4/17/2023

Lab1

Embedded C Lesson 2

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

Without debug information & Makefile.

Name:

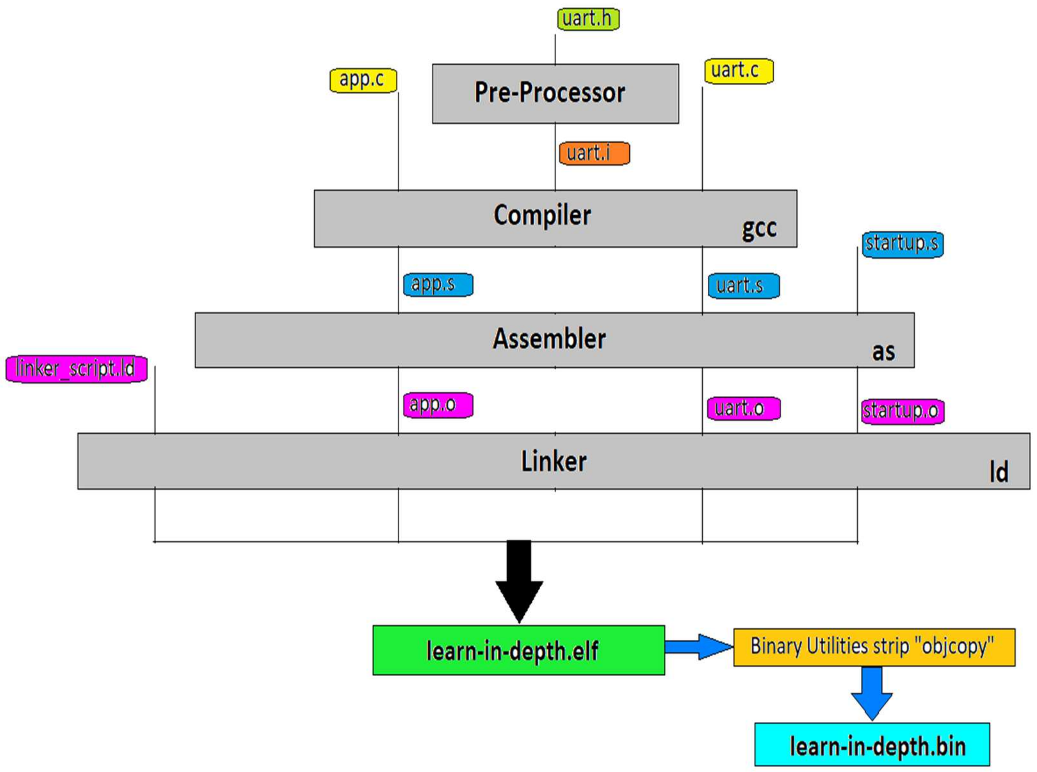
Mostafa Mohamed Edrees

Supervisor:

Eng. Keroles Shenouda

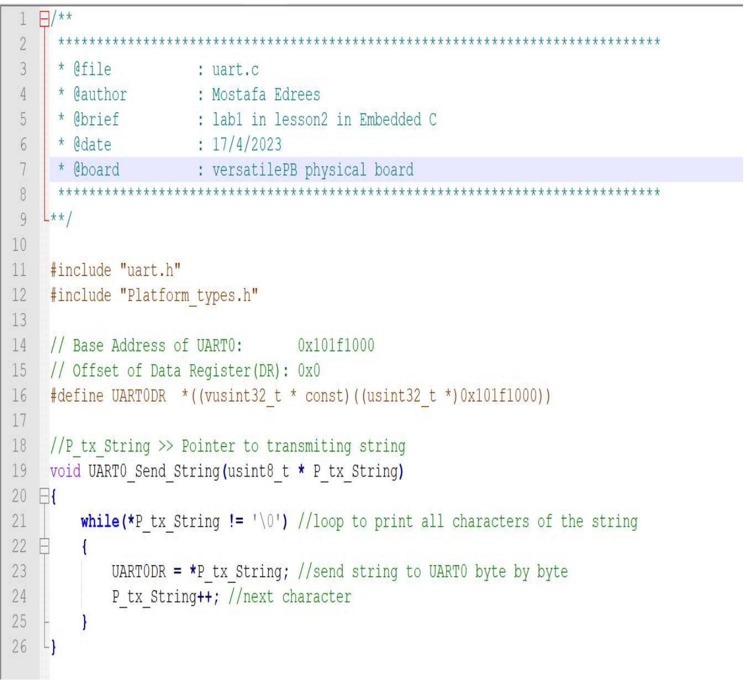
Steps:

- 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.



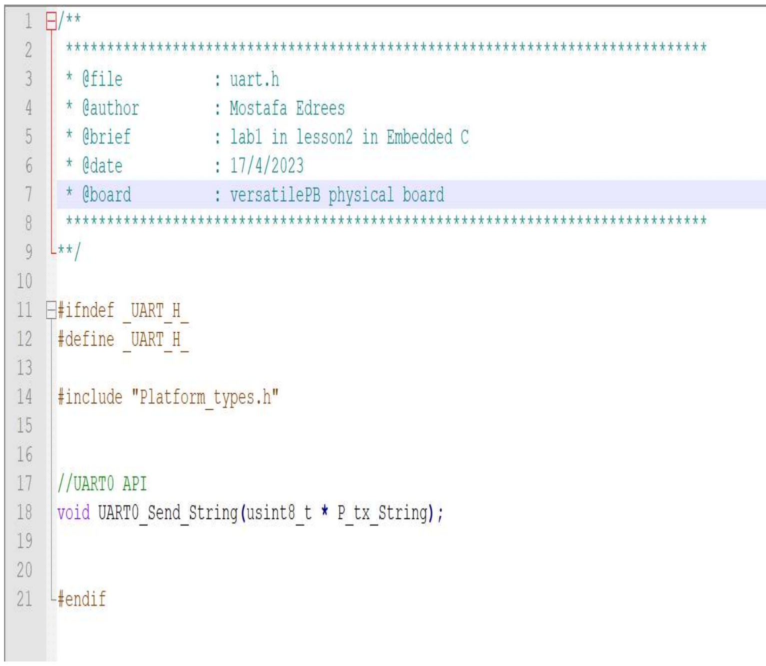
Create C code files:

Uart.c



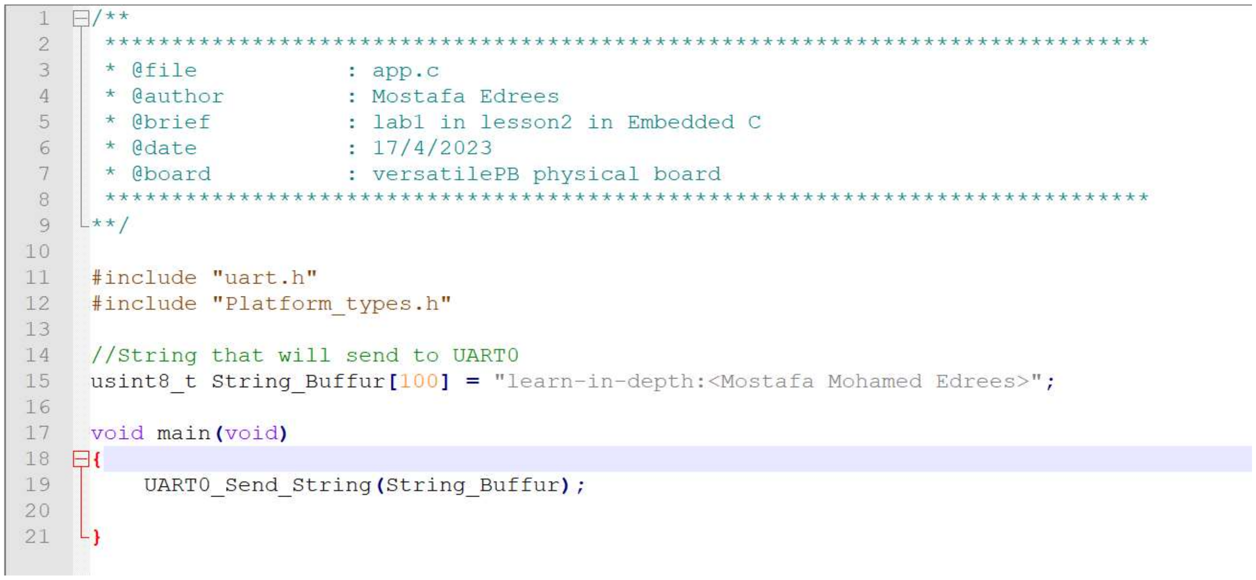
```c
/**
 ******************************************************************************
 * @file          : uart.c
 * @author        : Mostafa Edrees
 * @brief         : lab1 in lesson2 in Embedded C
 * @date          : 17/4/2023
 * @board         : versatilePB physical board
 ******************************************************************************
**/

#include "uart.h"
#include "Platform_types.h"

// Base Address of UART0:      0x101f1000
// Offset of Data Register(DR): 0x0
#define UART0DR  *((vusint32_t * const)((usint32_t *)0x101f1000))

//P_tx_String >> Pointer to transmiting string
void UART0_Send_String(usint8_t * P_tx_String)
{
    while(*P_tx_String != '\0') //loop to print all characters of the string
    {
        UART0DR = *P_tx_String; //send string to UART0 byte by byte
        P_tx_String++; //next character
    }
}
```

Uart.h



```c
/**
 ******************************************************************************
 * @file          : uart.h
 * @author        : Mostafa Edrees
 * @brief         : lab1 in lesson2 in Embedded C
 * @date          : 17/4/2023
 * @board         : versatilePB physical board
 ******************************************************************************
**/

#ifndef _UART_H_
#define _UART_H_

#include "Platform_types.h"


//UART0 API
void UART0_Send_String(usint8_t * P_tx_String);


#endif
```

App.c

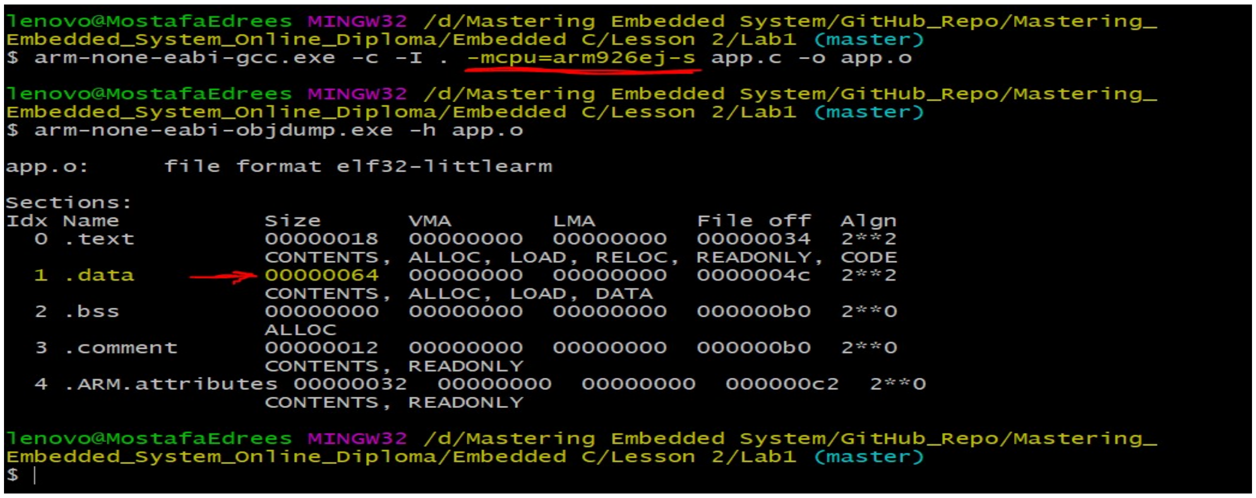


```c
/**
 ******************************************************************************
 * @file          : app.c
 * @author        : Mostafa Edrees
 * @brief         : lab1 in lesson2 in Embedded C
 * @date          : 17/4/2023
 * @board         : versatilePB physical board
 ******************************************************************************
**/

#include "uart.h"
#include "Platform_types.h"

//String that will send to UART0
usint8_t String_Buffur[100]  = "learn-in-depth:<Mostafa Mohamed Edrees>";

void main(void)
{
    UART0_Send_String(String_Buffur);

}
```

Generate object files:

Processor: arm926ej-s

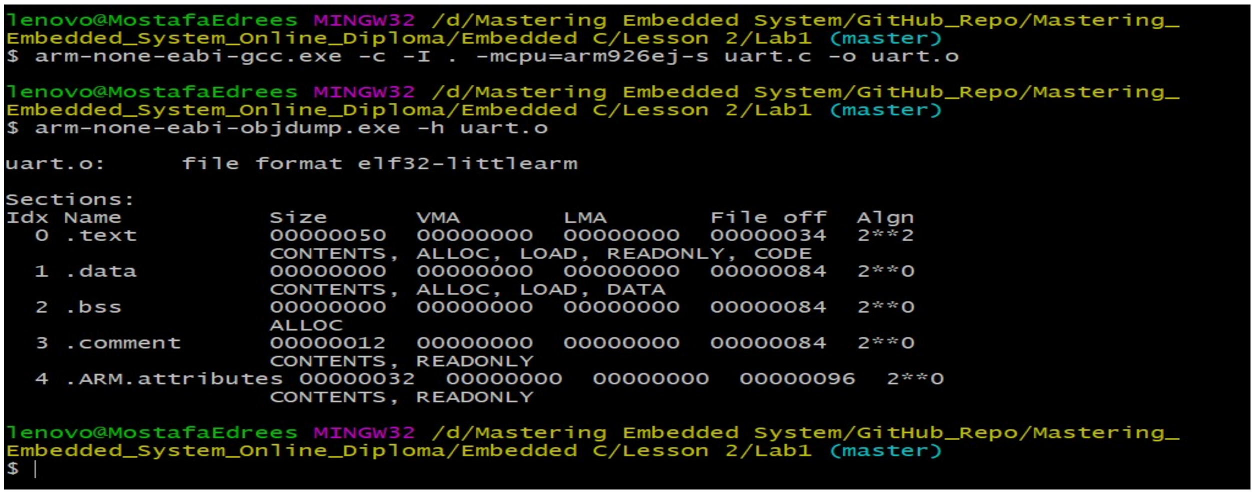
App.o



```
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
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000064  00000000  00000000  0000004c  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b0  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  000000b0  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000032  00000000  00000000  000000c2  2**0
                  CONTENTS, READONLY

lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ |
```

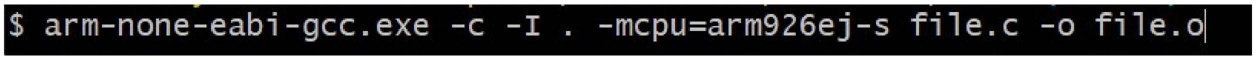
Sizeof .data section = 100 (64 in hexa) bytes because we have array of characters consist of 100 characters.

