Unit 3 Embedded C
Lecture 3 Report

Part 1] Debug BareMetal SW on ARM VersatilePB (Lab1)

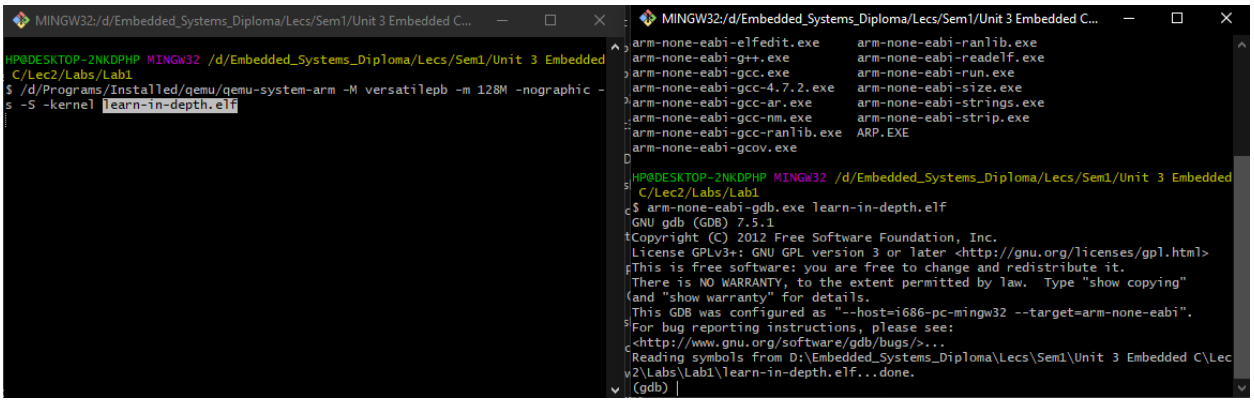
Part 2] Write BareMetal SW on STM32MPU ARM CortexM3 (Lab2)

By: Yara Ashraf

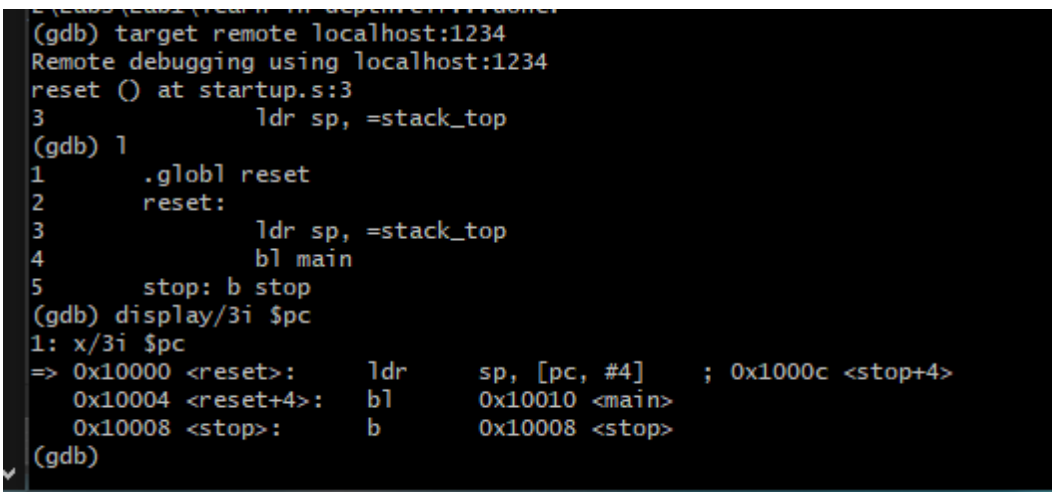
Debug BareMetal SW on ARM VersatilePB (Lab1)

A] Debug via terminal (gdb)

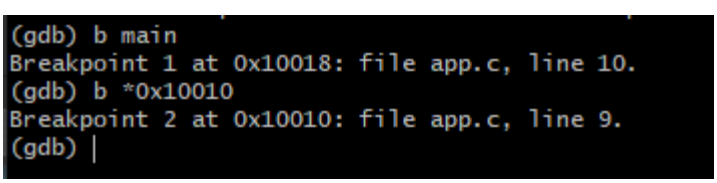
- Run QEMU and pass options -s -S to debug
- Call arm-none-eabi-gdb.exe



- Connecting to localhost target on port 1234
- l (list current file)
- PC at entry point (reset section in startup)



- Adding breakpoints to main (0x10018) and to 0x10010 to include context switching to main()



- si (step assembly instruction)
- s (step c line)
- c (continue to next breakpoint)

```
(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, lsl #4
(gdb) s

Breakpoint 2, main () at app.c:9
9       {
1: x/3i $pc
=> 0x10010 <main>:      push    {r11, lr}
   0x10014 <main+4>:    add     r11, sp, #4
   0x10018 <main+8>:    ldr     r0, [pc, #4]    ; 0x10024 <main+20>
(gdb) c
Continuing.

Breakpoint 1, main () at app.c:10
10              Uart_Send_String (&string_buffer[0]);
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) c
Continuing.
```

- Output on QEMU, string was sent via UART.

```
MINGW32:/d/Embedded_Systems_Diploma/Lecs/Sem1/Unit 3 Embedded C/Lec2/Labs/Lab1

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Lecs/Sem1/Unit 3 Embedded
 C/Lec2/Labs/Lab1
$ /d/Programs/Installed/qemu/qemu-system-arm -M versatilepb -m 128M -nographic -
s -S -kernel learn-in-depth.elf
learn-in-depth:yara
```

