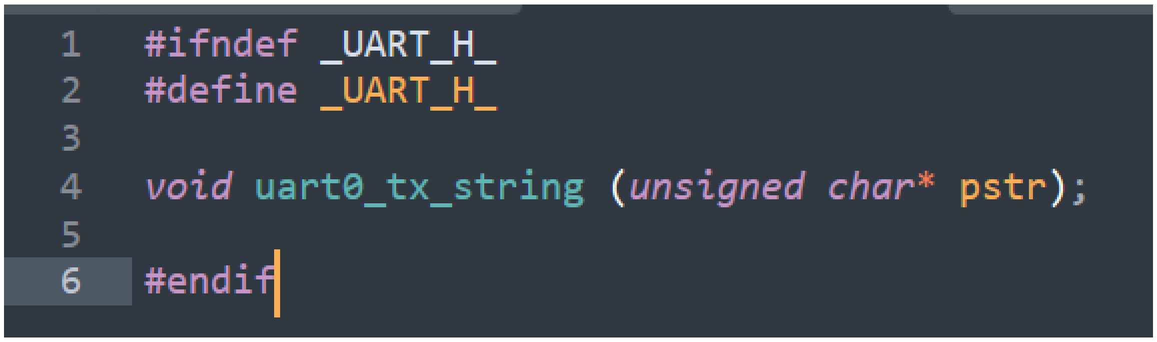
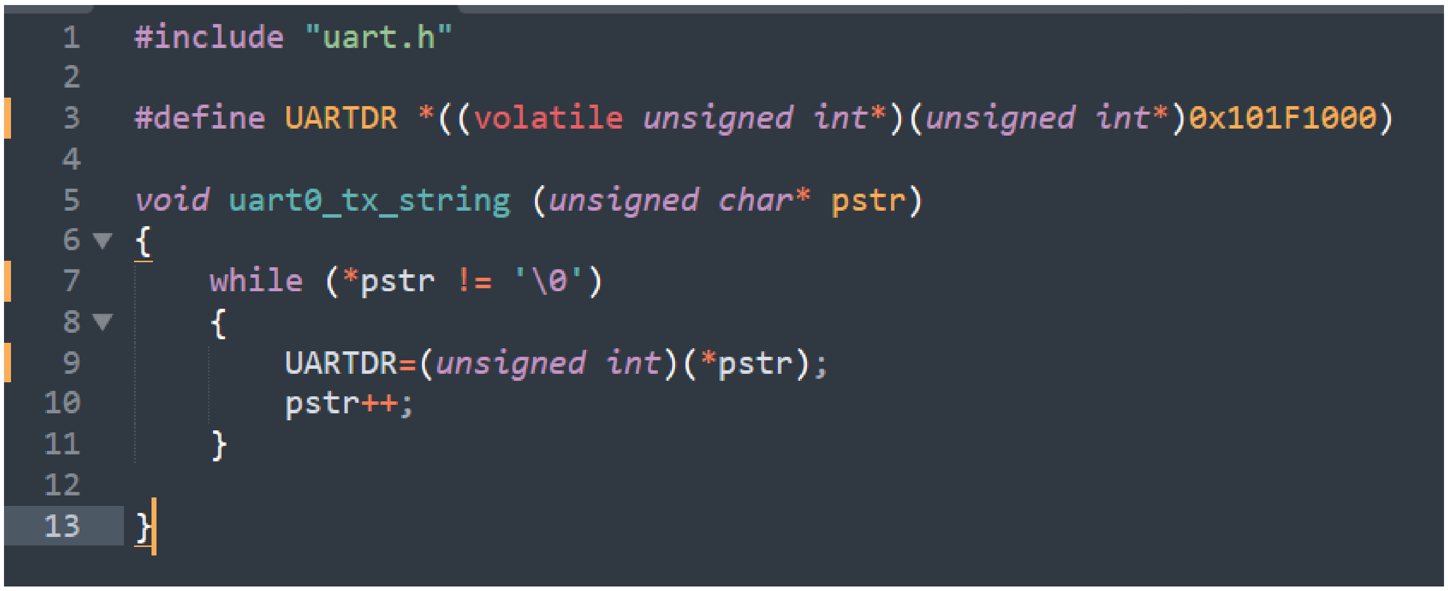
uart.h



