Report on Lab1 in Embedded C lesson 2

Course title	**Learn in depth**
Student Name	**علي عادل محمد علي**
Student Level	**First star**

Under supervision/ Eng. Karolos Shinoda

Table of contents:

1. Introduction.

2. Code files.

3. Startup file.

4. Object files.

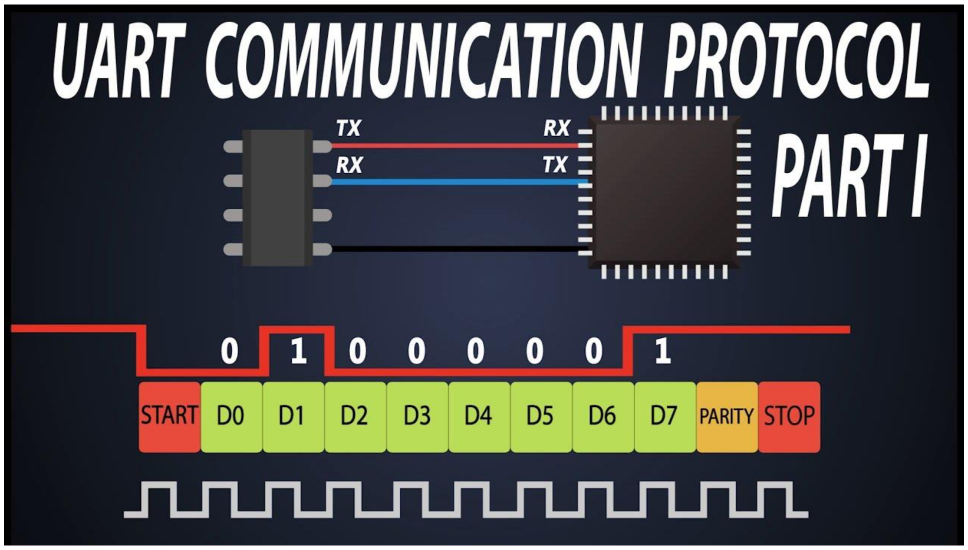
5. Linker Script file.

6. Execute the binary file.

7.View the sections and their contents.

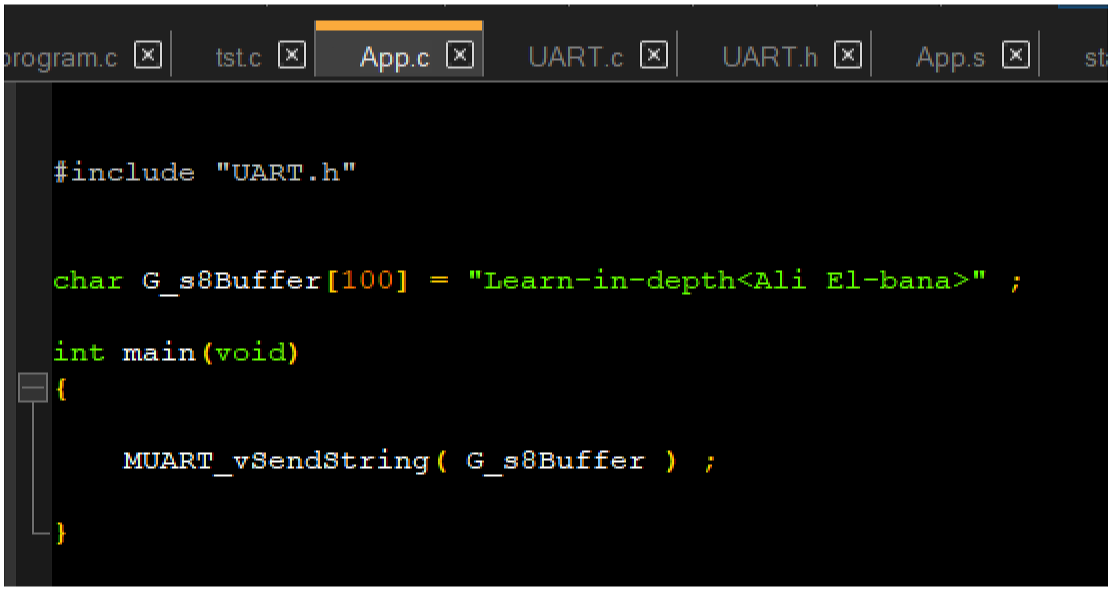
1. Introduction:

In this lab we want to send a string (Learn-in-depth<Ali El-bana>) using the UART communication protocol of the ARM VersatilePB board.