- watch (monitor variable for any change, then branch to change)
- where (print current function/file/line)
- backtrace (print stack trace of function calls)
- info breakpoints (list all breakpoints with their hit count)
- adding breakpoint to **Uart_Send_String** function

```
(gdb) watch string_buffer
Hardware watchpoint 3: string_buffer
(gdb) backtrace
#0  stop () at startup.s:5
#1  0x00010020 in main () at app.c:10
(gdb) where
#0  stop () at startup.s:5
#1  0x00010020 in main () at app.c:10
(gdb) info breakpoints
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x00010018 in main at app.c:10
        breakpoint already hit 1 time
2       breakpoint     keep y   0x00010010 in main at app.c:9
        breakpoint already hit 1 time
3       hw watchpoint  keep y              string_buffer
(gdb) b Uart_Send_String
Breakpoint 4 at 0x10038: file uart.c, line 7.
```

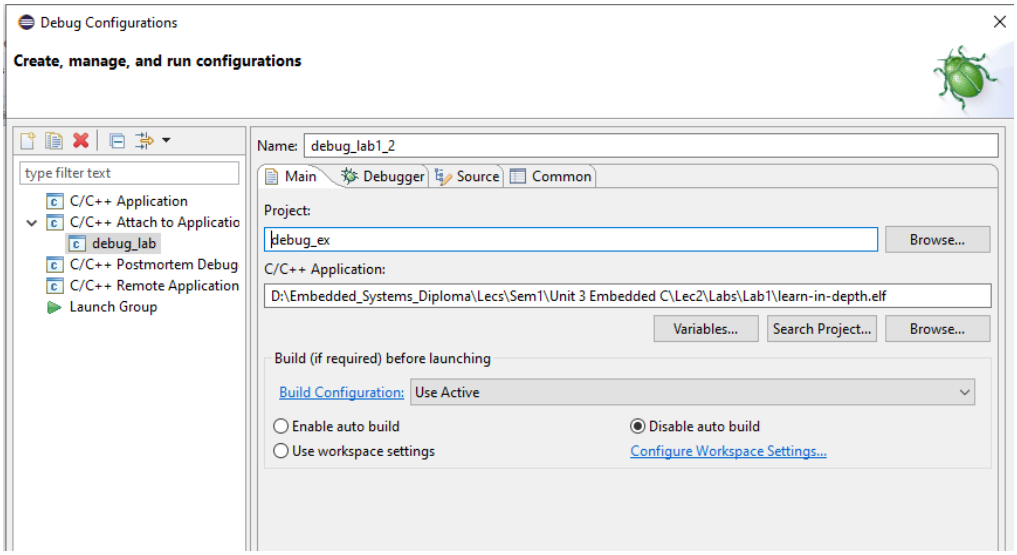
- step to **Uart_Send_String** function in while loop to send the string (char by char)

```
MINGW32:/d/Embedded_Systems_Diploma/Lecs/Sem1/Unit 3 Embedded C/Lec2/Labs/Lab1

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Lecs/Sem1/Unit 3 Embedded
 C/Lec2/Labs/Lab1
$ /d/Programs/Installed/qemu/qemu-system-arm -M versatilepb -m 128M -nographic -s -S -kernel l
earn-in-depth.elf
learn-in-depth:ya
```