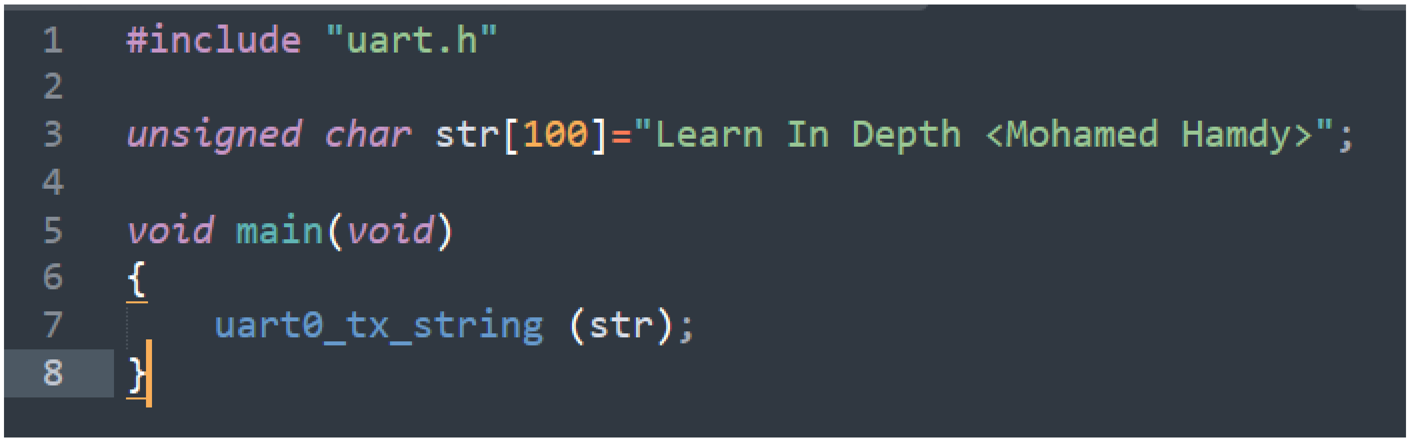
```c
#ifndef _UART_H_
#define _UART_H_

void uart0_tx_string (unsigned char* pstr);

#endif
```

uart.c



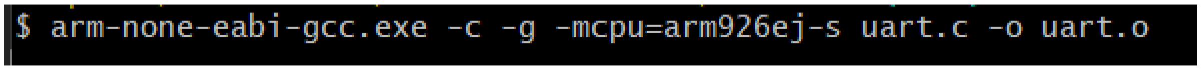
```c
#include "uart.h"

#define UARTDR *((volatile unsigned int*)(unsigned int*)0x101F1000)

void uart0_tx_string (unsigned char* pstr)
{
    while (*pstr != '\0')
    {
        UARTDR=(unsigned int)(*pstr);
        pstr++;
    }

}
```

app.c

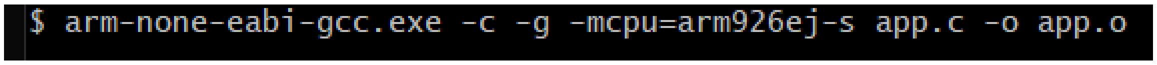


```c
#include "uart.h"

unsigned char str[100]="Learn In Depth <Mohamed Hamdy>";

void main(void)
{
    uart0_tx_string (str);
}
```