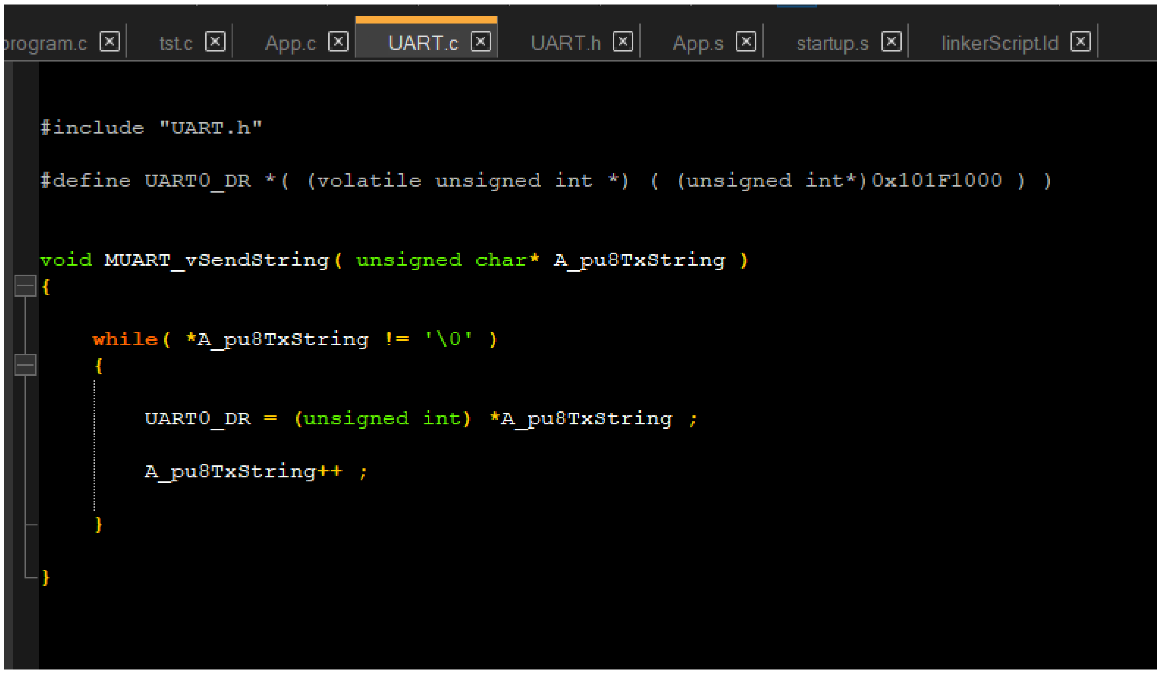
2. Code files:

App source code:



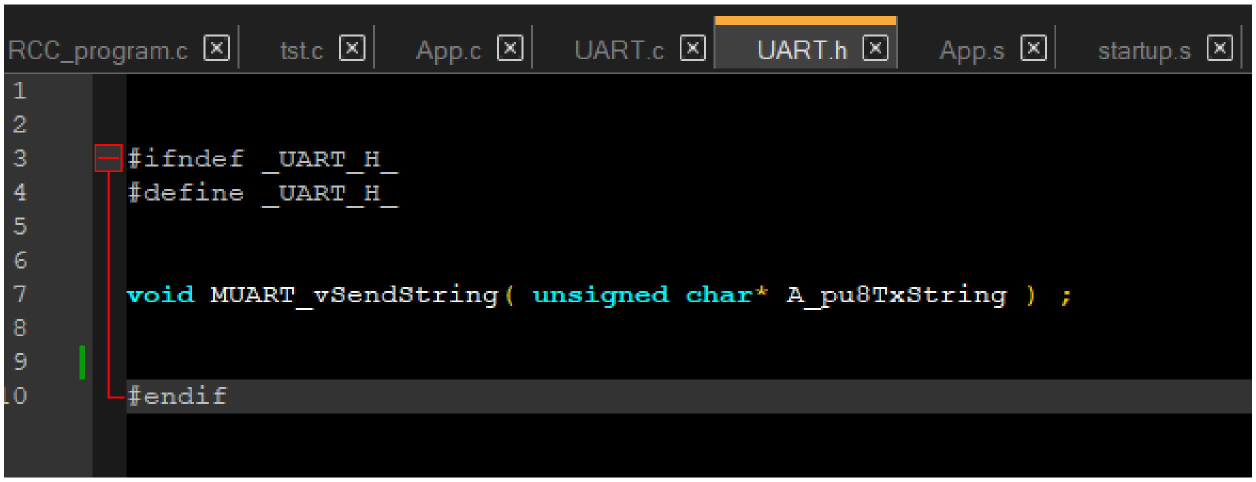
```c
#include "UART.h"

char G_s8Buffer[100] = "Learn-in-depth<Ali El-bana>" ;
int main(void)
{
    MUART_vSendString( G_s8Buffer ) ;
}
```

UART program:



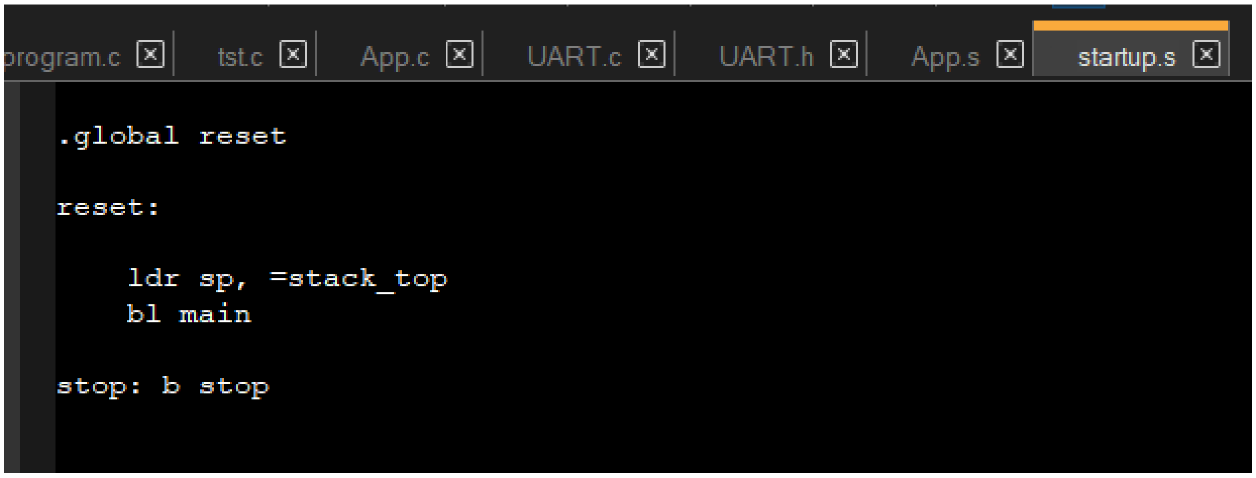
```c
#include "UART.h"
#define UART0_DR *( (volatile unsigned int *) ( (unsigned int*)0x101F1000 ) )

void MUART_vSendString( unsigned char* A_pu8TxString )
{
    while( *A_pu8TxString != '\0' )
    {
        UART0_DR = (unsigned int) *A_pu8TxString ;
        A_pu8TxString++ ;
    }
}
```

UART header file:



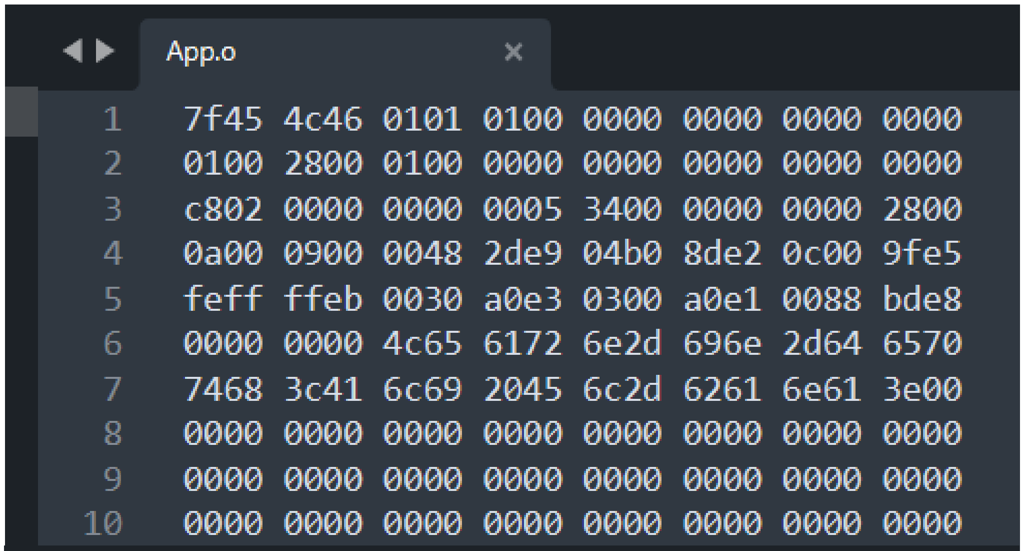
```
#ifndef _UART_H_
#define _UART_H_


void MUART_vSendString( unsigned char* A_pu8TxString ) ;


#endif
```

3. Startup file:

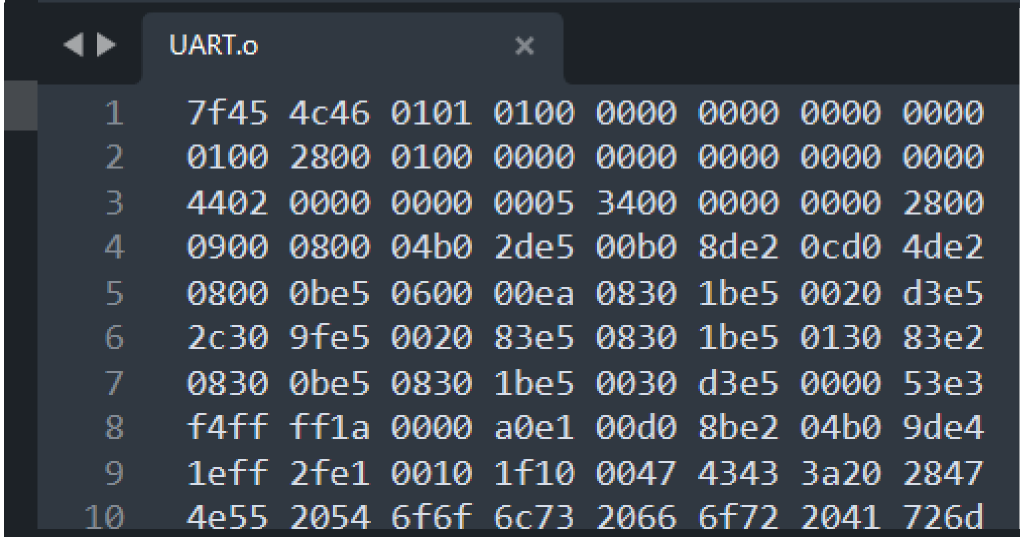


```
.global reset

reset:

    ldr sp, =stack_top
    bl main

stop: b stop
```

