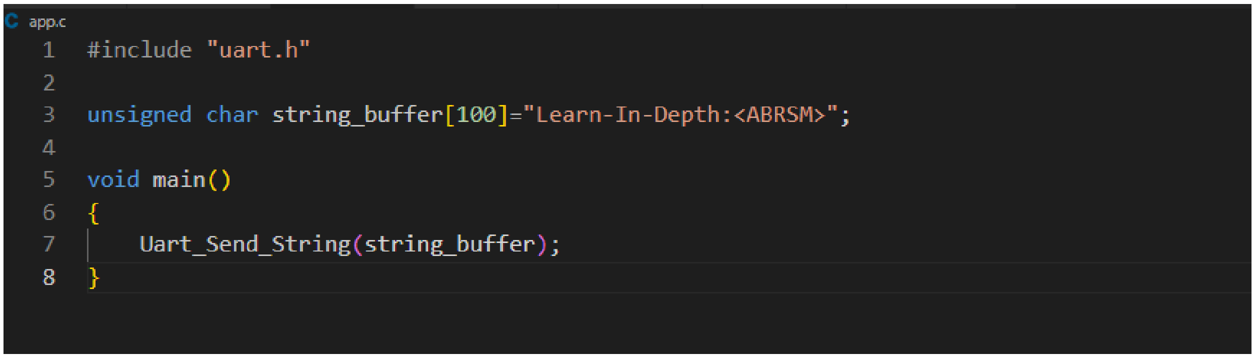
Assignment Lecture 2

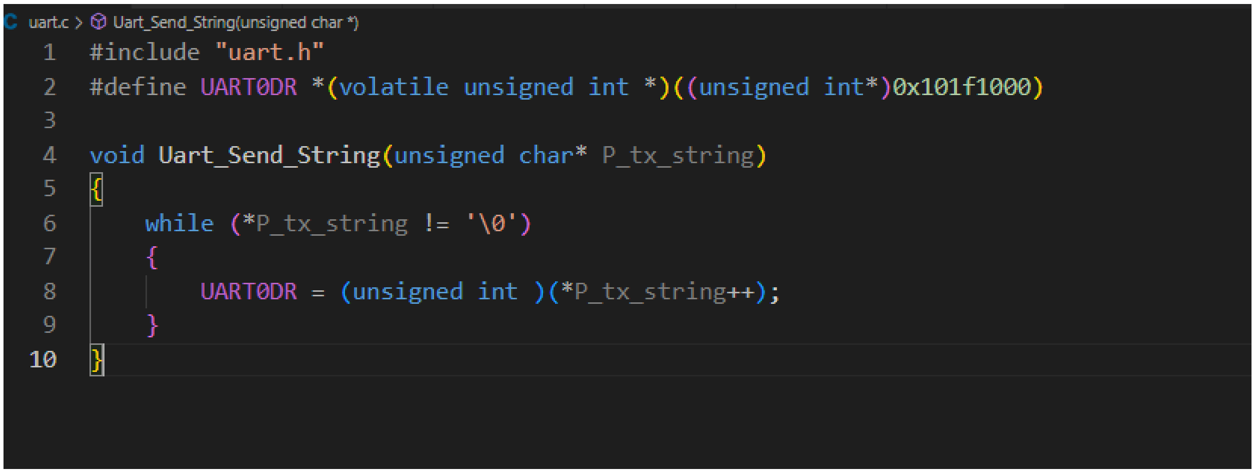
1. Writing Codess

I. app.c code



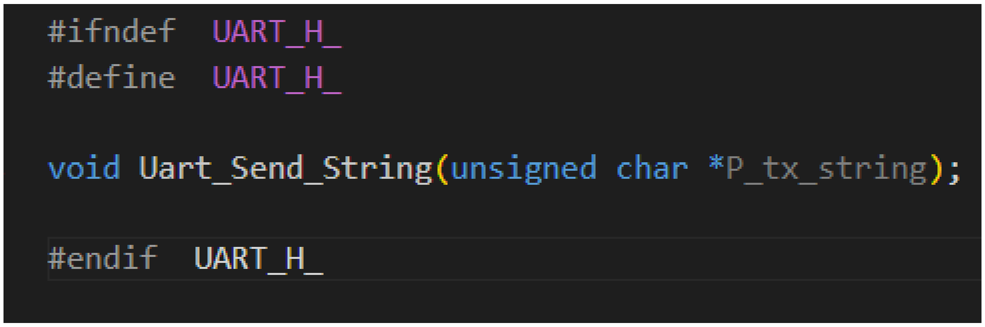
```c
#include "uart.h"

unsigned char string_buffer[100]="Learn-In-Depth:<ABRSM>";

void main()
{
    Uart_Send_String(string_buffer);
}
```