Uart.o



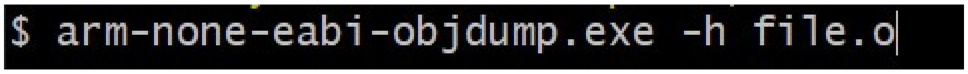
```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s uart.c -o uart.o

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
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000084  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000084  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  00000084  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000032  00000000  00000000  00000096  2**0
                  CONTENTS, READONLY

lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ |
```

Command Line to Create object file:



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

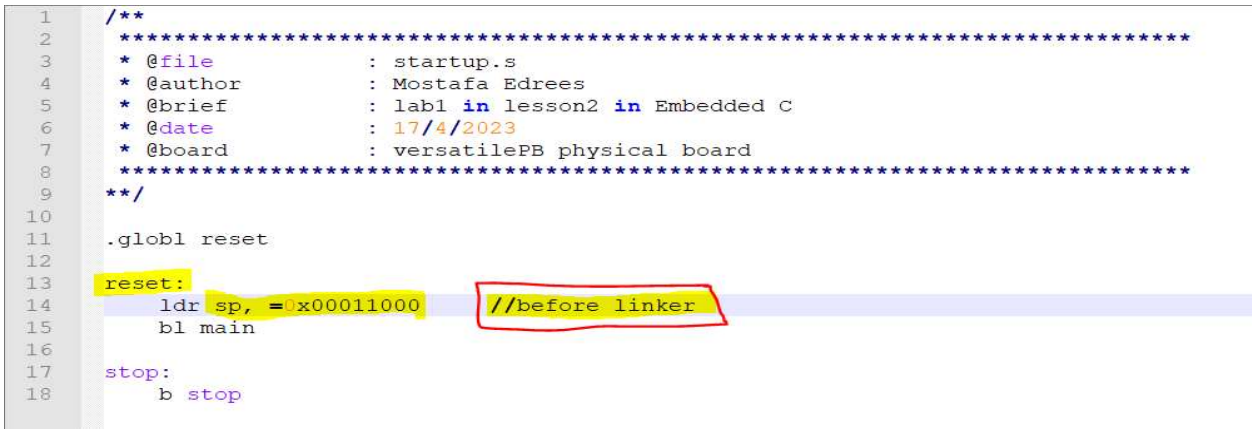
Command Line to show sections of object file:



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

Write Startup code file:

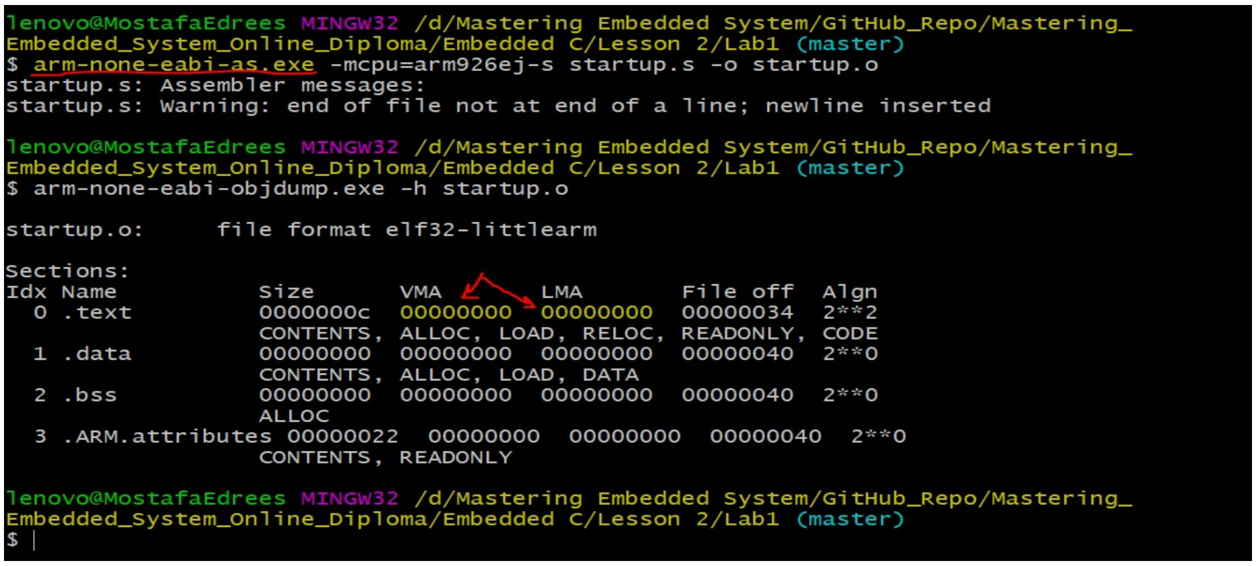
startup.s:



```
/**
 *****************************************************************************
 * @file                : startup.s
 * @author              : Mostafa Edrees
 * @brief               : lab1 in lesson2 in Embedded C
 * @date                : 17/4/2023
 * @board               : versatilePB physical board
 *****************************************************************************
**/

.globl reset

reset:
    ldr sp, =0x00011000         //before linker
    bl main

stop:
    b stop
```

Generate object file for startup

startup.o

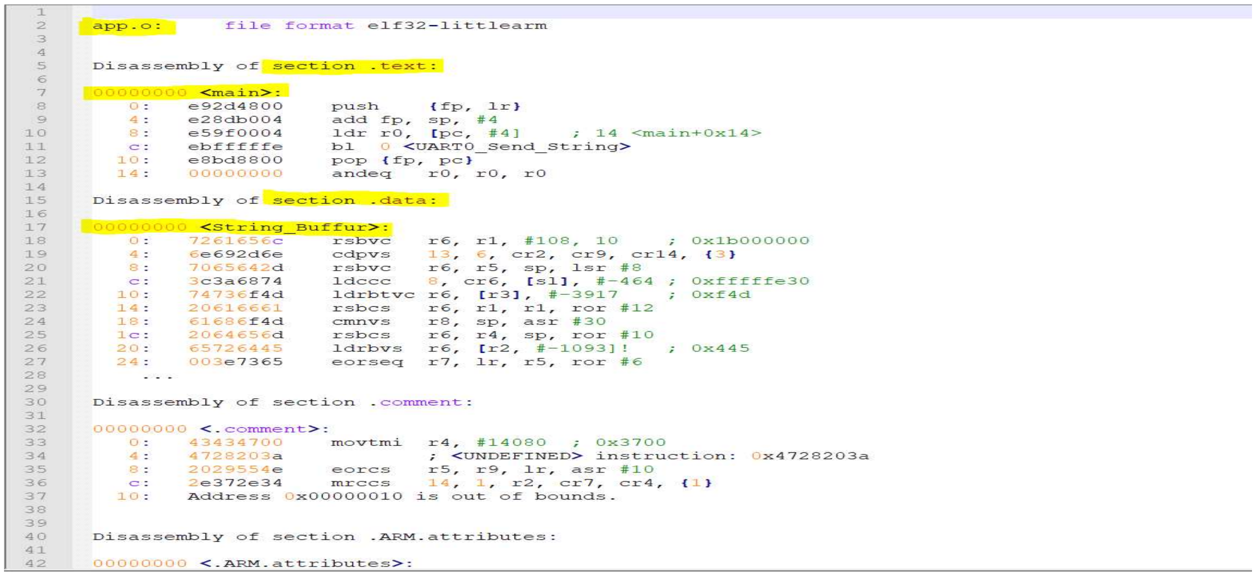


```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o startup.o
startup.s: Assembler messages:
startup.s: Warning: end of file not at end of a line; newline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

lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ |
```

Disassembly of object files:

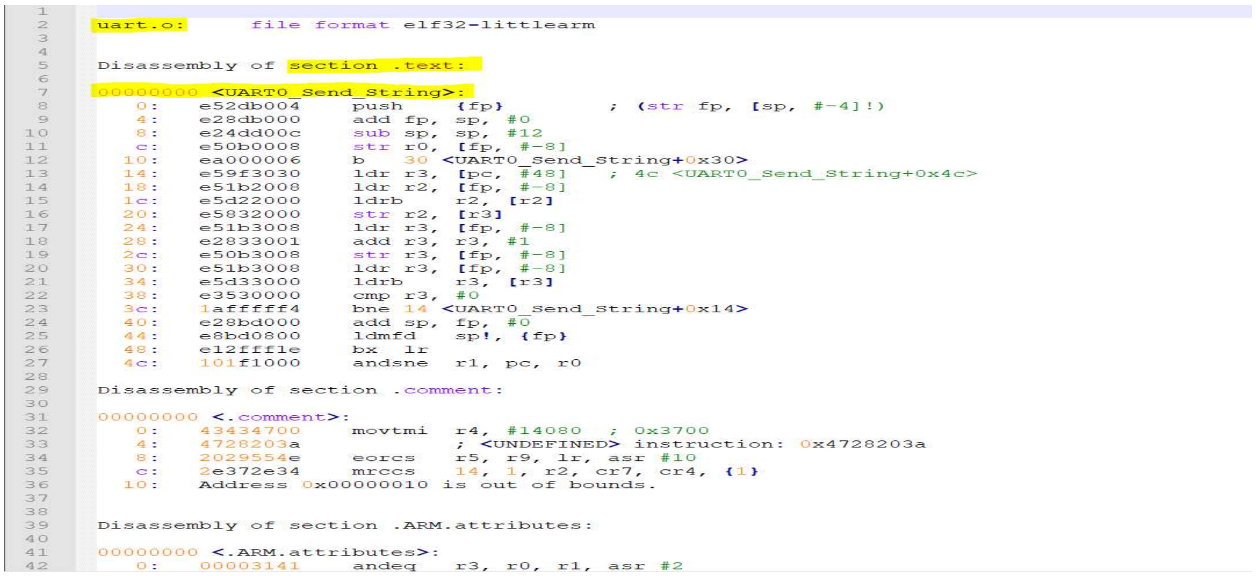
Addresses are virtual not SOC physical addresses.

App.s



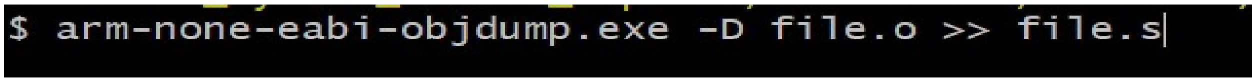
```
app.o:     file format elf32-littlearm


Disassembly of section .text:

00000000 <main>:
   0:	e92d4800 	push	{fp, lr}
   4:	e28db004 	add	fp, sp, #4
   8:	e59f0004 	ldr	r0, [pc, #4]	; 14 <main+0x14>
   c:	ebfffffe 	bl	0 <UART0_Send_String>
  10:	e8bd8800 	pop	{fp, pc}
  14:	00000000 	andeq	r0, r0, r0

Disassembly of section .data:

00000000 <String_Buffur>:
   0:	7261656c 	rsbvc	r6, r1, #108, 10	; 0x1b000000
   4:	6e692d6e 	cdpvs	13, 6, cr2, cr9, cr14, {3}
   8:	7065642d 	rsbvc	r6, r5, sp, lsr #8
   c:	3c3a6874 	ldccc	8, cr6, [sl], #-464	; 0xfffffe30
  10:	74736f4d 	ldrbtvc	r6, [r3], #-3917	; 0xf4d
  14:	20616661 	rsbcs	r6, r1, r1, ror #12
  18:	61686f4d 	cmnvs	r8, sp, asr #30
  1c:	2064656d 	rsbcs	r6, r4, sp, ror #10
  20:	65726445 	ldrbvs	r6, [r2, #-1093]!	; 0x445
  24:	003e7365 	eorseq	r7, lr, r5, ror #6
	...

Disassembly of section .comment:

00000000 <.comment>:
   0:	43434700 	movtmi	r4, #14080	; 0x3700
   4:	4728203a 			; <UNDEFINED> instruction: 0x4728203a
   8:	2029554e 	eorcs	r5, r9, lr, asr #10
   c:	2e372e34 	mrccs	14, 1, r2, cr7, cr4, {1}
  10:	Address 0x00000010 is out of bounds.


Disassembly of section .ARM.attributes:

00000000 <.ARM.attributes>:
```

Uart.s



```
uart.o:     file format elf32-littlearm


Disassembly of section .text:

00000000 <UART0_Send_String>:
   0:	e52db004 	push	{fp}		; (str fp, [sp, #-4]!)
   4:	e28db000 	add	fp, sp, #0
   8:	e24dd00c 	sub	sp, sp, #12
   c:	e50b0008 	str	r0, [fp, #-8]
  10:	ea000006 	b	30 <UART0_Send_String+0x30>
  14:	e59f3030 	ldr	r3, [pc, #48]	; 4c <UART0_Send_String+0x4c>
  18:	e51b2008 	ldr	r2, [fp, #-8]
  1c:	e5d22000 	ldrb	r2, [r2]
  20:	e5832000 	str	r2, [r3]
  24:	e51b3008 	ldr	r3, [fp, #-8]
  28:	e2833001 	add	r3, r3, #1
  2c:	e50b3008 	str	r3, [fp, #-8]
  30:	e51b3008 	ldr	r3, [fp, #-8]
  34:	e5d33000 	ldrb	r3, [r3]
  38:	e3530000 	cmp	r3, #0
  3c:	1afffff4 	bne	14 <UART0_Send_String+0x14>
  40:	e28bd000 	add	sp, fp, #0
  44:	e8bd0800 	ldmfd	sp!, {fp}
  48:	e12fff1e 	bx	lr
  4c:	101f1000 	andsne	r1, pc, r0

Disassembly of section .comment:

00000000 <.comment>:
   0:	43434700 	movtmi	r4, #14080	; 0x3700
   4:	4728203a 			; <UNDEFINED> instruction: 0x4728203a
   8:	2029554e 	eorcs	r5, r9, lr, asr #10
   c:	2e372e34 	mrccs	14, 1, r2, cr7, cr4, {1}
  10:	Address 0x00000010 is out of bounds.


Disassembly of section .ARM.attributes:

00000000 <.ARM.attributes>:
   0:	00003141 	andeq	r3, r0, r1, asr #2
```