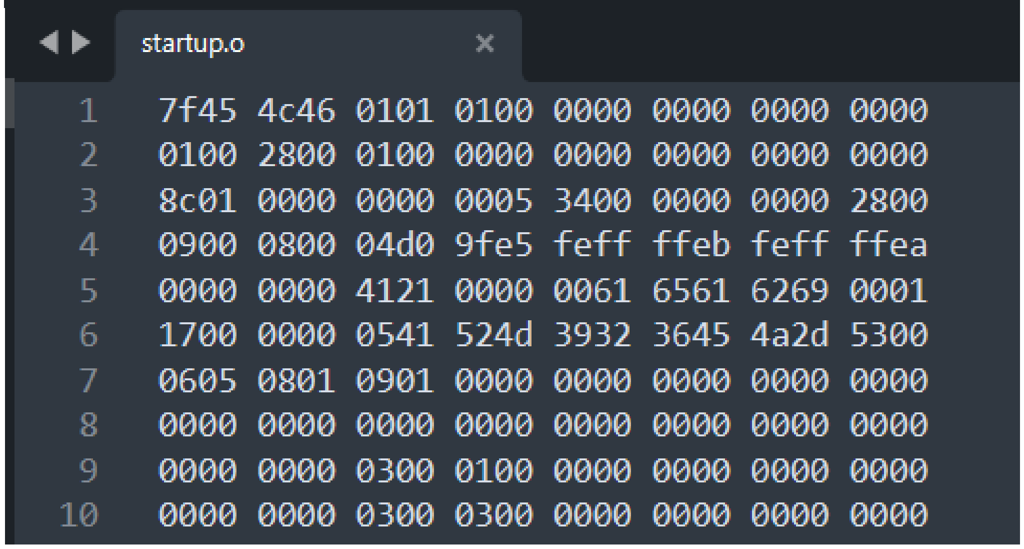
4. Object files:



```
7f45 4c46 0101 0100 0000 0000 0000 0000
0100 2800 0100 0000 0000 0000 0000 0000
c802 0000 0000 0005 3400 0000 0000 2800
0a00 0900 0048 2de9 04b0 8de2 0c00 9fe5
feff ffeb 0030 a0e3 0300 a0e1 0088 bde8
0000 0000 4c65 6172 6e2d 696e 2d64 6570
7468 3c41 6c69 2045 6c2d 6261 6e61 3e00
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

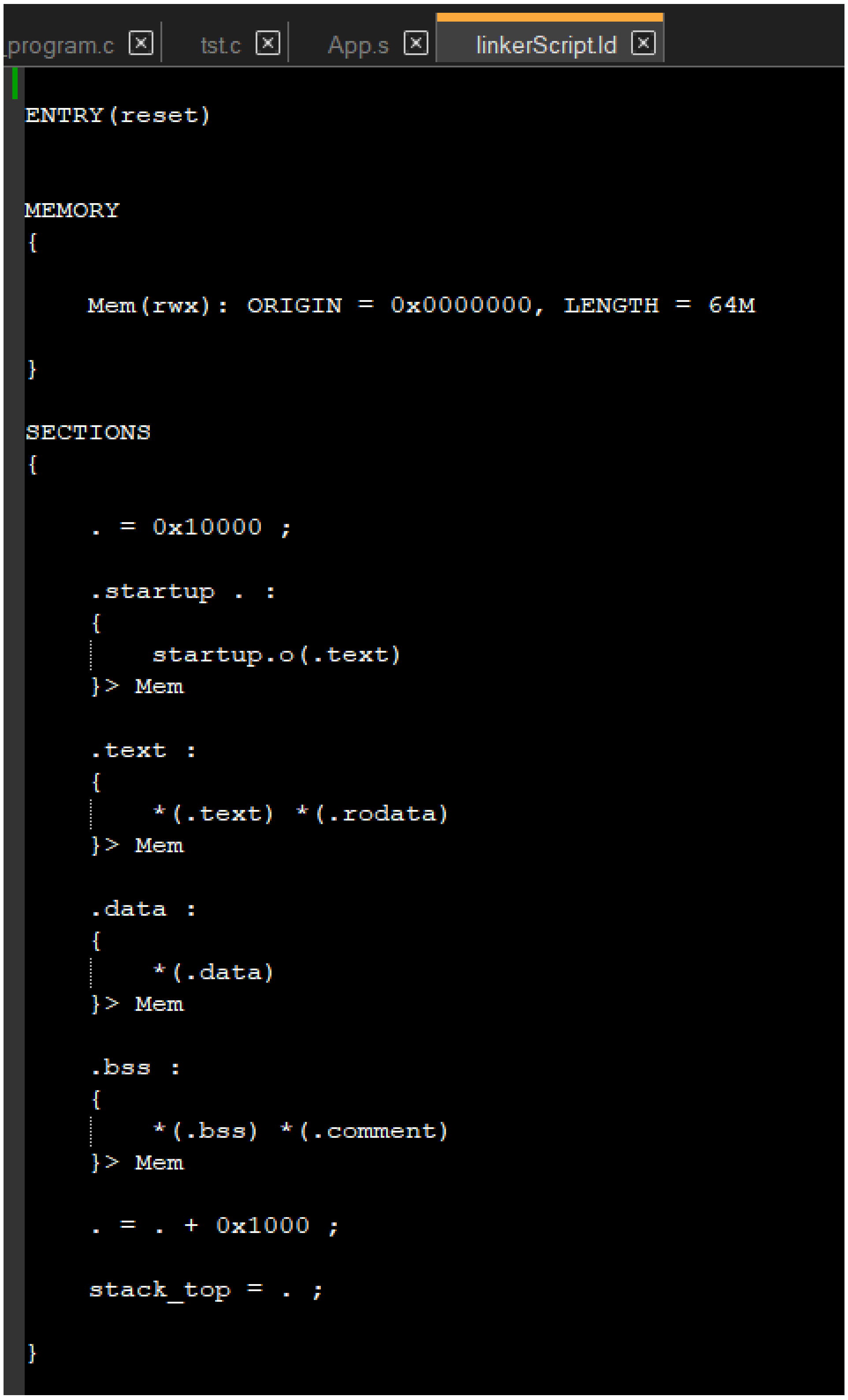