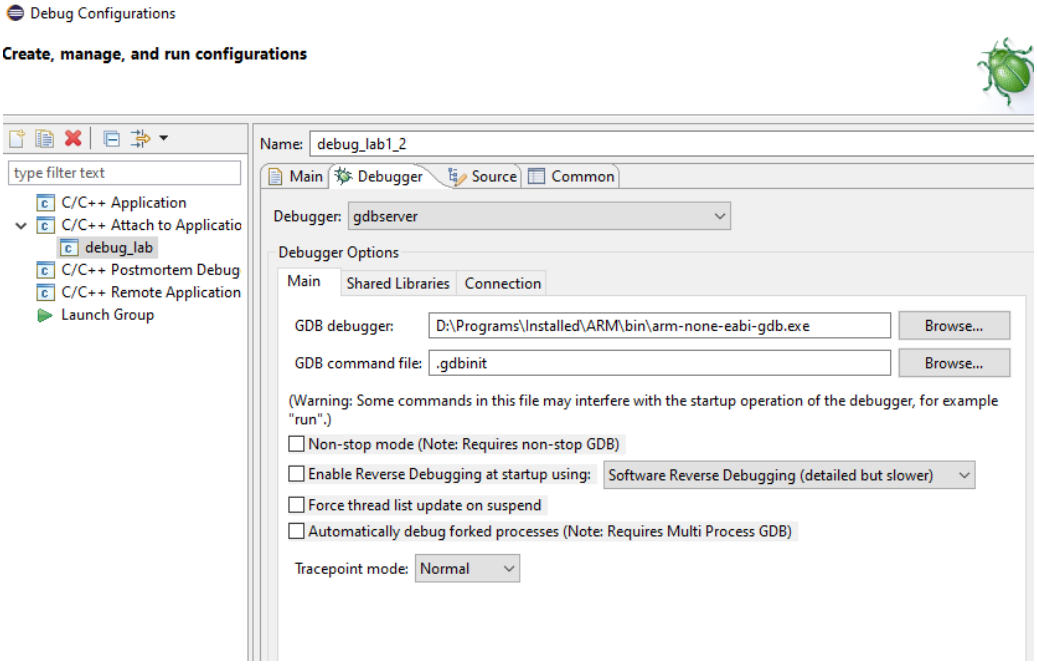
```
MINGW32:/d/Embedded_Systems_Diploma/Lecs/Sem1/Unit 3 Embedded C/Lec2/Labs/Lab1
1: x/3i $pc
=> 0x10058 <Uart_Send_String+48>:       ldr     r3, [r11, #-8]
   0x1005c <Uart_Send_String+52>:       ldrb    r3, [r3]
   0x10060 <Uart_Send_String+56>:       cmp     r3, #0
(gdb) s

Breakpoint 4, Uart_Send_String (P_tx_string=0x101b3 <string_buffer+15> "yara")
    at uart.c:9
9                       UART0DR = (unsigned int)(*P_tx_string);         //Transmit char
1: x/3i $pc
=> 0x1003c <Uart_Send_String+20>:
    ldr r3, [pc, #48]   ; 0x10074 <Uart_Send_String+76>
   0x10040 <Uart_Send_String+24>:       ldr     r2, [r11, #-8]
   0x10044 <Uart_Send_String+28>:       ldrb    r2, [r2]
(gdb) s
10                      P_tx_string++;
1: x/3i $pc
=> 0x1004c <Uart_Send_String+36>:       ldr     r3, [r11, #-8]
   0x10050 <Uart_Send_String+40>:       add     r3, r3, #1
   0x10054 <Uart_Send_String+44>:       str     r3, [r11, #-8]
(gdb) s
7               while(*P_tx_string !='\0')
1: x/3i $pc
=> 0x10058 <Uart_Send_String+48>:       ldr     r3, [r11, #-8]
   0x1005c <Uart_Send_String+52>:       ldrb    r3, [r3]
   0x10060 <Uart_Send_String+56>:       cmp     r3, #0
(gdb) s

Breakpoint 4, Uart_Send_String (P_tx_string=0x101b4 <string_buffer+16> "ara")
    at uart.c:9
9                       UART0DR = (unsigned int)(*P_tx_string);         //Transmit char
1: x/3i $pc
=> 0x1003c <Uart_Send_String+20>:
    ldr r3, [pc, #48]   ; 0x10074 <Uart_Send_String+76>
   0x10040 <Uart_Send_String+24>:       ldr     r2, [r11, #-8]
   0x10044 <Uart_Send_String+28>:       ldrb    r2, [r2]
(gdb) s
10                      P_tx_string++;
1: x/3i $pc
=> 0x1004c <Uart_Send_String+36>:       ldr     r3, [r11, #-8]
   0x10050 <Uart_Send_String+40>:       add     r3, r3, #1
   0x10054 <Uart_Send_String+44>:       str     r3, [r11, #-8]
(gdb) s
7               while(*P_tx_string !='\0')
1: x/3i $pc
=> 0x10058 <Uart_Send_String+48>:       ldr     r3, [r11, #-8]
   0x1005c <Uart_Send_String+52>:       ldrb    r3, [r3]
   0x10060 <Uart_Send_String+56>:       cmp     r3, #0
(gdb) s
```

B] Debug via eclipse

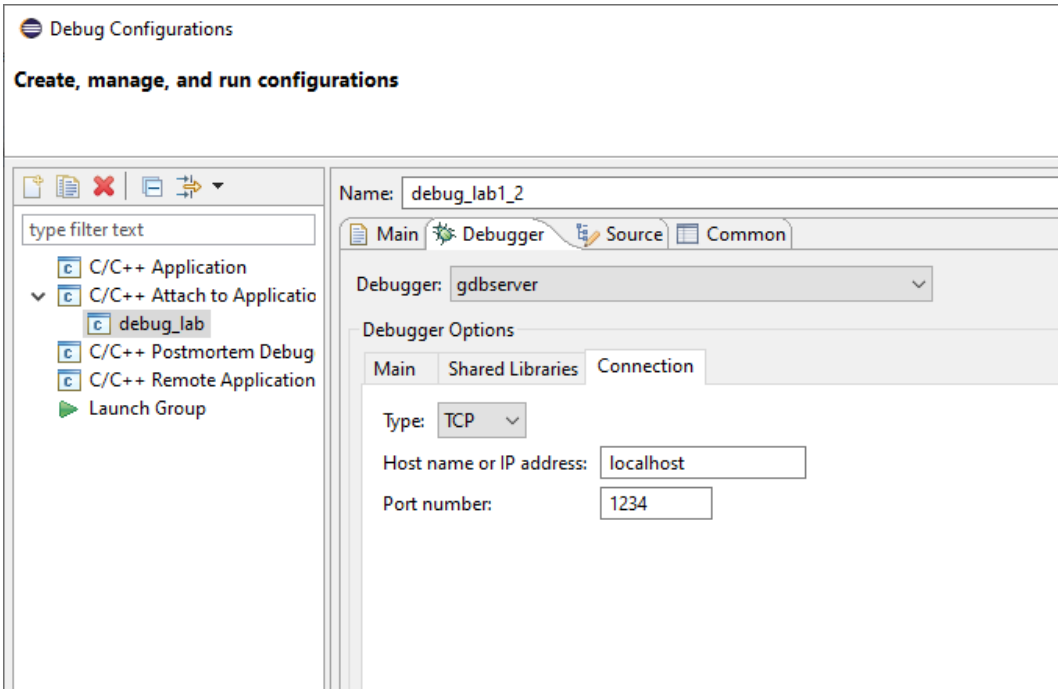
- Attach .elf file to debug project, disabling auto build for the project as we are concerned only with the .elf