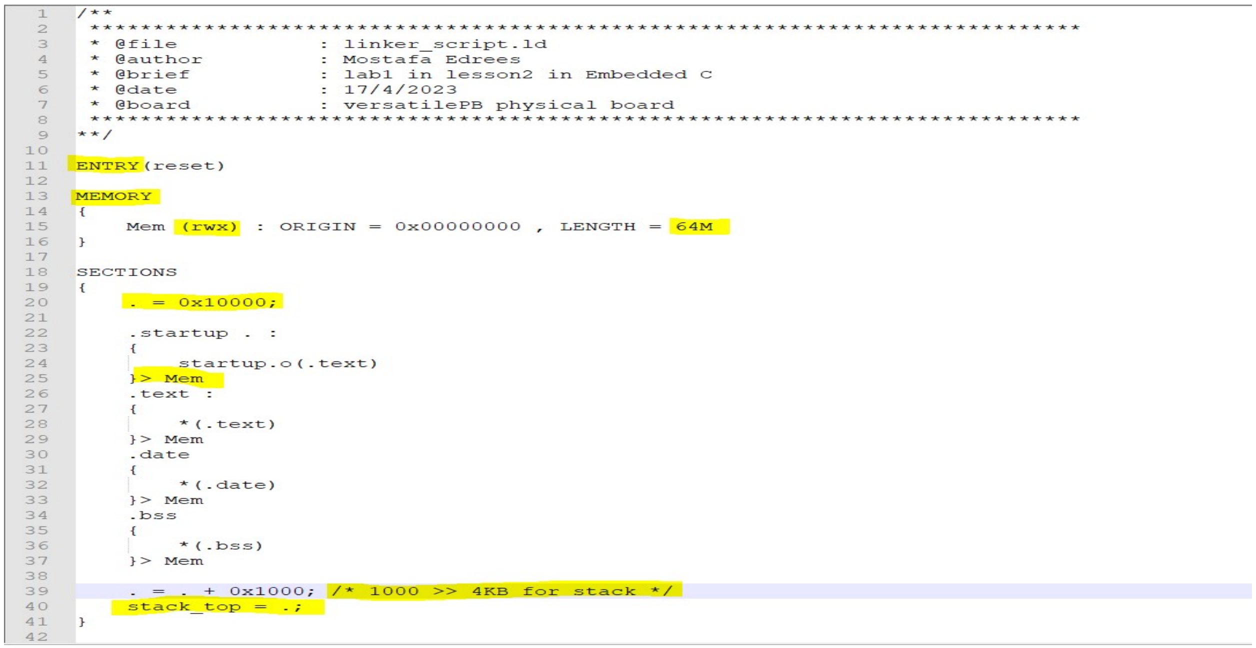
Command Line to create disassembly file of object file:



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

Write Linker Script file:

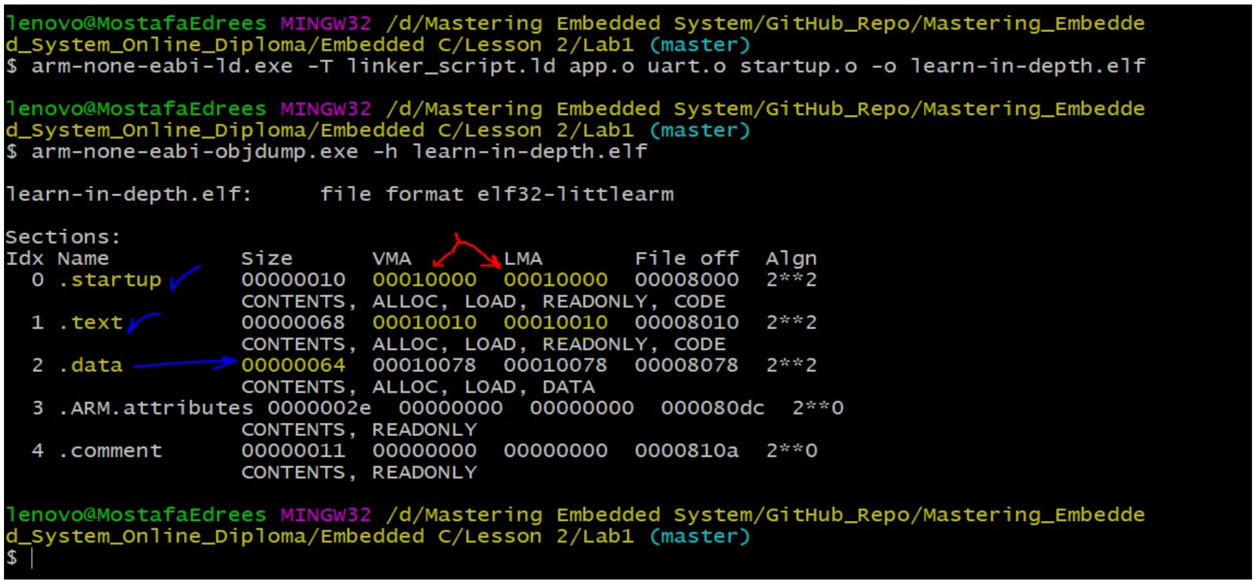
Linker_script.ld



```
/**
 ******************************************************************************
 * @file              : linker_script.ld
 * @author            : Mostafa Edrees
 * @brief             : lab1 in lesson2 in Embedded C
 * @date              : 17/4/2023
 * @board             : versatilePB physical board
 ******************************************************************************
**/

ENTRY(reset)

MEMORY
{
	Mem (rwx) : ORIGIN = 0x00000000 , LENGTH = 64M
}

SECTIONS
{
	. = 0x10000;

	.startup . :
	{
		startup.o(.text)
	}> Mem
	.text :
	{
		*(.text)
	}> Mem
	.date
	{
		*(.date)
	}> Mem
	.bss
	{
		*(.bss)
	}> Mem

	. = . + 0x1000; /* 1000 >> 4KB for stack */
	stack_top = .;
}
```

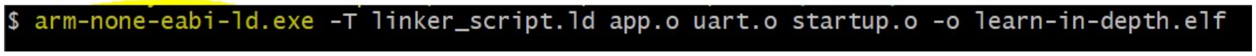
After linker_script the addresses will be physical with SOC

