

Unit 3 Embedded C

Lecture 3 Report

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

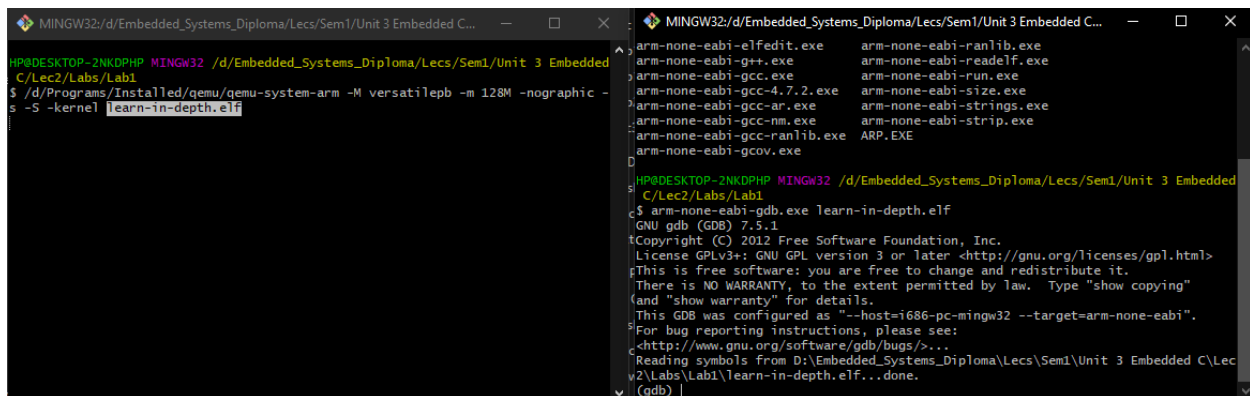
Part 2] Write BareMetal SW on STM32MPU ARM
CortexM3 (Lab2)

By: Yara Ashraf

Debug BareMetal SW on ARM VersatilePB (Lab1)

A] Debug via terminal (gdb)

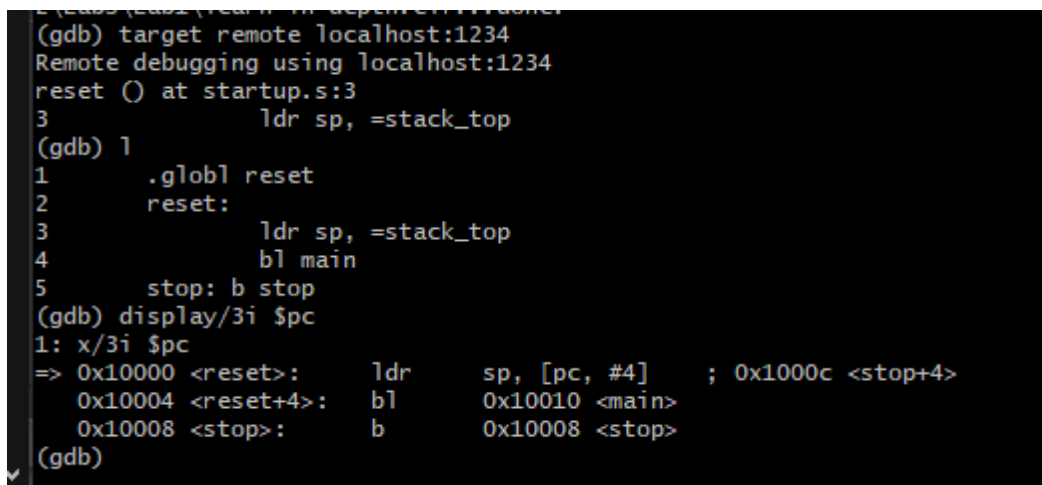
- Run QEMU and pass options -s -S to debug
- Call arm-none-eabi-gdb.exe



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

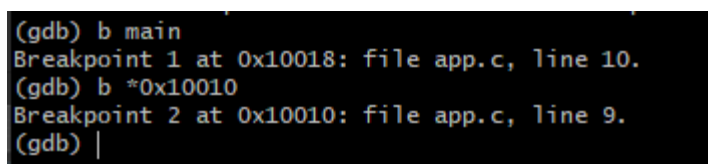
HP@DESKTOP-2NKDHP MINGW32 /d/Embedded_Systems_Diploma/Lecs/Sem1/Unit 3 Embedded C...
$ arm-none-eabi-gdb.exe learn-in-depth.elf
GNU gdb (GDB) 7.5.1
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you 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 "x86_64-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from D:\Embedded_Systems_Diploma\Lecs\Sem1\Unit 3 Embedded C\Lab1\2\Labs\Lab1\learn-in-depth.elf...done.
(gdb)
```

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



```
(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)
```

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



```
(gdb) b main
Breakpoint 1 at 0x10018: file app.c, line 10.
(gdb) b *0x10010
Breakpoint 2 at 0x10010: file app.c, line 9.
(gdb) |
```

- 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.
|
```

