

A dark blue vertical bar on the left side of the slide. A blue arrow points to the right from the bar, containing the date.

4/17/2023

Lab1

Embedded C Lesson 2

Several thin, curved lines in dark blue and light grey originate from the bottom left corner and sweep upwards and to the right.

Mostafa Mohamed Edrees
LEARN-IN-DEPTH

Lab1

Required:

You have to create a bare metal Software to send a “learn-in-depth :< Your_Name >” using UART.

Physical Board:

VersatilePB

Processor:

Arm926ej-s

Name:

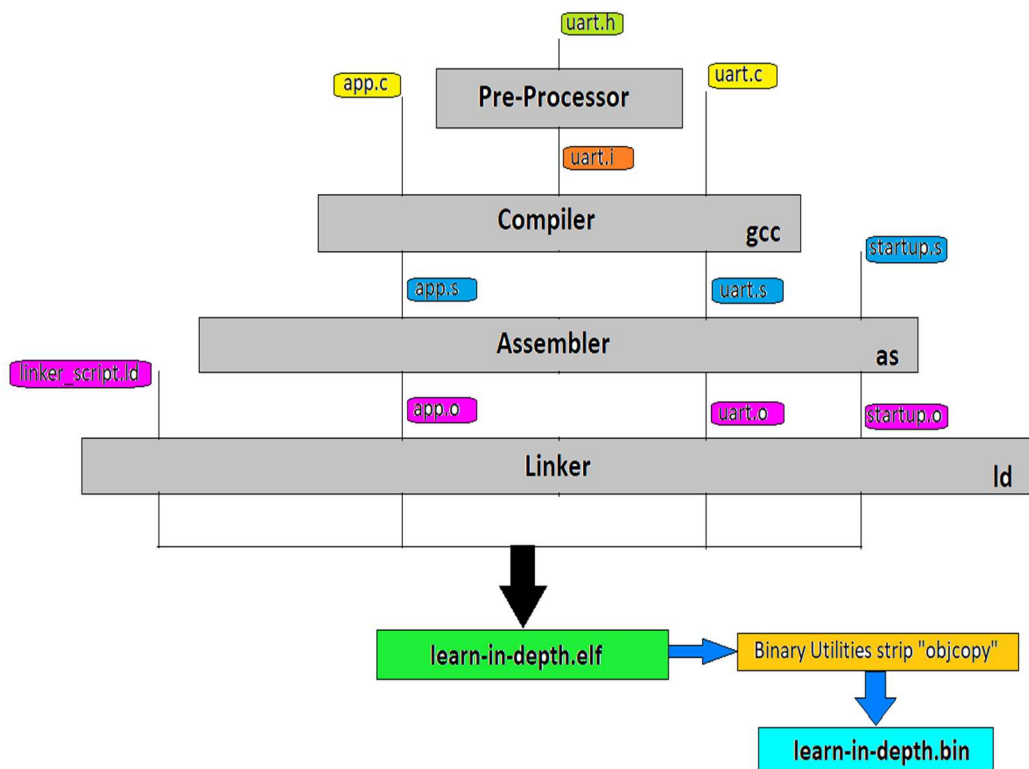
Mostafa Mohamed Edrees

Supervisor:

Eng. Keroles Shenouda

Steps:

- Create C code files. >> app.c , uart.c , uart.h
- Generate object files. >> app.o , uart.o
- Write Startup code file. >> startup.s
- Generate object file for startup. >> startup.o
- Write Linker Script file. >> linker_script.ld
- Get the executable file. >> learn-in-depth.elf
- Analyze the executable file.
- Get the binary file. >> learn-in-depth.bin
- Run the program in the QEMU Simulator.