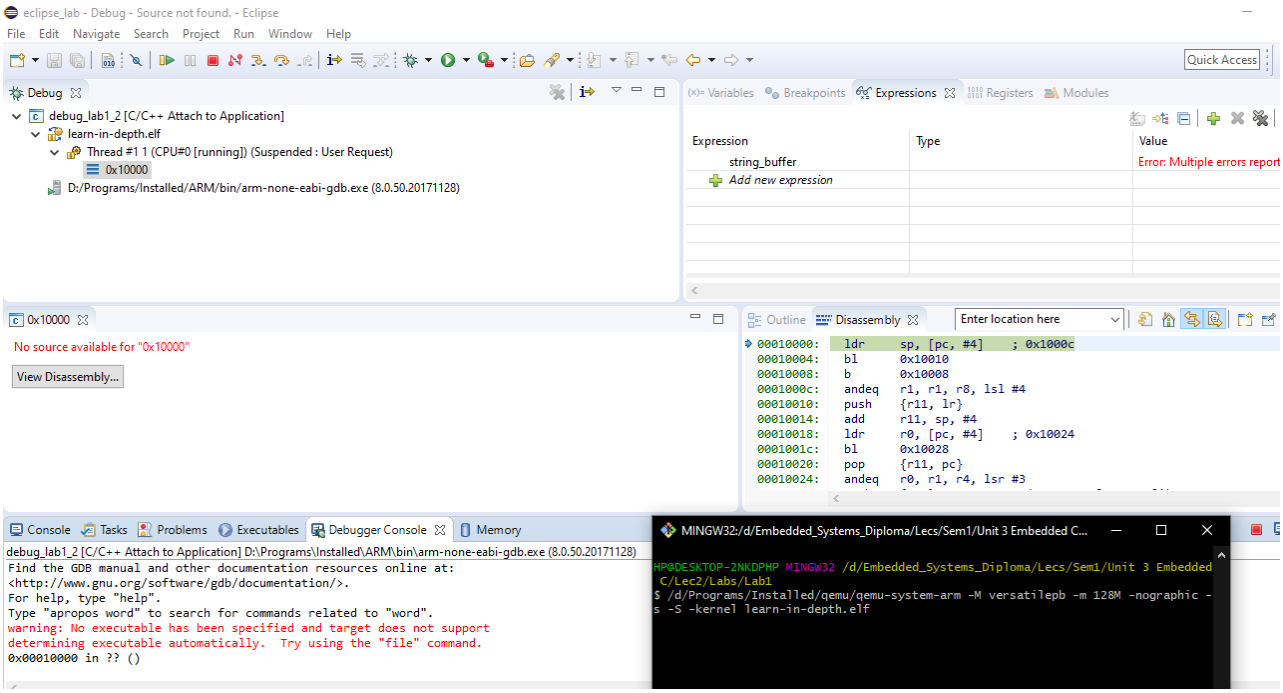
- Select gdbserver for the debugger



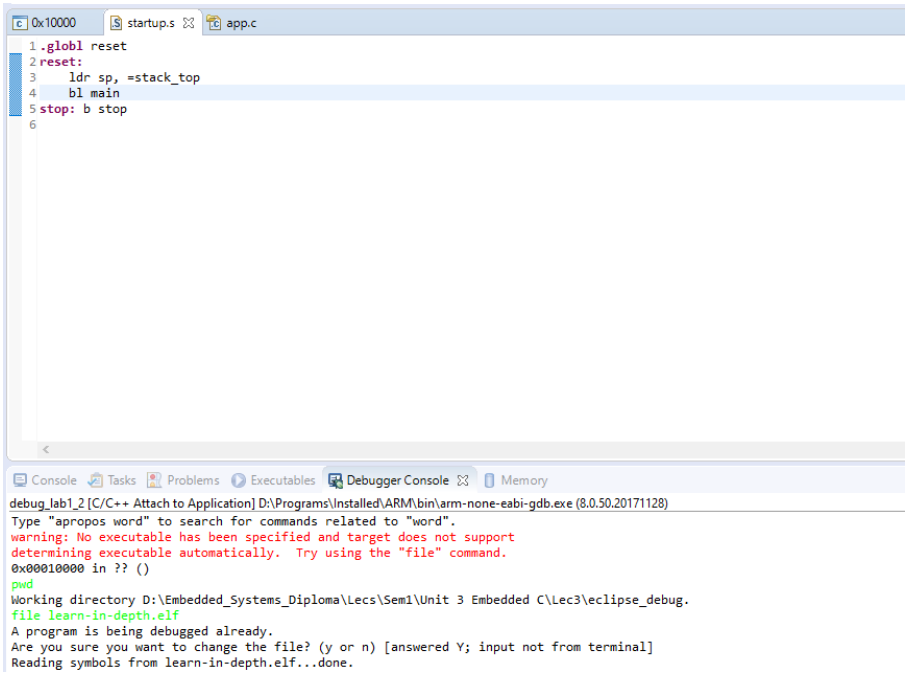
- Connect via TCP port 1234 to localhost



