# FILE CONTENT :

- app.c code
- uart.c code
- uart.h code
- startup.s code
- linker_script.ld code
- app.o Informations
- uart.o Informations
- startup.o Informations
- app.o disassemble
- app.o symbols
- uart.o symbols
- learn-in-depth.elf symbols
- learn-in-depth.elf Informations
- map file Informations
- Program Output

## app.c “ Code ” :

```
/**=========================================================
--    Author      :   YOUSSEF ADEL YOUSSEF
--  Description  :   Write a C Program to send a “learn-in-depth:<your_Name>”
                     by using UART
=========================================================**/

#include<string.h>
#include"uart.h"

unsigned char Buffer1_str[100] = "Leran-in-depth : Youssef Adel";
unsigned char const Buffer2_str[100] = "for test only";

void main(void)
{
	unsigned int c , x=0;
	while(Buffer1_str[x] != '\0')
	{
		if(Buffer1_str[x] == ' ')
			c++;
		else
			c++;
		x++;
	}
	Uart_Send_String(c,Buffer1_str);
}
```

## uart.c " Code " :

```
#include"uart.h"

#define UARTDR *((volatile unsigned int* const)((unsigned int*)(0x101f1000)))

void Uart_Send_String(unsigned int i , unsigned char *P_str)
{
        while(i != 0)
        {
                UARTDR = (unsigned int)(*P_str);
                *P_str++;
                i--;
        }
}
```

## uart.h “ Code ” :

```
#ifndef _UART_H_
#define _UART_H_

void Uart_Send_String(unsigned int i , unsigned char *P_str);

#endif
```

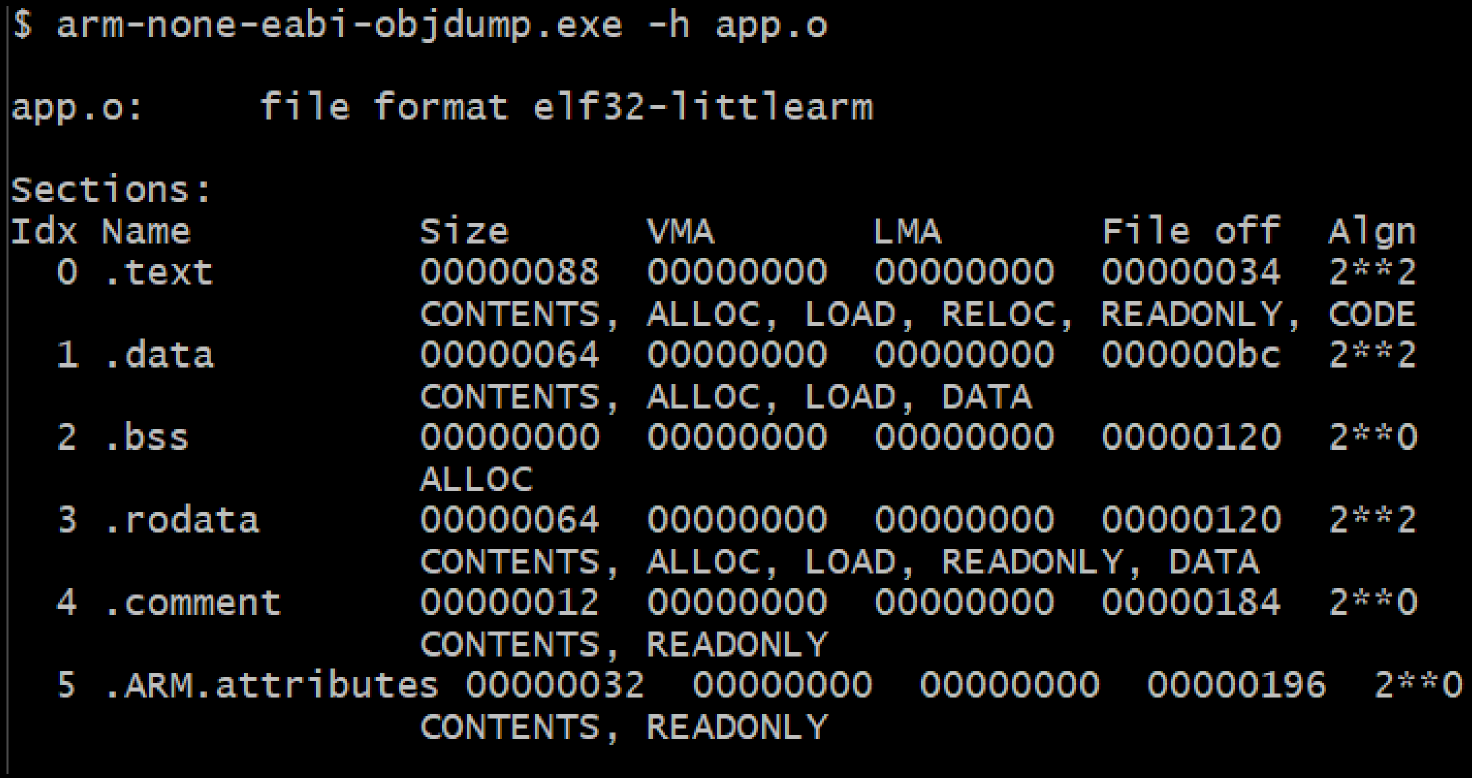
## startup.s “ Code ” :

```
.globl reset
reset:
    ldr sp, =stack_top
    bl main
stop: b stop
```