```
7f45 4c46 0101 0100 0000 0000 0000 0000
0100 2800 0100 0000 0000 0000 0000 0000
4402 0000 0000 0005 3400 0000 0000 2800
0900 0800 04b0 2de5 00b0 8de2 0cd0 4de2
0800 0be5 0600 00ea 0830 1be5 0020 d3e5
2c30 9fe5 0020 83e5 0830 1be5 0130 83e2
0830 0be5 0830 1be5 0030 d3e5 0000 53e3
f4ff ff1a 0000 a0e1 00d0 8be2 04b0 9de4
1eff 2fe1 0010 1f10 0047 4343 3a20 2847
4e55 2054 6f6f 6c73 2066 6f72 2041 726d
```



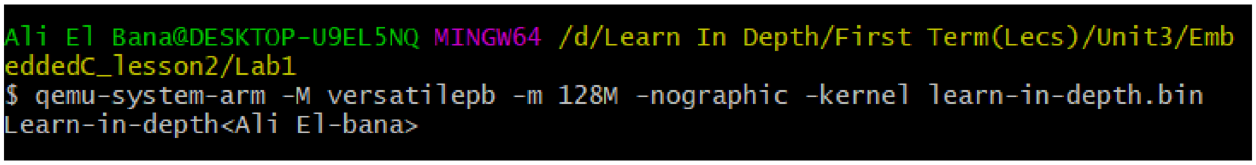
```
7f45 4c46 0101 0100 0000 0000 0000 0000
0100 2800 0100 0000 0000 0000 0000 0000
8c01 0000 0000 0005 3400 0000 0000 2800
0900 0800 04d0 9fe5 feff ffeb feff ffea
0000 0000 4121 0000 0061 6561 6269 0001
1700 0000 0541 524d 3932 3645 4a2d 5300
0605 0801 0901 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0300 0100 0000 0000 0000 0000
0000 0000 0300 0300 0000 0000 0000 0000
```

5. Linker Script file:



```
ENTRY(reset)


MEMORY
{

    Mem(rwx): ORIGIN = 0x0000000, LENGTH = 64M

}

SECTIONS
{

    . = 0x10000 ;

    .startup . :
    {
        startup.o(.text)
    }> Mem

    .text :
    {
        *(.text) *(.rodata)
    }> Mem

    .data :
    {
        *(.data)
    }> Mem

    .bss :
    {
        *(.bss) *(.comment)
    }> Mem

    . = . + 0x1000 ;

    stack_top = . ;

}
```

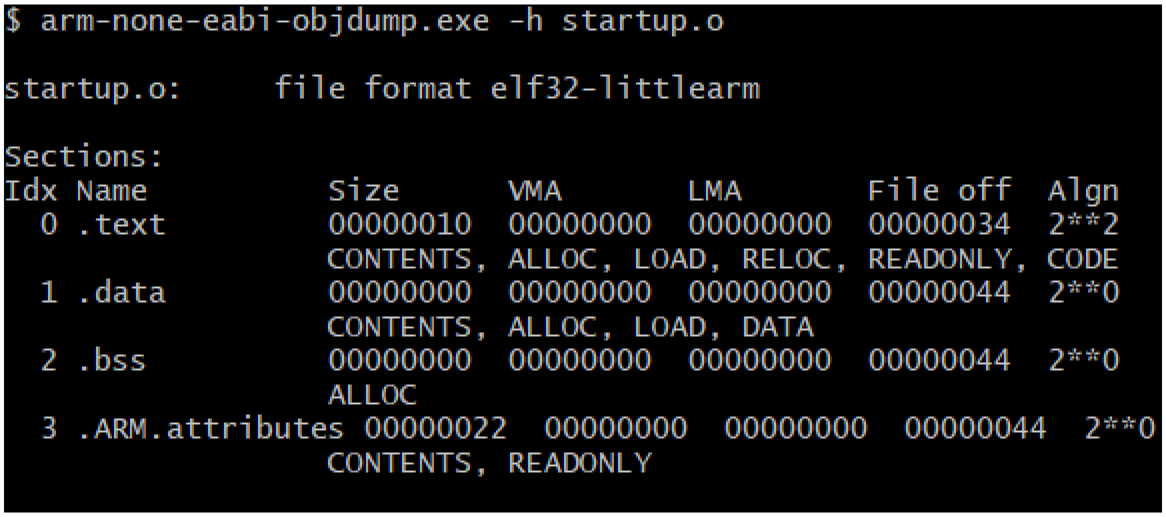
6. Execute the binary file:



```
Ali El Bana@DESKTOP-U9EL5NQ MINGW64 /d/Learn In Depth/First Term(Lecs)/Unit3/Emb
eddedC_lesson2/Lab1
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin
Learn-in-depth<Ali El-bana>
```

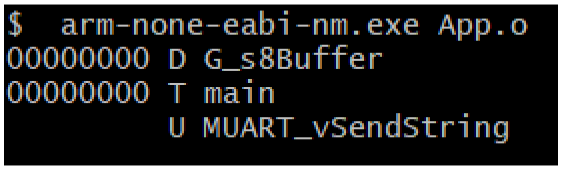
7.View the sections and their contents:

Startup header sections:

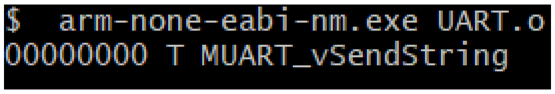


```
$ arm-none-eabi-objdump.exe -h startup.o

startup.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000010  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000044  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000044  2**0
                  ALLOC
  3 .ARM.attributes 00000022  00000000  00000000  00000044  2**0
                  CONTENTS, READONLY
```

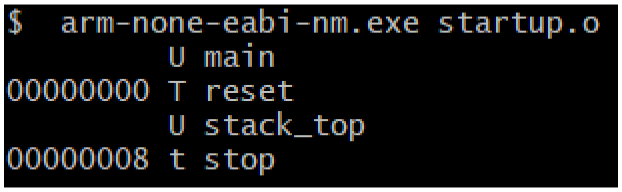
To read the symbol table contents of the obj files:



```
$  arm-none-eabi-nm.exe App.o
00000000 D G_s8Buffer
00000000 T main
         U MUART_vSendString
```



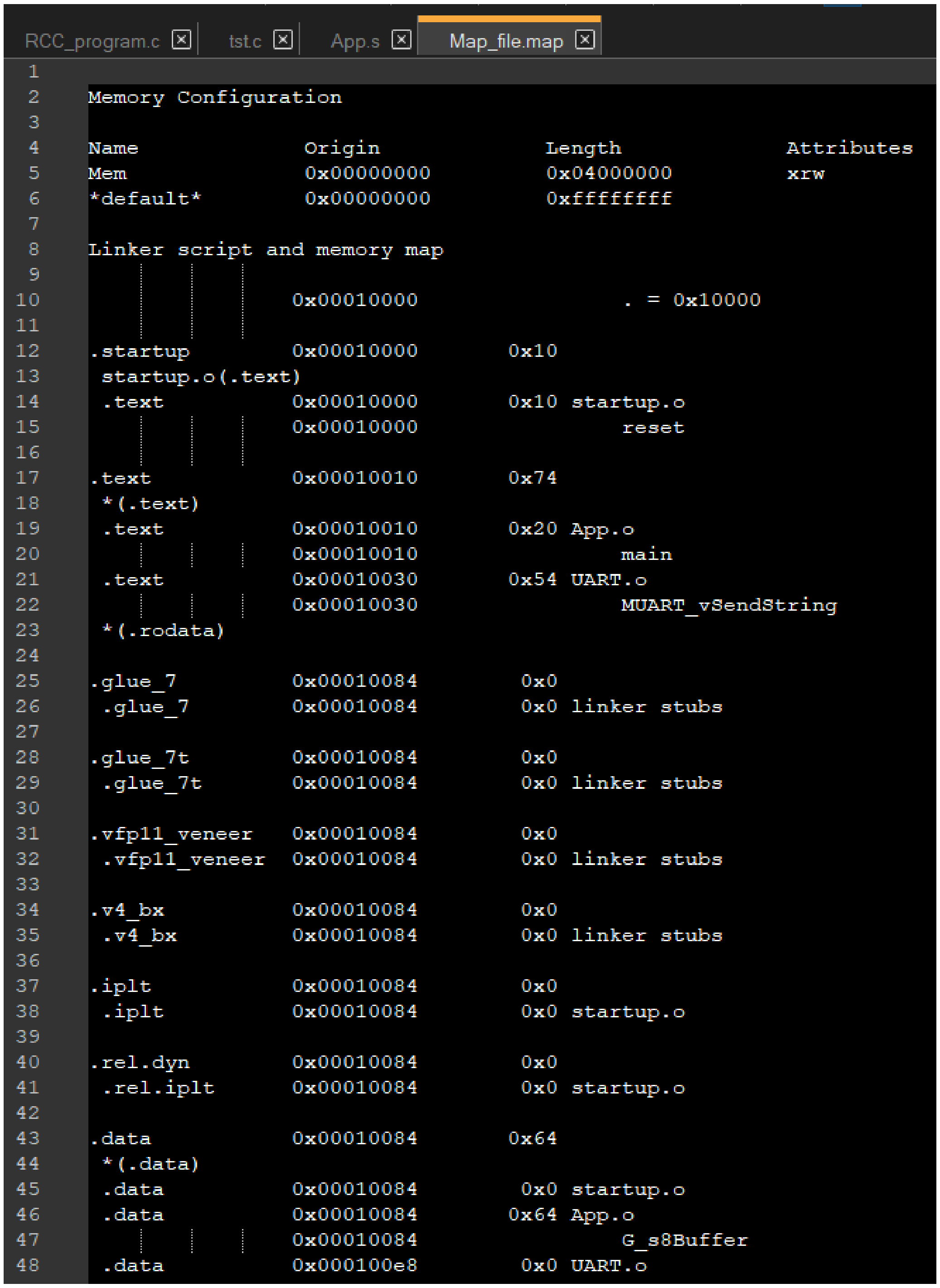
```
$  arm-none-eabi-nm.exe UART.o
00000000 T MUART_vSendString
```



