Learn-In-Depth

Be Professional in Embedded System

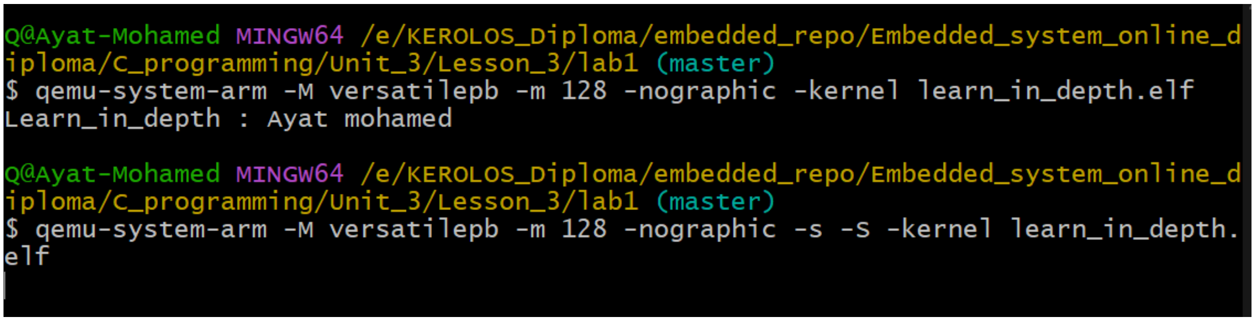
Assignment-3

Lesson-3

Name : Ayat Mohamed

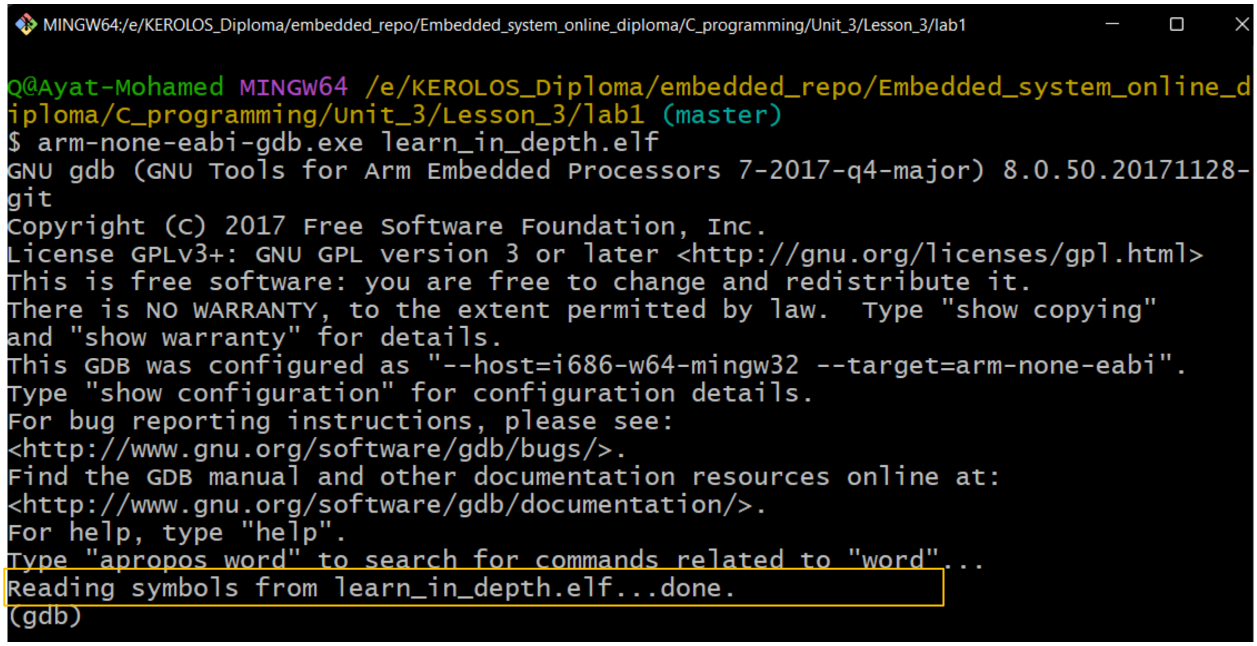
LAB_1:

- GDB
  - first ,we will run a versatilepb board, and then use -s -S for debug the board



```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_3/lab1 (master)
$ qemu-system-arm -M versatilepb -m 128 -nographic -kernel learn_in_depth.elf
Learn_in_depth : Ayat mohamed

Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_3/lab1 (master)
$ qemu-system-arm -M versatilepb -m 128 -nographic -s -S -kernel learn_in_depth.
elf
```

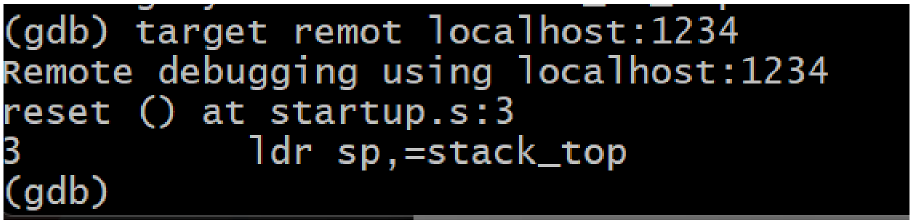
  - then open an another terminal to run the gdb server
  - use file.elf no file.bin ,because the file.elf contains the debug info.


MINGW64:/e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_diploma/C_programming/Unit_3/Lesson_3/lab1

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_3/lab1 (master)
$ arm-none-eabi-gdb.exe learn_in_depth.elf
GNU gdb (GNU Tools for Arm Embedded Processors 7-2017-q4-major) 8.0.50.20171128-
git
Copyright (C) 2017 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-w64-mingw32 --target=arm-none-eabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from learn_in_depth.elf...done.
(gdb)
```

-to connect to gdb server we should have the IP address and port #

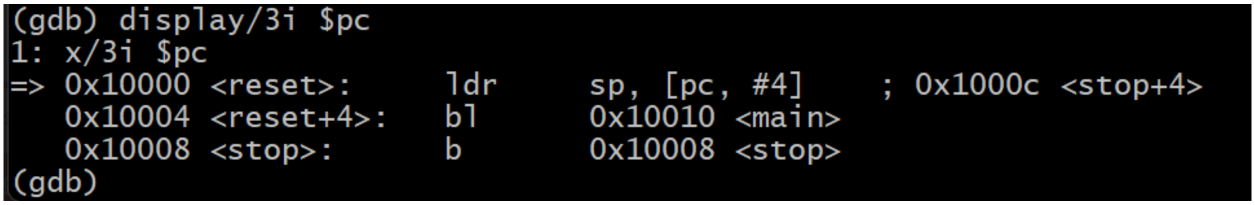
-in this case the server will be our localhost and port # is 1234



```
(gdb) target remot localhost:1234
Remote debugging using localhost:1234
reset () at startup.s:3
3           ldr sp,=stack_top
(gdb)
```

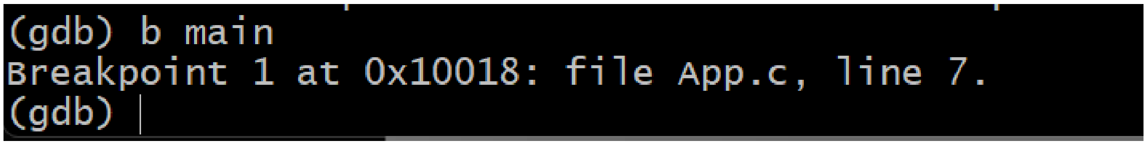
-to display the current pc

-this command displays no of instructions and the location of current pc ,it located at rest symbol.



```
(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)
```

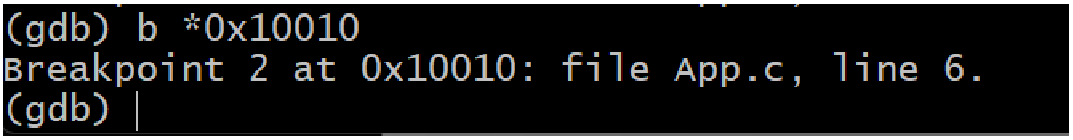
-to but a break point at main



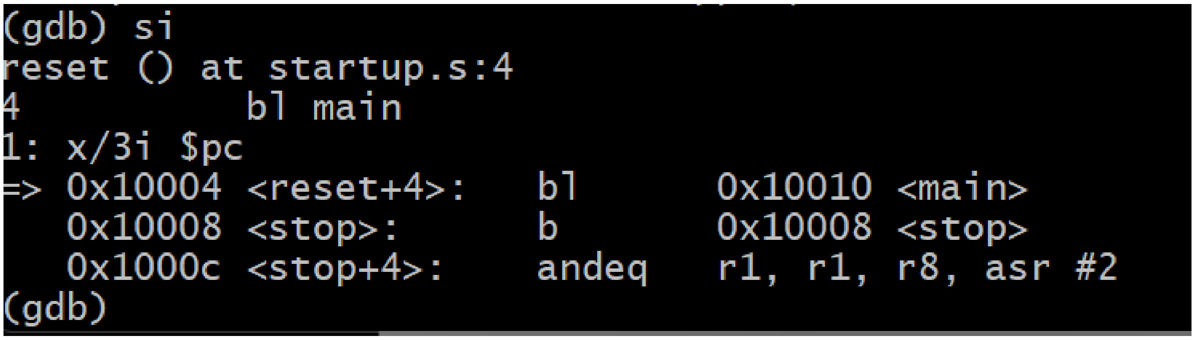
```
(gdb) b main
Breakpoint 1 at 0x10018: file App.c, line 7.
(gdb) |
```

-if we want to add a break point at a certain address

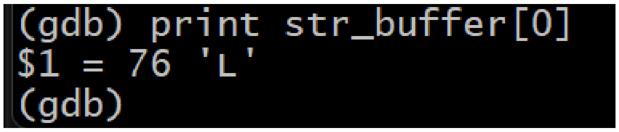
-this address is related with context switch before main