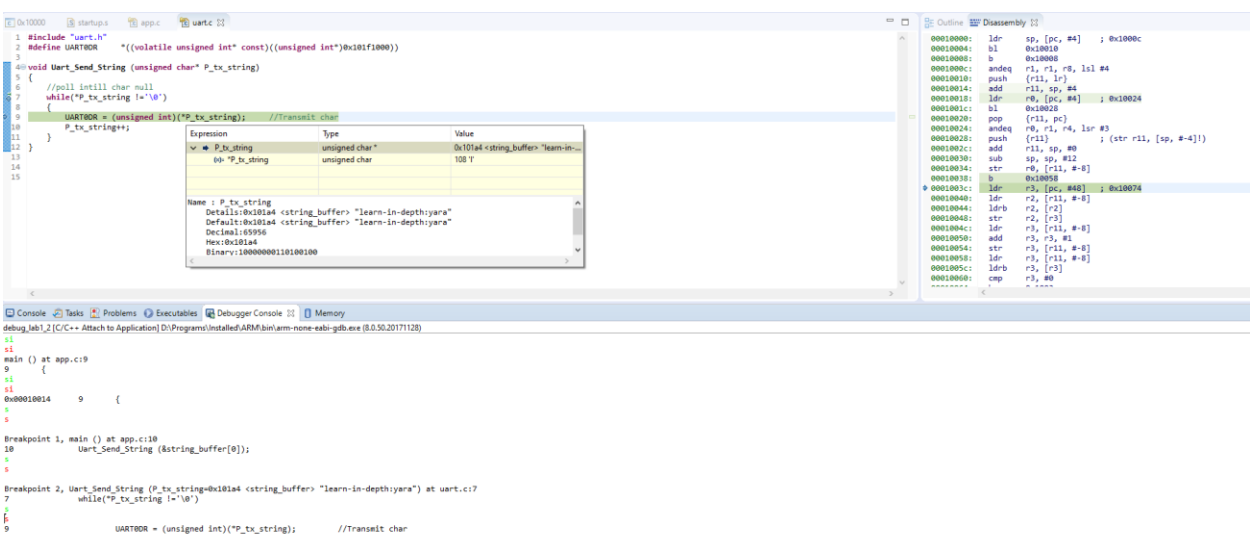
- After running the debugger circuit on QEMU board and connecting gdb client to gdbserver on eclipse, we can see that the .elf was not read and address 0x1000 was not resolved



- After copying .elf to the working directory (eclipse project) and calling it via file command , the gdb was able to read the symbols



- We can view disassembly and execute gdb commands via the console as in the terminal
- Hovering over the pointer, we can see the value its pointing to



Writing a BareMetal SW on STM32MPU ARM CortexM3

`

Main.c

File Edit Selection Find View Goto Tools Project Preferences Help

startup.s-orig | main.c

```c
 *                          opensource.org/licenses/bsd-3-clause
 *
 ******************************************************************************
 */
#include "stdint.h"

typedef volatile unsigned int vint32_t;

#define RCC_BASE 0x40021000
#define PORTA_BASE 0x40010800

#define RCC_APB2ENR     *(volatile uint32_t*)(RCC_BASE + 0x18)
#define GPIOA_CRH       *(volatile uint32_t*)(PORTA_BASE + 0x04)
#define GPIOA_ODR       *(volatile uint32_t*)(PORTA_BASE + 0x0C)

typedef union {
    vint32_t all_fields;
    struct {
        vint32_t reserved:13;
        vint32_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};
/*volatile unsigned char bss_var[3] ;*/

int main (void)
{

    RCC_APB2ENR |= 1<<2; //enable RCC CLK to GPIOA, bit 2
    GPIOA_CRH &=0xFF0FFFFF; //clear 20 to 24
    GPIOA_CRH |=0x00200000; //write 2

    int i;
    while (1)
    {

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

        R_ODR->pin.pin13=1;
        for(i=0;i<5000;i++);
        R_ODR->pin.pin13=0;
        for(i=0;i<5000;i++);

    }

    return 0;
}

```

Makefile

- gdwarf-2 option is passed to compiler and assembler to output debug sections to be able to debug on Proteus

D:\Embedded_Systems_Diploma\Git\Learn-in-Depth-Diploma\Unit 3 Embedded C\Lec 3 Lab Assignment\Makefile - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

main.c | startup.s-orig | Makefile

```makefile
#@copyright : Yara
CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -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

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

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

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

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

clean_all:
	rm *.o *.elf *.bin *.map
clean:
	rm *.elf *.bin
```

A] startup.s

- defining .vectors section to include the vector handlers

- the .text section includes section labels : reset (to branch to main) and vector_handler (to branch to reset section)

main.c startup.s-orig

```
/* startup_cortexM3.s
Eng.Yara
*/

/*SRAM 0x20000000 */

.section .vectors

.word 0x20001000        /*stack top address*/
.word _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 reserved */
.word Vector_handler    /* 13 Reserved */
.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
Vector_handler:
    b _reset
```

- Calling the cross toolchain via **make** command

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embedded C/Lec 3 Lab Assignment (main)
$ ls
linker_script.ld  main.c  Makefile  startup.s

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diplom
a/Unit 3 Embedded C/Lec 3 Lab Assignment (main)
$ make
arm-none-eabi-as.exe -mcpu=cortex-m3 -gdwarf-2 startup.s -o startup.o
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -I . -c main.c -o main.o
arm-none-eabi-ld.exe -T linker_script.ld  startup.o main.o -o learn-in-depth_cor
tex_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

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diplom
a/Unit 3 Embedded C/Lec 3 Lab Assignment (main)
$ ls
learn-in-depth_cortex_m3.bin  learn-in-depth_cortex_m3.elf  linker_script.ld  main.c  main.o  Makefile  Map_file.map  startup.o  startup.s
```

- main.o headers sections

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit
$ 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         0000007c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000007  00000000  00000000  000000b0  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b7  2**0
                  ALLOC
  3 .rodata       00000003  00000000  00000000  000000b8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .debug_info   000001a5  00000000  00000000  000000bb  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_abbrev 000000e3  00000000  00000000  00000260  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_loc    00000038  00000000  00000000  00000343  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  0000037b  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_line   000001e3  00000000  00000000  0000039b  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  9 .debug_str    000001c2  00000000  00000000  0000057e  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .comment      0000007c  00000000  00000000  00000740  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000002c  00000000  00000000  000007bc  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 12 .ARM.attributes 00000033  00000000  00000000  000007e8  2**0
                  CONTENTS, READONLY
```

