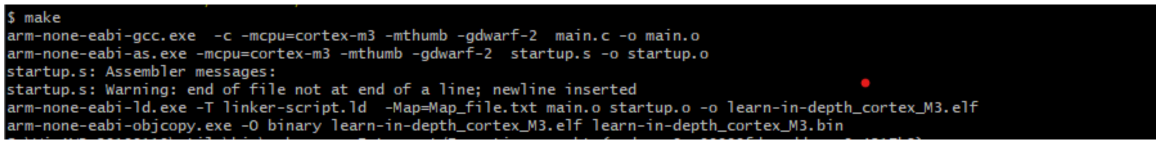
```
/*Hassan Attia*/




MEMORY{

    flash(rx): ORIGIN = 0x8000000, LENGTH = 128k
    sram(rwx): ORIGIN = 0x20000000, LENGTH = 20k


}



SECTIONS{

.text :{
    *(.vectors*)
    *(.text*)
    *(.rodata*)

}>flash

.data :{
    *(.data*)
}>flash

.bss :{
    *(.bss*)

}>sram



}
```

4-Makefile:



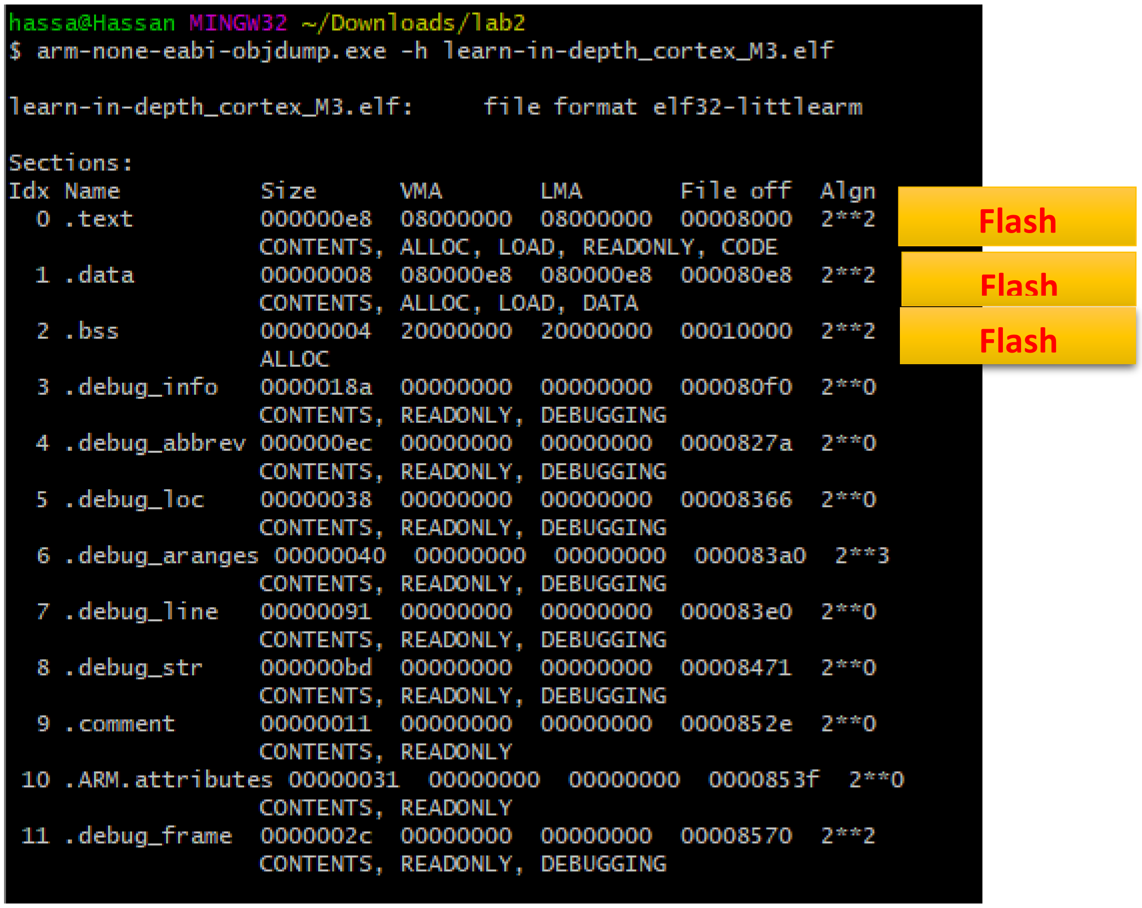
```
#Author: Hassan Attia

CC=arm-none-eabi-
CFLAGS= -mcpu=cortex-m3 -mthumb -gdwarf-2
INCS=
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********"

startup.o: startup.s
	$(CC)as.exe $(CFLAGS)  $< -o $@

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

$(Project_Name).bin: $(Project_Name).elf

	$(CC)objcopy.exe -O binary $< $@

$(Project_Name).elf: $(OBJ) $(AsOBJ)
	$(CC)ld.exe -T linker-script.ld $(LIBS) -Map=Map_file.txt $(OBJ) $(AsOBJ) -o $@








clean:
	rm *.o

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