II. uart.c code



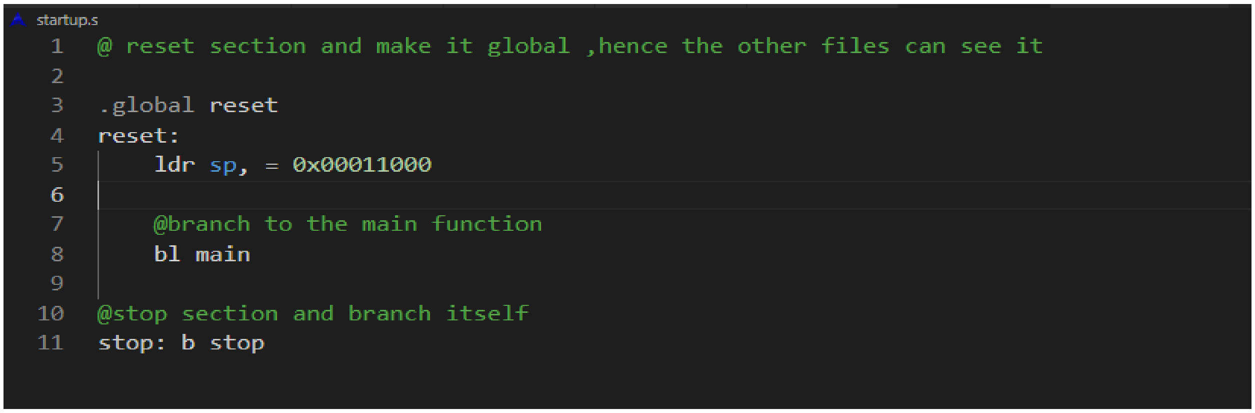
```c
#include "uart.h"
#define UART0DR *(volatile unsigned int *)((unsigned int*)0x101f1000)

void Uart_Send_String(unsigned char* P_tx_string)
{
    while (*P_tx_string != '\0')
    {
        UART0DR = (unsigned int )(*P_tx_string++);
    }
}
```

III. uart.h code



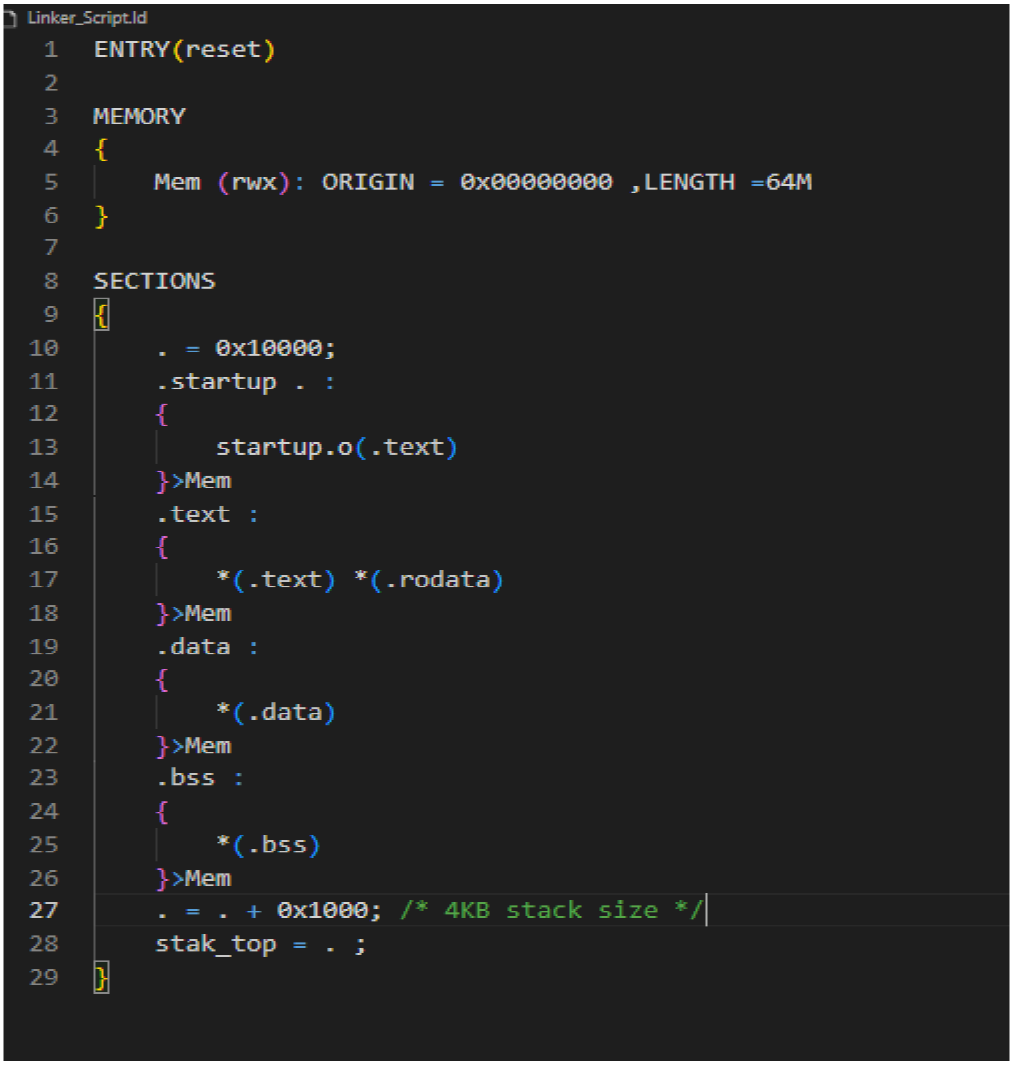
```c
#ifndef  UART_H_
#define  UART_H_

void Uart_Send_String(unsigned char *P_tx_string);

#endif  UART_H_
```