## linker_script.ld “ Code ” :

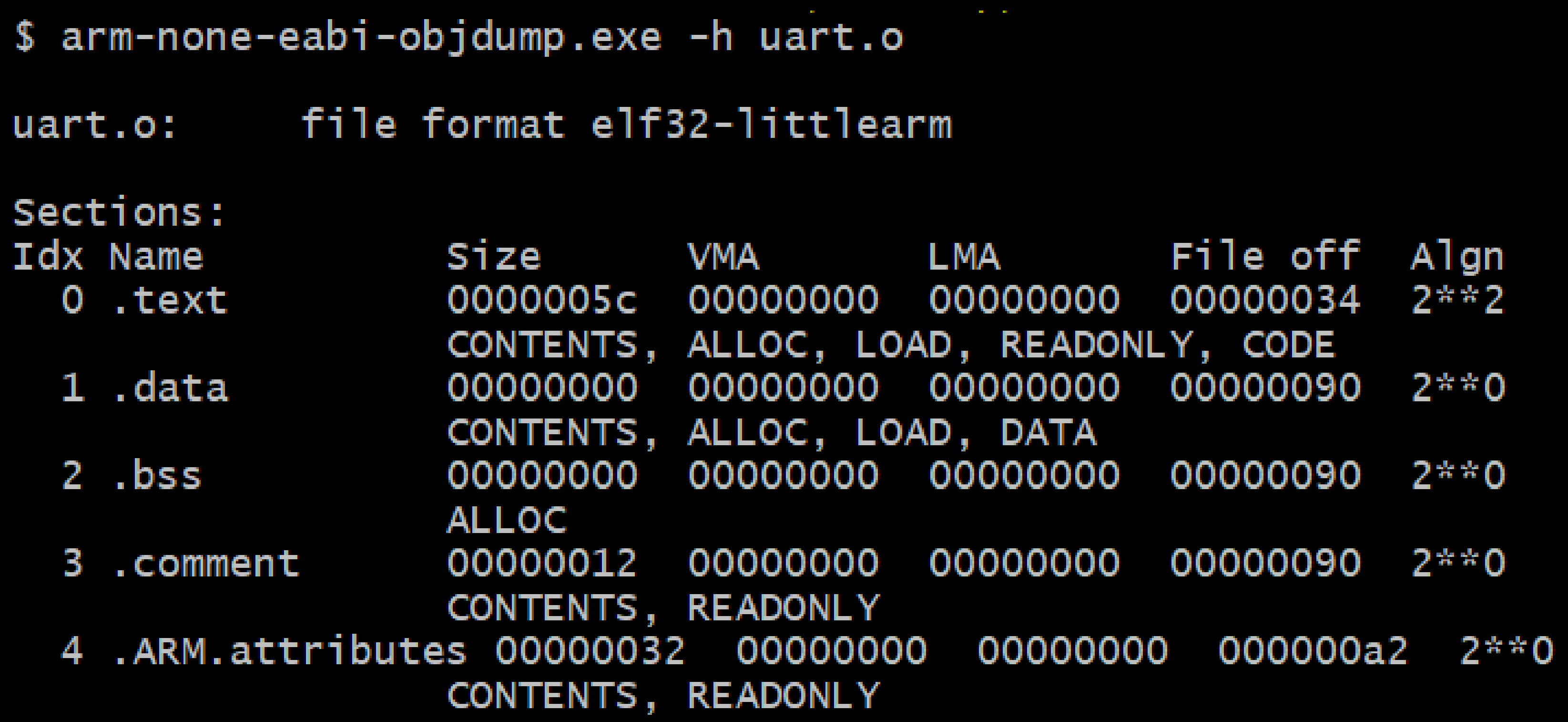
```
ENTRY(reset)

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

SECTIONS
{
        . = 0x10000;
        .startup . :
        {
           startup.o(.text)
        }> Mem
        .text :
        {
           *(.text) *(.rodata)
        }> Mem
        .data :
        {
           *(.data)
        }> Mem
        .bss :
        {
           *(.bss) *(COMMON)
        }> Mem
        . = . + 0x1000; /* 1KB of Stack Memory */
        stack_top = . ;
}
```