Create C code files:

Uart.c

```
1  /**
2  ****
3  * @file      : uart.c
4  * @author    : Mostafa Edrees
5  * @brief     : lab1 in lesson2 in Embedded C
6  * @date      : 17/4/2023
7  * @board     : versatilePB physical board
8  ****
9  **/
10
11 #include "uart.h"
12 #include "Platform_types.h"
13
14 // Base Address of UART0:      0x101f1000
15 // Offset of Data Register(DR): 0x0
16 #define UART0DR *((volatile uint32_t *) (0x101f1000))
17
18 //P_tx_String >> Pointer to transmitting string
19 void UART0_Send_String(uint8_t * P_tx_String)
20 {
21     while(*P_tx_String != '\0') //loop to print all characters of the string
22     {
23         UART0DR = *P_tx_String; //send string to UART0 byte by byte
24         P_tx_String++; //next character
25     }
26 }
```

Uart.h

```
1  /**
2  ****
3  * @file      : uart.h
4  * @author    : Mostafa Edrees
5  * @brief     : lab1 in lesson2 in Embedded C
6  * @date      : 17/4/2023
7  * @board     : versatilePB physical board
8  ****
9  **/
10
11 #ifndef _UART_H_
12 #define _UART_H_
13
14 #include "Platform_types.h"
15
16 //UART0 API
17 void UART0_Send_String(uint8_t * P_tx_String);
18
19
20
21 #endif
```

App.c

```
1  /**
2  ****
3  * @file      : app.c
4  * @author    : Mostafa Edrees
5  * @brief     : lab1 in lesson2 in Embedded C
6  * @date      : 17/4/2023
7  * @board     : versatilePB physical board
8  ****
9  **/
10
11 #include "uart.h"
12 #include "Platform_types.h"
13
14 //String that will send to UART0
15 uint8_t String_Buffur[100] = "learn-in-depth:<Mostafa Mohamed Edrees>";
16
17 void main(void)
18 {
19     UART0_Send_String(String_Buffur);
20 }
21
```

Generate object files:

Processor: arm926ej-s

App.o

```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s app.c -o app.o

lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-objdump.exe -h app.o

app.o:          file format elf32-littlearm

Sections:
Idx Name              Size              VMA               LMA               File off        Algn
 0 .text              00000018          00000000          00000000          00000034        2**2
 1 .data              00000064          00000000          00000000          0000004c        2**2
 2 .bss               00000000          00000000          00000000          000000b0        2**0
 3 .comment           00000012          00000000          00000000          000000b0        2**0
 4 .ARM.attributes    00000032          00000000          00000000          000000c2        2**0
```

Sizeof .data section = 100 (64 in hexa) bytes because we have array of characters consist of 100 characters.

Uart.o

```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s uart.c -o uart.o

lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-objdump.exe -h uart.o

uart.o:          file format elf32-littlearm

Sections:
Idx Name              Size              VMA               LMA               File off        Algn
 0 .text              00000050          00000000          00000000          00000034        2**2
 1 .data              00000000          00000000          00000000          00000084        2**0
 2 .bss               00000000          00000000          00000000          00000084        2**0
 3 .comment           00000012          00000000          00000000          00000084        2**0
 4 .ARM.attributes    00000032          00000000          00000000          00000096        2**0
```

Command Line to Create object file:

```
$ arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s file.c -o file.o
```

Command Line to show sections of object file:

```
$ arm-none-eabi-objdump.exe -h file.o
```

Write Startup code file:

startup.s:

```
1  /**
2  * *****
3  * @file      : startup.s
4  * @author    : Mostafa Edrees
5  * @brief     : lab1 in lesson2 in Embedded C
6  * @date      : 17/4/2023
7  * @board     : versatilePB physical board
8  * *****
9  */
10
11 .globl reset
12
13 reset:
14     ldr sp, =x00011000 //before linker
15     bl main
16
17 stop:
18     b stop
```

Generate object file for startup

startup.o

```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o startup.o
startup.s: Assembler messages:
startup.s: Warning: end of file not at end of a line; newline inserted

lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ 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              0000000c  00000000  00000000  00000034  2**2
CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data              00000000  00000000  00000000  00000040  2**0
CONTENTS, ALLOC, LOAD, DATA
 2 .bss              00000000  00000000  00000000  00000040  2**0
ALLOC
 3 .ARM.attributes    00000022  00000000  00000000  00000040  2**0
CONTENTS, READONLY
```


Disassembly of object files:

Addresses are virtual not SOC physical addresses.