IV. startup.s code



```
@ reset section and make it global ,hence the other files can see it

.global reset
reset:
    ldr sp, = 0x00011000

    @branch to the main function
    bl main

@stop section and branch itself
stop: b stop
```

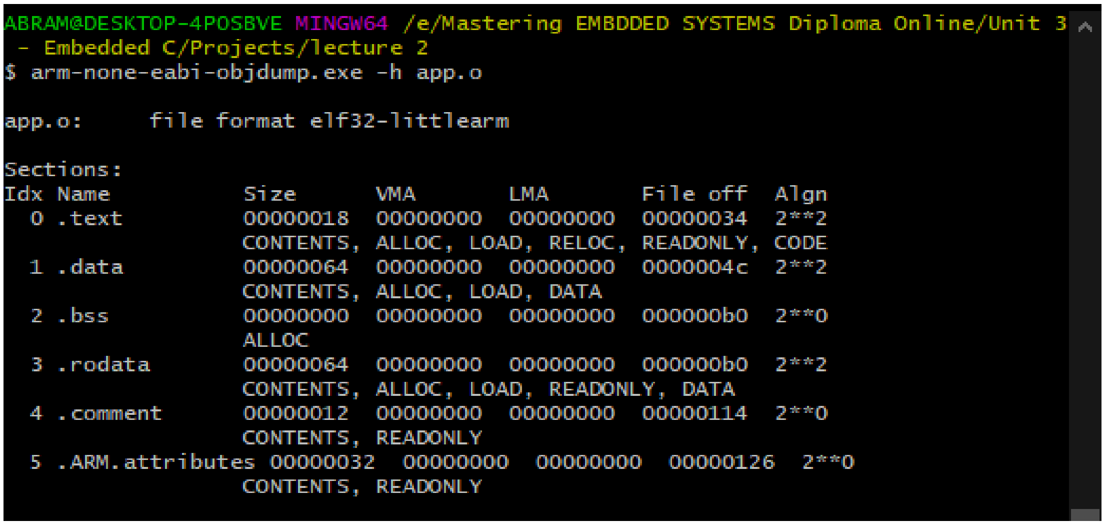
V. Linker_Script code



```
ENTRY(reset)

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

SECTIONS
{
    . = 0x10000;
    .startup . :
    {
        startup.o(.text)
    }>Mem
    .text :
    {
        *(.text) *(.rodata)
    }>Mem
    .data :
    {
        *(.data)
    }>Mem
    .bss :
    {
        *(.bss)
    }>Mem
    . = . + 0x1000; /* 4KB stack size */
    stak_top = . ;
}
```

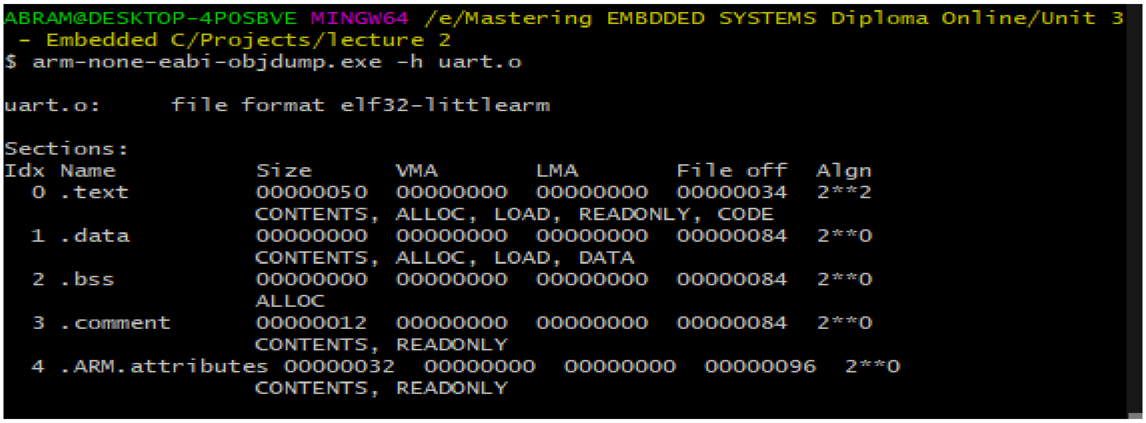
2. Object files sections for above codes