## app.o Informations :



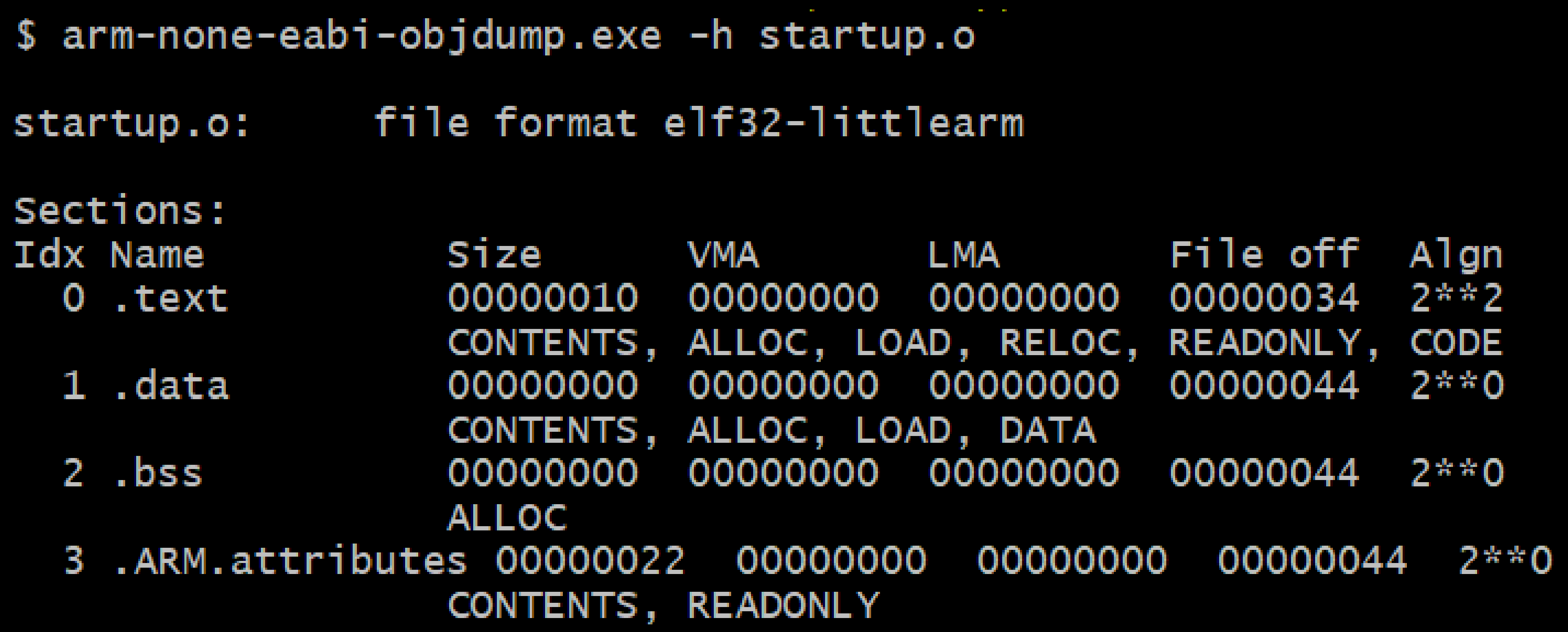
```
$ arm-none-eabi-objdump.exe -h app.o

app.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000088  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000064  00000000  00000000  000000bc  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000120  2**0
                  ALLOC
  3 .rodata       00000064  00000000  00000000  00000120  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      00000012  00000000  00000000  00000184  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 00000032  00000000  00000000  00000196  2**0
                  CONTENTS, READONLY
```

## uart.o Informations :



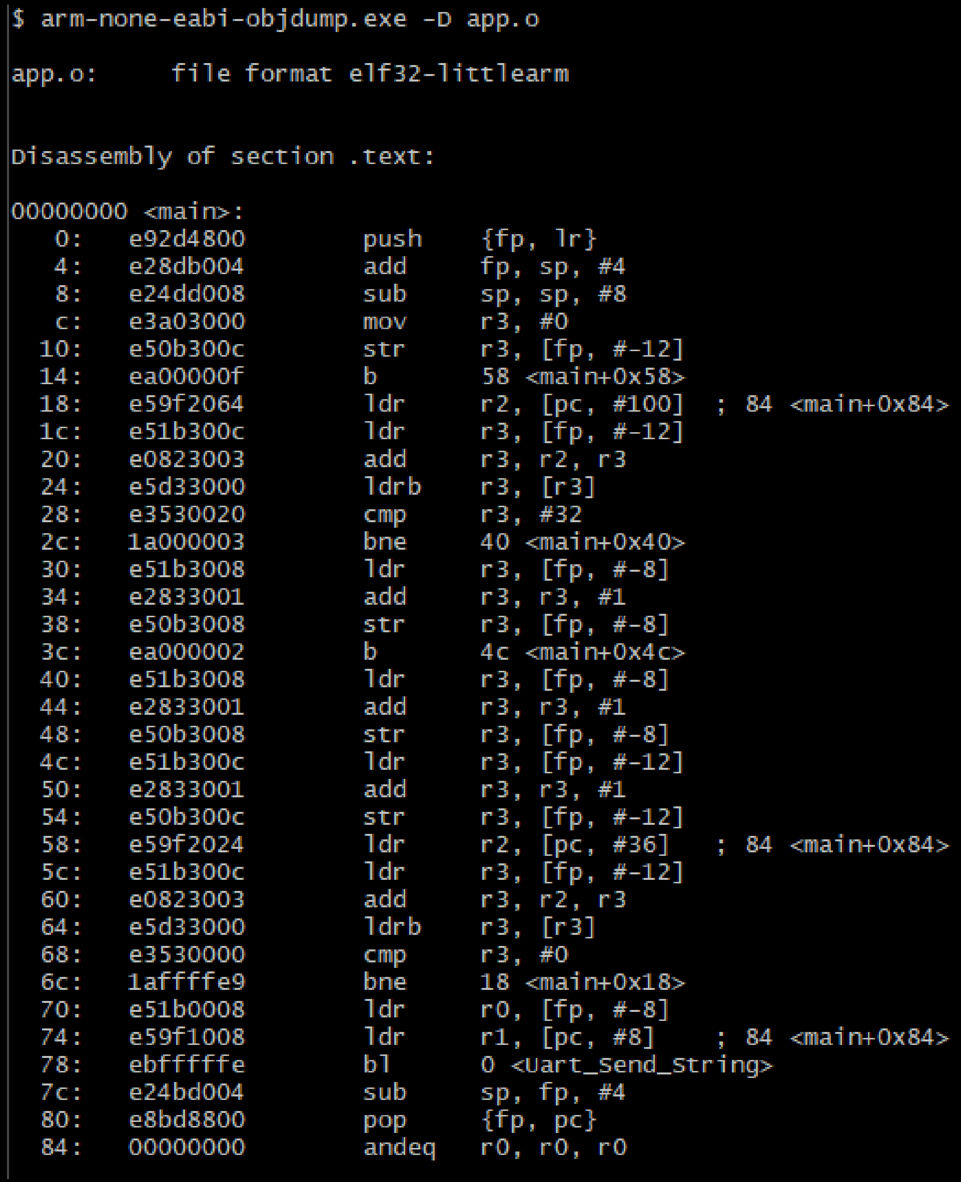
```
$ arm-none-eabi-objdump.exe -h uart.o

uart.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000005c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000090  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000090  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  00000090  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000032  00000000  00000000  000000a2  2**0
                  CONTENTS, READONLY
```