- main.o symbols

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma
$ arm-none-eabi-nm.exe main.o
00000003 C bss_var
00000000 R const_variables
00000004 D g_variables
00000000 T main
00000000 D R_ODR
```

- startup.o sections

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Emb
$ arm-none-eabi-objdump.exe -h startup.o

startup.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000008  00000000  00000000  00000034  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  0000003c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  0000003c  2**0
                  ALLOC
  3 .vectors      00000050  00000000  00000000  0000003c  2**0
                  CONTENTS, RELOC, READONLY
  4 .debug_line   0000003b  00000000  00000000  0000008c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_info   00000026  00000000  00000000  000000c7  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  6 .debug_abbrev 00000014  00000000  00000000  000000ed  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  00000108  2**3
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000076  00000000  00000000  00000128  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .ARM.attributes 00000021  00000000  00000000  0000019e  2**0
                  CONTENTS, READONLY
```

- startup.o symbols

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/
$ arm-none-eabi-nm.exe startup.o
00000000 t _reset
         U main
00000006 t Vector_handler

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/
```

Part B] startup.c

- initial startup.c , each handler is defined seperately

D:\Embedded_Systems_Diploma\Git\Learn-in-Depth-Diploma\Unit 3 Embedded C\Lec 3 Lab Assignment\startup.c - Sublime Text (U

File Edit Selection Find View Goto Tools Project Preferences Help

linker_script.ld — Git\...\Lec 3 Lab Assignment | linker_script.ld — Lecs\...\Lab2 | startup.c | Makefile

```c
//startup.c
//Eng. Yara
//Learn-in-depth

#include "stdint.h"
#define stack_top_SP 0x20001000
extern int main ();

void Reset_Handler (void)
{
    main();
}
void NMI_Handler (void)
{
    Reset_Handler();
}
void H_Fault_Handler (void)
{
    Reset_Handler();
}
void MM_Fault_Handler (void)
{
    Reset_Handler();
}
void Bus_Fault (void)
{
    Reset_Handler();
}
void Usage_Fault_Handler (void)
{
    Reset_Handler();
}


//define .vectors section
//addresses dependent on MCU vector table
uint32_t vectors[] __attribute__((section(".vectors"))) = {

    stack_top_SP,
    (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

};
```

- .elf resolved symbols, each symbol has its own address

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 3 Lab Assignment (main)
$ make
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -I . -c startup.c -o startup.o
arm-none-eabi-ld.exe -T linker_script.ld   main.o startup.o -o learn-in-depth_cortex_m3.elf -M
ap=Map_file.map
arm-none-eabi-objcopy.exe -O binary learn-in-depth_cortex_m3.elf learn-in-depth_cortex_m3.bin

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 3 Lab Assignment (main)
$ arm-none-eabi-nm.exe learn-in-depth_cortex_m3.elf
20000000 B bss_var
080000c8 T Bus_Fault
080000e0 T const_variables
080000e8 D g_variables
080000b0 T H_Fault_Handler
0800001c T main
080000bc T MM_Fault_Handler
080000a4 T NMI_Handler
080000e4 D R_ODR
08000098 T Reset_Handler
080000d4 T Usage_Fault_Handler
08000000 T vectors
```

- After defining the vector handlers via **weak** (to allow redefinition) and **alias** attributes (to refer to the Default_Handler Symbol)

D:\Embedded_Systems_Diploma\Git\Learn-in-Depth-Diploma\Unit 3 Embedded C\Lec 3 Lab Assignment\startup.c - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

linker_script.ld — Git\...\Lec 3 Lab Assignment | linker_script.ld — Lecs\...\Lab2 | startup.c | Makefile

```c
//startup.c
//Eng. Yara
//Learn-in-depth

#include "stdint.h"
#define stack_top_SP 0x20001000
extern int main ();

void Reset_Handler (void);
void Default_Handler (void)
{
    Reset_Handler();
}

//passing weak attribute to compiler to be able to redefine handlers and alias so that it refers to Default_Handler symbol
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")));


//define .vectors section
//addresses dependent on MCU vector table
uint32_t vectors[] __attribute__((section(".vectors"))) = {

    stack_top_SP,
    (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

};

void Reset_Handler (void)
{
    main();
}

```

- Now the symbols that are alias to Default_Handler refer to its same address

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 3 Lab Assignment (main)
$ make
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -I . -c startup.c -o startup.o
arm-none-eabi-ld.exe -T linker_script.ld   main.o startup.o -o learn-in-depth_cortex_m3.elf -M
ap=Map_file.map
arm-none-eabi-objcopy.exe -O binary learn-in-depth_cortex_m3.elf learn-in-depth_cortex_m3.bin

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 3 Lab Assignment (main)
$ arm-none-eabi-nm.exe learn-in-depth_cortex_m3.elf
20000000 B bss_var
08000098 W Bus_Fault
080000b0 T const_variables
08000098 T Default_Handler
080000b8 D g_variables
08000098 W H_Fault_Handler
0800001c T main
08000098 W MM_Fault_Handler
08000098 W NMI_Handler
080000b4 D R_ODR
080000a4 T Reset_Handler
08000098 W Usage_Fault_Handler
08000000 T vectors
```

- main.c after redefining the NMI & Bus_Fault vector handlers

linker_script.ld — Git\...\Lec 3 Lab Assignme | linker_script.ld — Lecs\...\Lab2 | startup.c | main.c

```c
 */
#include "stdint.h"

typedef volatile unsigned int vint32_t;