App.s

```
1
2 app.o: file format elf32-littlearm
3
4
5 Disassembly of section .text:
6
7 00000000 <main>:
8 0: e92d4800 push {fp, lr}
9 4: e28db004 add fp, sp, #4
10 8: e59f0004 ldr r0, [pc, #4] ; 14 <main+0x14>
11 c: ebfffffe bl 0 <UART0_Send_String>
12 10: e8bd8800 pop {fp, pc}
13 14: 00000000 andeq r0, r0, r0
14
15 Disassembly of section .data:
16
17 00000000 <String_Buffur>:
18 0: 72e1c5ec rsbvc r6, r1, #108, 10 ; 0x1b000000
19 4: ee692d6e cdpvs 13, 6, cr2, cr9, cr14, {3}
20 8: 7065642d rsbvc r6, r5, sp, lsr #8
21 c: 3c3a6874 ldccc 8, cr6, [s1], #-464 ; 0xfffffe30
22 10: 7473cf4d ldrtbvc r6, [r3], #-3917
23 14: 20616681 rsbcs r6, r1, r1, ror #12 ; 0xf4d
24 18: 6168ef4d cmnvs r8, sp, asr #30
25 1c: 2064656d rsbcs r6, r4, sp, ror #10
26 20: 65726445 ldrtbvs r6, [r2, #-1093]! ; 0x445
27 24: 003e7365 eorseq r7, lr, r5, ror #6
28 ...
29
30 Disassembly of section .comment:
31
32 00000000 <.comment>:
33 0: 43434700 movtmi r4, #14080 ; 0x3700
34 4: 4728203a ; <UNDEFINED> instruction: 0x4728203a
35 8: 2029554e eorcs r5, r9, lr, asr #10
36 c: 2e372e34 mrrcs 14, 1, r2, cr7, cr4, {1}
37 10: Address 0x00000010 is out of bounds.
38
39 Disassembly of section .ARM.attributes:
40
41 00000000 <.ARM.attributes>:
42
```

Uart.s

```
1
2 uart.o: file format elf32-littlearm
3
4
5 Disassembly of section .text:
6
7 00000000 <UART0_Send_String>:
8 0: e52db004 push {fp} ; (str fp, [sp, #-4]!)
9 4: e28db000 add fp, sp, #0
10 8: e24dd00c sub sp, sp, #12
11 c: e50b0008 str r0, [fp, #-8]
12 10: ea000006 b 30 <UART0_Send_String+0x30>
13 14: e59f3030 ldr r3, [pc, #48] ; 4c <UART0_Send_String+0x4c>
14 18: e51b2008 ldr r2, [fp, #-8]
15 1c: e5d22000 ldrtb r2, [r2]
16 20: e5832000 str r2, [r3]
17 24: e51b3008 ldr r3, [fp, #-8]
18 28: e2833001 add r3, r3, #1
19 2c: e50b3008 str r3, [fp, #-8]
20 30: e51b3008 ldr r3, [fp, #-8]
21 34: e5d33000 ldrtb r3, [r3]
22 38: e3530000 cmp r3, #0
23 3c: 1afffff4 bne 14 <UART0_Send_String+0x14>
24 40: e28bd000 add sp, fp, #0
25 44: e8bd0800 ldmfd sp!, {fp}
26 48: e12ffffe bx lr
27 4c: 101ff100 andsne r1, pc, r0
28
29 Disassembly of section .comment:
30
31 00000000 <.comment>:
32 0: 43434700 movtmi r4, #14080 ; 0x3700
33 4: 4728203a ; <UNDEFINED> instruction: 0x4728203a
34 8: 2029554e eorcs r5, r9, lr, asr #10
35 c: 2e372e34 mrrcs 14, 1, r2, cr7, cr4, {1}
36 10: Address 0x00000010 is out of bounds.
37
38 Disassembly of section .ARM.attributes:
39
40 00000000 <.ARM.attributes>:
41 0: 00003141 andeq r3, r0, r1, asr #2
42
```