## startup.o Informations :

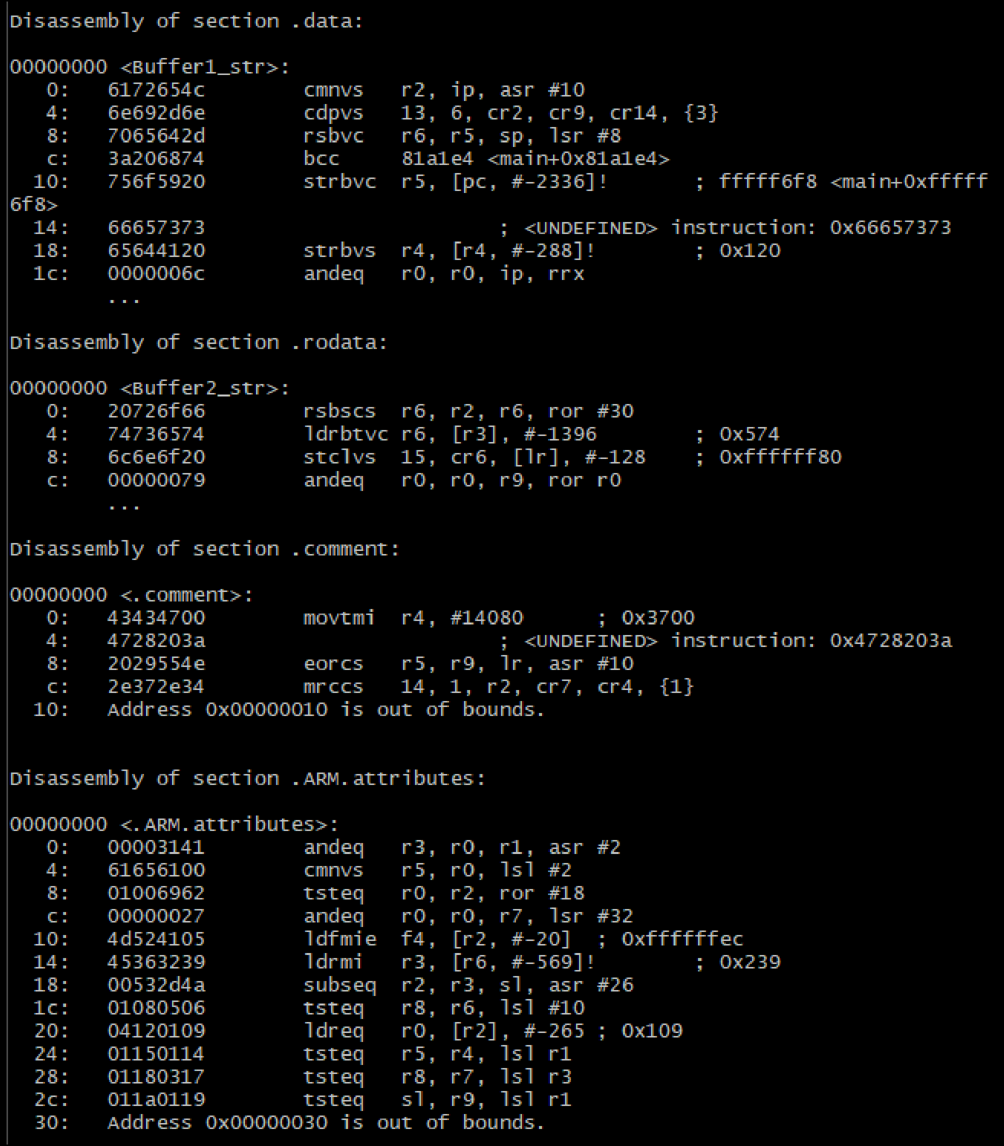


```
$ arm-none-eabi-objdump.exe -h startup.o

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
```