```
(gdb) b *0x10010
Breakpoint 2 at 0x10010: file App.c, line 6.
(gdb) |
```

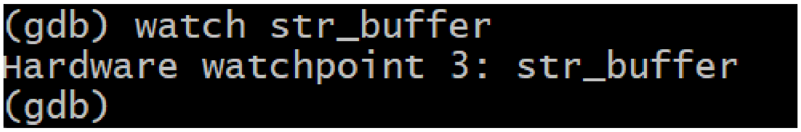


```
(gdb) si
reset () at startup.s:4
4           bl main
1: x/3i $pc
=> 0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      b       0x10008 <stop>
   0x1000c <stop+4>:    andeq   r1, r1, r8, asr #2
(gdb)
```



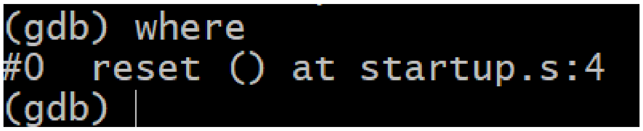
```
(gdb) print str_buffer[0]
$1 = 76 'L'
(gdb)
```

-if you want to watch the variable ,use watch “var_name”

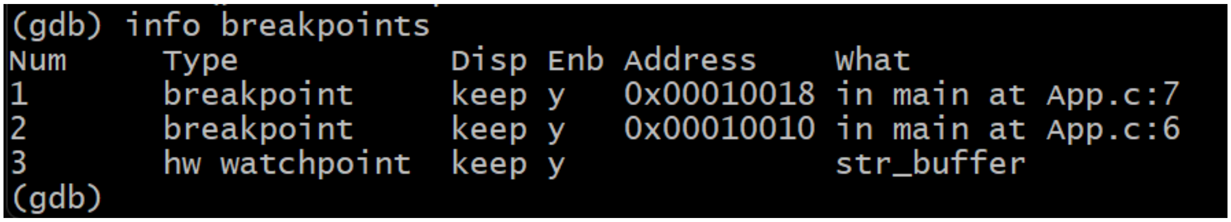


```
(gdb) watch str_buffer
Hardware watchpoint 3: str_buffer
(gdb)
```

-if we want to know our location ,use “where”

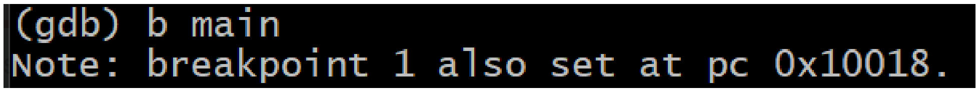


```
(gdb) where
#0  reset () at startup.s:4
(gdb)
```



```
(gdb) info breakpoints
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x00010018 in main at App.c:7
2       breakpoint     keep y   0x00010010 in main at App.c:6
3       hw watchpoint  keep y              str_buffer
(gdb)
```

-if you want to delete a specific breakpoint, use “delete bp_name”.

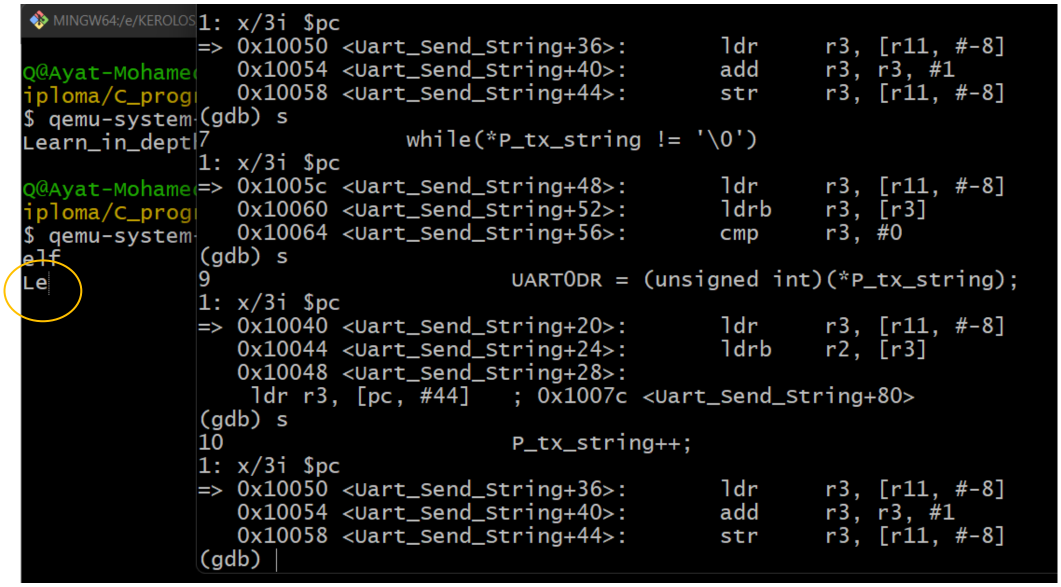


```
(gdb) b main
Note: breakpoint 1 also set at pc 0x10018.
```

-if we want to continue to our break point use

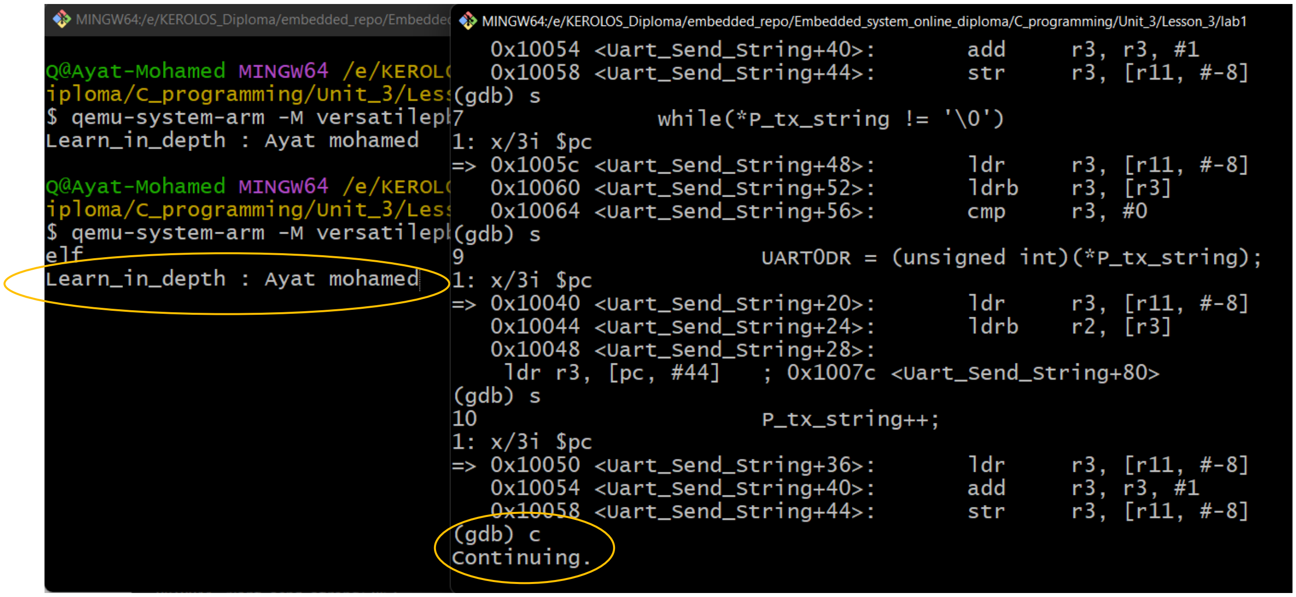
“c”->continue

-and we use “s” to step until UART.c finish ,after this we will find the string is transmitted and printed in terminal