uart.o & app.o commands

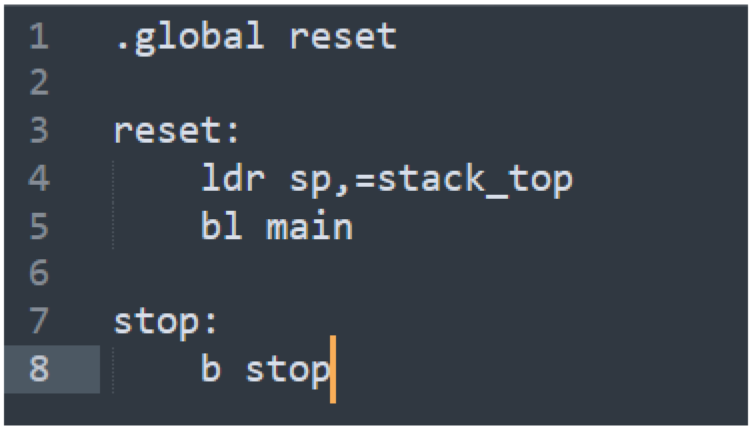


```
$ arm-none-eabi-gcc.exe -c -g -mcpu=arm926ej-s uart.c -o uart.o
```



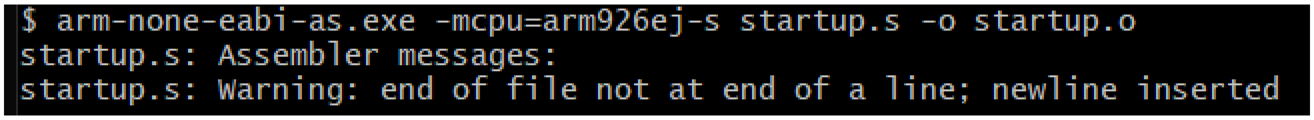
```
$ arm-none-eabi-gcc.exe -c -g -mcpu=arm926ej-s app.c -o app.o
```

startup.s



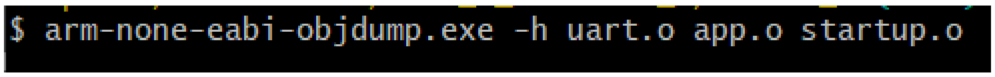
```
.global reset

reset:
    ldr sp,=stack_top
    bl main

stop:
    b stop
```

startup.o command

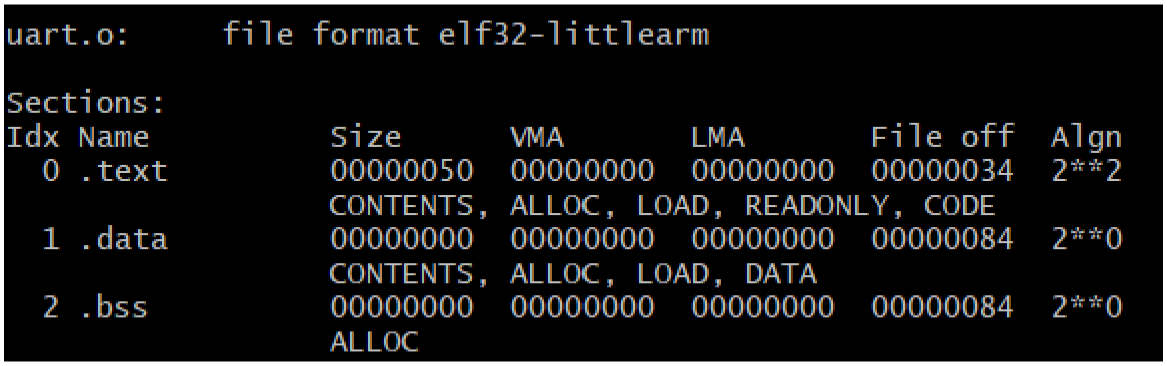


```
$ arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o startup.o
startup.s: Assembler messages:
startup.s: Warning: end of file not at end of a line; newline inserted
```