# app.o “ Disassemble Code ” :

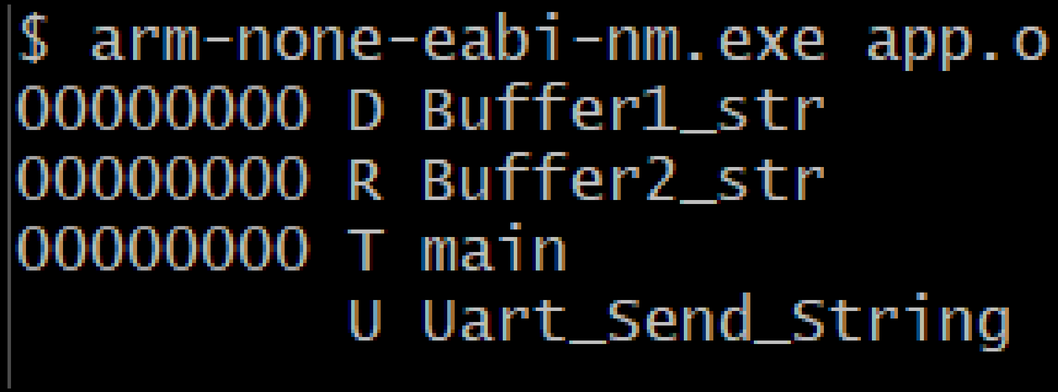


```
$ arm-none-eabi-objdump.exe -D app.o

app.o:     file format elf32-littlearm


Disassembly of section .text:

00000000 <main>:
   0:   e92d4800        push    {fp, lr}
   4:   e28db004        add     fp, sp, #4
   8:   e24dd008        sub     sp, sp, #8
   c:   e3a03000        mov     r3, #0
  10:   e50b300c        str     r3, [fp, #-12]
  14:   ea00000f        b       58 <main+0x58>
  18:   e59f2064        ldr     r2, [pc, #100]  ; 84 <main+0x84>
  1c:   e51b300c        ldr     r3, [fp, #-12]
  20:   e0823003        add     r3, r2, r3
  24:   e5d33000        ldrb    r3, [r3]
  28:   e3530020        cmp     r3, #32
  2c:   1a000003        bne     40 <main+0x40>
  30:   e51b3008        ldr     r3, [fp, #-8]
  34:   e2833001        add     r3, r3, #1
  38:   e50b3008        str     r3, [fp, #-8]
  3c:   ea000002        b       4c <main+0x4c>
  40:   e51b3008        ldr     r3, [fp, #-8]
  44:   e2833001        add     r3, r3, #1
  48:   e50b3008        str     r3, [fp, #-8]
  4c:   e51b300c        ldr     r3, [fp, #-12]
  50:   e2833001        add     r3, r3, #1
  54:   e50b300c        str     r3, [fp, #-12]
  58:   e59f2024        ldr     r2, [pc, #36]   ; 84 <main+0x84>
  5c:   e51b300c        ldr     r3, [fp, #-12]
  60:   e0823003        add     r3, r2, r3
  64:   e5d33000        ldrb    r3, [r3]
  68:   e3530000        cmp     r3, #0
  6c:   1affffe9        bne     18 <main+0x18>
  70:   e51b0008        ldr     r0, [fp, #-8]
  74:   e59f1008        ldr     r1, [pc, #8]    ; 84 <main+0x84>
  78:   ebfffffe        bl      0 <Uart_Send_String>
  7c:   e24bd004        sub     sp, fp, #4
  80:   e8bd8800        pop     {fp, pc}
  84:   00000000        andeq   r0, r0, r0
```



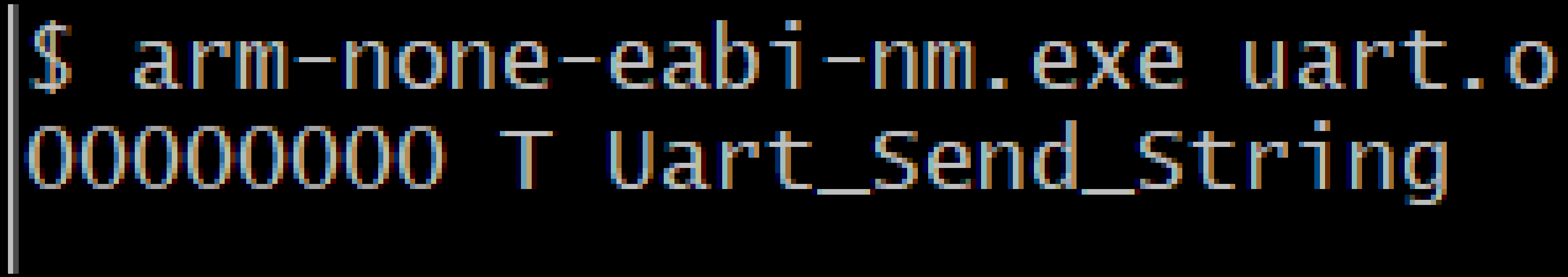
```
Disassembly of section .data:

00000000 <Buffer1_str>:
   0:   6172654c        cmnvs   r2, ip, asr #10
   4:   6e692d6e        cdpvs   13, 6, cr2, cr9, cr14, {3}
   8:   7065642d        rsbvc   r6, r5, sp, lsr #8
   c:   3a206874        bcc     81a1e4 <main+0x81a1e4>
  10:   756f5920        strbvc  r5, [pc, #-2336]!       ; fffff6f8 <main+0xfffff
6f8>
  14:   66657373                        ; <UNDEFINED> instruction: 0x66657373
  18:   65644120        strbvs  r4, [r4, #-288]!        ; 0x120
  1c:   0000006c        andeq   r0, r0, ip, rrx
        ...

Disassembly of section .rodata:

00000000 <Buffer2_str>:
   0:   20726f66        rsbscs  r6, r2, r6, ror #30
   4:   74736574        ldrbtvc r6, [r3], #-1396        ; 0x574
   8:   6c6e6f20        stclvs  15, cr6, [lr], #-128    ; 0xffffff80
   c:   00000079        andeq   r0, r0, r9, ror 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
   4:   61656100        cmnvs   r5, r0, lsl #2
   8:   01006962        tsteq   r0, r2, ror #18
   c:   00000027        andeq   r0, r0, r7, lsr #32
  10:   4d524105        ldfmie  f4, [r2, #-20]  ; 0xffffffec
  14:   45363239        ldrmi   r3, [r6, #-569]!        ; 0x239
  18:   00532d4a        subseq  r2, r3, sl, asr #26
  1c:   01080506        tsteq   r8, r6, lsl #10
  20:   04120109        ldreq   r0, [r2], #-265 ; 0x109
  24:   01150114        tsteq   r5, r4, lsl r1
  28:   01180317        tsteq   r8, r7, lsl r3
  2c:   011a0119        tsteq   sl, r9, lsl r1
  30:   Address 0x00000030 is out of bounds.
```