I. app



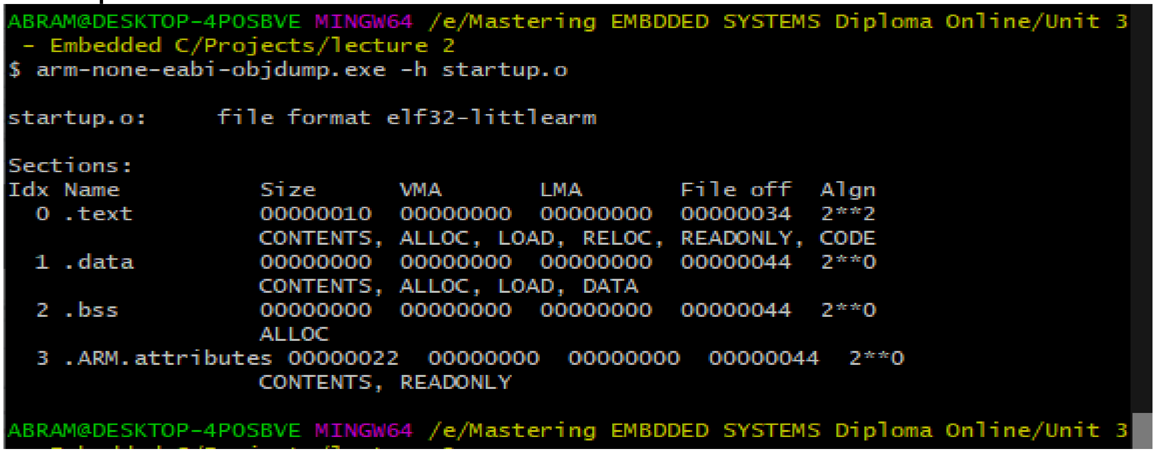
```
ABRAM@DESKTOP-4POSBVE MINGW64 /e/Mastering EMBDDED SYSTEMS Diploma Online/Unit 3
 - Embedded C/Projects/lecture 2
$ arm-none-eabi-objdump.exe -h app.o

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 .rodata       00000064  00000000  00000000  000000b0  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      00000012  00000000  00000000  00000114  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 00000032  00000000  00000000  00000126  2**0
                  CONTENTS, READONLY
```

II. uart



```
ABRAM@DESKTOP-4POSBVE MINGW64 /e/Mastering EMBDDED SYSTEMS Diploma Online/Unit 3
 - Embedded C/Projects/lecture 2
$ 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
```

III. startup



```
ABRAM@DESKTOP-4POSBVE MINGW64 /e/Mastering EMBDDED SYSTEMS Diploma Online/Unit 3
 - Embedded C/Projects/lecture 2
$ 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

ABRAM@DESKTOP-4POSBVE MINGW64 /e/Mastering EMBDDED SYSTEMS Diploma Online/Unit 3
```