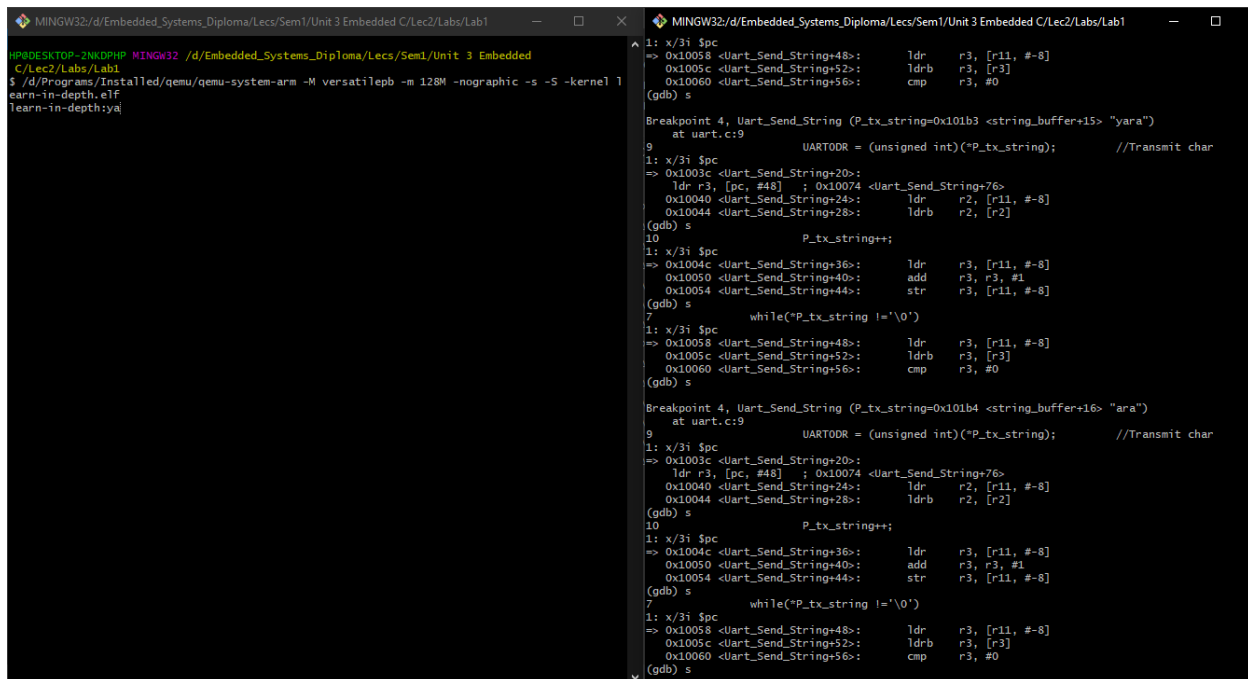
- Output on QEMU, string was sent via UART.

```
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)

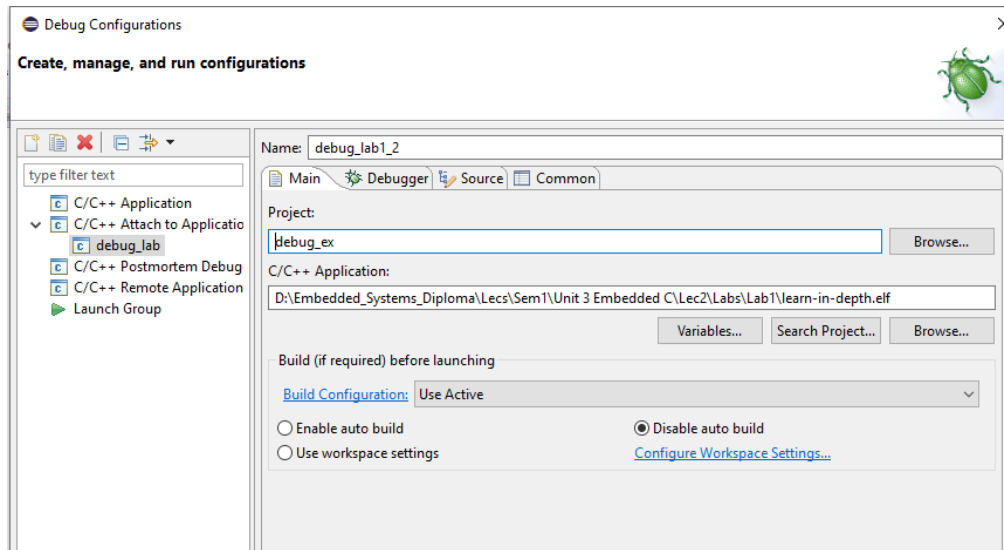


```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Lecs/Sem1/Unit 3 Embedded C/Lec2/Labs/Lab1
$ /d/Programs/Installed/qemu/qemu-system-arm -M versatilepb -m 128M -nographic -s -S -kernel 1
learn-in-depth.ya