```
$  arm-none-eabi-nm.exe startup.o
         U main
00000000 T reset
         U stack_top
00000008 t stop
```

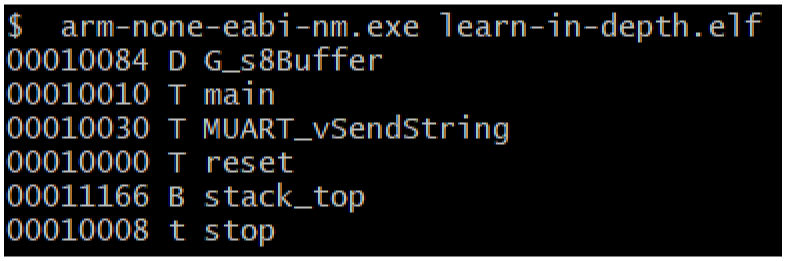
To read the map table contents:

arm-none-eabi-ld.exe -T linkerScript.ld App.o UART.o startup.o -o learn-in-depth.elf -Map=Map_file.map



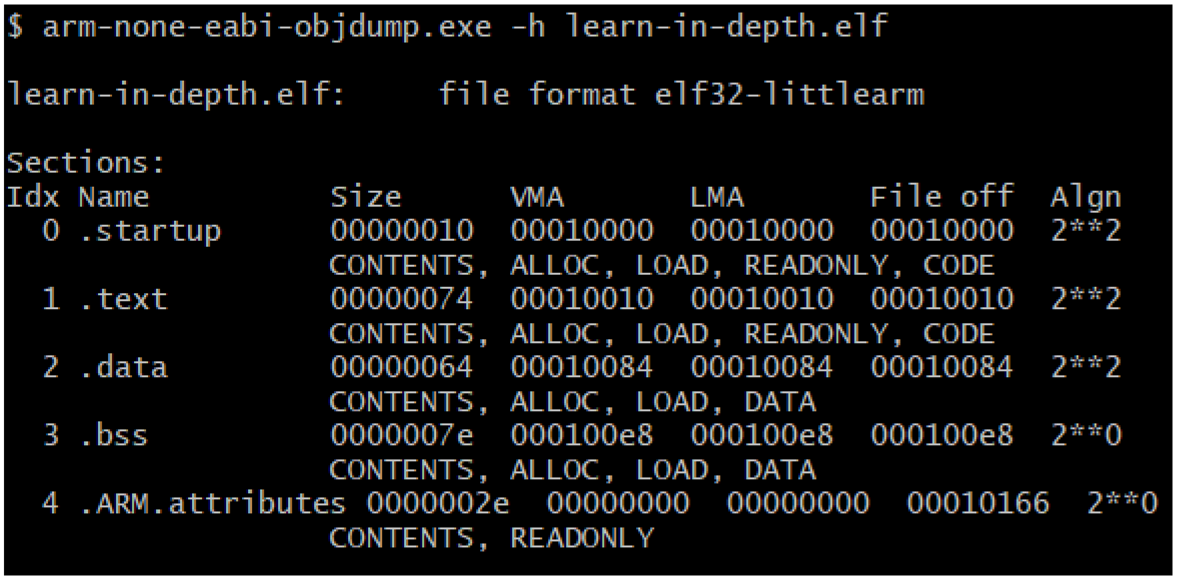
```

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       0x74
 *(.text)
 .text          0x00010010       0x20 App.o
                0x00010010                main
 .text          0x00010030       0x54 UART.o
                0x00010030                MUART_vSendString
 *(.rodata)

.glue_7         0x00010084        0x0
 .glue_7        0x00010084        0x0 linker stubs

.glue_7t        0x00010084        0x0
 .glue_7t       0x00010084        0x0 linker stubs

.vfp11_veneer   0x00010084        0x0
 .vfp11_veneer  0x00010084        0x0 linker stubs

.v4_bx          0x00010084        0x0
 .v4_bx         0x00010084        0x0 linker stubs

.iplt           0x00010084        0x0
 .iplt          0x00010084        0x0 startup.o

.rel.dyn        0x00010084        0x0
 .rel.iplt      0x00010084        0x0 startup.o