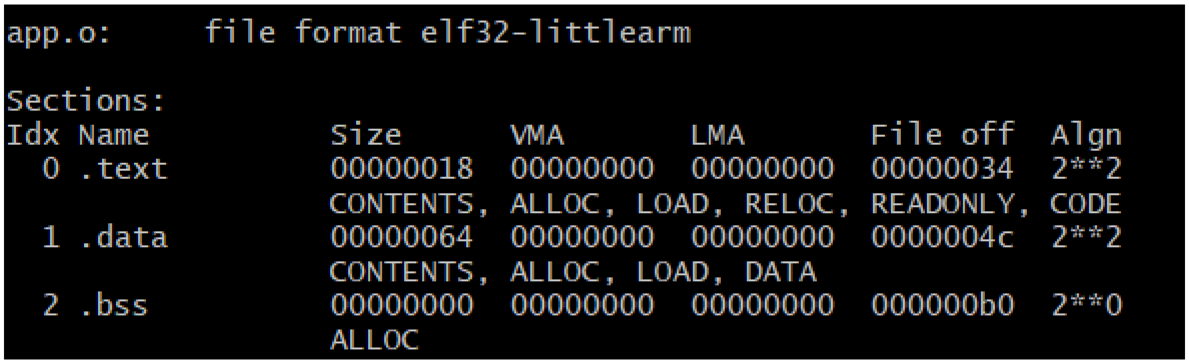
Sections in uart.o & app.o & startup.o



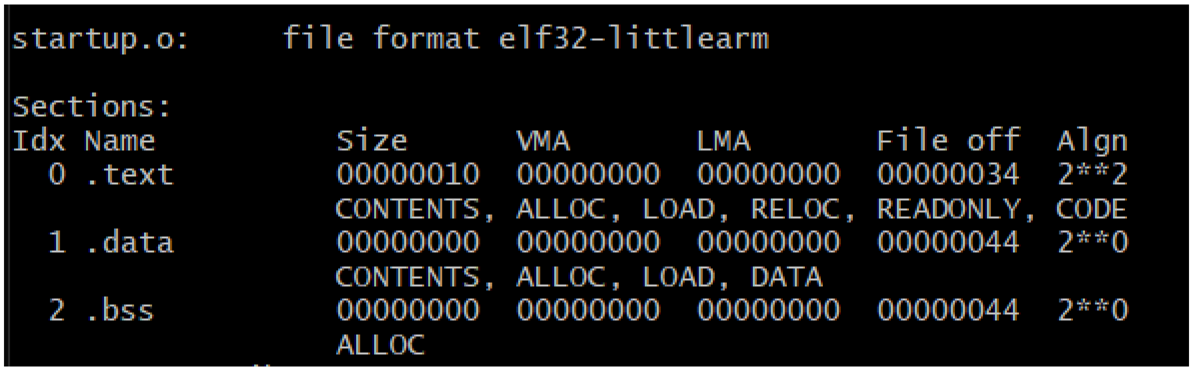
```
$ arm-none-eabi-objdump.exe -h uart.o app.o startup.o
```



```
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
```

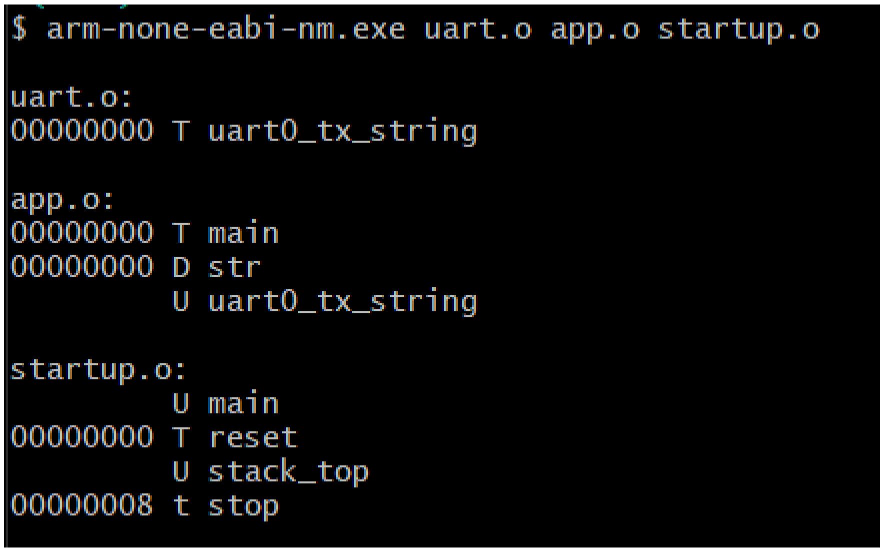


```
app.o:     file format elf32-littlearm

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
```