## app.o Symbols :



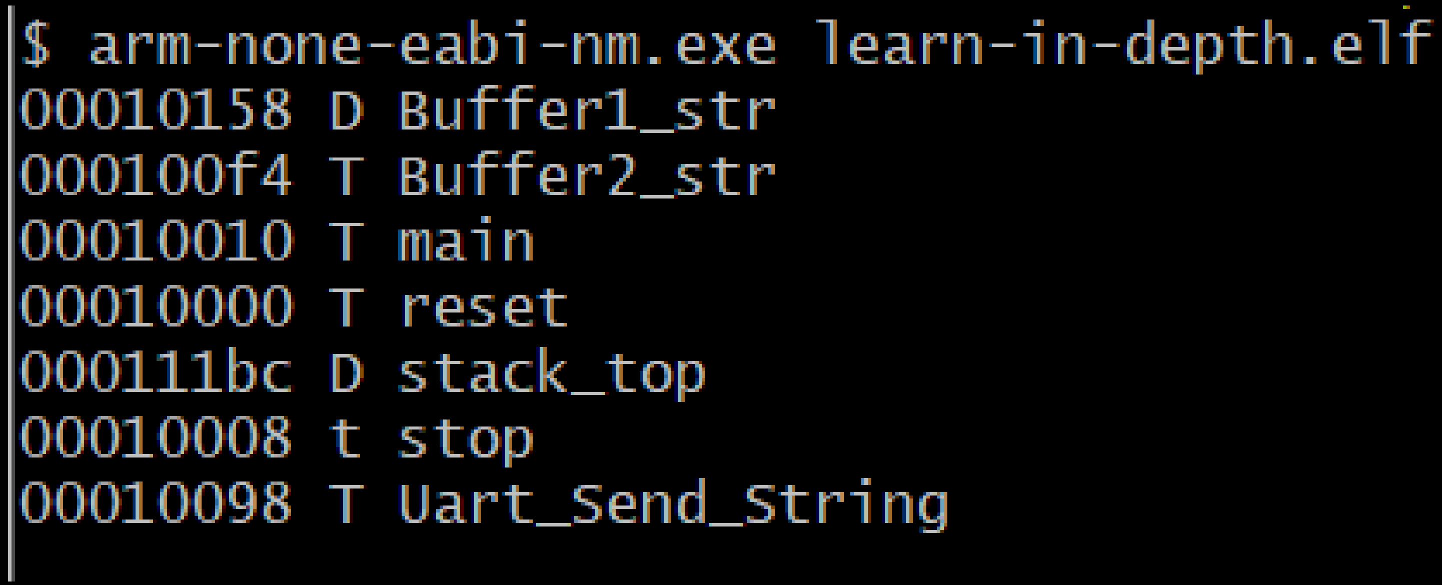
```
$ arm-none-eabi-nm.exe app.o
00000000 D Buffer1_str
00000000 R Buffer2_str
00000000 T main
         U Uart_Send_String
```

## uart.o Symbols :



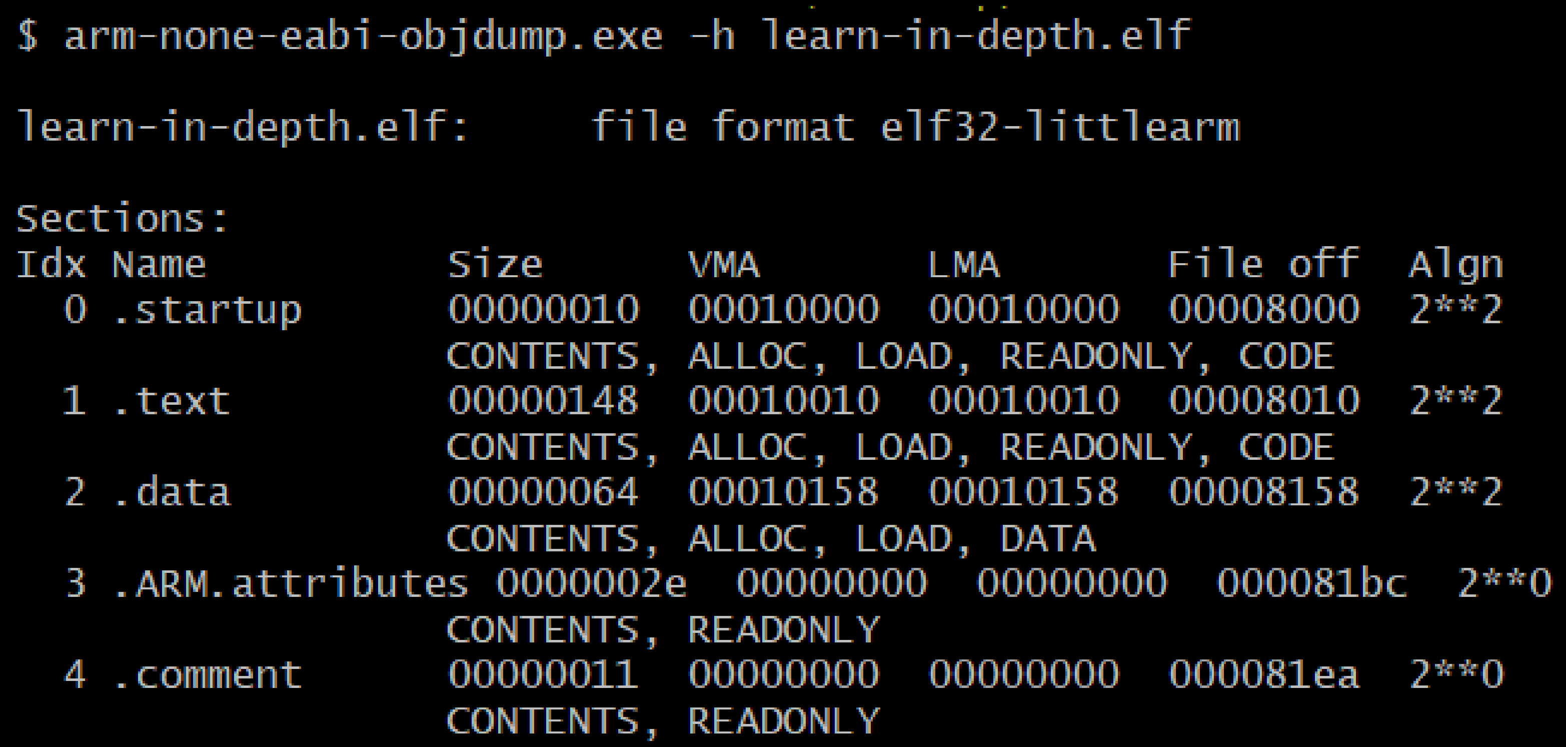
```
$ arm-none-eabi-nm.exe uart.o
00000000 T Uart_Send_String
```