3. Obj files for app.c, uart.c and startup.c

I. app.o

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

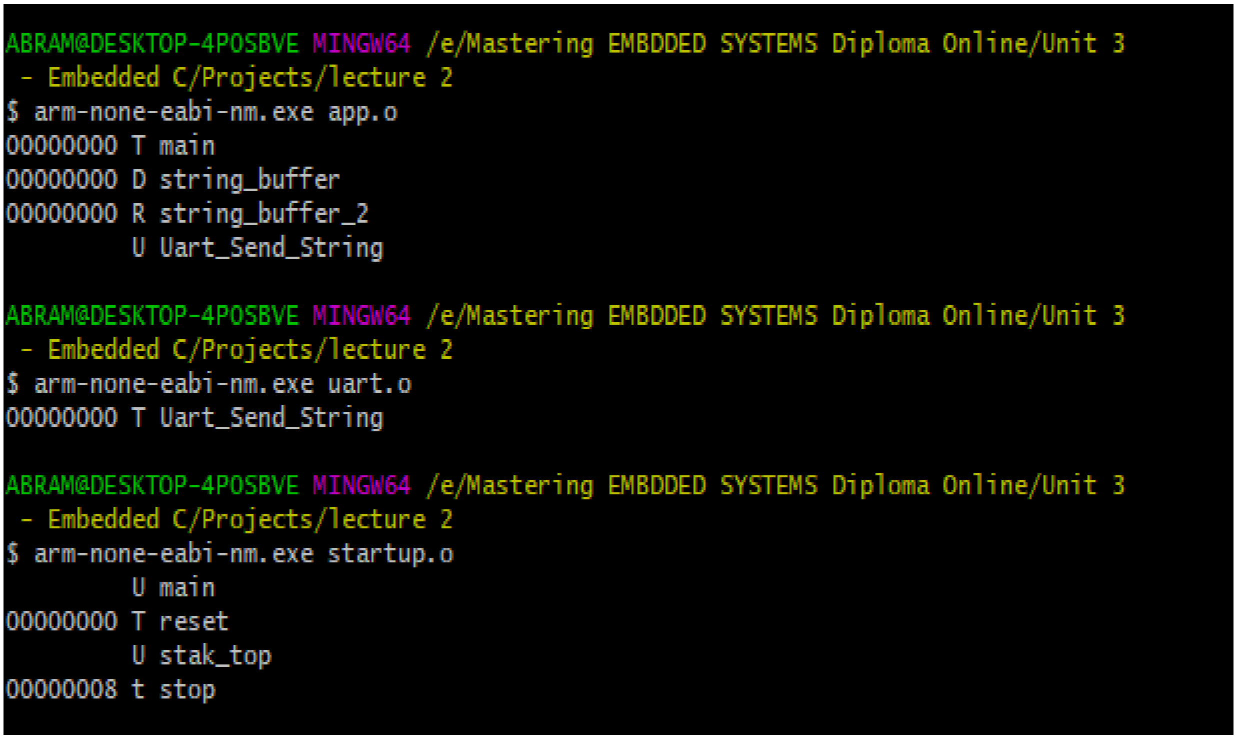
II. uart.o

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

III. startup.o

```
arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o startup.o
```

4. Symbols of app.o, uart.o and startup.o

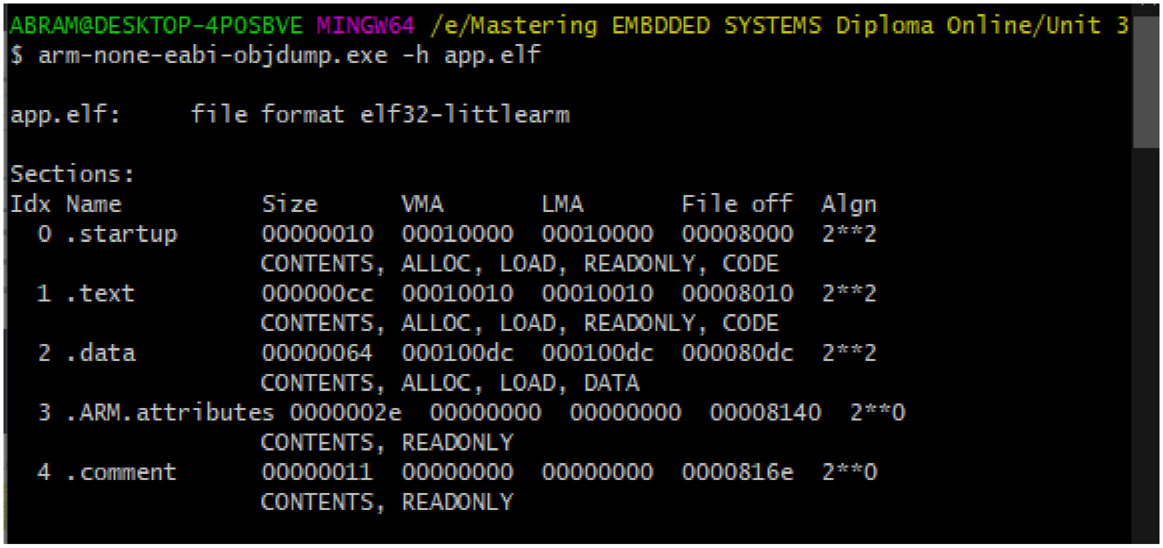


```
ABRAM@DESKTOP-4POSBVE MINGW64 /e/Mastering EMBDDED SYSTEMS Diploma Online/Unit 3
 - Embedded C/Projects/lecture 2
$ arm-none-eabi-nm.exe app.o
00000000 T main
00000000 D string_buffer
00000000 R string_buffer_2
         U Uart_Send_String

ABRAM@DESKTOP-4POSBVE MINGW64 /e/Mastering EMBDDED SYSTEMS Diploma Online/Unit 3
 - Embedded C/Projects/lecture 2
$ arm-none-eabi-nm.exe uart.o
00000000 T Uart_Send_String

ABRAM@DESKTOP-4POSBVE MINGW64 /e/Mastering EMBDDED SYSTEMS Diploma Online/Unit 3
 - Embedded C/Projects/lecture 2
$ arm-none-eabi-nm.exe startup.o
         U main
00000000 T reset
         U stak_top
00000008 t stop
```