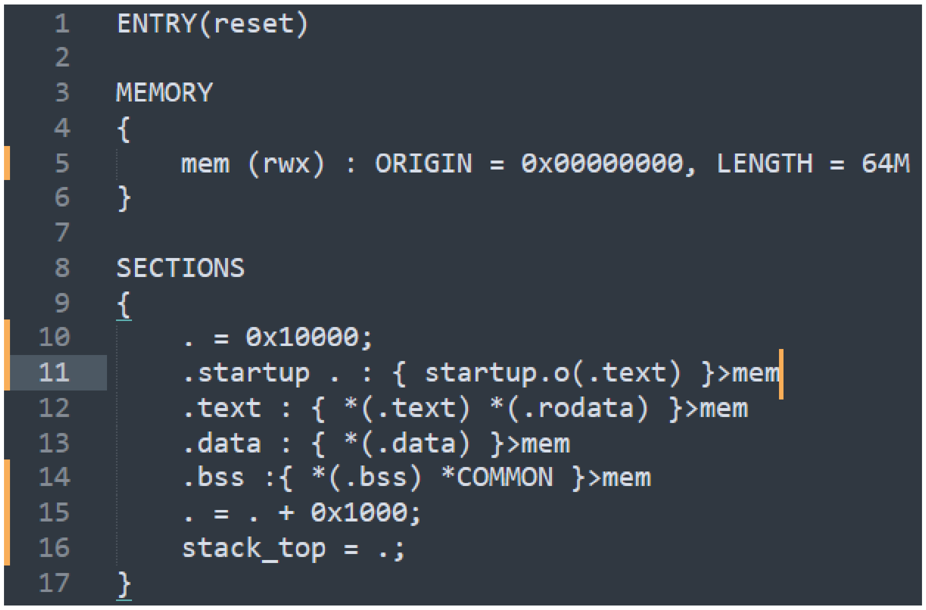
```
startup.o:     file format elf32-littlearm

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
```

Symbols in uart.o & app.o & startup.o



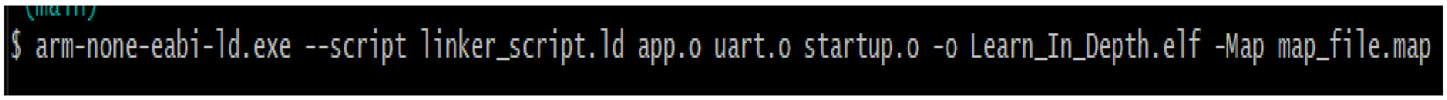
```
$ arm-none-eabi-nm.exe uart.o app.o startup.o

uart.o:
00000000 T uart0_tx_string

app.o:
00000000 T main
00000000 D str
         U uart0_tx_string

startup.o:
         U main
00000000 T reset
         U stack_top
00000008 t stop
```

linker_script.ld



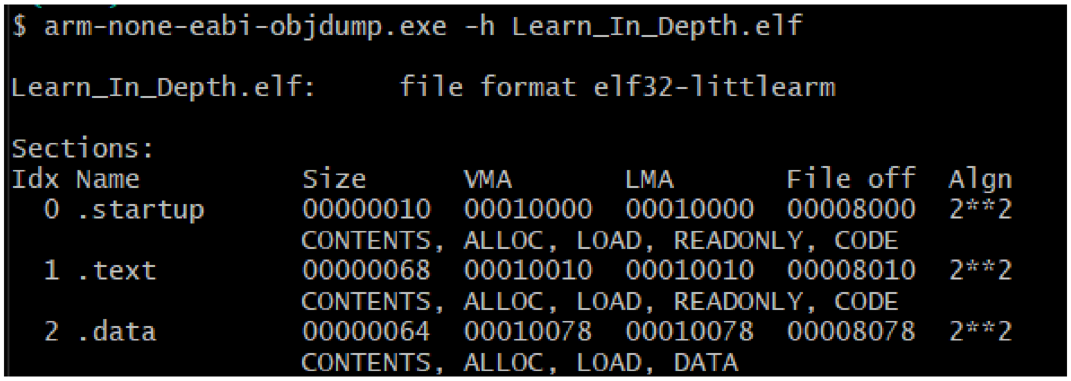
```
ENTRY(reset)

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;
    stack_top = .;
}
```