.data           0x00010084       0x64
 *(.data)
 .data          0x00010084        0x0 startup.o
 .data          0x00010084       0x64 App.o
                0x00010084                G_s8Buffer
 .data          0x000100e8        0x0 UART.o
```

To read the symbol table contents of the elf file:



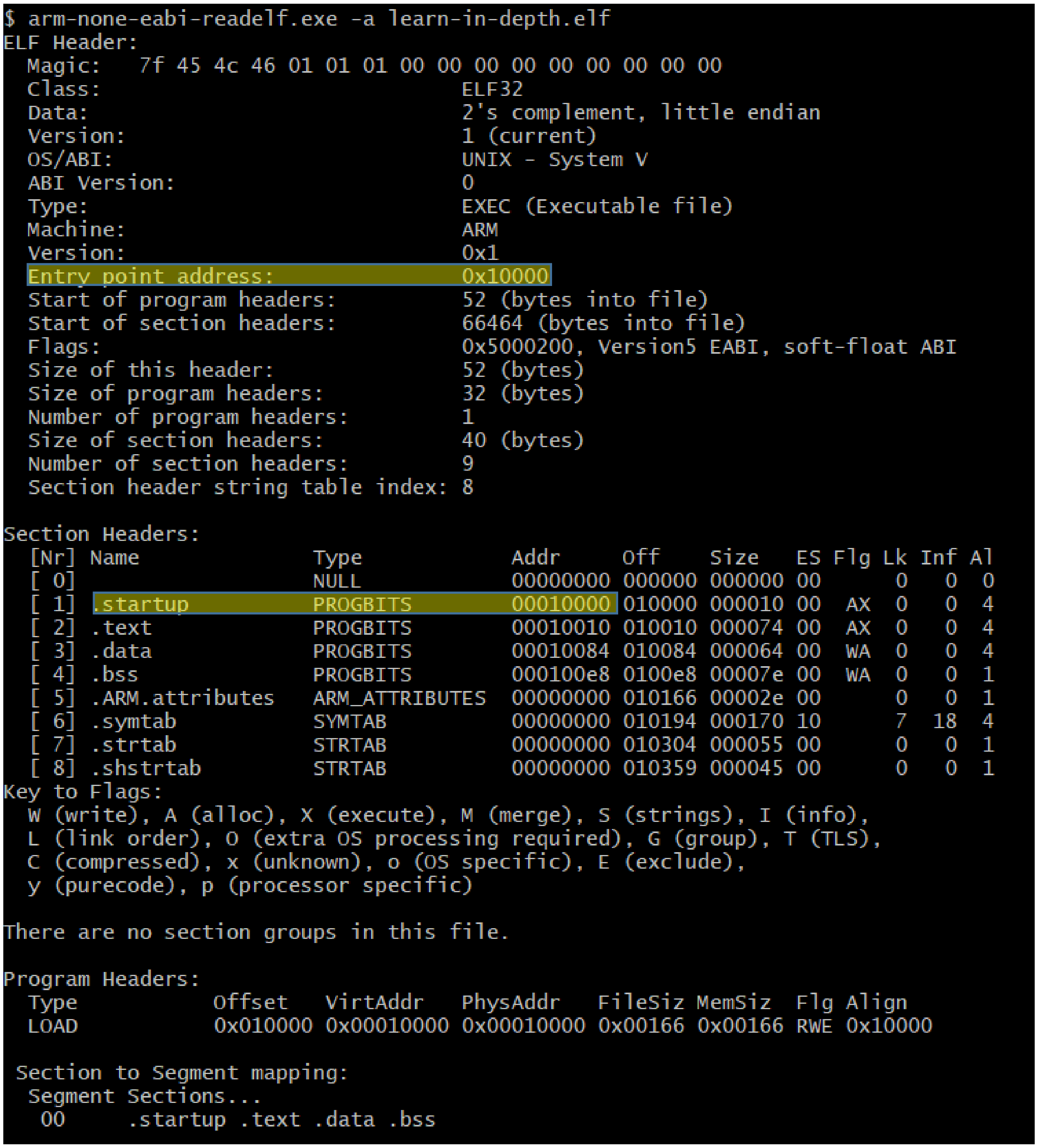
```
$  arm-none-eabi-nm.exe learn-in-depth.elf
00010084 D G_s8Buffer
00010010 T main
00010030 T MUART_vSendString
00010000 T reset
00011166 B stack_top
00010008 t stop
```

Elf file header sections:



```
$ 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  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         00000074  00010010  00010010  00010010  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  00010084  00010084  00010084  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .bss          0000007e  000100e8  000100e8  000100e8  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  4 .ARM.attributes 0000002e  00000000  00000000  00010166  2**0
                  CONTENTS, READONLY
```

To read the sections and contents of the elf file:



```
$ 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:          66464 (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:         1
  Size of section headers:           40 (bytes)
  Number of section headers:         9
  Section header string table index: 8

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 010000 000010 00  AX  0   0  4
  [ 2] .text             PROGBITS        00010010 010010 000074 00  AX  0   0  4
  [ 3] .data             PROGBITS        00010084 010084 000064 00  WA  0   0  4
  [ 4] .bss              PROGBITS        000100e8 0100e8 00007e 00  WA  0   0  1
  [ 5] .ARM.attributes   ARM_ATTRIBUTES  00000000 010166 00002e 00      0   0  1
  [ 6] .symtab           SYMTAB          00000000 010194 000170 10      7  18  4
  [ 7] .strtab           STRTAB          00000000 010304 000055 00      0   0  1
  [ 8] .shstrtab         STRTAB          00000000 010359 000045 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)

There are no section groups in this file.

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  LOAD           0x010000 0x00010000 0x00010000 0x00166 0x00166 RWE 0x10000

 Section to Segment mapping:
  Segment Sections...
   00     .startup .text .data .bss
```