5. Using Linker_Script.ld to get the executable file (app.elf)

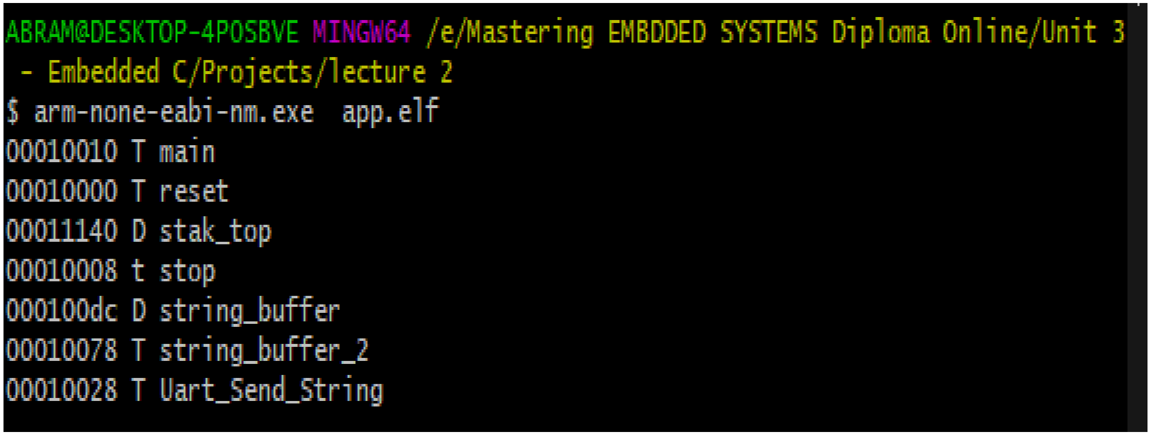
I. arm-none-eabi-ld -T test.ld -Map=output.map app.o uart.o startup.o -o app.elf

II. app.elf sections



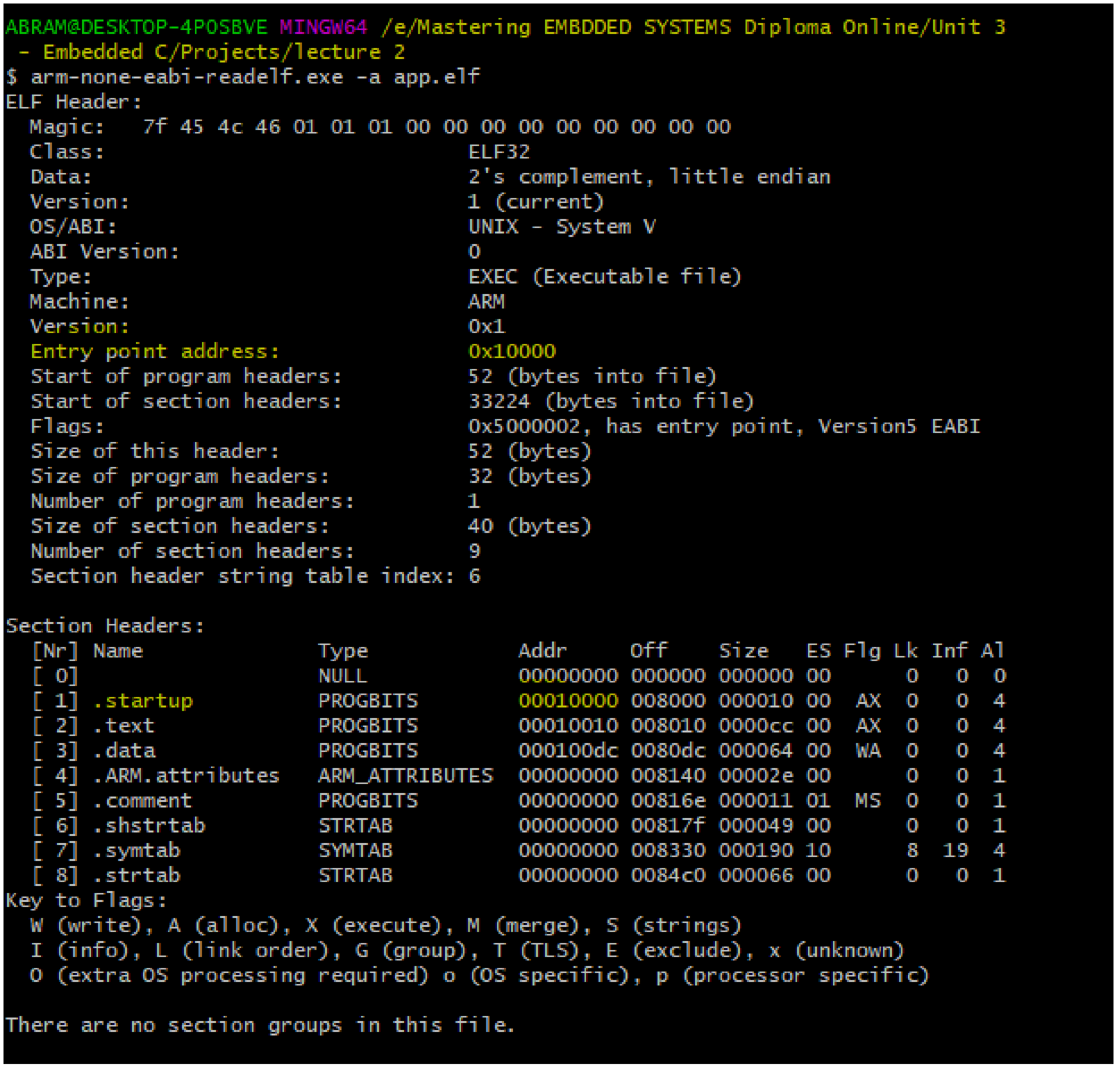
```
ABRAM@DESKTOP-4POSBVE MINGW64 /e/Mastering EMBDDED SYSTEMS Diploma Online/Unit 3
$ arm-none-eabi-objdump.exe -h app.elf

app.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         000000cc  00010010  00010010  00008010  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  000100dc  000100dc  000080dc  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 0000002e  00000000  00000000  00008140  2**0
                  CONTENTS, READONLY
  4 .comment      00000011  00000000  00000000  0000816e  2**0
                  CONTENTS, READONLY
```