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)
$ |
```

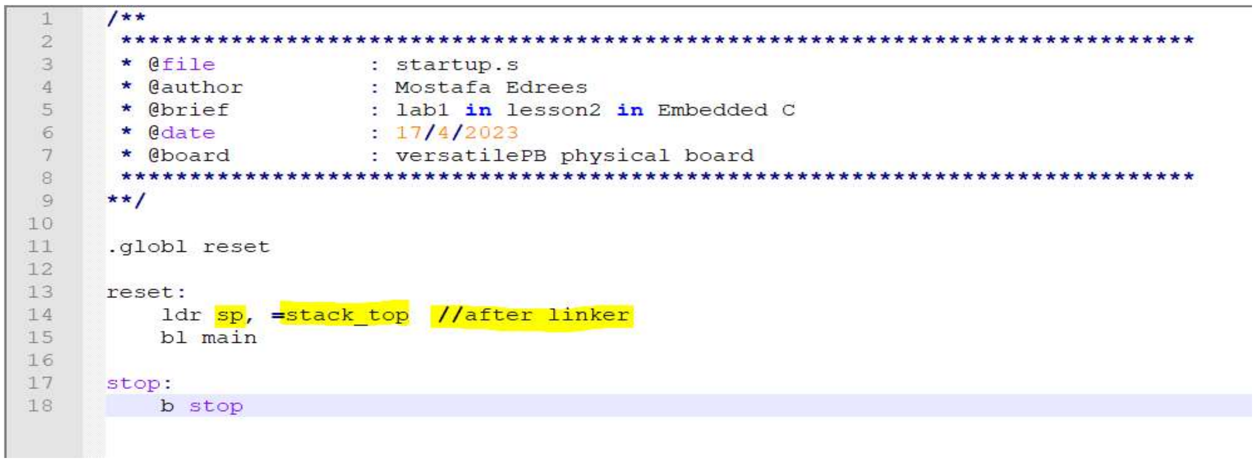
Command Line to link object files and get the executable:



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

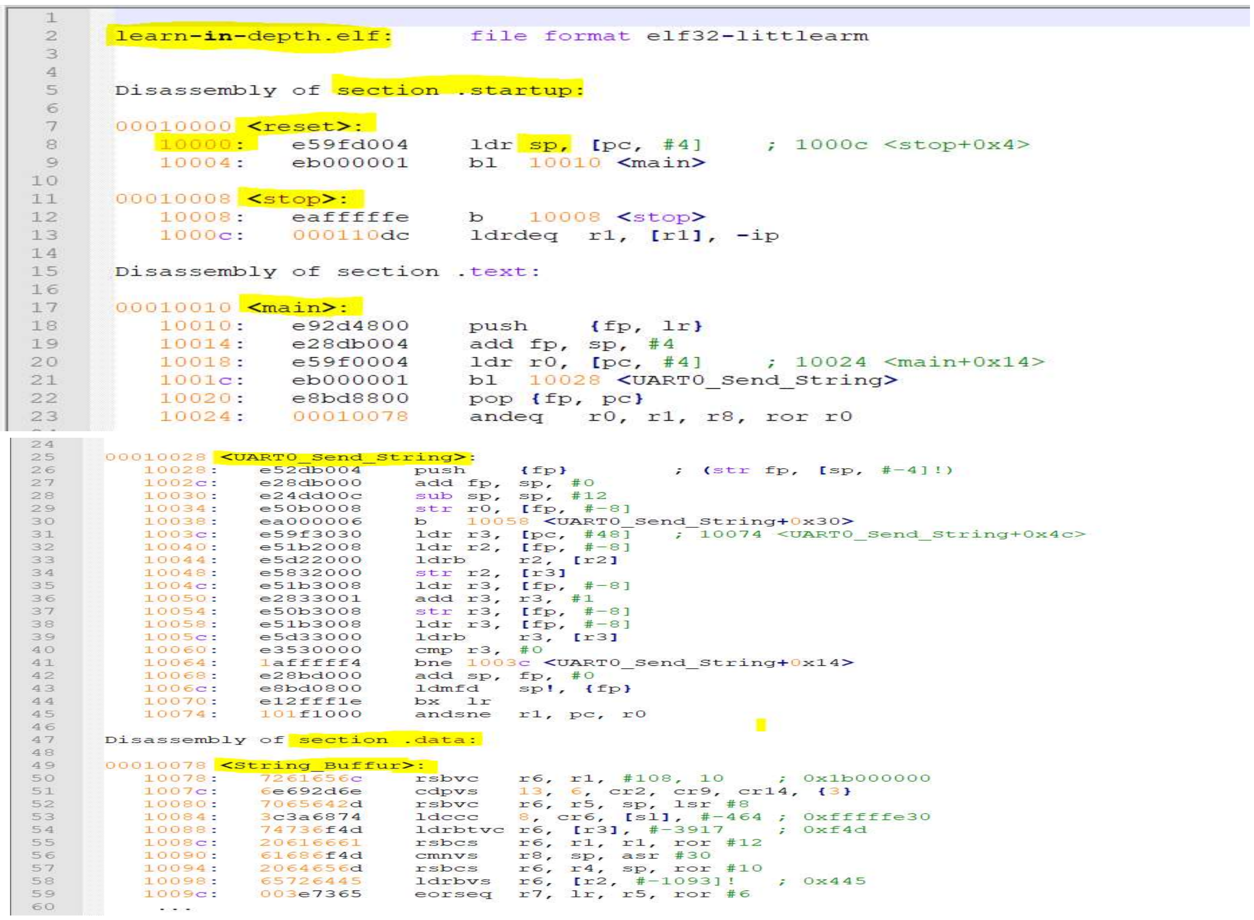
We put address of stack_top at startup.s to put this address in stack pointer register (PC).

Startup.s



```
/**
 *****************************************************************************
 * @file            : startup.s
 * @author          : Mostafa Edrees
 * @brief           : lab1 in lesson2 in Embedded C
 * @date            : 17/4/2023
 * @board           : versatilePB physical board
 *****************************************************************************
**/

.globl reset

reset:
    ldr sp, =stack_top  //after linker
    bl main

stop:
    b stop
```

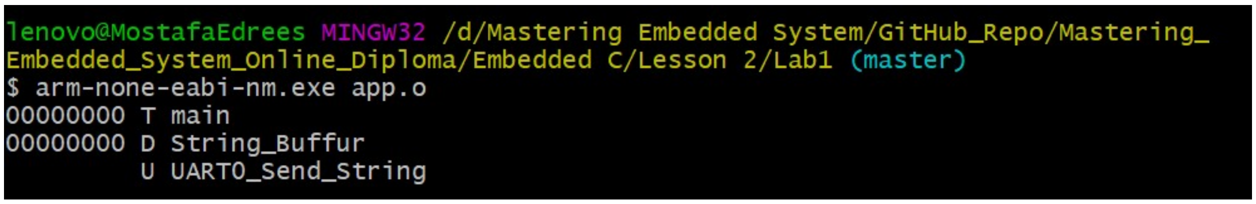
Let's create the disassembly of **learn-in-depth.elf**



```

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


Disassembly of section .startup:

00010000 <reset>:
   10000:	e59fd004 	ldr	sp, [pc, #4]	; 1000c <stop+0x4>
   10004:	eb000001 	bl	10010 <main>

00010008 <stop>:
   10008:	eafffffe 	b	10008 <stop>
   1000c:	000110dc 	ldrdeq	r1, [r1], -ip

Disassembly of section .text:

00010010 <main>:
   10010:	e92d4800 	push	{fp, lr}
   10014:	e28db004 	add	fp, sp, #4
   10018:	e59f0004 	ldr	r0, [pc, #4]	; 10024 <main+0x14>
   1001c:	eb000001 	bl	10028 <UART0_Send_String>
   10020:	e8bd8800 	pop	{fp, pc}
   10024:	00010078 	andeq	r0, r1, r8, ror r0

00010028 <UART0_Send_String>:
   10028:	e52db004 	push	{fp}		; (str fp, [sp, #-4]!)
   1002c:	e28db000 	add	fp, sp, #0
   10030:	e24dd00c 	sub	sp, sp, #12
   10034:	e50b0008 	str	r0, [fp, #-8]
   10038:	ea000006 	b	10058 <UART0_Send_String+0x30>
   1003c:	e59f3030 	ldr	r3, [pc, #48]	; 10074 <UART0_Send_String+0x4c>
   10040:	e51b2008 	ldr	r2, [fp, #-8]
   10044:	e5d22000 	ldrb	r2, [r2]
   10048:	e5832000 	str	r2, [r3]
   1004c:	e51b3008 	ldr	r3, [fp, #-8]
   10050:	e2833001 	add	r3, r3, #1
   10054:	e50b3008 	str	r3, [fp, #-8]
   10058:	e51b3008 	ldr	r3, [fp, #-8]
   1005c:	e5d33000 	ldrb	r3, [r3]
   10060:	e3530000 	cmp	r3, #0
   10064:	1afffff4 	bne	1003c <UART0_Send_String+0x14>
   10068:	e28bd000 	add	sp, fp, #0
   1006c:	e8bd0800 	ldmfd	sp!, {fp}
   10070:	e12fff1e 	bx	lr
   10074:	101f1000 	andsne	r1, pc, r0

Disassembly of section .data:

00010078 <String_Buffur>:
   10078:	7261656c 	rsbvc	r6, r1, #108, 10	; 0x1b000000
   1007c:	6e692d6e 	cdpvs	13, 6, cr2, cr9, cr14, {3}
   10080:	7065642d 	rsbvc	r6, r5, sp, lsr #8
   10084:	3c3a6874 	ldccc	8, cr6, [sl], #-464	; 0xfffffe30
   10088:	74736f4d 	ldrbtvc	r6, [r3], #-3917	; 0xf4d
   1008c:	20616661 	rsbcs	r6, r1, r1, ror #12
   10090:	61686f4d 	cmnvs	r8, sp, asr #30
   10094:	2064656d 	rsbcs	r6, r4, sp, ror #10
   10098:	65726445 	ldrbvs	r6, [r2, #-1093]!	; 0x445
   1009c:	003e7365 	eorseq	r7, lr, r5, ror #6
	...
```

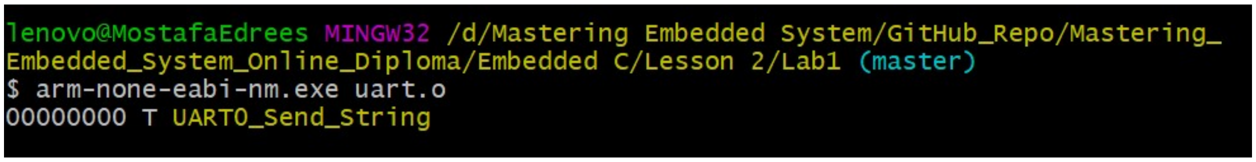
See Symbols in all object files:

app.o



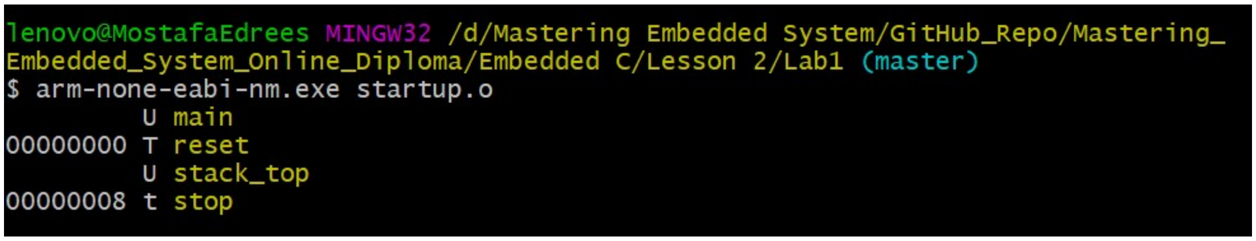
```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-nm.exe app.o
00000000 T main
00000000 D String_Buffur
         U UART0_Send_String
```

uart.o



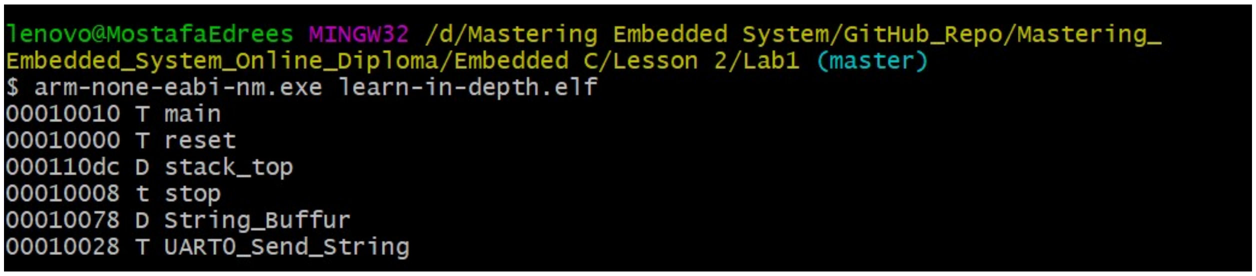
```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-nm.exe uart.o
00000000 T UART0_Send_String
```

startup.o



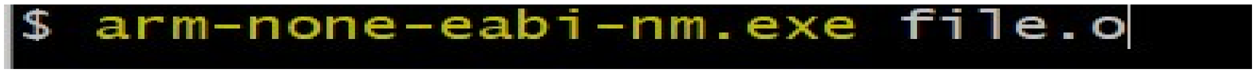
```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-nm.exe startup.o
         U main
00000000 T reset
         U stack_top
00000008 t stop
```

learn-in-depth.elf



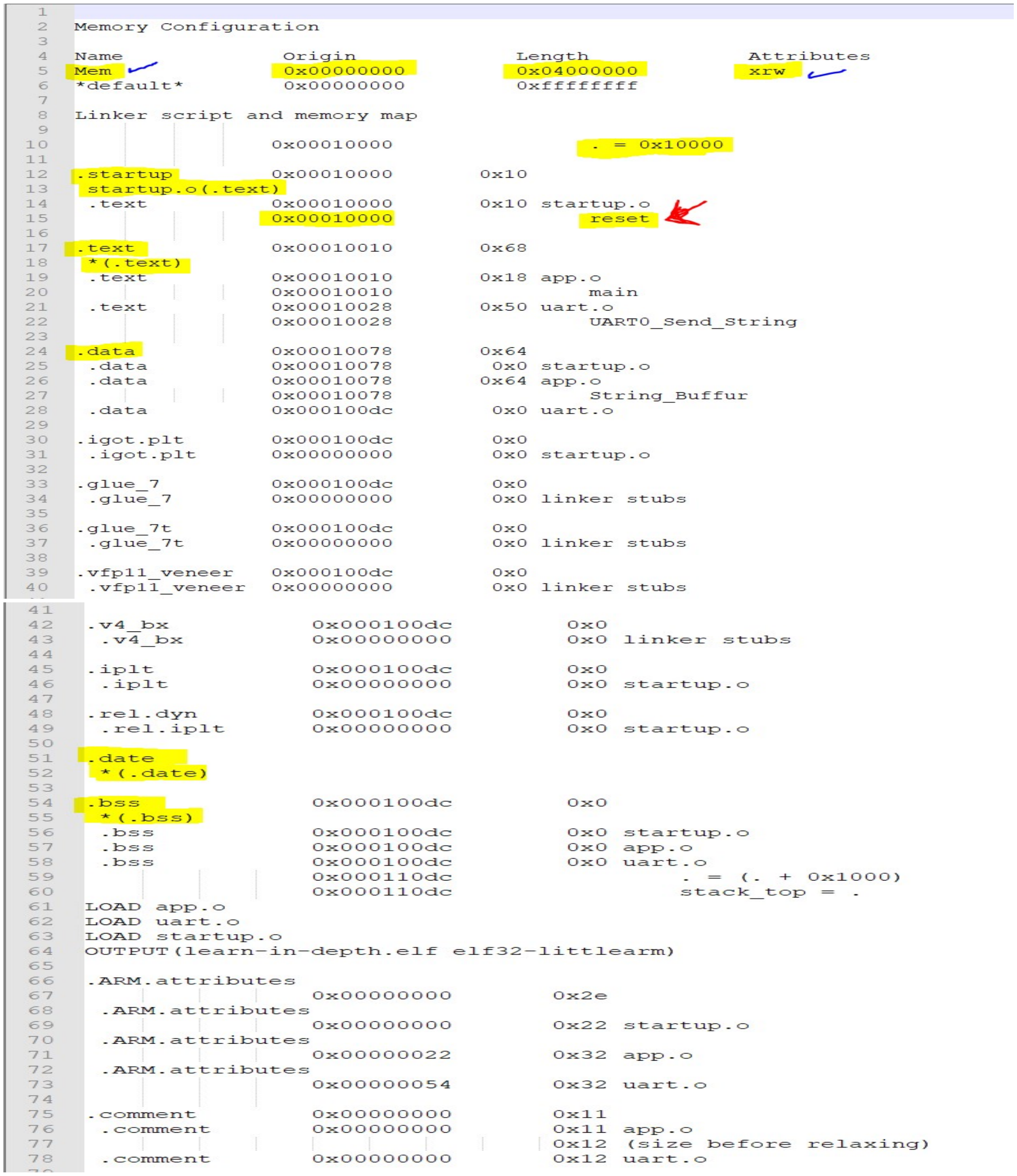
```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_
Embedded_System_Online_Diploma/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-nm.exe learn-in-depth.elf
00010010 T main
00010000 T reset
000110dc D stack_top
00010008 t stop
00010078 D String_Buffur
00010028 T UART0_Send_String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