```

5-Compiling using Make



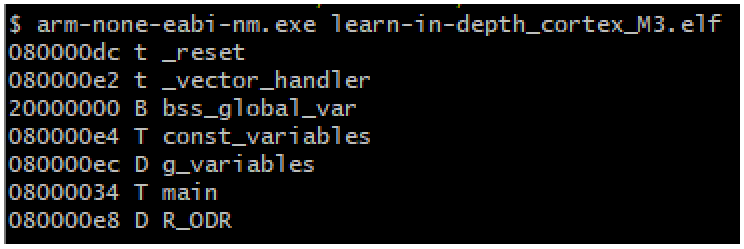
```
$ make
arm-none-eabi-gcc.exe  -c -mcpu=cortex-m3 -mthumb -gdwarf-2  main.c -o main.o
arm-none-eabi-as.exe -mcpu=cortex-m3 -mthumb -gdwarf-2  startup.s -o startup.o
startup.s: Assembler messages:
startup.s: Warning: end of file not at end of a line; newline inserted
arm-none-eabi-ld.exe -T linker-script.ld  -Map=Map_file.txt main.o startup.o -o learn-in-depth_cortex_M3.elf
arm-none-eabi-objcopy.exe -O binary learn-in-depth_cortex_M3.elf learn-in-depth_cortex_M3.bin
```

6-Excutable file sections:



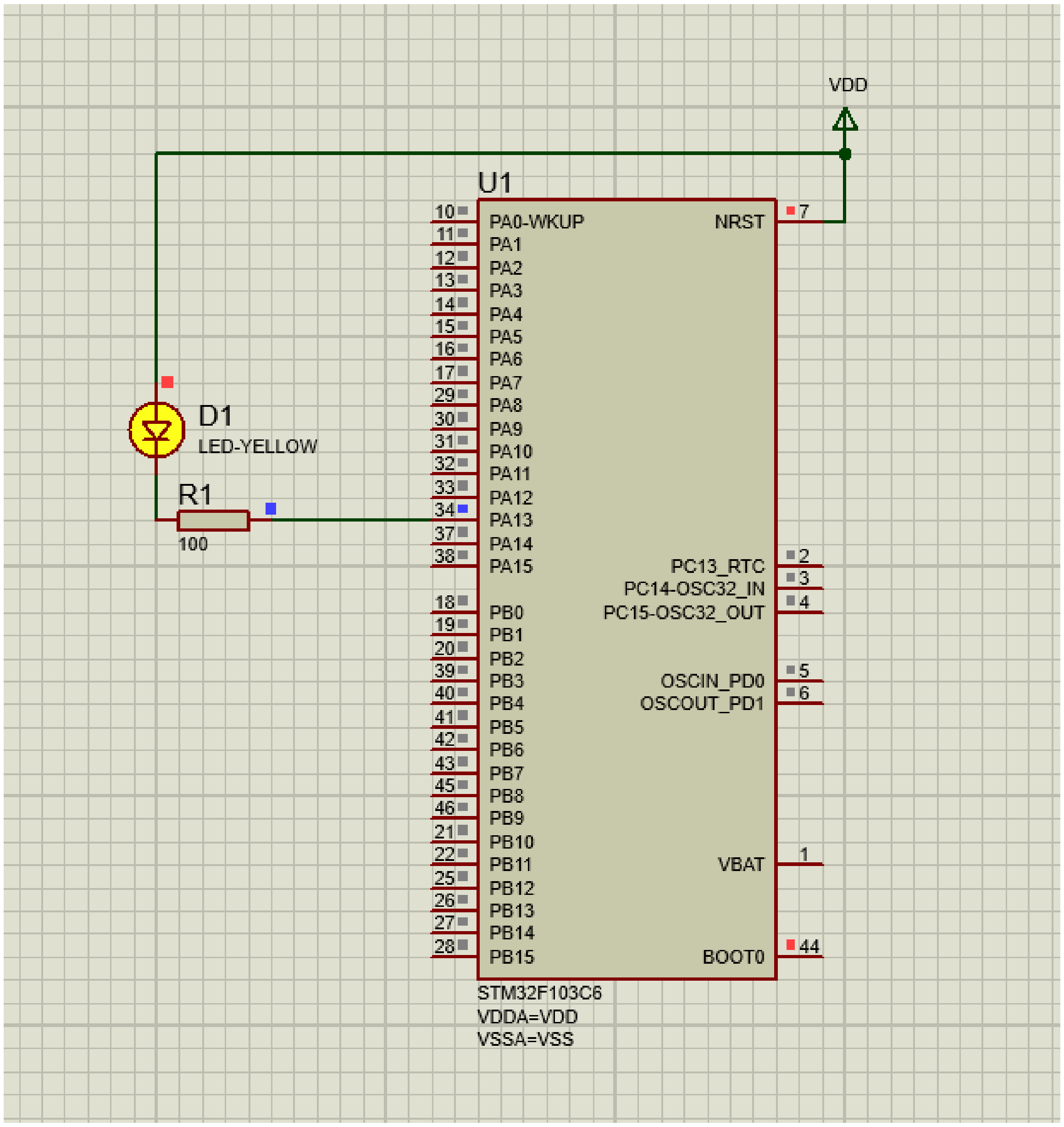
```
hassa@Hassan MINGW32 ~/Downloads/lab2
$ arm-none-eabi-objdump.exe -h learn-in-depth_cortex_M3.elf

learn-in-depth_cortex_M3.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000e8  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000008  080000e8  080000e8  000080e8  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  20000000  20000000  00010000  2**2
                  ALLOC
  3 .debug_info   0000018a  00000000  00000000  000080f0  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 000000ec  00000000  00000000  0000827a  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000038  00000000  00000000  00008366  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000040  00000000  00000000  000083a0  2**3
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000091  00000000  00000000  000083e0  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    000000bd  00000000  00000000  00008471  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  0000852e  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000031  00000000  00000000  0000853f  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000002c  00000000  00000000  00008570  2**2
                  CONTENTS, READONLY, DEBUGGING
```

7-Excutable file symbols:

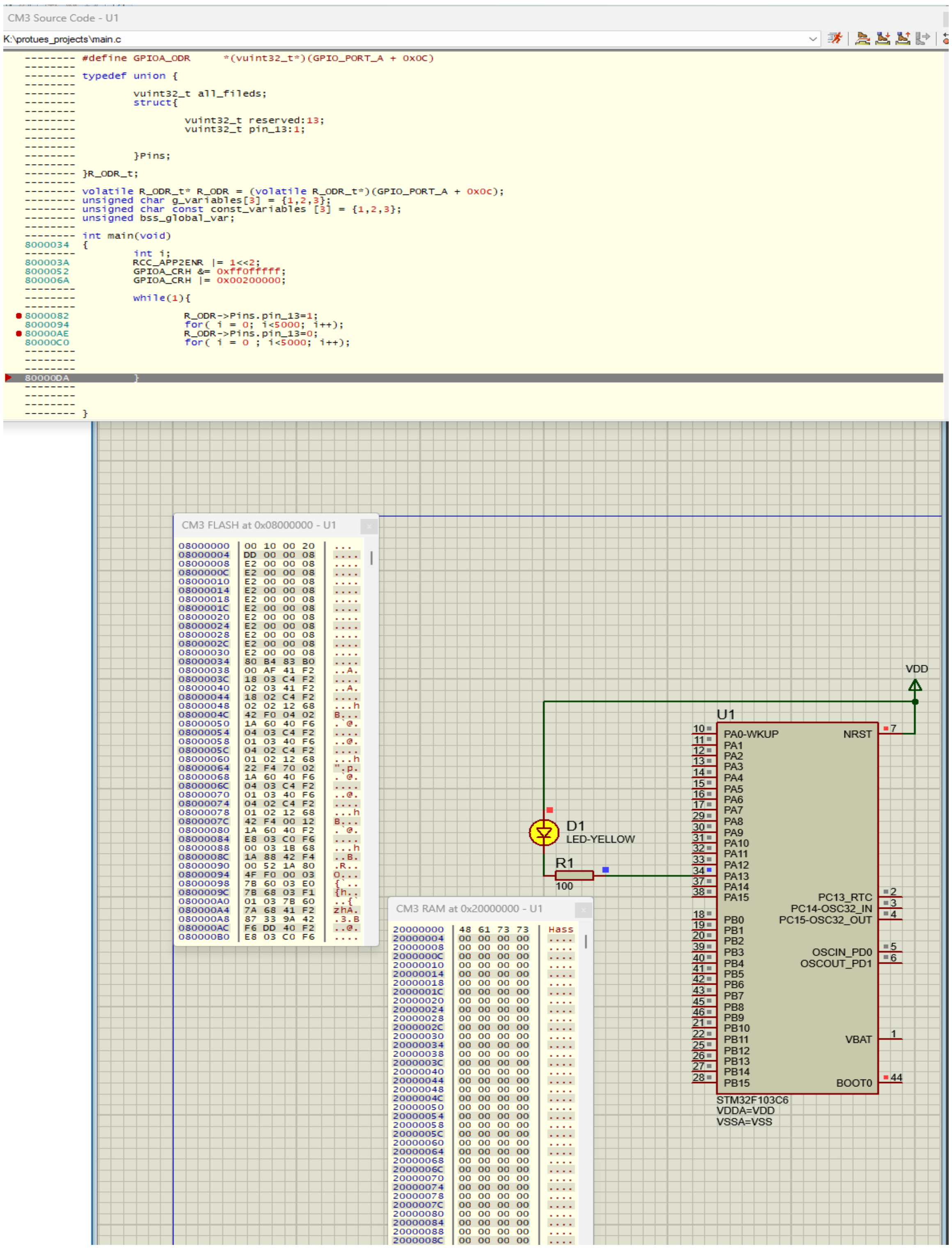


```
$ arm-none-eabi-nm.exe learn-in-depth_cortex_M3.elf
080000dc t _reset
080000e2 t _vector_handler
20000000 B bss_global_var
080000e4 T const_variables
080000ec D g_variables
08000034 T main
080000e8 D R_ODR
```

8-simulating the executable file on proteus:



9-Debugging:



CM3 Source Code - U1

K:\protues_projects\main.c

```
-------- #define GPIOA_ODR     *(vuint32_t*)(GPIO_PORT_A + 0x0C)
--------
-------- typedef union {
--------
--------         vuint32_t all_fileds;
--------         struct{
--------
--------                 vuint32_t reserved:13;
--------                 vuint32_t pin_13:1;
--------
--------
--------         }Pins;
--------
-------- }R_ODR_t;
--------
-------- volatile R_ODR_t* R_ODR = (volatile R_ODR_t*)(GPIO_PORT_A + 0x0c);
-------- unsigned char g_variables[3] = {1,2,3};
-------- unsigned char const const_variables [3] = {1,2,3};
-------- unsigned bss_global_var;
--------
-------- int main(void)
8000034  {
--------         int i;
800003A          RCC_APP2ENR  |= 1<<2;
8000052          GPIOA_CRH &= 0xff0fffff;
800006A          GPIOA_CRH  |= 0x00200000;
--------
--------         while(1){
--------
● 8000082                R_ODR->Pins.pin_13=1;
8000094                  for( i = 0; i<5000; i++);
● 80000AE                R_ODR->Pins.pin_13=0;
80000C0                  for( i = 0 ; i<5000; i++);
--------
--------
--------
▶ 80000DA        }
--------
--------
--------
-------- }
```