Learn_IN_Depth.elf command & map_file.map



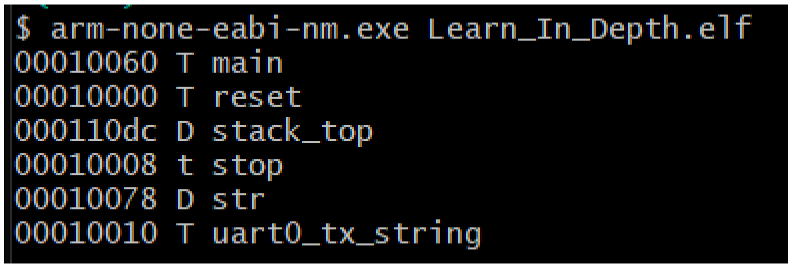
```
$ arm-none-eabi-ld.exe --script linker_script.ld app.o uart.o startup.o -o Learn_In_Depth.elf -Map map_file.map
```

Sections in Learn_IN_Depth.elf



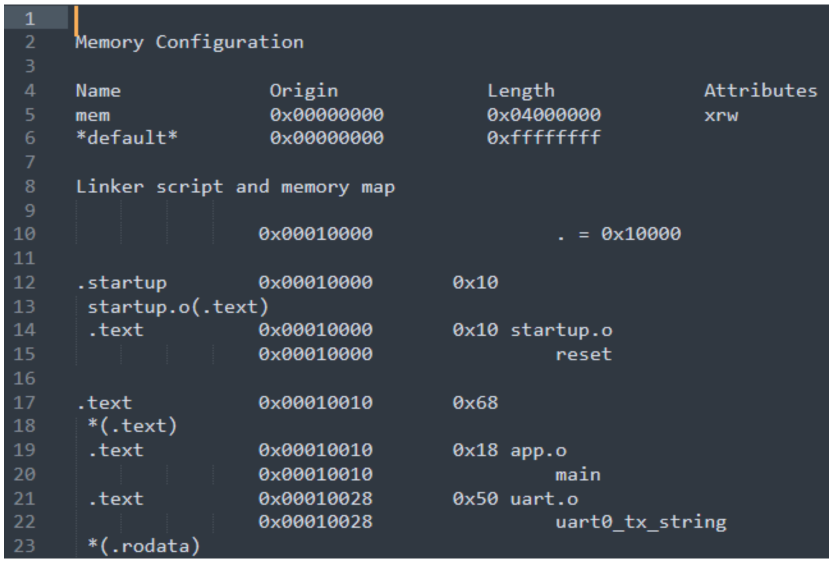
```
$ arm-none-eabi-objdump.exe -h Learn_In_Depth.elf

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
```

Symbols in Learn_IN_Depth.elf

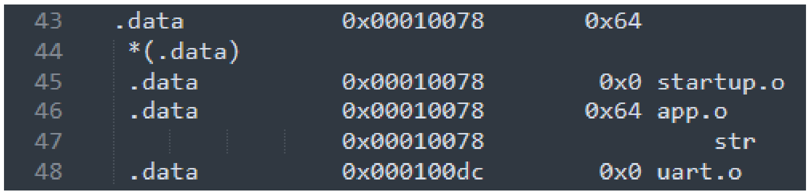


```
$ arm-none-eabi-nm.exe Learn_In_Depth.elf
00010060 T main
00010000 T reset
000110dc D stack_top
00010008 t stop
00010078 D str
00010010 T uart0_tx_string
```