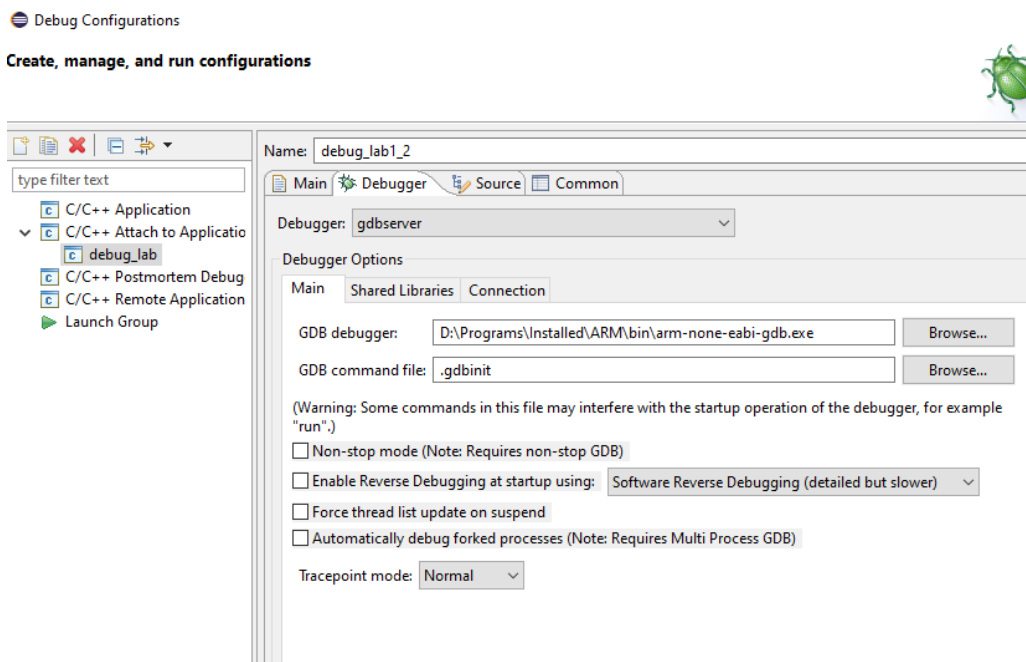
1: x/3i $pc
=> 0x10058 <Uart_Send_String+48>: ldr r3, [r11, #-8]
0x1005c <Uart_Send_String+52>: ldrb r3, [r3]
0x10060 <Uart_Send_String+56>: cmp r3, #0
(gdb) s
Breakpoint 4, Uart_Send_String (P_tx_string=0x101b3 <string_buffer+15> "yara")
at uart.c:9
9          UART0DR = (unsigned int)(P_tx_string); //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

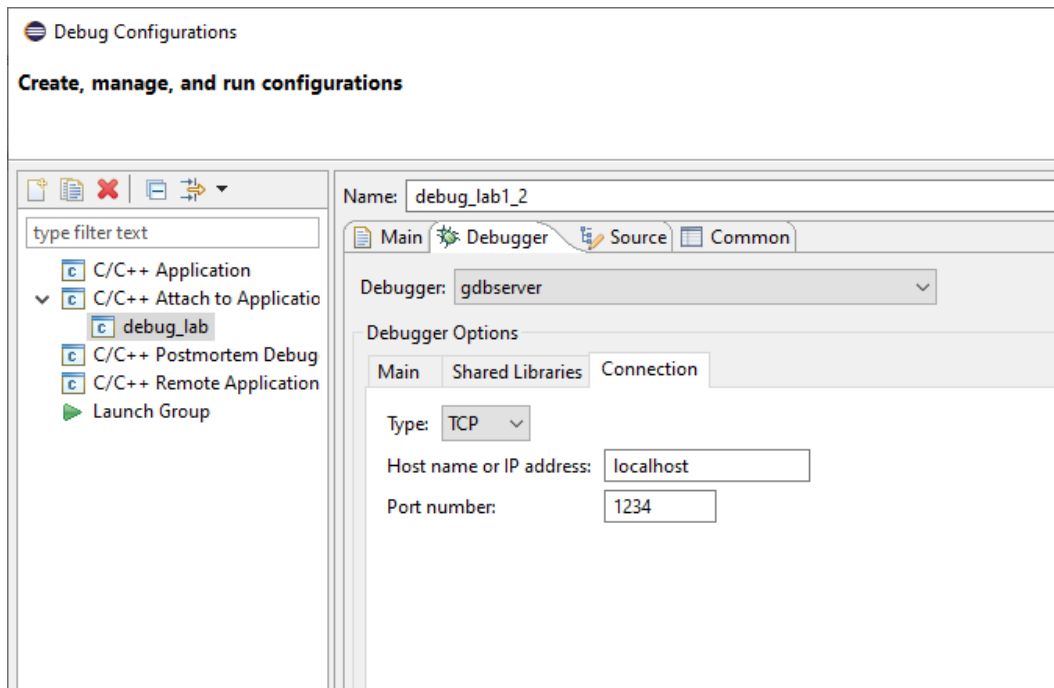
- Attach .elf file to debug project, disabling auto build for the project as we are concerned only with the .elf