## Learn-in-depth.elf Symbols :



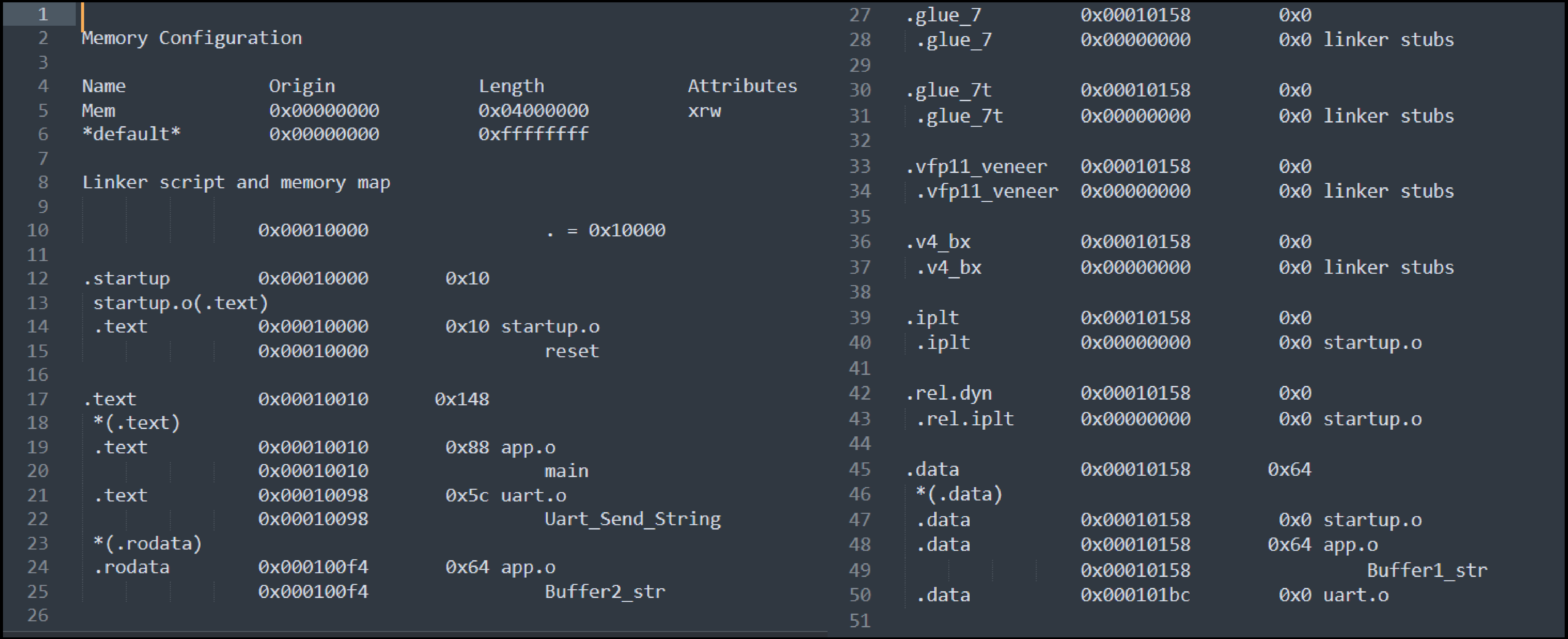
```
$ arm-none-eabi-nm.exe learn-in-depth.elf
00010158 D Buffer1_str
000100f4 T Buffer2_str
00010010 T main
00010000 T reset
000111bc D stack_top
00010008 t stop
00010098 T Uart_Send_String
```