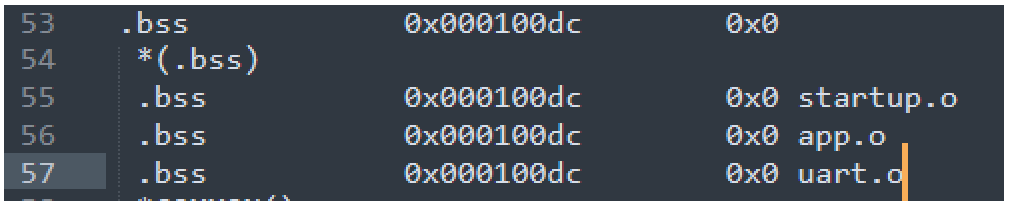
map_file.map



```

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       0x68
 *(.text)
 .text          0x00010010       0x18 app.o
                0x00010010                main
 .text          0x00010028       0x50 uart.o
                0x00010028                uart0_tx_string
 *(.rodata)
```

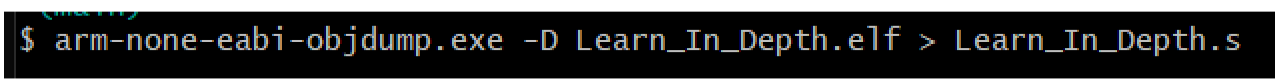


```
.data           0x00010078       0x64
 *(.data)
 .data          0x00010078        0x0 startup.o
 .data          0x00010078       0x64 app.o
                0x00010078                str
 .data          0x000100dc        0x0 uart.o
```

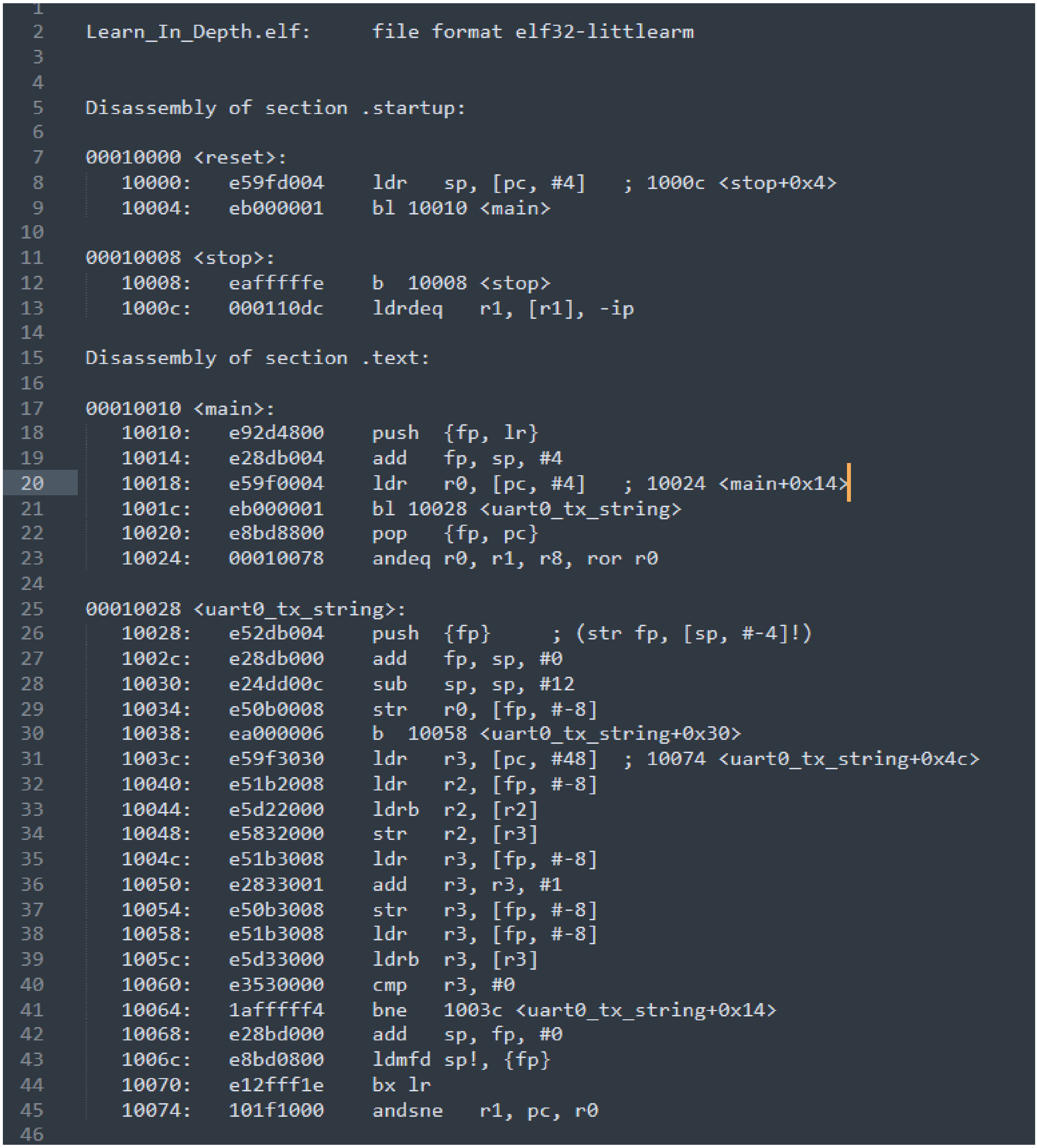


```
.bss            0x000100dc        0x0
 *(.bss)
 .bss           0x000100dc        0x0 startup.o
 .bss           0x000100dc        0x0 app.o
 .bss           0x000100dc        0x0 uart.o
```