III. app.elf symbols



```
ABRAM@DESKTOP-4POSBVE MINGW64 /e/Mastering EMBDDED SYSTEMS Diploma Online/Unit 3
 - Embedded C/Projects/lecture 2
$ arm-none-eabi-nm.exe  app.elf
00010010 T main
00010000 T reset
00011140 D stak_top
00010008 t stop
000100dc D string_buffer
00010078 T string_buffer_2
00010028 T Uart_Send_String
```

IV. Make sure ENTRY point at address 0x10000

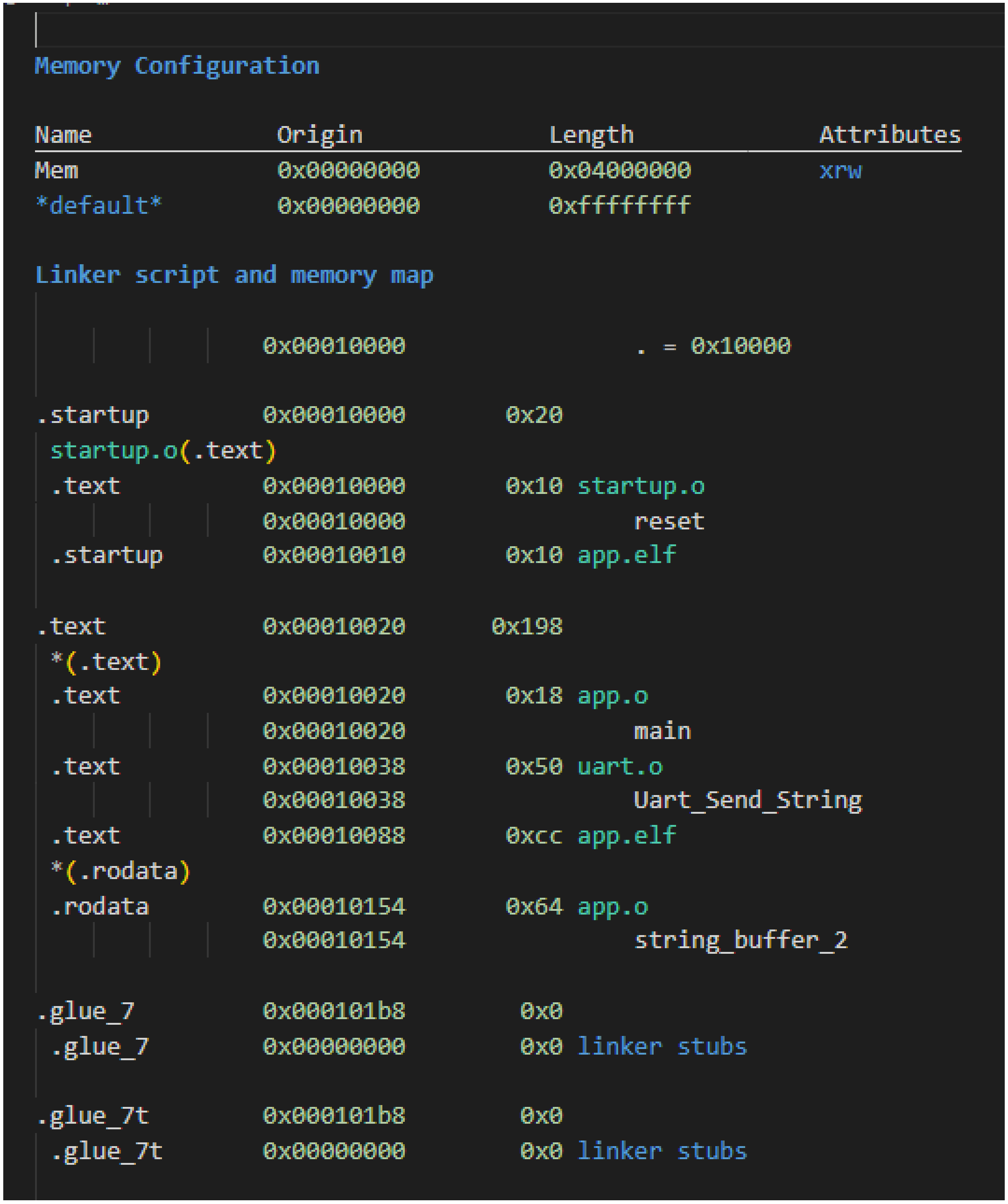


```
ABRAM@DESKTOP-4POSBVE MINGW64 /e/Mastering EMBDDED SYSTEMS Diploma Online/Unit 3
 - Embedded C/Projects/lecture 2
$ arm-none-eabi-readelf.exe -a app.elf
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              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:          33224 (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 0000cc 00  AX  0   0  4
  [ 3] .data             PROGBITS        000100dc 0080dc 000064 00  WA  0   0  4
  [ 4] .ARM.attributes   ARM_ATTRIBUTES  00000000 008140 00002e 00      0   0  1
  [ 5] .comment          PROGBITS        00000000 00816e 000011 01  MS  0   0  1
  [ 6] .shstrtab         STRTAB          00000000 00817f 000049 00      0   0  1
  [ 7] .symtab           SYMTAB          00000000 008330 000190 10      8  19  4
  [ 8] .strtab           STRTAB          00000000 0084c0 000066 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)

There are no section groups in this file.
```

6. Map file