## Learn-in-depth.elf Informations :



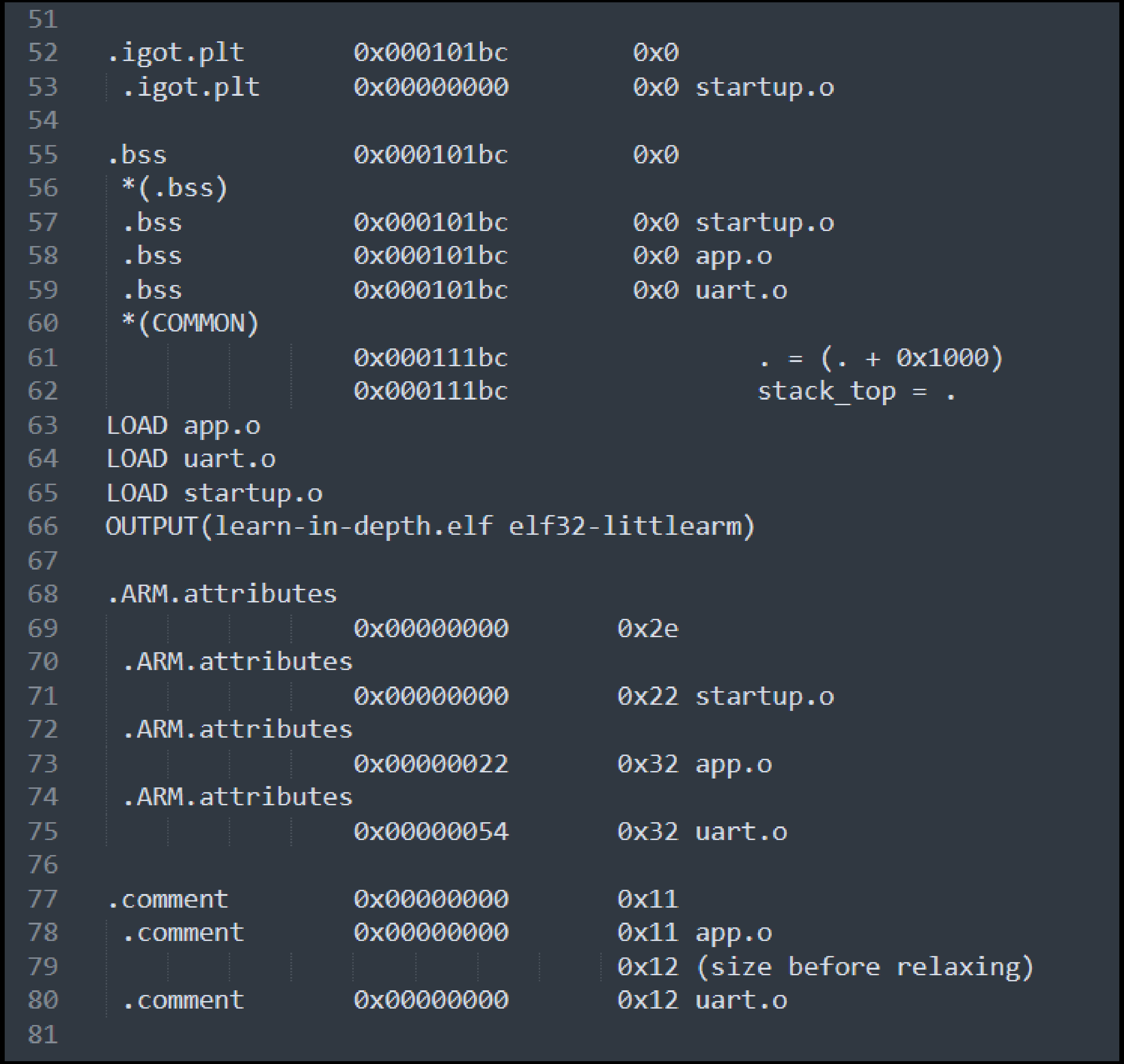
```
$ arm-none-eabi-objdump.exe -h learn-in-depth.elf

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         00000148  00010010  00010010  00008010  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  00010158  00010158  00008158  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 0000002e  00000000  00000000  000081bc  2**0
                  CONTENTS, READONLY
  4 .comment      00000011  00000000  00000000  000081ea  2**0
                  CONTENTS, READONLY
```

# map file Informations :



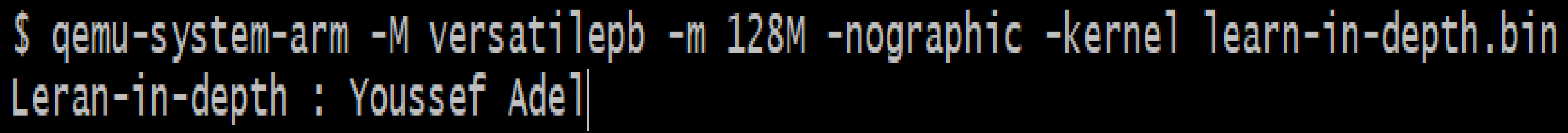
```

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      0x148
 *(.text)
 .text          0x00010010       0x88 app.o
                0x00010010                main
 .text          0x00010098       0x5c uart.o
                0x00010098                Uart_Send_String
 *(.rodata)
 .rodata        0x000100f4       0x64 app.o
                0x000100f4                Buffer2_str

.glue_7         0x00010158        0x0
 .glue_7        0x00000000        0x0 linker stubs

.glue_7t        0x00010158        0x0
 .glue_7t       0x00000000        0x0 linker stubs

.vfp11_veneer   0x00010158        0x0
 .vfp11_veneer  0x00000000        0x0 linker stubs

.v4_bx          0x00010158        0x0
 .v4_bx         0x00000000        0x0 linker stubs

.iplt           0x00010158        0x0
 .iplt          0x00000000        0x0 startup.o

.rel.dyn        0x00010158        0x0
 .rel.iplt      0x00000000        0x0 startup.o

.data           0x00010158       0x64
 *(.data)
 .data          0x00010158        0x0 startup.o
 .data          0x00010158       0x64 app.o
                0x00010158                Buffer1_str
 .data          0x000101bc        0x0 uart.o

```

## map file Informations Cont. :



```

 .igot.plt       0x000101bc        0x0
  .igot.plt      0x00000000        0x0 startup.o

 .bss            0x000101bc        0x0
  *(.bss)
  .bss           0x000101bc        0x0 startup.o
  .bss           0x000101bc        0x0 app.o
  .bss           0x000101bc        0x0 uart.o
  *(COMMON)
                 0x000111bc                . = (. + 0x1000)
                 0x000111bc                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

```

## Program Output :



```
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin
Leran-in-depth : Youssef Adel
```