```
MINGW64:/e/KEROLOS

Q@Ayat-Mohamed
iploma/C_progr
$ qemu-system-
Learn_in_depth

Q@Ayat-Mohamed
iploma/C_progr
$ qemu-system-
elf
Le
```

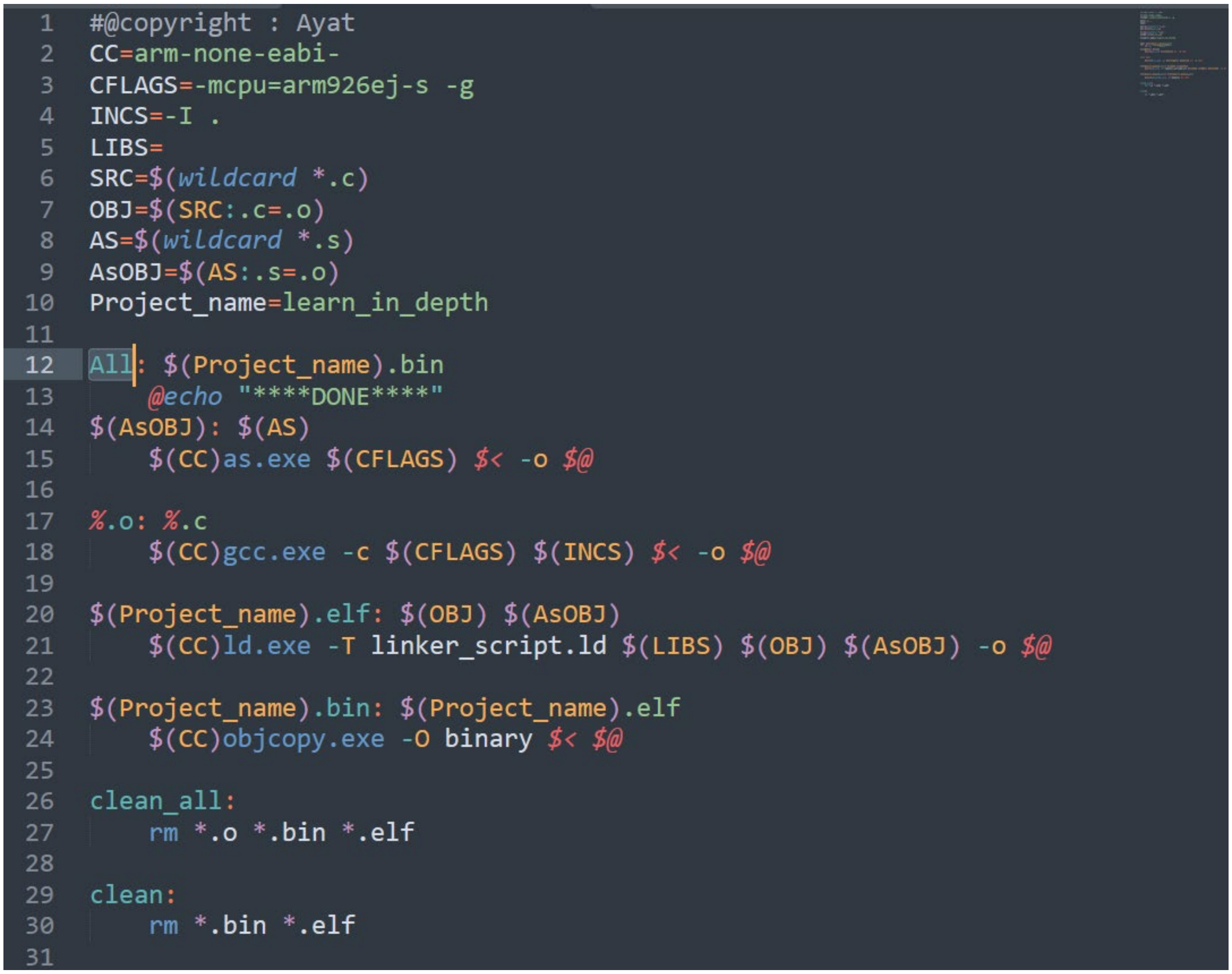
```
1: x/3i $pc
=> 0x10050 <Uart_Send_String+36>:       ldr     r3, [r11, #-8]
   0x10054 <Uart_Send_String+40>:       add     r3, r3, #1
   0x10058 <Uart_Send_String+44>:       str     r3, [r11, #-8]
(gdb) s
7               while(*P_tx_string != '\0')
1: x/3i $pc
=> 0x1005c <Uart_Send_String+48>:       ldr     r3, [r11, #-8]
   0x10060 <Uart_Send_String+52>:       ldrb    r3, [r3]
   0x10064 <Uart_Send_String+56>:       cmp     r3, #0
(gdb) s
9                       UART0DR = (unsigned int)(*P_tx_string);
1: x/3i $pc
=> 0x10040 <Uart_Send_String+20>:       ldr     r3, [r11, #-8]
   0x10044 <Uart_Send_String+24>:       ldrb    r2, [r3]
   0x10048 <Uart_Send_String+28>:
    ldr r3, [pc, #44]   ; 0x1007c <Uart_Send_String+80>
(gdb) s
10                      P_tx_string++;
1: x/3i $pc
=> 0x10050 <Uart_Send_String+36>:       ldr     r3, [r11, #-8]
   0x10054 <Uart_Send_String+40>:       add     r3, r3, #1
   0x10058 <Uart_Send_String+44>:       str     r3, [r11, #-8]
(gdb)
```



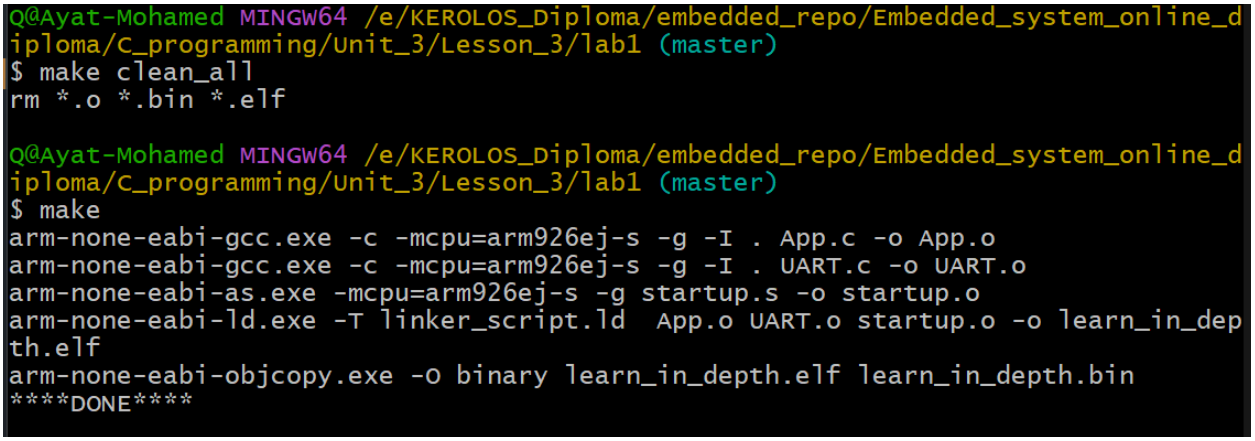
```
MINGW64:/e/KEROLOS_Diploma/embedded_repo/Embedded

Q@Ayat-Mohamed MINGW64 /e/KEROL
iploma/C_programming/Unit_3/Less
$ qemu-system-arm -M versatilepb
Learn_in_depth : Ayat mohamed

Q@Ayat-Mohamed MINGW64 /e/KEROL
iploma/C_programming/Unit_3/Less
$ qemu-system-arm -M versatilepb
elf
Learn_in_depth : Ayat mohamed
```

```
MINGW64:/e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_diploma/C_programming/Unit_3/Lesson_3/lab1
   0x10054 <Uart_Send_String+40>:       add     r3, r3, #1
   0x10058 <Uart_Send_String+44>:       str     r3, [r11, #-8]
(gdb) s
7               while(*P_tx_string != '\0')
1: x/3i $pc
=> 0x1005c <Uart_Send_String+48>:       ldr     r3, [r11, #-8]
   0x10060 <Uart_Send_String+52>:       ldrb    r3, [r3]
   0x10064 <Uart_Send_String+56>:       cmp     r3, #0
(gdb) s
9                       UART0DR = (unsigned int)(*P_tx_string);
1: x/3i $pc
=> 0x10040 <Uart_Send_String+20>:       ldr     r3, [r11, #-8]
   0x10044 <Uart_Send_String+24>:       ldrb    r2, [r3]
   0x10048 <Uart_Send_String+28>:
    ldr r3, [pc, #44]   ; 0x1007c <Uart_Send_String+80>
(gdb) s
10                      P_tx_string++;
1: x/3i $pc
=> 0x10050 <Uart_Send_String+36>:       ldr     r3, [r11, #-8]
   0x10054 <Uart_Send_String+40>:       add     r3, r3, #1
   0x10058 <Uart_Send_String+44>:       str     r3, [r11, #-8]
(gdb) c
Continuing.
```

- *MakeFile*



```
#@copyright : Ayat
CC=arm-none-eabi-
CFLAGS=-mcpu=arm926ej-s -g
INCS=-I .
LIBS=
SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)
AS=$(wildcard *.s)
AsOBJ=$(AS:.s=.o)
Project_name=learn_in_depth

All: $(Project_name).bin
	@echo "****DONE****"
$(AsOBJ): $(AS)
	$(CC)as.exe $(CFLAGS) $< -o $@

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

$(Project_name).elf: $(OBJ) $(AsOBJ)
	$(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $@

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

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

clean:
	rm *.bin *.elf

```



```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_3/lab1 (master)
$ make clean_all
rm *.o *.bin *.elf

Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_3/lab1 (master)
$ make
arm-none-eabi-gcc.exe -c -mcpu=arm926ej-s -g -I . App.c -o App.o
arm-none-eabi-gcc.exe -c -mcpu=arm926ej-s -g -I . UART.c -o UART.o
arm-none-eabi-as.exe -mcpu=arm926ej-s -g startup.s -o startup.o
arm-none-eabi-ld.exe -T linker_script.ld  App.o UART.o startup.o -o learn_in_dep
th.elf
arm-none-eabi-objcopy.exe -O binary learn_in_depth.elf learn_in_depth.bin
****DONE****
```

LAB_2 :part(1)with startup.s

-Board name : STM32f103c8t6(Cortex-M3)

- the entry point to this board is 0x08000000

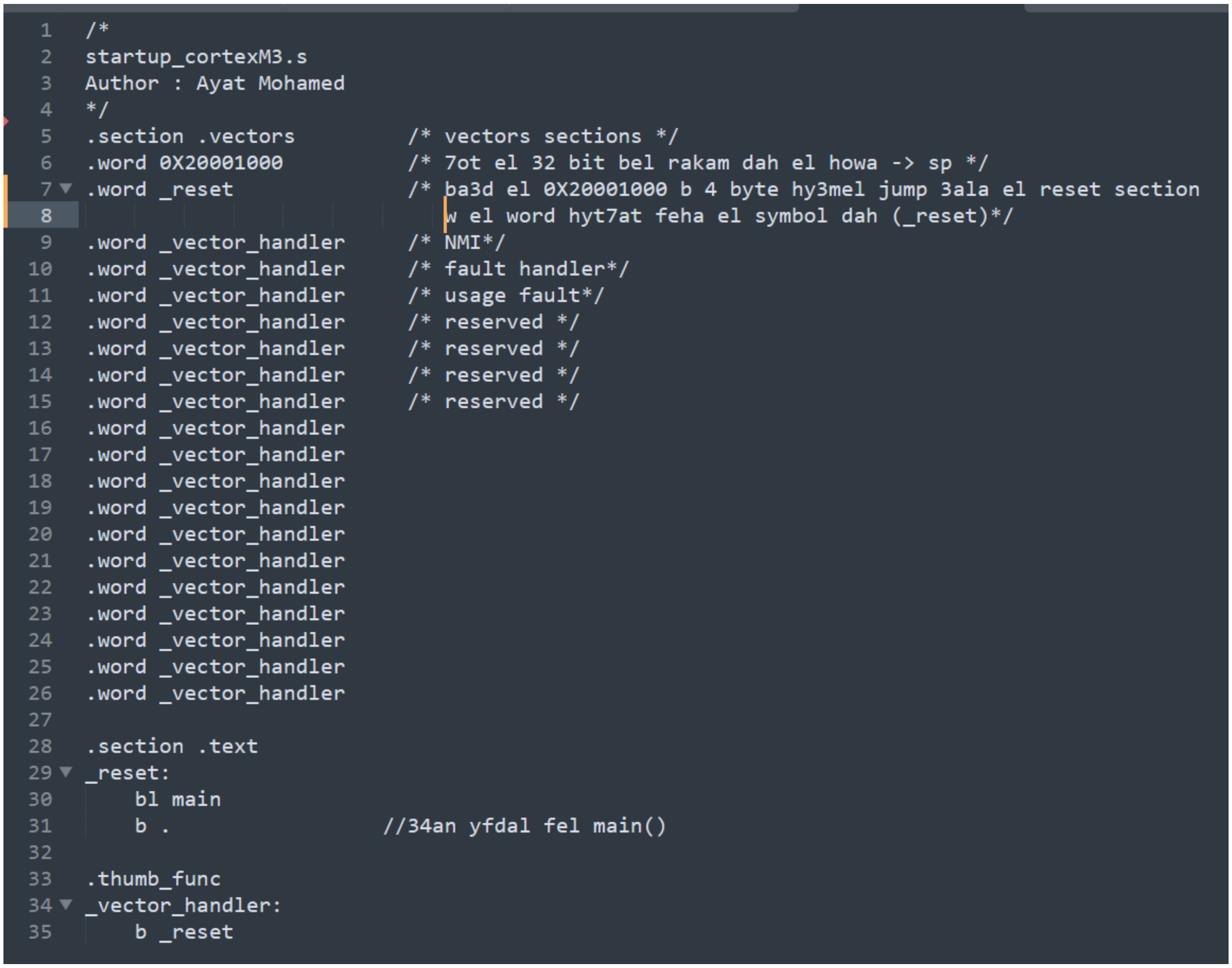
-this address contains the SP value if address that points to stack in (SRAM)

-in the begin we should define all vector Handlers in vector table in section called ".vector"

-in the first we will defined first word as a value of SP is 0x20001000 within the range of SRAM.

-According to the specs , the vector table must start after SP, by defining vector handlers after SP word.

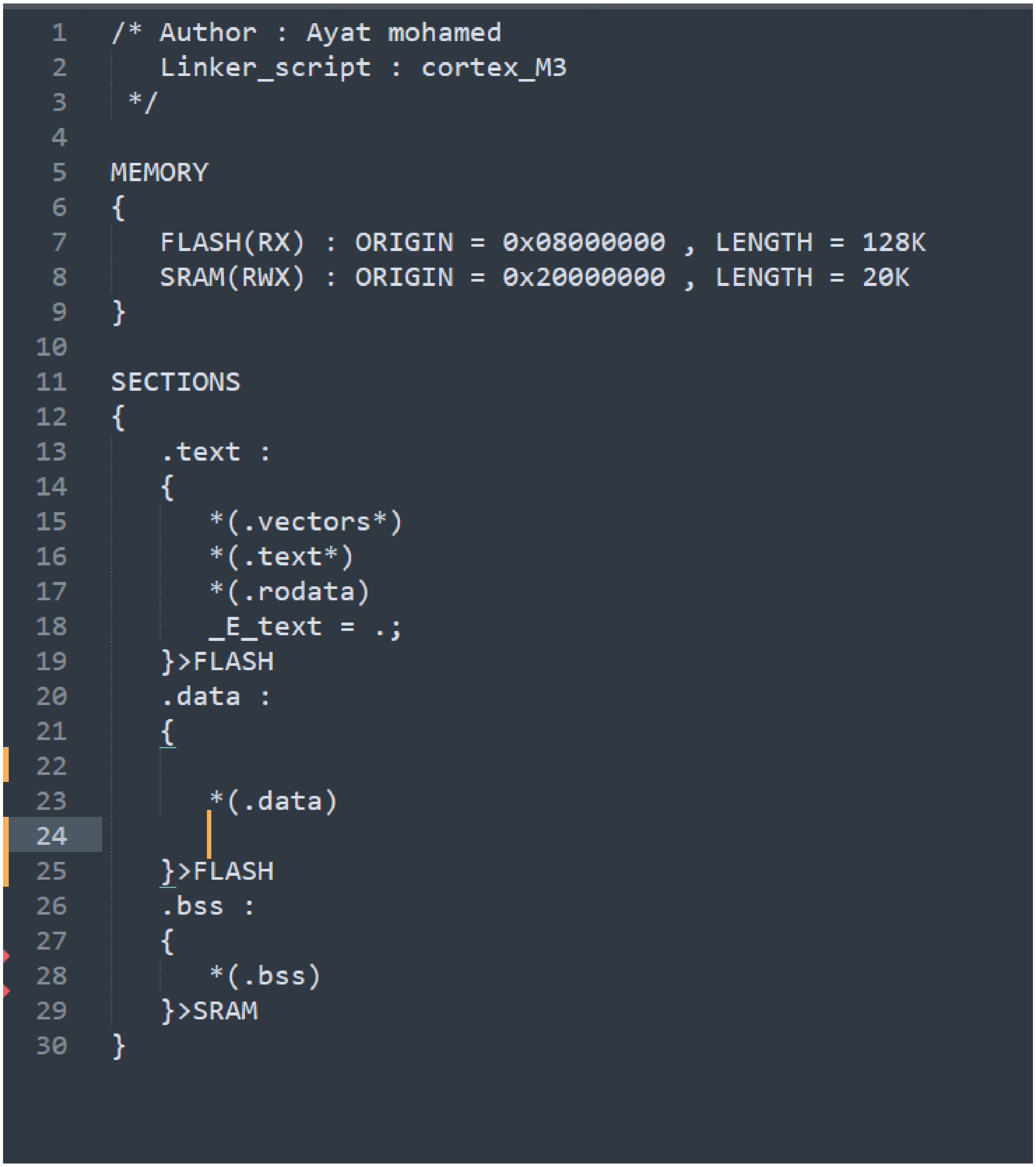
-Startup.s



```
/*
startup_cortexM3.s
Author : Ayat Mohamed
*/
.section .vectors          /* vectors sections */
.word 0X20001000           /* 7ot el 32 bit bel rakam dah el howa -> sp */
.word _reset               /* ba3d el 0X20001000 b 4 byte hy3mel jump 3ala el reset section
                              w el word hyt7at feha el symbol dah (_reset)*/
.word _vector_handler      /* NMI*/
.word _vector_handler      /* fault handler*/
.word _vector_handler      /* usage fault*/
.word _vector_handler      /* reserved */
.word _vector_handler      /* reserved */
.word _vector_handler      /* reserved */
.word _vector_handler      /* reserved */
.word _vector_handler
.word _vector_handler
.word _vector_handler
.word _vector_handler
.word _vector_handler
.word _vector_handler
.word _vector_handler
.word _vector_handler
.word _vector_handler
.word _vector_handler
.word _vector_handler
.word _vector_handler

.section .text
_reset:
    bl main
    b .                    //34an yfdal fel main()

.thumb_func
_vector_handler:
    b _reset
```

-linker_script.ld

-According to specs the FLASH memory start with 0x08000000 and the SRAM ox02000000.



```
/* Author : Ayat mohamed
   Linker_script : cortex_M3
 */

MEMORY
{
   FLASH(RX) : ORIGIN = 0x08000000 , LENGTH = 128K
   SRAM(RWX) : ORIGIN = 0x20000000 , LENGTH = 20K
}

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

      *(.data)