Disassembly of Learn_IN_Depth.elf

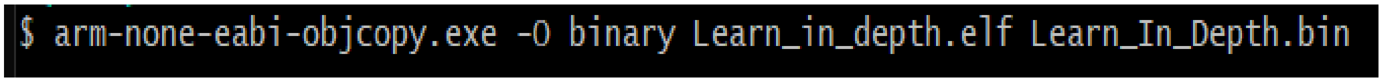


```
$ arm-none-eabi-objdump.exe -D Learn_In_Depth.elf > Learn_In_Depth.s
```



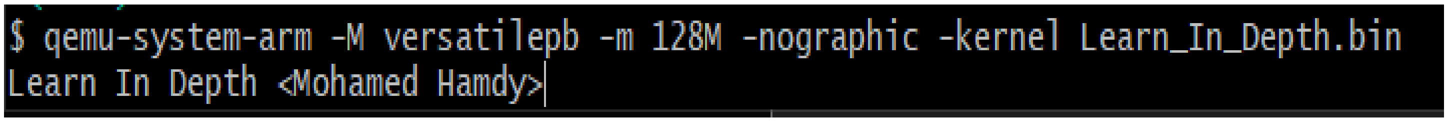
```

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_tx_string>
   10020:	e8bd8800 	pop	{fp, pc}
   10024:	00010078 	andeq	r0, r1, r8, ror r0

00010028 <uart0_tx_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_tx_string+0x30>
   1003c:	e59f3030 	ldr	r3, [pc, #48]	; 10074 <uart0_tx_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_tx_string+0x14>
   10068:	e28bd000 	add	sp, fp, #0
   1006c:	e8bd0800 	ldmfd	sp!, {fp}
   10070:	e12fff1e 	bx	lr
   10074:	101f1000 	andsne	r1, pc, r0

```

Strip binary from Learn_IN_Depth.elf



```
$ arm-none-eabi-objcopy.exe -O binary Learn_in_depth.elf Learn_In_Depth.bin
```

Run Learn_IN_Depth.bin using QEMU



```
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel Learn_In_Depth.bin
Learn In Depth <Mohamed Hamdy>
```