I. arm-none-eabi-ld.exe -T Linker_Script.ld -Map=app.map

II. Analyze map_file.map

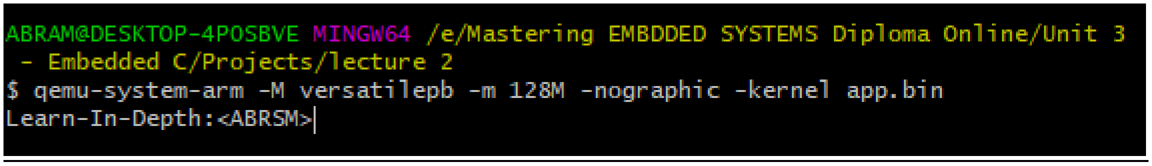


```
Memory Configuration

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

Linker script and memory map

                0x00010000                . = 0x10000

.startup        0x00010000       0x20
 startup.o(.text)
 .text          0x00010000       0x10 startup.o
                0x00010000                reset
 .startup       0x00010010       0x10 app.elf

.text           0x00010020      0x198
 *(.text)
 .text          0x00010020       0x18 app.o
                0x00010020                main
 .text          0x00010038       0x50 uart.o
                0x00010038                Uart_Send_String
 .text          0x00010088       0xcc app.elf
 *(.rodata)
 .rodata        0x00010154       0x64 app.o
                0x00010154                string_buffer_2

.glue_7         0x000101b8        0x0
 .glue_7        0x00000000        0x0 linker stubs

.glue_7t        0x000101b8        0x0
 .glue_7t       0x00000000        0x0 linker stubs
```

7. Get the binary file app.bin

```
arm-none-eabi-objcopy.exe -O binary app.elf app.bin
```

8. Burn the binary file using qemu



```
ABRAM@DESKTOP-4POSBVE MINGW64 /e/Mastering EMBDDED SYSTEMS Diploma Online/Unit 3
 - Embedded C/Projects/lecture 2
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel app.bin
Learn-In-Depth:<ABRSM>
```