Command Line to create disassembly file of object file:

```
$ arm-none-eabi-objdump.exe -D file.o >> file.s
```

Write Linker Script file:

Linker_script.ld

```
1  /**
2  *
3  * @file      : linker_script.ld
4  * @author    : Mostafa Edrees
5  * @brief     : lab1 in lesson2 in Embedded C
6  * @date      : 17/4/2023
7  * @board     : versatilePB physical board
8  *
9  */
10
11 ENTRY(reset)
12
13 MEMORY
14 {
15     Mem (rwx) : ORIGIN = 0x00000000 , LENGTH = 64M
16 }
17
18 SECTIONS
19 {
20     . = 0x10000;
21
22     .startup . :
23     {
24         startup.o(.text)
25     } > Mem
26     .text :
27     {
28         *(.text)
29     } > Mem
30     .data :
31     {
32         *(.data)
33     } > Mem
34     .bss :
35     {
36         *(.bss)
37     } > Mem
38
39     . = . + 0x1000; /* 1000 >> 4KB for stack */
40     stack_top = .;
41 }
42
```

After linker_script the addresses will be physical with SOC

Get the executable file:

```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_Embedde
d_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-ld.exe -T linker_script.ld app.o uart.o startup.o -o learn-in-depth.elf

lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_Embedde
d_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-objdump.exe -h learn-in-depth.elf

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

Sections:
Idx Name          Size      VMA      LMA      File off  Algn
 0  .startup ✓      00000010  00010000  00010000  00008000  2**2
    CONTENTS, ALLOC, LOAD, READONLY, CODE
 1  .text ✓         00000068  00010010  00010010  00008010  2**2
    CONTENTS, ALLOC, LOAD, READONLY, CODE
 2  .data →         00000064  00010078  00010078  00008078  2**2
    CONTENTS, ALLOC, LOAD, DATA
 3  .ARM.attributes 0000002e  00000000  00000000  000080dc  2**0
    CONTENTS, READONLY
 4  .comment        00000011  00000000  00000000  0000810a  2**0
    CONTENTS, READONLY

lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_Embedde
d_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$
```

Command Line to link object files and get the executable:

```
$ arm-none-eabi-ld.exe -T linker_script.ld app.o uart.o startup.o -o learn-in-depth.elf
```


We put address of stack_top at startup.s to put this address in stack pointer register (PC).

Startup.s

```
1  /**
2  ****
3  * @file      : startup.s
4  * @author    : Mostafa Edrees
5  * @brief     : lab1 in lesson2 in Embedded C
6  * @date      : 17/4/2023
7  * @board     : versatilePB physical board
8  ****
9  **/
10
11 .globl reset
12
13 reset:
14     ldr sp, =stack_top //after linker
15     bl main
16
17 stop:
18     b stop
```