#define RCC_BASE 0x40021000
#define PORTA_BASE 0x40010800

#define RCC_APB2ENR     *(volatile uint32_t*)(RCC_BASE + 0x18)
#define GPIOA_CRH       *(volatile uint32_t*)(PORTA_BASE + 0x04)
#define GPIOA_ODR       *(volatile uint32_t*)(PORTA_BASE + 0x0C)

extern void NMI_Handler (void)
{

}
extern void Bus_Fault (void)
{

}

typedef union {
    vint32_t all_fields;
    struct {
        vint32_t reserved:13;
        vint32_t pin13:1;

    }pin;

}R_ODR_t;

```

- NMI & Bus_Fault vector handlers have different symbols than the Defualt_Handler after their redefinition even though they were alias to it as they were defined as **weak attributes**

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 3 Lab Assignment (main)
$ make
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -I . -c startup.c -o startup.o
arm-none-eabi-ld.exe -T linker_script.ld   main.o startup.o -o learn-in-depth_cortex_m3.elf -M
ap=Map_file.map
arm-none-eabi-objcopy.exe -O binary learn-in-depth_cortex_m3.elf learn-in-depth_cortex_m3.bin

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 3 Lab Assignment (main)
$ arm-none-eabi-nm.exe learn-in-depth_cortex_m3.elf
20000000 B bss_var
08000028 T Bus_Fault
080000c8 T const_variables
080000b0 T Default_Handler
080000d0 D g_variables
080000b0 W H_Fault_Handler
08000034 T main
080000b0 W MM_Fault_Handler
0800001c T NMI_Handler
080000cc D R_ODR
080000bc T Reset_Handler
080000b0 W Usage_Fault_Handler
08000000 T vectors
```

- Linker_script before Memory Alignment

D:\Embedded_Systems_Diploma\Git\Learn-in-Depth-Diploma\Unit 3 Embedded C\Lec 3 Lab Assi

File Edit Selection Find View Goto Tools Project Preferences Help

linker_script.ld | Makefile | main.c

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

        . = . + 0x1000 ;
        _stack_top = . ;
    } > sram
}
```

- Map_file before Memory Alignment
- _E_data = 0x20000007

```
Linker script and memory map


 .text           0x0000000008000000      0x12b
  *(.vectors*)
  .vectors       0x0000000008000000       0x1c startup.o
                 0x0000000008000000                vectors
  *(.text*)
  .text          0x000000000800001c       0x7c main.o
                 0x000000000800001c                main
  .text          0x0000000008000098       0x90 startup.o
                 0x0000000008000098                H_Fault_Handler
                 0x0000000008000098                MM_Fault_Handler
                 0x0000000008000098                Usage_Fault_Handler
                 0x0000000008000098                Bus_Fault
                 0x0000000008000098                Default_Handler
                 0x0000000008000098                NMI_Handler
                 0x00000000080000a4                Reset_Handler
  *(.rodata*)
  .rodata        0x0000000008000128        0x3 main.o
                 0x0000000008000128                const_variables
                 0x000000000800012b                _E_text = .

 .glue_7         0x000000000800012c        0x0
  .glue_7        0x000000000800012c        0x0 linker stubs

 .glue_7t        0x000000000800012c        0x0
  .glue_7t       0x000000000800012c        0x0 linker stubs

 .vfp11_veneer   0x000000000800012c        0x0
  .vfp11_veneer  0x000000000800012c        0x0 linker stubs

 .v4_bx          0x000000000800012c        0x0
  .v4_bx         0x000000000800012c        0x0 linker stubs

 .iplt           0x000000000800012c        0x0
  .iplt          0x000000000800012c        0x0 main.o

 .rel.dyn        0x000000000800012c        0x0
  .rel.iplt      0x000000000800012c        0x0 main.o

 .data           0x0000000020000000        0x7 load address 0x000000000800012b
                 0x0000000020000000                _S_data = .
  *(.data)
  .data          0x0000000020000000        0x7 main.o
                 0x0000000020000000                R_ODR
                 0x0000000020000004                g_variables
  .data          0x0000000020000007        0x0 startup.o
                 0x0000000020000007                _E_data = .

 .igot.plt       0x0000000020000008        0x0 load address 0x0000000008000132
  .igot.plt      0x0000000020000008        0x0 main.o

 .bss            0x0000000020000007     0x1000 load address 0x0000000008000132
                 0x0000000020000007                _S_bss = .
  *(.bss*)
  .bss           0x0000000020000007        0x0 main.o
  .bss           0x0000000020000007        0x0 startup.o
                 0x0000000020000007                _E_bss = .
                 0x0000000020001007                . = (. + 0x1000)
  *fill*         0x0000000020000007     0x1000
                 0x0000000020001007                _stack_top = .
```

- Aligning .data section by 4 Bytes

linker_script.ld | Map_file.map | Makefile | main.c

```
/*
Learn-in-depth
Unit2_Lec3_Lab2
Eng. Yara Ashraf
*/

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 = . ;
        . = . + 0x1000 ;
        _stack_top = . ;
    } > sram
}
```

- Map_file after alignment, _E_data = 0x20000008

```
Linker script and memory map


.text           0x0000000008000000      0x12b
 *(.vectors*)
 .vectors       0x0000000008000000       0x1c startup.o
                0x0000000008000000                vectors
 *(.text*)
 .text          0x000000000800001c       0x7c main.o
                0x000000000800001c                main
 .text          0x0000000008000098       0x90 startup.o
                0x0000000008000098                H_Fault_Handler
                0x0000000008000098                MM_Fault_Handler
                0x0000000008000098                Usage_Fault_Handler
                0x0000000008000098                Bus_Fault
                0x0000000008000098                Default_Handler
                0x0000000008000098                NMI_Handler
                0x00000000080000a4                Reset_Handler
 *(.rodata*)
 .rodata        0x0000000008000128        0x3 main.o
                0x0000000008000128                const_variables
                0x000000000800012b                _E_text = .