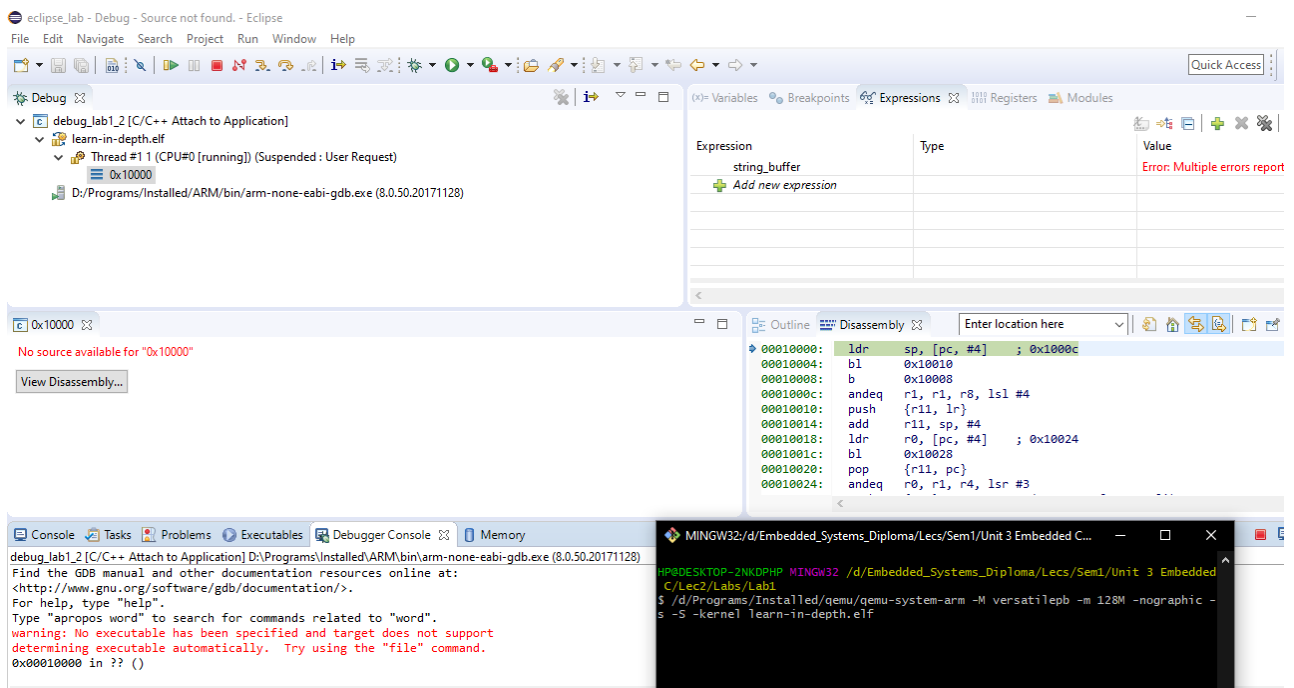
- Select gdbserver for the debugger



- 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

The screenshot shows the Eclipse IDE with the 'startup.s' file open. The code in the file is as follows:

```

1 .globl reset
2 reset:
3     ldr sp, =stack_top
4     bl main
5 stop: b stop
6

```

The debugger console at the bottom shows the following output:

```

debug:lab1_2 [C/C++ Attach to Application] D:\Programs\Installed\ARM\bin\arm-none-eabi-gdb.exe (8.0.50.20171128)
Type "apropos word" to search for commands related to "word".
warning: No executable has been specified and target does not support
determining executable automatically. Try using the "file" command.
0x00010000 in ?? ()
pwd
Working directory D:\Embedded_Systems_Diploma\Lecs\Sem1\Unit 3 Embedded C\Lec3\eclipse_debug.
file learn-in-depth.elf
A program is being debugged already.
Are you sure you want to change the file? (y or n) [answered Y; input not from terminal]
Reading symbols from learn-in-depth.elf...done.

```

- 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

The screenshot shows the Eclipse IDE with the 'app.c' file open. The code in the file is as follows:

```

1 #include "uart.h"
2 #define UARTDR (*(volatile unsigned int *)0x1010000)
3
4 void Uart_Send_String (unsigned char* P_tx_string)
5 {
6     //poll intill char null
7     while(*P_tx_string != '\0')
8     {
9         UARTDR = (unsigned int)*P_tx_string; //Transmit char
10        P_tx_string++;
11    }
12 }
13
14
15

```

A hover tooltip is shown over the variable `P_tx_string` in line 9, displaying the following information:

Expression	Type	Value
* P_tx_string	unsigned char*	0x10104 <string_buffer> "learn-in-...
0x- P_tx_string	unsigned char	108 T

The tooltip also shows the variable's details:

```

Name : P_tx_string
Detail:0x10104 <string_buffer> "learn-in-deptiyara"
Default:0x10104 <string_buffer> "learn-in-deptiyara"
Decimal:65536
Hex:0x10104
Binary:10000000110100100

```

The assembly view on the right shows the disassembly of the code, with the following instructions:

```

00010000: ldr    sp, [pc, #4] ; 0x1000c
00010004: bl     0x10010
00010008: b      0x10000
0001000c: andeq  r1, r1, r8, lsl #4
00010010: push  {r11, lr}
00010014: add    r11, sp, #4
00010018: ldr    r0, [pc, #4] ; 0x10024
0001001c: bl     0x10028
00010020: pop    {r11, pc}
00010024: andeq  r0, r1, r4, lsr #3
00010028: push  {r11}
0001002c: add    r11, sp, #0 ; (str r11, [sp, #-4]!)
00010030: sub    sp, sp, #12
00010034: str    r0, [r11, #-8]
00010038: b      0x10038
0001003c: ldr    r3, [pc, #4] ; 0x10074
00010040: ldr    r2, [r11, #-8]
00010044: ldrb   r2, [r2]
00010048: str    r2, [r3]
0001004c: ldr    r3, [r11, #-8]
00010050: add    r3, r3, #1
00010054: str    r3, [r11, #-8]
00010058: ldr    r3, [r11, #-8]
0001005c: ldrb   r3, [r3]
00010060: cmp    r3, #0