Let's create the disassembly of learn-in-depth.elf

```
1  learn-in-depth.elf:      file format elf32-littlearm
2
3
4
5  Disassembly of section .startup:
6
7  00010000 <reset>:
8      10000: e59fd004    ldr sp, [pc, #4]      ; 1000c <stop+0x4>
9      10004: eb000001    bl 10010 <main>
10
11 00010008 <stop>:
12      10008: eaffffff    b 10008 <stop>
13      1000c: 000110dc    ldrdeq r1, [r1], -ip
14
15 Disassembly of section .text:
16
17 00010010 <main>:
18      10010: e92d4800    push {fp, lr}
19      10014: e28db004    add fp, sp, #4
20      10018: e59f0004    ldr r0, [pc, #4]      ; 10024 <main+0x14>
21      1001c: eb000001    bl 10028 <UART0_Send_String>
22      10020: e8bd8800    pop {fp, pc}
23      10024: 00010078    andeq r0, r1, r8, ror r0
24
25 00010028 <UART0_Send_String>:
26      10028: e52db004    push {fp}             ; (str fp, [sp, #-4]!)
27      1002c: e28db000    add fp, sp, #0
28      10030: e24dd00c    sub sp, sp, #12
29      10034: e50b0008    str r0, [fp, #-8]
30      10038: ea000006    b 10058 <UART0_Send_String+0x30>
31      1003c: e59f3030    ldr r3, [pc, #48]      ; 10074 <UART0_Send_String+0x4c>
32      10040: e51b2008    ldr r2, [fp, #-8]
33      10044: e5d22000    ldrb r2, [r2]
34      10048: e5832000    str r2, [r3]
35      1004c: e51b3008    ldr r3, [fp, #-8]
36      10050: e2833001    add r3, r3, #1
37      10054: e50b3008    str r3, [fp, #-8]
38      10058: e51b3008    ldr r3, [fp, #-8]
39      1005c: e5d33000    ldrb r3, [r3]
40      10060: e3530000    cmp r3, #0
41      10064: 1affffff    bne 1003c <UART0_Send_String+0x14>
42      10068: e28bd000    add sp, fp, #0
43      1006c: e8bd0800    ldmfd sp, {fp}
44      10070: e12ffffe    bx lr
45      10074: 101f1000    andsne r1, pc, r0
46
47
48 Disassembly of section .data:
49
50 00010078 <String_Buffur>:
51      10078: 7261656e    rsbvc r6, r1, #108, 10 ; 0x1b000000
52      1007c: 6e692d6e    cdpvs 13, 6, cr2, cr9, cr14, {3}
53      10080: 706b642d    rsbvc r6, r5, sp, lsr #8
54      10084: 3c3a6874    ldccc 8, cr6, [r1], #-464 ; 0xfffffe30
55      10088: 74736f4d    ldrbtvc r6, [r3], #-3917 ; 0xf4d
56      1008c: 20616661    rsbcs r6, r1, r1, ror #12
57      10090: 61686f4d    cmnvs r8, sp, asr #30
58      10094: 2064656d    rsbcs r6, r4, sp, ror #10
59      10098: 65726445    ldrbvs r6, [r2, #-1093]! ; 0x445
60      1009c: 003e7365    eorseq r7, lr, r5, ror #6
61      ...
```

See Symbols in all object files:

app.o

```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-nm.exe app.o
00000000 T main
00000000 D String_Buffur
          U UART0_Send_String
```

uart.o

```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-nm.exe uart.o
00000000 T UART0_Send_String
```

startup.o

```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-nm.exe startup.o
          U main
00000000 T reset
          U stack_top
00000008 t stop
```

learn-in-depth.elf

```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-nm.exe learn-in-depth.elf
00010010 T main
00010000 T reset
000110dc D stack_top
00010008 t stop
00010078 D String_Buffur
00010028 T UART0_Send_String
```

Command Line to get symbols of an object file:

```
$ arm-none-eabi-nm.exe file.o
```