.glue_7         0x000000000800012c        0x0
 .glue_7        0x000000000800012c        0x0 linker stubs

.glue_7t        0x000000000800012c        0x0
 .glue_7t       0x000000000800012c        0x0 linker stubs

.vfp11_veneer   0x000000000800012c        0x0
 .vfp11_veneer  0x000000000800012c        0x0 linker stubs

.v4_bx          0x000000000800012c        0x0
 .v4_bx         0x000000000800012c        0x0 linker stubs

.iplt           0x000000000800012c        0x0
 .iplt          0x000000000800012c        0x0 main.o

.rel.dyn        0x000000000800012c        0x0
 .rel.iplt      0x000000000800012c        0x0 main.o

.data           0x0000000020000000        0x8 load address 0x000000000800012b
                0x0000000020000000                _S_data = .
 *(.data)
 .data          0x0000000020000000        0x7 main.o
                0x0000000020000000                R_ODR
                0x0000000020000004                g_variables
 .data          0x0000000020000007        0x0 startup.o
                0x0000000020000008                . = ALIGN (0x4)
 *fill*         0x0000000020000007        0x1
                0x0000000020000008                _E_data = .

.igot.plt       0x0000000020000008        0x0 load address 0x0000000008000133
 .igot.plt      0x0000000020000008        0x0 main.o

.bss            0x0000000020000008     0x1000 load address 0x0000000008000133
                0x0000000020000008                _S_bss = .
 *(.bss*)
 .bss           0x0000000020000008        0x0 main.o
 .bss           0x0000000020000008        0x0 startup.o
                0x0000000020000008                _E_bss = .
                0x0000000020001008                . = (. + 0x1000)
 *fill*         0x0000000020000008     0x1000
                0x0000000020001008                _stack_top = .
LOAD main.o
```

- main.o objdump

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-D
ded C/Lec 3 Lab Assignment (main)
$ 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         0000007c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000007  00000000  00000000  000000b0  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b7  2**0
                  ALLOC
  3 .rodata       00000003  00000000  00000000  000000b8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .debug_info   00000179  00000000  00000000  000000bb  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_abbrev 000000e3  00000000  00000000  00000234  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_loc    00000038  00000000  00000000  00000317  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  0000034f  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_line   000001e3  00000000  00000000  0000036f  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  9 .debug_str    000001ba  00000000  00000000  00000552  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .comment      0000007c  00000000  00000000  0000070c  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000002c  00000000  00000000  00000788  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 12 .ARM.attributes 00000033  00000000  00000000  000007b4  2**0
                  CONTENTS, READONLY
```

- .elf objdump

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/U
ded C/Lec 3 Lab Assignment (main)
$ 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         0000012b  08000000  08000000  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000008  20000000  0800012b  00020000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001000  20000008  08000133  00020008  2**0
                  ALLOC
  3 .debug_info   000002e6  00000000  00000000  00020008  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 000001a7  00000000  00000000  000202ee  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000000b4  00000000  00000000  00020495  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000040  00000000  00000000  00020549  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   000003d7  00000000  00000000  00020589  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    000001fd  00000000  00000000  00020960  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007b  00000000  00000000  00020b5d  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00020bd8  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000007c  00000000  00000000  00020c0c  2**2
                  CONTENTS, READONLY, DEBUGGING
```

- Readelf shows entry point = 0x08000000

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3
ded C/Lec 3 Lab Assignment (main)
$ arm-none-eabi-readelf.exe -a learn-in-depth_cortex_m3.elf
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x8000000
  Start of program headers:          52 (bytes into file)
  Start of section headers:          135304 (bytes into file)
  Flags:                             0x5000200, Version5 EABI, soft-float ABI
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         2
  Size of section headers:           40 (bytes)
  Number of section headers:         16
  Section header string table index: 15

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .text             PROGBITS        08000000 010000 00012b 00  AX  0   0  4
  [ 2] .data             PROGBITS        20000000 020000 000008 00  WA  0   0  4
  [ 3] .bss              NOBITS          20000008 020008 001000 00  WA  0   0  1
  [ 4] .debug_info       PROGBITS        00000000 020008 0002e6 00      0   0  1
  [ 5] .debug_abbrev     PROGBITS        00000000 0202ee 0001a7 00      0   0  1
  [ 6] .debug_loc        PROGBITS        00000000 020495 0000b4 00      0   0  1
  [ 7] .debug_aranges    PROGBITS        00000000 020549 000040 00      0   0  1
  [ 8] .debug_line       PROGBITS        00000000 020589 0003d7 00      0   0  1
  [ 9] .debug_str        PROGBITS        00000000 020960 0001fd 01  MS  0   0  1
  [10] .comment          PROGBITS        00000000 020b5d 00007b 01  MS  0   0  1
  [11] .ARM.attributes   ARM_ATTRIBUTES  00000000 020bd8 000033 00      0   0  1
  [12] .debug_frame      PROGBITS        00000000 020c0c 00007c 00      0   0  4
  [13] .symtab           SYMTAB          00000000 020c88 000280 10     14  22  4
  [14] .strtab           STRTAB          00000000 020f08 0000e1 00      0   0  1
  [15] .shstrtab         STRTAB          00000000 020fe9 00009d 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
  L (link order), O (extra OS processing required), G (group), T (TLS),
  C (compressed), x (unknown), o (OS specific), E (exclude),
  y (purecode), p (processor specific)
```