```

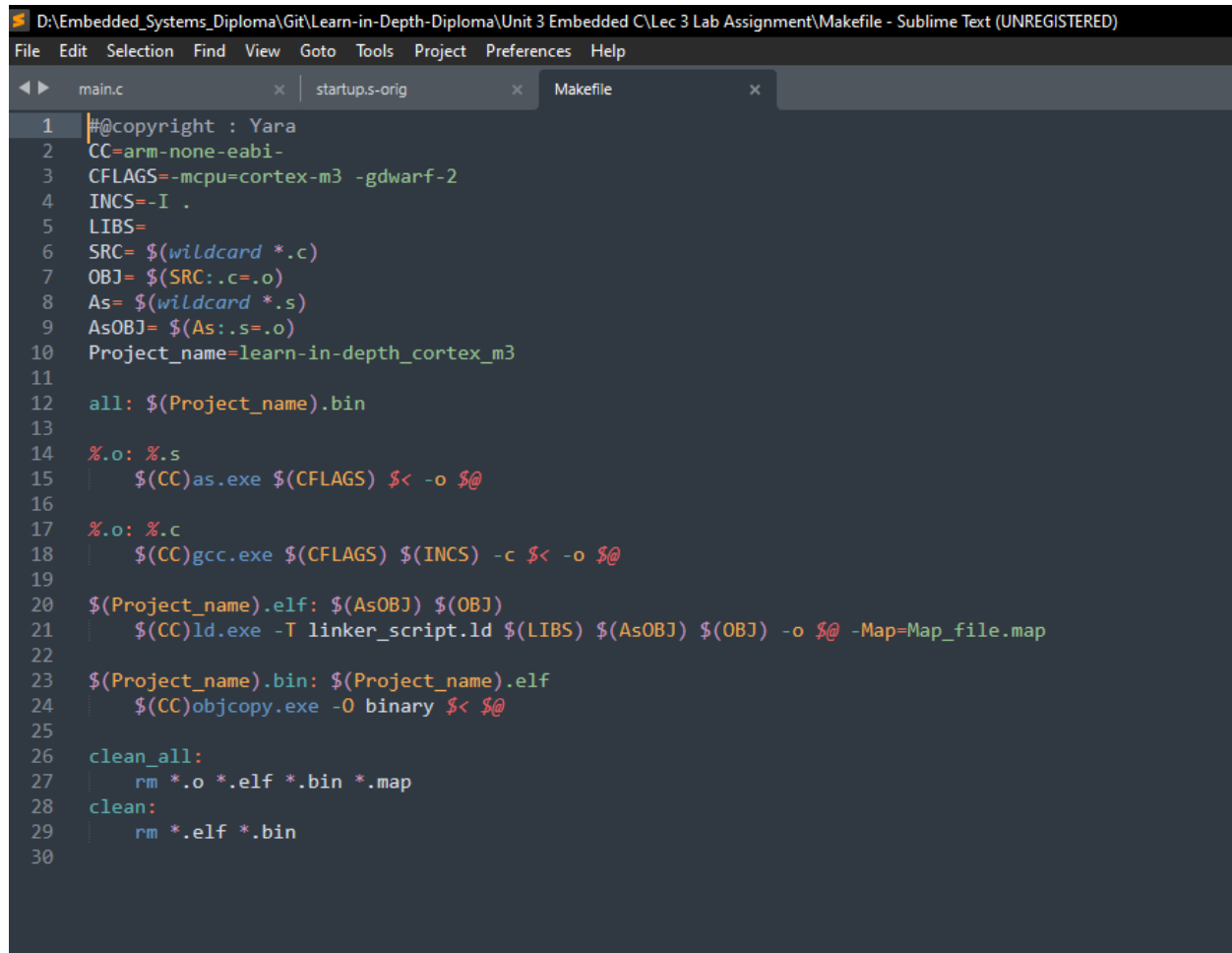
Writing a BareMetal SW on STM32MPU ARM CortexM3