   }>FLASH
   .bss :
   {
      *(.bss)
   }>SRAM
}
```

LAB_2:part(2)with startup·c

-As we said before, when the entry point is the address of SP, we can write the startup·c

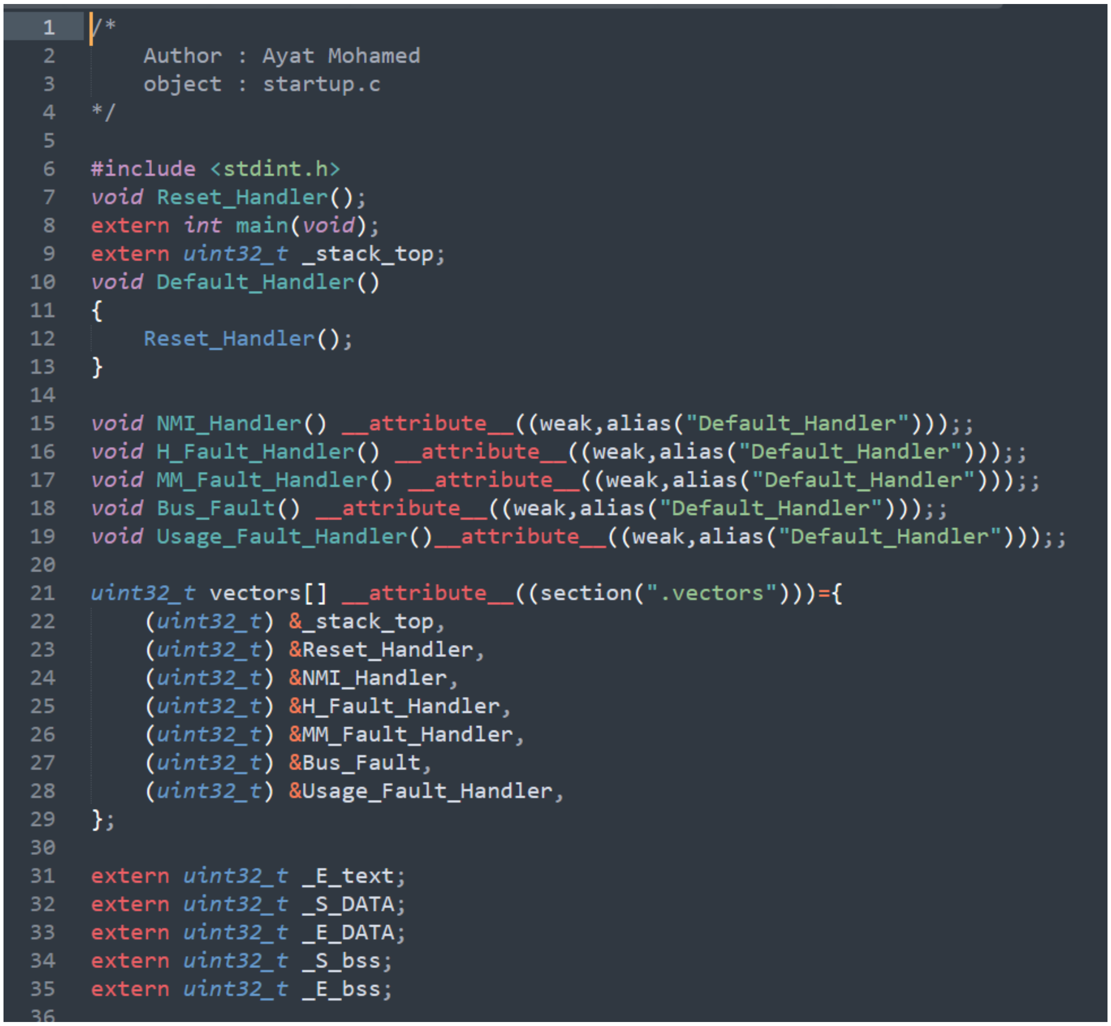
- Define Interrupt vectors Section 2

-Copy Data from ROM to RAM

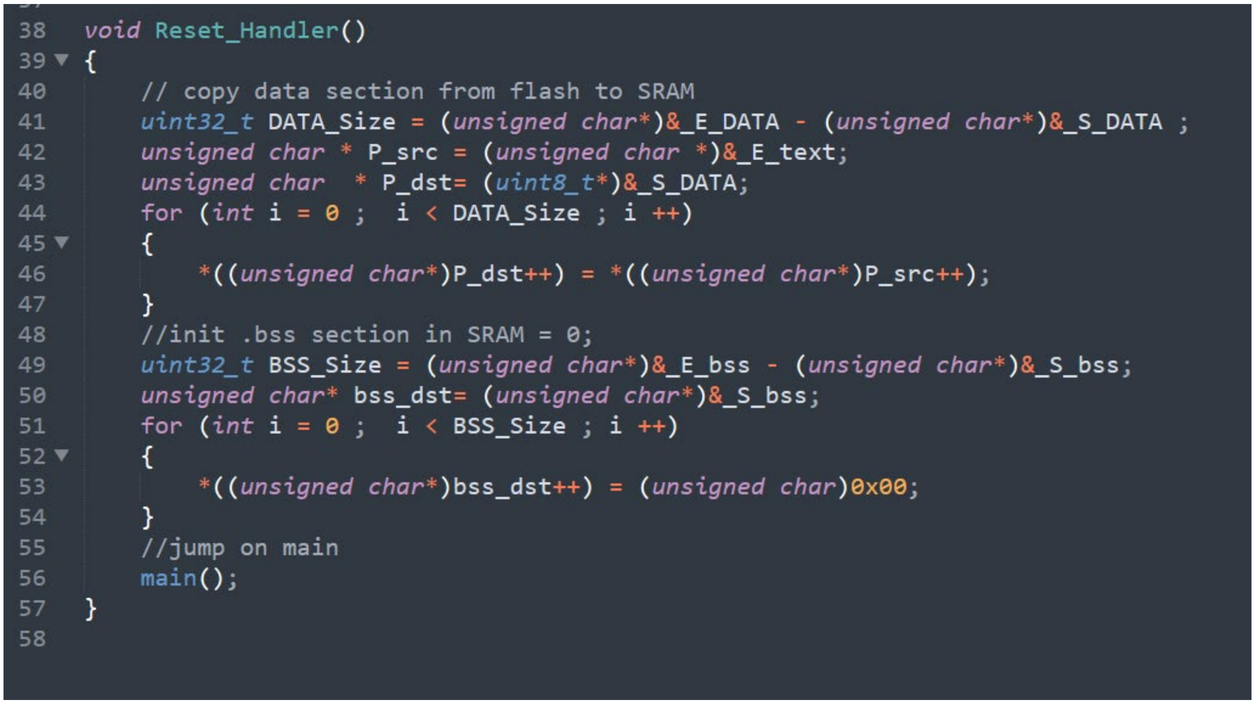
-Initialize Data Area and Initialize Stack

-Create a reset section and Call main( )

-Startup·c

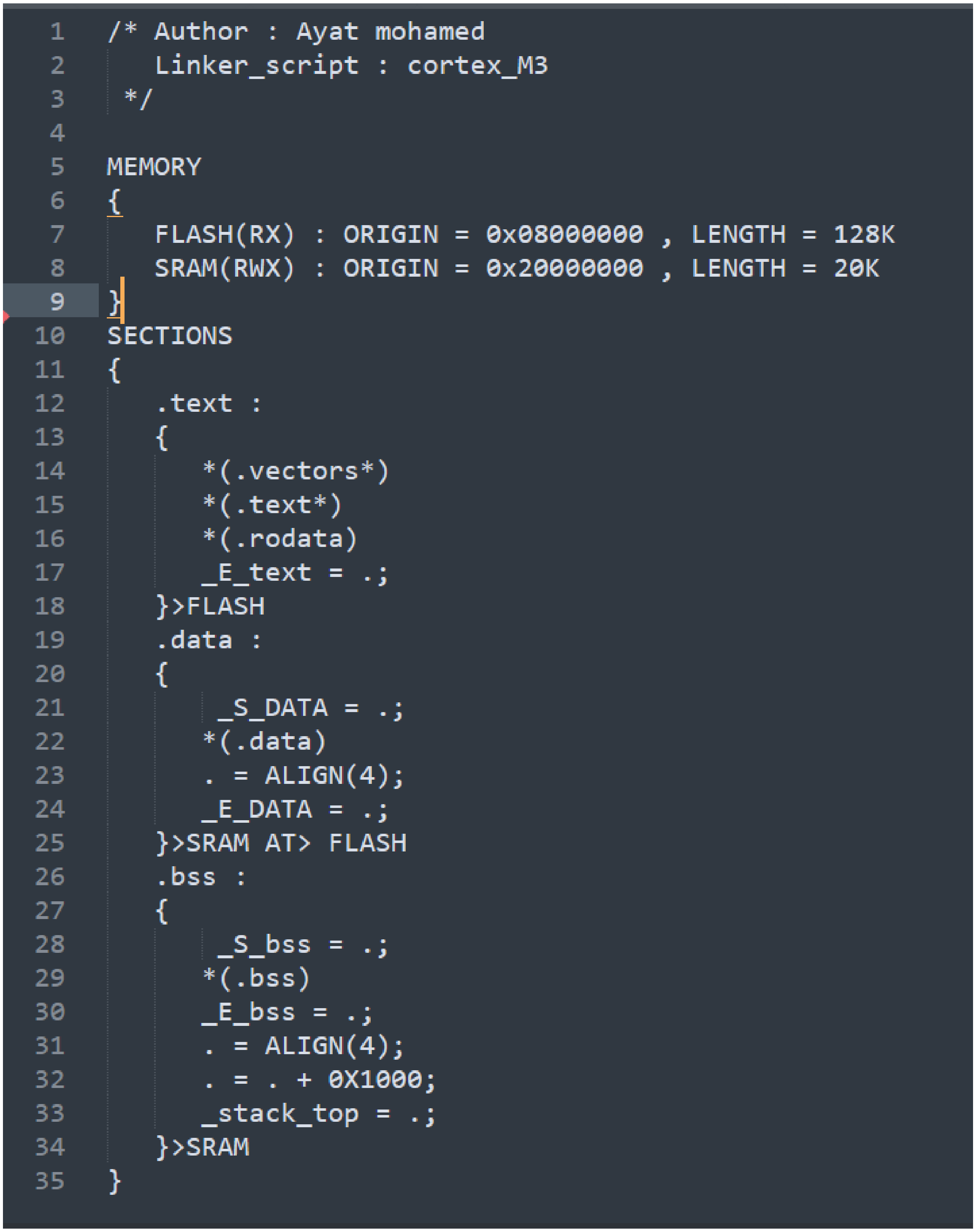


```c
/*
    Author : Ayat Mohamed
    object : startup.c
*/

#include <stdint.h>
void Reset_Handler();
extern int main(void);
extern uint32_t _stack_top;
void Default_Handler()
{
    Reset_Handler();
}

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")));;

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,
};