See map file of learn-in-depth.elf:

```

1
2 Memory Configuration
3
4 Name          Origin          Length          Attributes
5 Mem ✓         0x00000000      0x04000000      xrw ✓
6 *default*     0x00000000      0xffffffff
7
8 Linker script and memory map
9
10              0x00010000      . = 0x10000
11
12 .startup      0x00010000      0x10
13 .startup.o(.text)
14 .text        0x00010000      0x10 startup.o
15              0x00010000      reset
16
17 .text        0x00010010      0x68
18 *(.text)
19 .text        0x00010010      0x18 app.o
20              0x00010010      main
21 .text        0x00010028      0x50 uart.o
22              0x00010028      UART0_Send_String
23
24 .data        0x00010078      0x64
25 .data        0x00010078      0x0 startup.o
26 .data        0x00010078      0x64 app.o
27              0x00010078      String_Buffur
28 .data        0x000100dc      0x0 uart.o
29
30 .igot.plt    0x000100dc      0x0
31 .igot.plt    0x00000000      0x0 startup.o
32
33 .glue_7      0x000100dc      0x0
34 .glue_7      0x00000000      0x0 linker stubs
35
36 .glue_7t     0x000100dc      0x0
37 .glue_7t     0x00000000      0x0 linker stubs
38
39 .vfp11_vneer 0x000100dc      0x0
40 .vfp11_vneer 0x00000000      0x0 linker stubs
41
42 .v4_bx       0x000100dc      0x0
43 .v4_bx       0x00000000      0x0 linker stubs
44
45 .iplt        0x000100dc      0x0
46 .iplt        0x00000000      0x0 startup.o
47
48 .rel.dyn     0x000100dc      0x0
49 .rel.iplt    0x00000000      0x0 startup.o
50
51 .date        0x000100dc      0x0
52 *(.date)
53
54 .bss         0x000100dc      0x0
55 *(.bss)
56 .bss         0x000100dc      0x0 startup.o
57 .bss         0x000100dc      0x0 app.o
58 .bss         0x000100dc      0x0 uart.o
59              0x000110dc      . = (. + 0x1000)
60              0x000110dc      stack_top = .
61 LOAD app.o
62 LOAD uart.o
63 LOAD startup.o
64 OUTPUT(learn-in-depth.elf elf32-littlearm)
65
66 .ARM.attributes 0x00000000      0x2e
67
68 .ARM.attributes 0x00000000      0x22 startup.o
69
70 .ARM.attributes 0x00000022      0x32 app.o
71
72 .ARM.attributes 0x00000054      0x32 uart.o
73
74
75 .comment      0x00000000      0x11
76 .comment      0x00000000      0x11 app.o
77              0x00000000      0x12 (size before relaxing)
78 .comment      0x00000000      0x12 uart.o

```

Command Line to get map file of an object file:

```
$ arm-none-eabi-ld.exe -T linker_script.ld -Map=map_file.map app.o uart.o startup.o -o learn-in-depth.elf
```

Check the entry point:

```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_Embedded_System_Online_Diploma
/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-readelf.exe -a learn-in-depth.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: 0x10000
  Start of program headers: 52 (bytes into file)
  Start of section headers: 33124 (bytes into file)
  Flags: 0x5000002, has entry point, Version5 EABI
  Size of this header: 52 (bytes)
  Size of program headers: 32 (bytes)
  Number of program headers: 1
  Size of section headers: 40 (bytes)
  Number of section headers: 9
  Section header string table index: 6

Section Headers:
 [Nr] Name                Type              Addr      Off      Size    ES Flg Lk Inf Al
 [ 0]                      NULL              00000000  000000  000000  00  0  0  0
 [ 1] .startup                PROGBITS          00010000  008000  000010  00  AX  0  0  4
 [ 2] .text                  PROGBITS          00010010  008010  000068  00  AX  0  0  4
 [ 3] .data                   PROGBITS          00010078  008078  000064  00  WA  0  0  4
 [ 4] .ARM.attributes        ARM_ATTRIBUTES    00000000  0080dc  00002e  00  0  0  1
 [ 5] .comment                PROGBITS          00000000  00810a  000011  01  MS  0  0  1
 [ 6] .shstrtab               STRTAB            00000000  00811b  000049  00  0  0  1
 [ 7] .symtab                 SYMTAB            00000000  0082cc  000170  10  8 18  4
 [ 8] .strtab                 STRTAB            00000000  00843c  000058  00  0  0  1

Key to Flags:
 W (write), A (alloc), X (execute), M (merge), S (strings)
 I (info), L (link order), G (group), T (TLS), E (exclude), x (unknown)
 O (extra OS processing required) o (OS specific), p (processor specific)
```

Get the binary file:

```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_Embedded_System_Online_Diploma
/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
```

Run the program in the QEMU Simulator:

```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_Embedded_System_Online_Diploma
/Embedded C/Lesson 2/Lab1 (master)
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin
learn-in-depth:<Mostafa Mohamed Edrees>
```

DONE ✓