Main.c

```
File Edit Selection Find View Goto Tools Project Preferences Help
startup.s-orig x main.c x
15 ~ opensource.org/licenses/bsd-3-clause
16 *
17 *****
18 */
19 #include "stdint.h"
20
21 typedef volatile unsigned int vint32_t;
22
23 #define RCC_BASE 0x40021000
24 #define PORTA_BASE 0x40010800
25
26 #define RCC_APB2ENR *(volatile uint32_t*)(RCC_BASE + 0x18)
27 #define GPIOA_CRH *(volatile uint32_t*)(PORTA_BASE + 0x04)
28 #define GPIOA_ODR *(volatile uint32_t*)(PORTA_BASE + 0x0C)
29
30 typedef union {
31     vint32_t all_fields;
32     struct {
33         vint32_t reserved:13;
34         vint32_t pin13:1;
35     };
36 }pin;
37
38 }R_ODR_t;
39
40 volatile R_ODR_t* R_ODR = (volatile R_ODR_t*)(PORTA_BASE + 0x0C);
41 unsigned char g_variables[3] = {1,2,3};
42 unsigned char const const_variables[3] = {1,2,3};
43 /*volatile unsigned char bss_var[3] ;*/
44
45 int main (void)
46 {
47
48     RCC_APB2ENR |= 1<<2; //enable RCC CLK to GPIOA, bit 2
49     GPIOA_CRH &= 0xFF0FFFFFFF; //clear 20 to 24
50     GPIOA_CRH |= 0x00200000; //write 2
51
52     int i;
53     while (1)
54     {
55
56         /*GPIOA_ODR |= 1<<13; //set bit 13
57         for(i=0;i<5000;i++);
58         GPIOA_ODR &= ~(1<<13); //clear bit 13
59         for(i=0;i<5000;i++);*/
60
61         R_ODR->pin.pin13=1;
62         for(i=0;i<5000;i++);
63         R_ODR->pin.pin13=0;
64         for(i=0;i<5000;i++);
65
66     }
67
68     return 0;
69 }
70
```


Makefile

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



```
1 #@copyright : Yara
2 CC=arm-none-eabi-
3 CFLAGS=-mcpu=cortex-m3 -gdwarf-2
4 INCS=-I .
5 LIBS=
6 SRC= $(wildcard *.c)
7 OBJ= $(SRC:.c=.o)
8 As= $(wildcard *.s)
9 AsOBJ= $(As:.s=.o)
10 Project_name=learn-in-depth_cortex_m3
11
12 all: $(Project_name).bin
13
14 %.o: %.s
15 | $(CC)as.exe $(CFLAGS) $< -o $@
16
17 %.o: %.c
18 | $(CC)gcc.exe $(CFLAGS) $(INCS) -c $< -o $@
19
20 $(Project_name).elf: $(AsOBJ) $(OBJ)
21 | $(CC)ld.exe -T linker_script.ld $(LIBS) $(AsOBJ) $(OBJ) -o $@ -Map=Map_file.map
22
23 $(Project_name).bin: $(Project_name).elf
24 | $(CC)objcopy.exe -O binary $< $@
25
26 clean_all:
27 | rm *.o *.elf *.bin *.map
28 clean:
29 | rm *.elf *.bin
30
```

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 x startup.s-orig x
1  /* startup_cortexM3.s
2  Eng.Yara
3  */
4
5  /*SRAM 0x20000000 */
6
7  .section .vectors
8
9  .word 0x20001000      /*stack top address*/
10 .word _reset
11 .word Vector_handler /* 2 NMI */
12 .word Vector_handler /* 3 Hard Fault */
13 .word Vector_handler /* 4 MM Fault */
14 .word Vector_handler /* 5 Bus Fault */
15 .word Vector_handler /* 6 Usage Fault */
16 .word Vector_handler /* 7 Reserved */
17 .word Vector_handler /* 8 Reserved */
18 .word Vector_handler /* 9 Reserved */
19 .word Vector_handler /* 10 Reserved */
20 .word Vector_handler /* 11 SV call */
21 .word Vector_handler /* 12 Debug reserved */
22 .word Vector_handler /* 13 Reserved */
23 .word Vector_handler /* 14 PendSV */
24 .word Vector_handler /* 15 SysTick */
25 .word Vector_handler /* 16 IRQ0 */
26 .word Vector_handler /* 17 IRQ1 */
27 .word Vector_handler /* 18 IRQ2 */
28 .word Vector_handler /* 19 ... */
29      /* on to IRQ67*/
30
31 .section .text
32
33 _reset:
34     bl main
35     b .
36
37 .thumb_func
38 Vector_handler:
39     b _reset
40
```

- 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-Diploma/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_cortex_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-Diploma/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 3 Embedded 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    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