CM3 FLASH at 0x08000000 - U1

```
08000000 | 00 10 00 20 | ...
08000004 | DD 00 00 08 | ....
08000008 | E2 00 00 08 | ....
0800000C | E2 00 00 08 | ....
08000010 | E2 00 00 08 | ....
08000014 | E2 00 00 08 | ....
08000018 | E2 00 00 08 | ....
0800001C | E2 00 00 08 | ....
08000020 | E2 00 00 08 | ....
08000024 | E2 00 00 08 | ....
08000028 | E2 00 00 08 | ....
0800002C | E2 00 00 08 | ....
08000030 | E2 00 00 08 | ....
08000034 | 80 B4 83 B0 | ....
08000038 | 00 AF 41 F2 | ..A.
0800003C | 18 03 C4 F2 | ....
08000040 | 02 03 41 F2 | ..A.
08000044 | 18 02 C4 F2 | ....
08000048 | 02 02 12 68 | ...h
0800004C | 42 F0 04 02 | B...
08000050 | 1A 60 40 F6 | .`@.
08000054 | 04 03 C4 F2 | ....
08000058 | 01 03 40 F6 | ..@.
0800005C | 04 02 C4 F2 | ....
08000060 | 01 02 12 68 | ...h
08000064 | 22 F4 70 02 | ".p.
08000068 | 1A 60 40 F6 | .`@.
0800006C | 04 03 C4 F2 | ....
08000070 | 01 03 40 F6 | ..@.
08000074 | 04 02 C4 F2 | ....
08000078 | 01 02 12 68 | ...h
0800007C | 42 F4 00 12 | B...
08000080 | 1A 60 40 F2 | .`@.
08000084 | E8 03 C0 F6 | ....
08000088 | 00 03 1B 68 | ...h
0800008C | 1A 88 42 F4 | ..B.
08000090 | 00 52 1A 80 | .R..
08000094 | 4F F0 00 03 | O...
08000098 | 7B 60 03 E0 | {`..
0800009C | 7B 68 03 F1 | {h..
080000A0 | 01 03 7B 60 | ..{`
080000A4 | 7A 68 41 F2 | zhA.
080000A8 | 87 33 9A 42 | .3.B
080000AC | F6 DD 40 F2 | ..@.
080000B0 | E8 03 C0 F6 | ....
```

CM3 RAM at 0x20000000 - U1

```
20000000 | 48 61 73 73 | Hass
20000004 | 00 00 00 00 | ....
20000008 | 00 00 00 00 | ....
2000000C | 00 00 00 00 | ....
20000010 | 00 00 00 00 | ....
20000014 | 00 00 00 00 | ....
20000018 | 00 00 00 00 | ....
2000001C | 00 00 00 00 | ....
20000020 | 00 00 00 00 | ....
20000024 | 00 00 00 00 | ....
20000028 | 00 00 00 00 | ....
2000002C | 00 00 00 00 | ....
20000030 | 00 00 00 00 | ....
20000034 | 00 00 00 00 | ....
20000038 | 00 00 00 00 | ....
2000003C | 00 00 00 00 | ....
20000040 | 00 00 00 00 | ....
20000044 | 00 00 00 00 | ....
20000048 | 00 00 00 00 | ....
2000004C | 00 00 00 00 | ....
20000050 | 00 00 00 00 | ....
20000054 | 00 00 00 00 | ....
20000058 | 00 00 00 00 | ....
2000005C | 00 00 00 00 | ....
20000060 | 00 00 00 00 | ....
20000064 | 00 00 00 00 | ....
20000068 | 00 00 00 00 | ....
2000006C | 00 00 00 00 | ....
20000070 | 00 00 00 00 | ....
20000074 | 00 00 00 00 | ....
20000078 | 00 00 00 00 | ....
2000007C | 00 00 00 00 | ....
20000080 | 00 00 00 00 | ....
20000084 | 00 00 00 00 | ....
20000088 | 00 00 00 00 | ....
2000008C | 00 00 00 00 | ....
```