extern uint32_t _E_text;
extern uint32_t _S_DATA;
extern uint32_t _E_DATA;
extern uint32_t _S_bss;
extern uint32_t _E_bss;
```



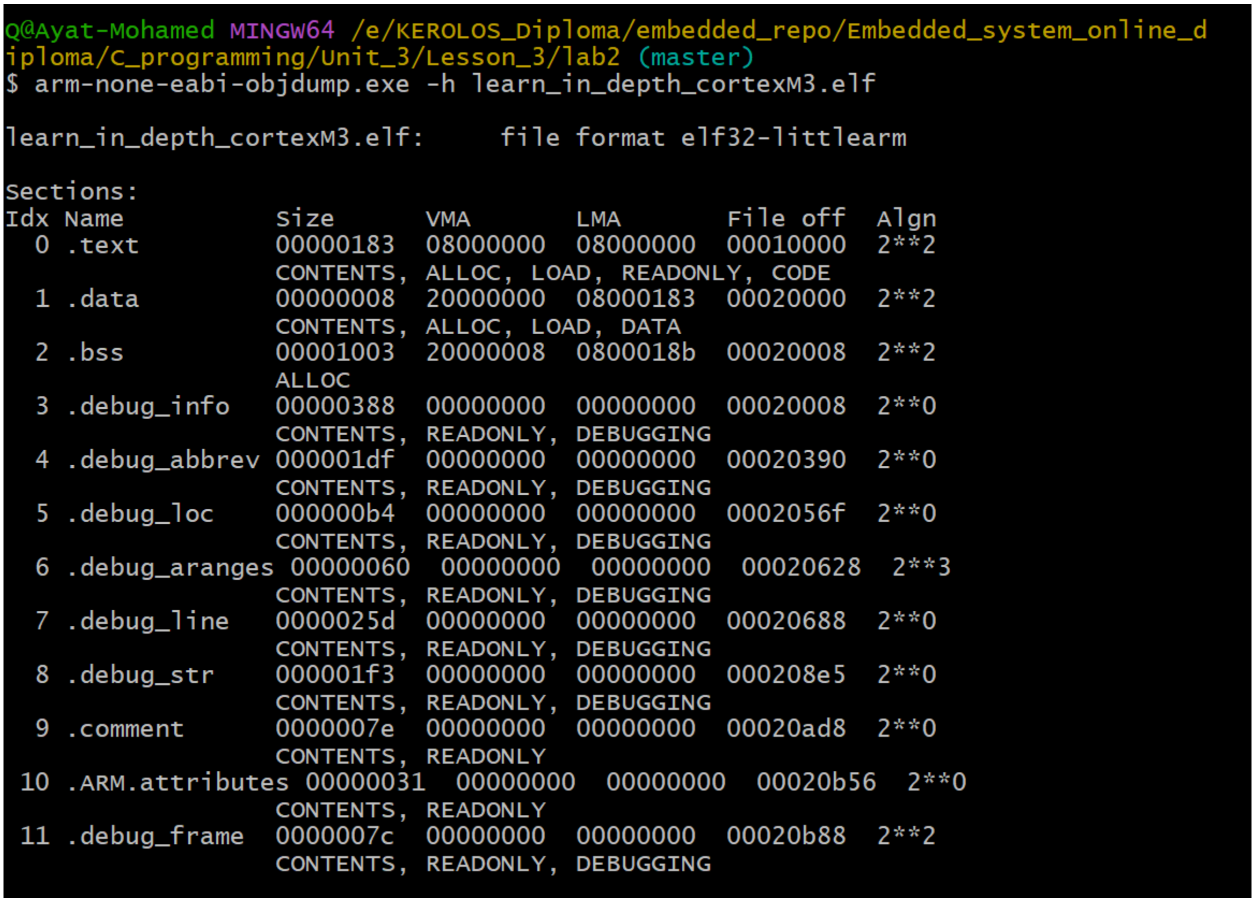
```
void Reset_Handler()
{
    // copy data section from flash to SRAM
    uint32_t DATA_Size = (unsigned char*)&_E_DATA - (unsigned char*)&_S_DATA ;
    unsigned char * P_src = (unsigned char *)&_E_text;
    unsigned char  * P_dst= (uint8_t*)&_S_DATA;
    for (int i = 0 ;  i < DATA_Size ; i ++)
    {
        *((unsigned char*)P_dst++) = *((unsigned char*)P_src++);
    }
    //init .bss section in SRAM = 0;
    uint32_t BSS_Size = (unsigned char*)&_E_bss - (unsigned char*)&_S_bss;
    unsigned char* bss_dst= (unsigned char*)&_S_bss;
    for (int i = 0 ;  i < BSS_Size ; i ++)
    {
        *((unsigned char*)bss_dst++) = (unsigned char)0x00;
    }
    //jump on main
    main();
}
```

-in linker we are defined some variables to make the memory boundaries at start and end od each section of the memory to know the size .



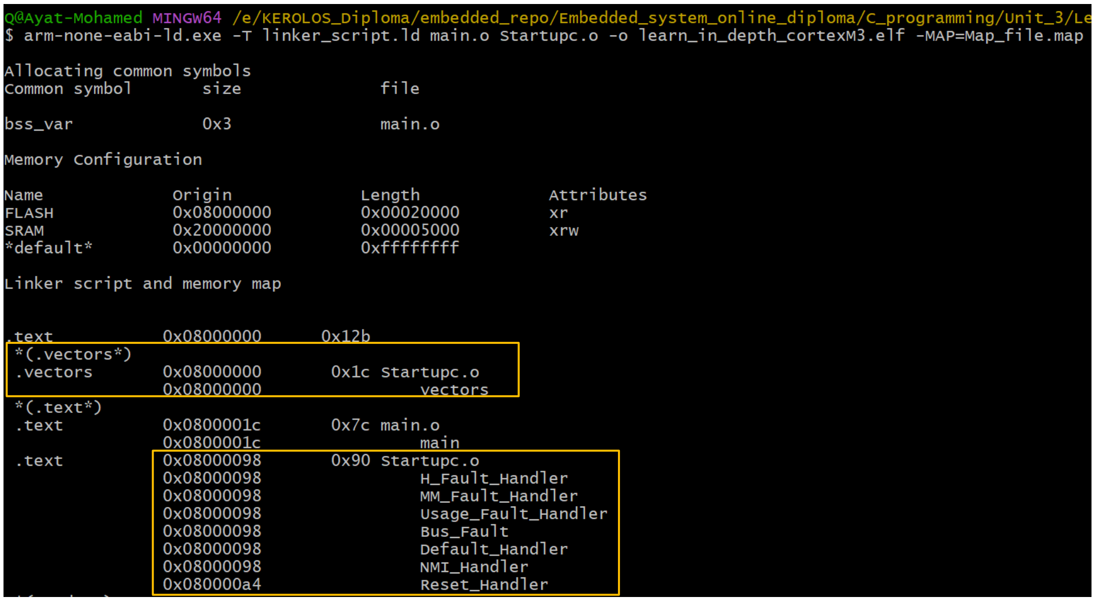
```
/* Author : Ayat mohamed
   Linker_script : cortex_M3
 */

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
}
```

-the sections of memories

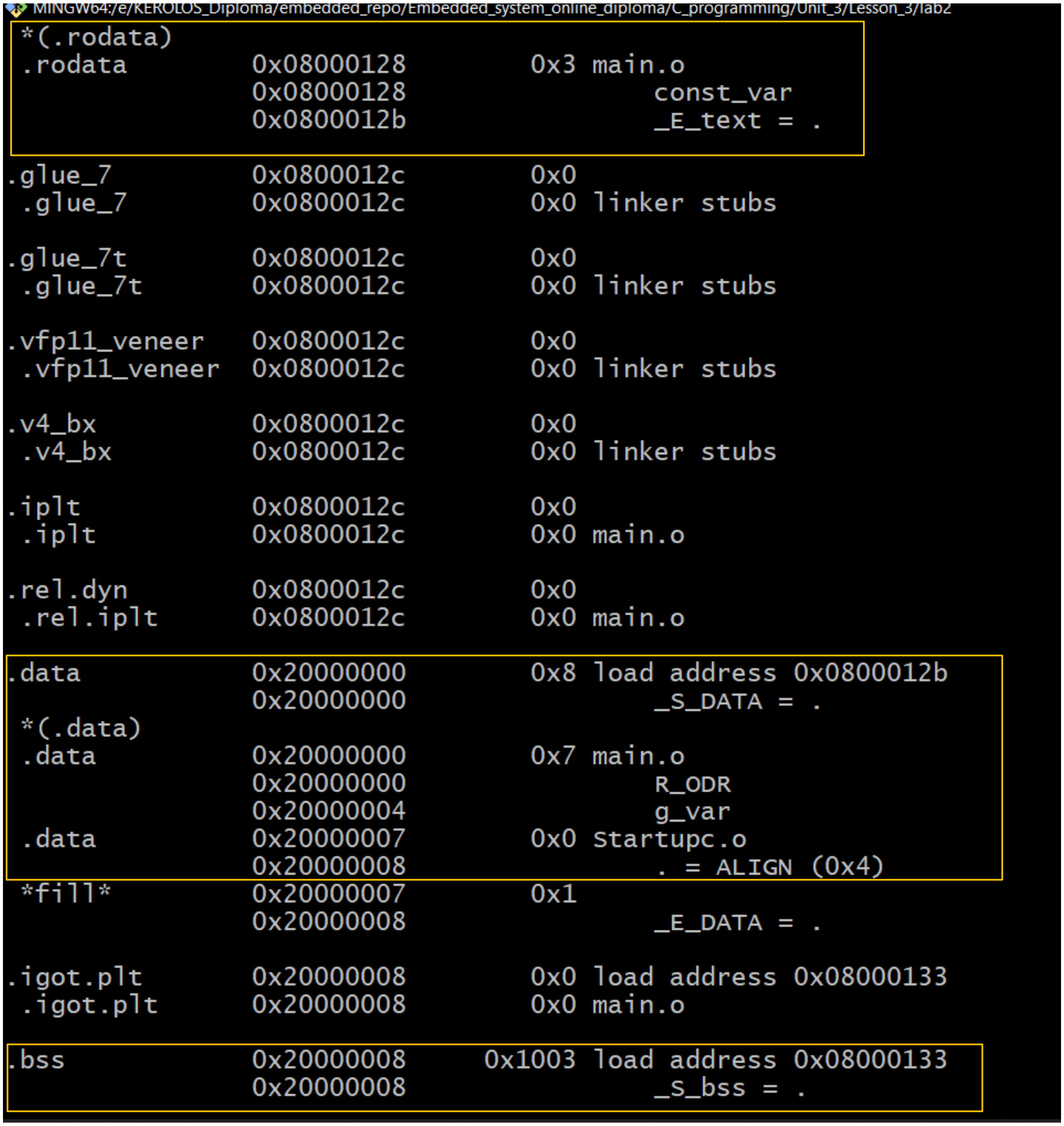


```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_3/lab2 (master)
$ arm-none-eabi-objdump.exe -h learn_in_depth_cortexM3.elf