```
1 //startup.c
2 //Eng. Yara
3 //Learn-in-depth
4
5 #include "stdint.h"
6 #define stack_top_SP 0x20001000
7 extern int main ();
8
9 void Reset_Handler (void)
10 {
11     main();
12 }
13 void NMI_Handler (void)
14 {
15     Reset_Handler();
16 }
17 void H_Fault_Handler (void)
18 {
19     Reset_Handler();
20 }
21 void MM_Fault_Handler (void)
22 {
23     Reset_Handler();
24 }
25 void Bus_Fault (void)
26 {
27     Reset_Handler();
28 }
29 void Usage_Fault_Handler (void)
30 {
31     Reset_Handler();
32 }
33
34
35 //define .vectors section
36 //addresses dependent on MCU vector table
37 uint32_t vectors[] __attribute__((section(".vectors"))) = {
38
39     stack_top_SP,
40     (uint32_t) &Reset_Handler,
41     (uint32_t) &NMI_Handler,
42     (uint32_t) &H_Fault_Handler,
43     (uint32_t) &MM_Fault_Handler,
44     (uint32_t) &Bus_Fault,
45     (uint32_t) &Usage_Fault_Handler
46
47 };
48
49
```

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

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embedded 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 -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-Diploma/Unit 3 Embedded 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 x linker_script.ld — Lec3\...Lab2 x startup.c x Makefile x

1 //startup.c
2 //Eng. Yara
3 //Learn-in-depth
4
5 #include "stdint.h"
6 #define stack_top_SP 0x20001000
7 extern int main ();
8
9 void Reset_Handler (void);
10 void Default_Handler (void)
11 {
12     Reset_Handler();
13 }
14
15 //passing weak attribute to compiler to be able to redefine handlers and alias so that it refers to Default_Handler symbol
16 void NMI_Handler (void) __attribute__((weak,alias("Default_Handler")));
17 void H_Fault_Handler (void) __attribute__((weak,alias("Default_Handler")));
18 void MM_Fault_Handler (void) __attribute__((weak,alias("Default_Handler")));
19 void Bus_Fault (void) __attribute__((weak,alias("Default_Handler")));
20 void Usage_Fault_Handler (void) __attribute__((weak,alias("Default_Handler")));
21
22
23 //define .vectors section
24 //addresses dependent on MCU vector table
25 uint32_t vectors[] __attribute__((section(".vectors"))) = {
26
27     stack_top_SP,
28     (uint32_t) &Reset_Handler,
29     (uint32_t) &NMI_Handler,
30     (uint32_t) &H_Fault_Handler,
31     (uint32_t) &MM_Fault_Handler,
32     (uint32_t) &Bus_Fault,
33     (uint32_t) &Usage_Fault_Handler
34
35 };
36
37 void Reset_Handler (void)
38 {
39     main();
40 }
41

```

- 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 Embedded 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 -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-Diploma/Unit 3 Embedded 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 — Lec3\...\Lab2 x | startup.c x | main.c x
17  /*
18  */
19  #include "stdint.h"
20
21  typedef volatile unsigned int vint32_t;
22
23  #define RCC_BASE 0x40021000
24  #define PORTA_BASE 0x40010800
25
26  #define RCC_APB2ENR  *(volatile uint32_t*)(RCC_BASE + 0x18)
27  #define GPIOA_CRH  *(volatile uint32_t*)(PORTA_BASE + 0x04)
28  #define GPIOA_ODR  *(volatile uint32_t*)(PORTA_BASE + 0x0C)
29
30  extern void NMI_Handler (void)
31  {
32
33  }
34  extern void Bus_Fault (void)
35  {
36
37  }
38
39  typedef union {
40      vint32_t all_fields;
41      struct {
42          vint32_t reserved:13;
43          vint32_t pin13:1;
44      }pin;
45  }R_ODR_t;
46
47  }R_ODR_t;
48
```


- NMI & Bus_Fault vector handlers have different symbols than the Default_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 Embedded 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 -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-Diploma/Unit 3 Embedded 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 x Makefile x main.c
6
7 MEMORY
8 {
9     flash (rx) : ORIGIN = 0x08000000 , LENGTH = 128K
10    sram (rwx) : ORIGIN = 0x20000000 , LENGTH = 20K
11 }
12
13 SECTIONS
14 {
15     .text :
16     {
17         *(.vectors*)
18         *(.text*)
19         *(.rodata*)
20         _E_text = . ;
21     } > flash
22     .data :
23     {
24         _S_data = . ;
25         *(.data)
26         _E_data = . ;
27     } > sram AT> flash
28     .bss :
29     {
30         _S_bss = . ;
31         *(.bss*)
32         _E_bss = . ;
33
34         . = . + 0x1000 ;
35         _stack_top = . ;
36     } > sram
37 }
```

- Map_file before Memory Alignment
- _E_data = 0x20000007

```

9  Linker script and memory map
10
11
12  .text          0x000000008000000  0x12b
13  *(.vectors*)
14  .vectors       0x000000008000000  0x1c startup.o
15                0x000000008000000  vectors
16  *(.text*)
17  .text          0x00000000800001c  0x7c main.o
18                0x00000000800001c  main
19  .text          0x000000008000098  0x90 startup.o
20                0x000000008000098  H_Fault_Handler
21                0x000000008000098  MM_Fault_Handler
22                0x000000008000098  Usage_Fault_Handler
23                0x000000008000098  Bus_Fault
24                0x000000008000098  Default_Handler
25                0x000000008000098  NMI_Handler
26                0x0000000080000a4  Reset_Handler
27  *(.rodata*)
28  .rodata        0x000000008000128  0x3 main.o
29                0x000000008000128  const_variables
30                0x00000000800012b  _E_text = .
31
32  .glue_7        0x00000000800012c  0x0
33  .glue_7        0x00000000800012c  0x0 linker stubs
34
35  .glue_7t       0x00000000800012c  0x0
36  .glue_7t       0x00000000800012c  0x0 linker stubs
37
38  .vfp11_veneer  0x00000000800012c  0x0
39  .vfp11_veneer  0x00000000800012c  0x0 linker stubs
40
41  .v4_bx         0x00000000800012c  0x0
42  .v4_bx         0x00000000800012c  0x0 linker stubs
43
44  .iplt          0x00000000800012c  0x0
45  .iplt          0x00000000800012c  0x0 main.o
46
47  .rel.dyn       0x00000000800012c  0x0
48  .rel.iplt      0x00000000800012c  0x0 main.o
49
50  .data          0x000000002000000  0x7 load address 0x00000000800012b
51                0x000000002000000  _S_data = .
52  *(.data)
53  .data          0x000000002000000  0x7 main.o
54                0x000000002000000  R_ODR
55                0x000000002000004  g_variables
56  .data          0x000000002000007  0x0 startup.o
57                0x000000002000007  _E_data = .
58
59  .igot.plt      0x000000002000008  0x0 load address 0x000000008000132
60  .igot.plt      0x000000002000008  0x0 main.o
61
62  .bss           0x000000002000007  0x100 load address 0x000000008000132
63                0x000000002000007  _S_bss = .
64  *(.bss*)
65  .bss           0x000000002000007  0x0 main.o
66  .bss           0x000000002000007  0x0 startup.o
67                0x000000002000007  _E_bss = .
68                0x000000002000107  . = ( . + 0x100)
69  *fill*         0x000000002000007  0x100
70                0x000000002000107  _stack_top = .
71  LOAD main.o