<u>Command Line to get symbols of an object file:</u>



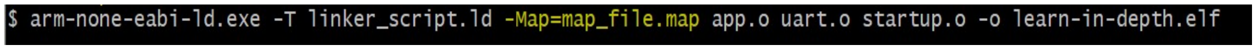
```
$ arm-none-eabi-nm.exe file.o
```

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



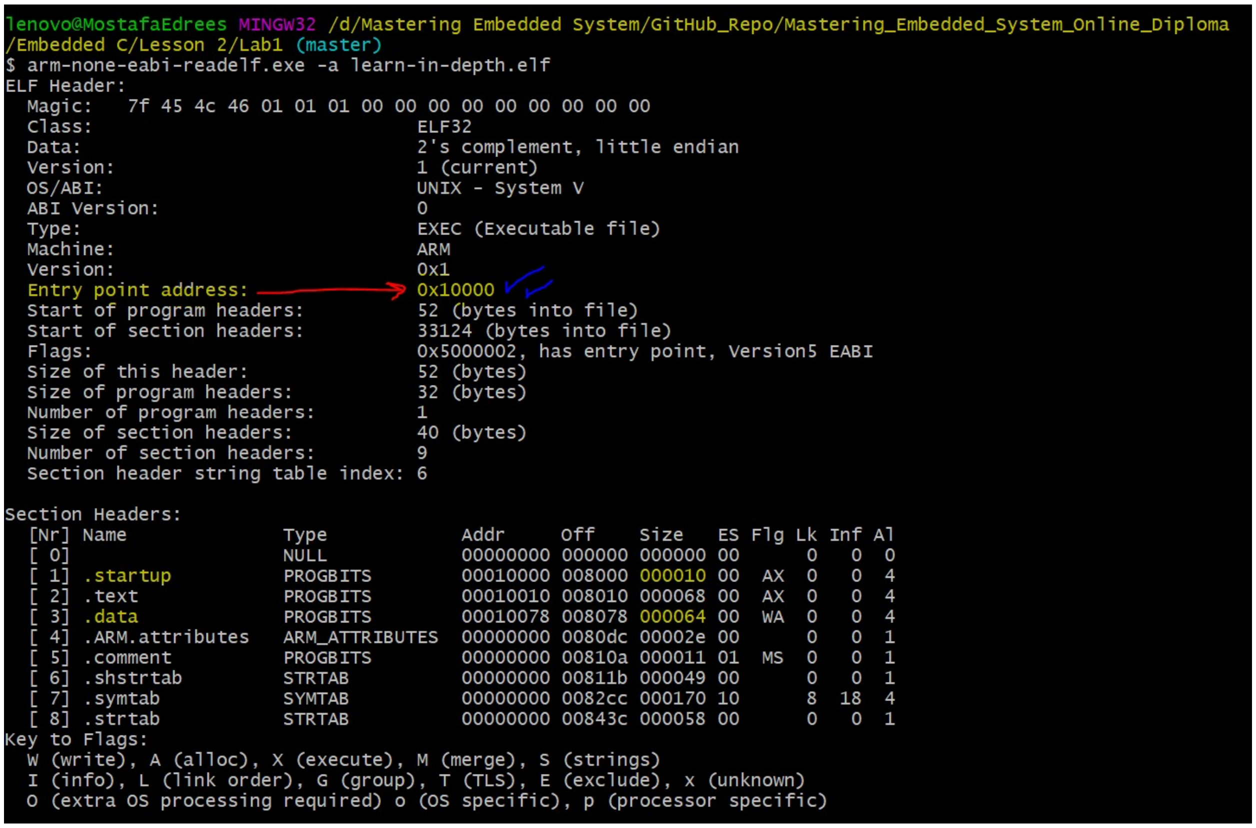
```

Memory Configuration

Name             Origin             Length             Attributes
Mem              0x00000000         0x04000000         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map

                0x00010000                . = 0x10000

.startup        0x00010000       0x10
 startup.o(.text)
 .text          0x00010000       0x10 startup.o
                0x00010000                reset

.text           0x00010010       0x68
 *(.text)
 .text          0x00010010       0x18 app.o
                0x00010010                main
 .text          0x00010028       0x50 uart.o
                0x00010028                UART0_Send_String

.data           0x00010078       0x64
 .data          0x00010078        0x0 startup.o
 .data          0x00010078       0x64 app.o
                0x00010078                String_Buffur
 .data          0x000100dc        0x0 uart.o

.igot.plt       0x000100dc        0x0
 .igot.plt      0x00000000        0x0 startup.o

.glue_7         0x000100dc        0x0
 .glue_7        0x00000000        0x0 linker stubs

.glue_7t        0x000100dc        0x0
 .glue_7t       0x00000000        0x0 linker stubs

.vfp11_veneer   0x000100dc        0x0
 .vfp11_veneer  0x00000000        0x0 linker stubs

.v4_bx          0x000100dc        0x0
 .v4_bx         0x00000000        0x0 linker stubs

.iplt           0x000100dc        0x0
 .iplt          0x00000000        0x0 startup.o

.rel.dyn        0x000100dc        0x0
 .rel.iplt      0x00000000        0x0 startup.o

.date
 *(.date)

.bss            0x000100dc        0x0
 *(.bss)
 .bss           0x000100dc        0x0 startup.o
 .bss           0x000100dc        0x0 app.o
 .bss           0x000100dc        0x0 uart.o
                0x000110dc                . = (. + 0x1000)
                0x000110dc                stack_top = .
LOAD app.o
LOAD uart.o
LOAD startup.o
OUTPUT(learn-in-depth.elf elf32-littlearm)

.ARM.attributes
                0x00000000       0x2e
 .ARM.attributes
                0x00000000       0x22 startup.o
 .ARM.attributes
                0x00000022       0x32 app.o
 .ARM.attributes
                0x00000054       0x32 uart.o

.comment        0x00000000       0x11
 .comment       0x00000000       0x11 app.o
                                 0x12 (size before relaxing)
 .comment       0x00000000       0x12 uart.o
```

Command Line to get map file of an object file:



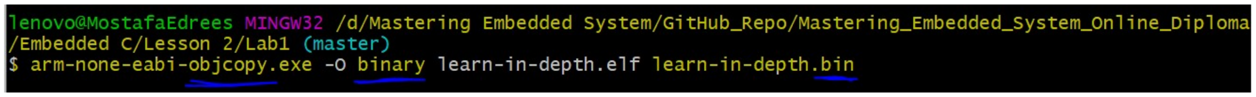
```
$ arm-none-eabi-ld.exe -T linker_script.ld -Map=map_file.map app.o uart.o startup.o -o learn-in-depth.elf
```

Check the entry point:



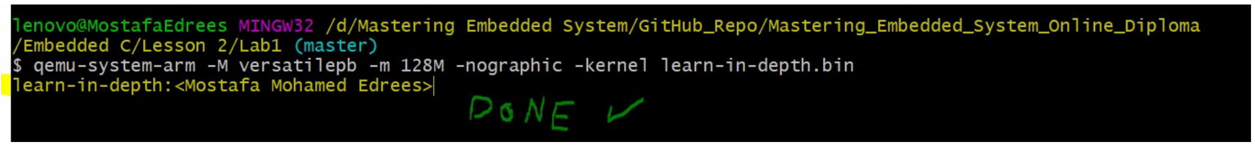
```
lenovo@MostafaEdrees MINGW32 /d/Mastering Embedded System/GitHub_Repo/Mastering_Embedded_System_Online_Diploma
/Embedded C/Lesson 2/Lab1 (master)
$ arm-none-eabi-readelf.exe -a learn-in-depth.elf
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             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 ✓