learn_in_depth_cortexM3.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000183  08000000  08000000  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000008  20000000  08000183  00020000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001003  20000008  0800018b  00020008  2**2
                  ALLOC
  3 .debug_info   00000388  00000000  00000000  00020008  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 000001df  00000000  00000000  00020390  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000000b4  00000000  00000000  0002056f  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000060  00000000  00000000  00020628  2**3
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   0000025d  00000000  00000000  00020688  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    000001f3  00000000  00000000  000208e5  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007e  00000000  00000000  00020ad8  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000031  00000000  00000000  00020b56  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000007c  00000000  00000000  00020b88  2**2
                  CONTENTS, READONLY, DEBUGGING
```

-Map File

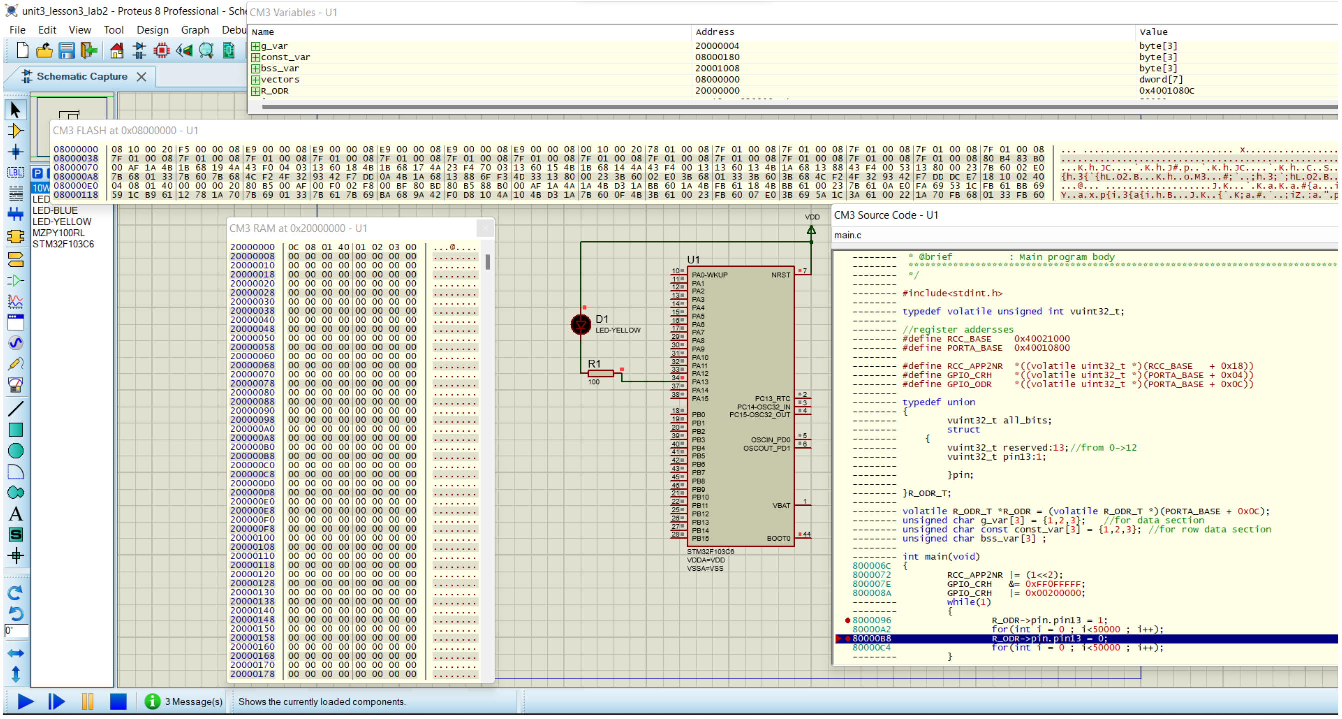


```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_diploma/C_programming/Unit_3/Le
$ arm-none-eabi-ld.exe -T linker_script.ld main.o Startupc.o -o learn_in_depth_cortexM3.elf -MAP=Map_file.map

Allocating common symbols
Common symbol       size              file

bss_var             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        0x12b
 *(.vectors*)
 .vectors       0x08000000         0x1c Startupc.o
                0x08000000                  vectors
 *(.text*)
 .text          0x0800001c         0x7c main.o
                0x0800001c                  main
 .text          0x08000098         0x90 Startupc.o
                0x08000098                  H_Fault_Handler
                0x08000098                  MM_Fault_Handler
                0x08000098                  Usage_Fault_Handler
                0x08000098                  Bus_Fault
                0x08000098                  Default_Handler
                0x08000098                  NMI_Handler
                0x080000a4                  Reset_Handler
```



```
MINGW64:/e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_diploma/C_programming/Unit_3/Lesson_3/lab2
 *(.rodata)
 .rodata        0x08000128       0x3 main.o
                0x08000128               const_var
                0x0800012b               _E_text = .

.glue_7         0x0800012c       0x0
 .glue_7        0x0800012c       0x0 linker stubs

.glue_7t        0x0800012c       0x0
 .glue_7t       0x0800012c       0x0 linker stubs

.vfp11_veneer   0x0800012c       0x0
 .vfp11_veneer  0x0800012c       0x0 linker stubs

.v4_bx          0x0800012c       0x0
 .v4_bx         0x0800012c       0x0 linker stubs

.iplt           0x0800012c       0x0
 .iplt          0x0800012c       0x0 main.o

.rel.dyn        0x0800012c       0x0
 .rel.iplt      0x0800012c       0x0 main.o

.data           0x20000000       0x8 load address 0x0800012b
                0x20000000               _S_DATA = .
 *(.data)
 .data          0x20000000       0x7 main.o
                0x20000000               R_ODR
                0x20000004               g_var
 .data          0x20000007       0x0 Startupc.o
                0x20000008                . = ALIGN (0x4)
 *fill*         0x20000007       0x1
                0x20000008               _E_DATA = .

.igot.plt       0x20000008       0x0 load address 0x08000133
 .igot.plt      0x20000008       0x0 main.o

.bss            0x20000008    0x1003 load address 0x08000133
                0x20000008               _S_bss = .
```

-Simulation



CM3 Variables - U1

Name	Address	Value
g_var	20000004	byte[3]
const_var	08000180	byte[3]
bss_var	20001008	byte[3]
vectors	08000000	dword[7]
R_ODR	20000000	0x4001080C

CM3 FLASH at 0x08000000 - U1

```
08000000 08 10 00 20 F5 00 00 08 E9 00 00 08 E9 00 00 08 E9 00 00 08 E9 00 00 08 E9 00 00 08 00 10 00 20 78 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08
08000038 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 80 B4 83 B0
08000070 00 AF 1A 4B 1B 68 19 4A 43 F0 04 03 13 60 18 4B 1B 68 17 4A 23 F4 70 03 13 60 15 4B 1B 68 14 4A 43 F4 00 13 13 60 13 4B 1A 68 13 88 43 F4 00 53 13 80 00 23 7B 60 02 E0
080000A8 7B 68 01 33 7B 60 7B 68 4C F2 4F 32 93 42 F7 DD 0A 4B 1A 68 13 88 6F F3 4D 33 13 80 00 23 3B 60 02 E0 3B 68 01 33 3B 60 3B 68 4C F2 4F 32 93 42 F7 DD DC E7 18 10 02 40
080000E0 04 08 01 40 00 00 00 20 80 B5 00 AF 00 F0 02 F8 00 BF 80 BD 80 B5 88 B0 00 AF 1A 4A 1A 4B D3 1A BB 60 1A 4B FB 61 18 4B BB 61 00 23 7B 61 0A E0 FA 69 53 1C FB 61 BB 69
08000118 59 1C B9 61 12 78 1A 70 7B 69 01 33 7B 61 7B 69 BA 68 9A 42 F0 D8 10 4A 10 4B D3 1A 7B 60 0F 4B 3B 61 00 23 FB 60 07 E0 3B 69 5A 1C 3A 61 00 22 1A 70 FB 68 01 33 FB 60
```

```
... ............................ x..............
...............................................
...K.h.JC....`.K.h.J#.p..`.K.h.JC....`.K.h..C..S..
{h.3{`{hL.O2.B...K.h..o.M3...#;`..;h.3;`;hL.O2.B..
...@... ...................J.K...`.K.a.K.a.#{a...i
Y..a.x.p{i.3{a{i.h.B...J.K..{`.K;a.#.`..;iZ.:a.".p
```

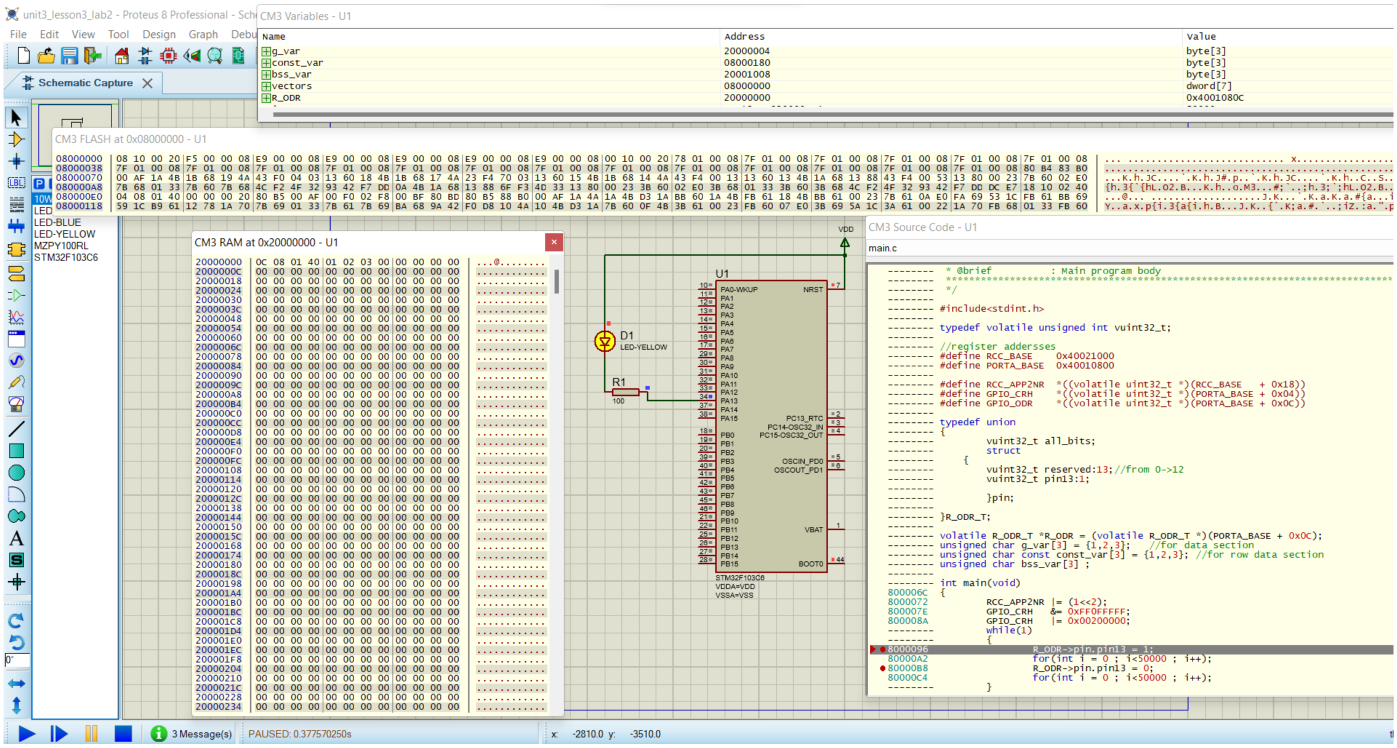
CM3 RAM at 0x20000000 - U1