```

- Aligning .data section by 4 Bytes

```
linker_script.ld  Map_file.map  Makefile  main.c
1  /*
2  Learn-in-depth
3  Unit2_Lec3_Lab2
4  Eng. Yara Ashraf
5  */
6
7  MEMORY
8  {
9      flash (rx) : ORIGIN = 0x08000000 , LENGTH = 128K
10     sram (rwx) : ORIGIN = 0x20000000 , LENGTH = 20K
11 }
12
13 SECTIONS
14 {
15     .text :
16     {
17         *(.vectors*)
18         *(.text*)
19         *(.rodata*)
20         _E_text = . ;
21     } > flash
22     .data :
23     {
24         _S_data = . ;
25         *(.data)
26         . = ALIGN(4);
27         _E_data = . ;
28     } > sram AT> flash
29     .bss :
30     {
31         _S_bss = . ;
32         *(.bss*)
33         _E_bss = . ;
34         . = . + 0x1000 ;
35         _stack_top = . ;
36     } > sram
37 }
```

- Map_file after alignment, _E_data = 0x20000008

```
Linker script and memory map

.text          0x000000008000000 0x12b
*(.vectors*)
.vectors       0x000000008000000 0x1c startup.o
               0x000000008000000 vectors
*(.text*)
.text          0x00000000800001c 0x7c main.o
               0x00000000800001c main
.text          0x000000008000098 0x90 startup.o
               0x000000008000098 H_Fault_Handler
               0x000000008000098 MM_Fault_Handler
               0x000000008000098 Usage_Fault_Handler
               0x000000008000098 Bus_Fault
               0x000000008000098 Default_Handler
               0x000000008000098 NMI_Handler
               0x0000000080000a4 Reset_Handler
*(.rodata*)
.rodata        0x000000008000128 0x3 main.o
               0x000000008000128 const_variables
               0x00000000800012b _E_text = .
.glue_7        0x00000000800012c 0x0
.glue_7        0x00000000800012c 0x0 linker stubs
.glue_7t       0x00000000800012c 0x0
.glue_7t       0x00000000800012c 0x0 linker stubs
.vfp11_veneer  0x00000000800012c 0x0
.vfp11_veneer  0x00000000800012c 0x0 linker stubs
.v4_bx         0x00000000800012c 0x0
.v4_bx         0x00000000800012c 0x0 linker stubs
.iplt          0x00000000800012c 0x0
.iplt          0x00000000800012c 0x0 main.o
.rel.dyn       0x00000000800012c 0x0
.rel.iplt      0x00000000800012c 0x0 main.o
.data          0x000000020000000 0x8 load address 0x00000000800012b
               0x000000020000000 _S_data = .
*(.data)
.data          0x000000020000000 0x7 main.o
               0x000000020000000 R_ODR
               0x000000020000004 g_variables
.data          0x000000020000007 0x0 startup.o
               0x000000020000008 . = ALIGN (0x4)
*fill*         0x000000020000007 0x1
               0x000000020000008 _E_data = .
.igot.plt      0x000000020000008 0x0 load address 0x000000008000133
.igot.plt      0x000000020000008 0x0 main.o
.bss           0x000000020000008 0x1000 load address 0x000000008000133
               0x000000020000008 _S_bss = .
*(.bss*)
.bss           0x000000020000008 0x0 main.o
.bss           0x000000020000008 0x0 startup.o
               0x000000020000008 _E_bss = .
               0x000000020001008 . = (. + 0x1000)
*fill*         0x000000020000008 0x1000
               0x000000020001008 _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  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  0  1
[ 5] .debug_abbrev           PROGBITS           00000000  0202ee  0001a7  00   0  0  0  1
[ 6] .debug_loc              PROGBITS           00000000  020495  0000b4  00   0  0  0  1
[ 7] .debug_aranges          PROGBITS           00000000  020549  000040  00   0  0  0  1
[ 8] .debug_line             PROGBITS           00000000  020589  0003d7  00   0  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  0  1
[12] .debug_frame             PROGBITS           00000000  020c0c  00007c  00   0  0  0  4
[13] .symtab                  SYMTAB             00000000  020c88  000280  10   14 22  4
[14] .strtab                  STRTAB             00000000  020f08  0000e1  00   0  0  0  1
[15] .shstrtab                STRTAB             00000000  020fe9  00009d  00   0  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)
```