```
20000000 | 0C 08 01 40 |01 02 03 00 | ...@....
20000008 | 00 00 00 00 |00 00 00 00 | ........
20000010 | 00 00 00 00 |00 00 00 00 | ........
20000018 | 00 00 00 00 |00 00 00 00 | ........
20000020 | 00 00 00 00 |00 00 00 00 | ........
20000028 | 00 00 00 00 |00 00 00 00 | ........
20000030 | 00 00 00 00 |00 00 00 00 | ........
20000038 | 00 00 00 00 |00 00 00 00 | ........
20000040 | 00 00 00 00 |00 00 00 00 | ........
20000048 | 00 00 00 00 |00 00 00 00 | ........
20000050 | 00 00 00 00 |00 00 00 00 | ........
20000058 | 00 00 00 00 |00 00 00 00 | ........
20000060 | 00 00 00 00 |00 00 00 00 | ........
20000068 | 00 00 00 00 |00 00 00 00 | ........
20000070 | 00 00 00 00 |00 00 00 00 | ........
20000078 | 00 00 00 00 |00 00 00 00 | ........
20000080 | 00 00 00 00 |00 00 00 00 | ........
20000088 | 00 00 00 00 |00 00 00 00 | ........
20000090 | 00 00 00 00 |00 00 00 00 | ........
20000098 | 00 00 00 00 |00 00 00 00 | ........
200000A0 | 00 00 00 00 |00 00 00 00 | ........
200000A8 | 00 00 00 00 |00 00 00 00 | ........
200000B0 | 00 00 00 00 |00 00 00 00 | ........
200000B8 | 00 00 00 00 |00 00 00 00 | ........
200000C0 | 00 00 00 00 |00 00 00 00 | ........
200000C8 | 00 00 00 00 |00 00 00 00 | ........
200000D0 | 00 00 00 00 |00 00 00 00 | ........
200000D8 | 00 00 00 00 |00 00 00 00 | ........
200000E0 | 00 00 00 00 |00 00 00 00 | ........
200000E8 | 00 00 00 00 |00 00 00 00 | ........
200000F0 | 00 00 00 00 |00 00 00 00 | ........
200000F8 | 00 00 00 00 |00 00 00 00 | ........
20000100 | 00 00 00 00 |00 00 00 00 | ........
20000108 | 00 00 00 00 |00 00 00 00 | ........
20000110 | 00 00 00 00 |00 00 00 00 | ........
20000118 | 00 00 00 00 |00 00 00 00 | ........
20000120 | 00 00 00 00 |00 00 00 00 | ........
20000128 | 00 00 00 00 |00 00 00 00 | ........
20000130 | 00 00 00 00 |00 00 00 00 | ........
20000138 | 00 00 00 00 |00 00 00 00 | ........
20000140 | 00 00 00 00 |00 00 00 00 | ........
20000148 | 00 00 00 00 |00 00 00 00 | ........
20000150 | 00 00 00 00 |00 00 00 00 | ........
20000158 | 00 00 00 00 |00 00 00 00 | ........
20000160 | 00 00 00 00 |00 00 00 00 | ........
20000168 | 00 00 00 00 |00 00 00 00 | ........
20000170 | 00 00 00 00 |00 00 00 00 | ........
20000178 | 00 00 00 00 |00 00 00 00 | ........
```

CM3 Source Code - U1

main.c

```
--------  * @brief          : Main program body
--------  ******************************************************************************
--------  */
--------
-------- #include<stdint.h>
--------
-------- typedef volatile unsigned int vuint32_t;
--------
-------- //register addersses
-------- #define RCC_BASE    0x40021000
-------- #define PORTA_BASE  0x40010800
--------
-------- #define RCC_APP2NR  *((volatile uint32_t *)(RCC_BASE   + 0x18))
-------- #define GPIO_CRH    *((volatile uint32_t *)(PORTA_BASE + 0x04))
-------- #define GPIO_ODR    *((volatile uint32_t *)(PORTA_BASE + 0x0C))
--------
-------- typedef union
-------- {
--------         vuint32_t all_bits;
--------         struct
--------     {
--------         vuint32_t reserved:13;//from 0->12
--------         vuint32_t pin13:1;
--------
--------         }pin;
--------
-------- }R_ODR_T;
--------
-------- volatile R_ODR_T *R_ODR = (volatile R_ODR_T *)(PORTA_BASE + 0x0C);
-------- unsigned char g_var[3] = {1,2,3};    //for data section
-------- unsigned char const const_var[3] = {1,2,3}; //for row data section
-------- unsigned char bss_var[3] ;
--------
-------- int main(void)
800006C  {
8000072         RCC_APP2NR |= (1<<2);
800007E         GPIO_CRH   &= 0xFF0FFFFF;
800008A         GPIO_CRH   |= 0x00200000;
--------        while(1)
--------        {
8000096                 R_ODR->pin.pin13 = 1;
80000A2                 for(int i = 0 ; i<50000 ; i++);
80000B8                 R_ODR->pin.pin13 = 0;
80000C4                 for(int i = 0 ; i<50000 ; i++);
--------        }
```



CM3 Variables - U1

Name	Address	Value
g_var	20000004	byte[3]
const_var	08000180	byte[3]
bss_var	20001008	byte[3]
vectors	08000000	dword[7]
R_ODR	20000000	0x4001080C

CM3 FLASH at 0x08000000 - U1

```
08000000 08 10 00 20 F5 00 00 08 E9 00 00 08 E9 00 00 08 E9 00 00 08 E9 00 00 08 E9 00 00 08 00 10 00 20 78 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08
08000038 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 7F 01 00 08 80 B4 83 B0
08000070 00 AF 1A 4B 1B 68 19 4A 43 F0 04 03 13 60 18 4B 1B 68 17 4A 23 F4 70 03 13 60 15 4B 1B 68 14 4A 43 F4 00 13 13 60 13 4B 1A 68 13 88 43 F4 00 53 13 80 00 23 7B 60 02 E0
080000A8 7B 68 01 33 7B 60 7B 68 4C F2 4F 32 93 42 F7 DD 0A 4B 1A 68 13 88 6F F3 4D 33 13 80 00 23 3B 60 02 E0 3B 68 01 33 3B 60 3B 68 4C F2 4F 32 93 42 F7 DD DC E7 18 10 02 40
080000E0 04 08 01 40 00 00 00 20 80 B5 00 AF 00 F0 02 F8 00 BF 80 BD 80 B5 88 B0 00 AF 1A 4A 1A 4B D3 1A BB 60 1A 4B FB 61 18 4B BB 61 00 23 7B 61 0A E0 FA 69 53 1C FB 61 BB 69
08000118 59 1C B9 61 12 78 1A 70 7B 69 01 33 7B 61 7B 69 BA 68 9A 42 F0 D8 10 4A 10 4B D3 1A 7B 60 0F 4B 3B 61 00 23 FB 60 07 E0 3B 69 5A 1C 3A 61 00 22 1A 70 FB 68 01 33 FB 60
```

```
... ............................ x..............
...............................................
...K.h.JC....`.K.h.J#.p..`.K.h.JC....`.K.h..C..S..
{h.3{`{hL.O2.B...K.h..o.M3...#;`..;h.3;`;hL.O2.B..
...@... ...................J.K...`.K.a.K.a.#{a...i
Y..a.x.p{i.3{a{i.h.B...J.K..{`.K;a.#.`..;iZ.:a.".p
```

CM3 RAM at 0x20000000 - U1

```
20000000 | 0C 08 01 40 |01 02 03 00 |00 00 00 00 | ...@........
2000000C | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000018 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000024 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000030 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
2000003C | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000048 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000054 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000060 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
2000006C | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000078 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000084 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000090 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
2000009C | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200000A8 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200000B4 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200000C0 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200000CC | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200000D8 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200000E4 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200000F0 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200000FC | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000108 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000114 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000120 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
2000012C | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000138 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000144 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000150 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
2000015C | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000168 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000174 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000180 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
2000018C | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000198 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200001A4 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200001B0 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200001BC | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200001C8 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200001D4 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200001E0 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200001EC | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
200001F8 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000204 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000210 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
2000021C | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000228 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
20000234 | 00 00 00 00 |00 00 00 00 |00 00 00 00 | ............
```

CM3 Source Code - U1

main.c

```
--------  * @brief          : Main program body
--------  ******************************************************************************
--------  */
--------
-------- #include<stdint.h>
--------
-------- typedef volatile unsigned int vuint32_t;
--------
-------- //register addersses
-------- #define RCC_BASE    0x40021000
-------- #define PORTA_BASE  0x40010800
--------
-------- #define RCC_APP2NR  *((volatile uint32_t *)(RCC_BASE   + 0x18))
-------- #define GPIO_CRH    *((volatile uint32_t *)(PORTA_BASE + 0x04))
-------- #define GPIO_ODR    *((volatile uint32_t *)(PORTA_BASE + 0x0C))
--------
-------- typedef union
-------- {
--------         vuint32_t all_bits;
--------         struct
--------     {
--------         vuint32_t reserved:13;//from 0->12
--------         vuint32_t pin13:1;
--------
--------         }pin;
--------
-------- }R_ODR_T;
--------
-------- volatile R_ODR_T *R_ODR = (volatile R_ODR_T *)(PORTA_BASE + 0x0C);
-------- unsigned char g_var[3] = {1,2,3};    //for data section
-------- unsigned char const const_var[3] = {1,2,3}; //for row data section
-------- unsigned char bss_var[3] ;
--------
-------- int main(void)
800006C  {
8000072         RCC_APP2NR |= (1<<2);
800007E         GPIO_CRH   &= 0xFF0FFFFF;
800008A         GPIO_CRH   |= 0x00200000;
--------        while(1)
--------        {
8000096                 R_ODR->pin.pin13 = 1;
80000A2                 for(int i = 0 ; i<50000 ; i++);
80000B8                 R_ODR->pin.pin13 = 0;
80000C4                 for(int i = 0 ; i<50000 ; i++);
--------        }
```