Embedded_C Lesson_2 LAP1

In this lab we need to create a **bare-metal** Software to send a

"learn-in-depth: Mario_Adel " string using UART

Runs on Board: (ARM VersatilePB) ARM926EJ-S core

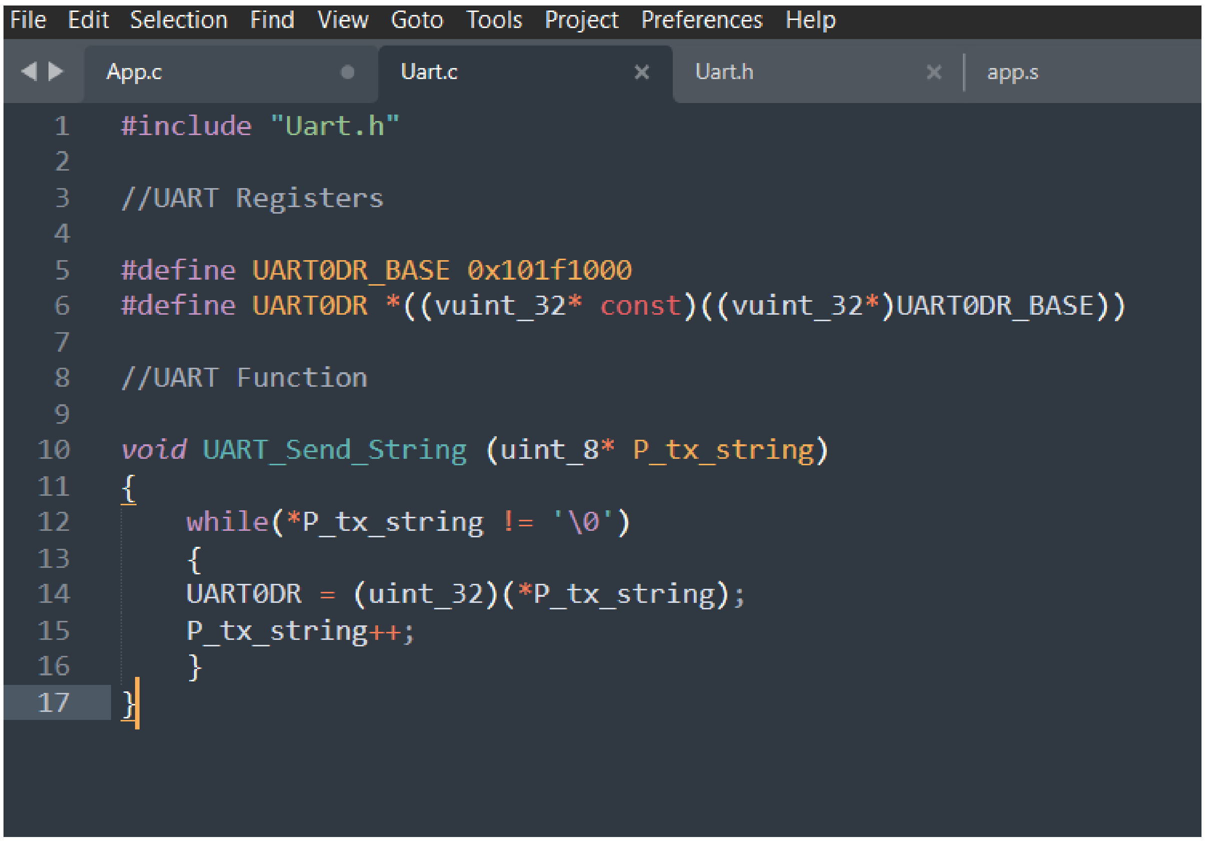
➢ **Write uart.c**, **uart.h** & **app.c**   files:

Create **uart.c**, **uart.h** & **app.c**   files using terminal command:

```
$ touch uart.c  uart.h  App.c
```

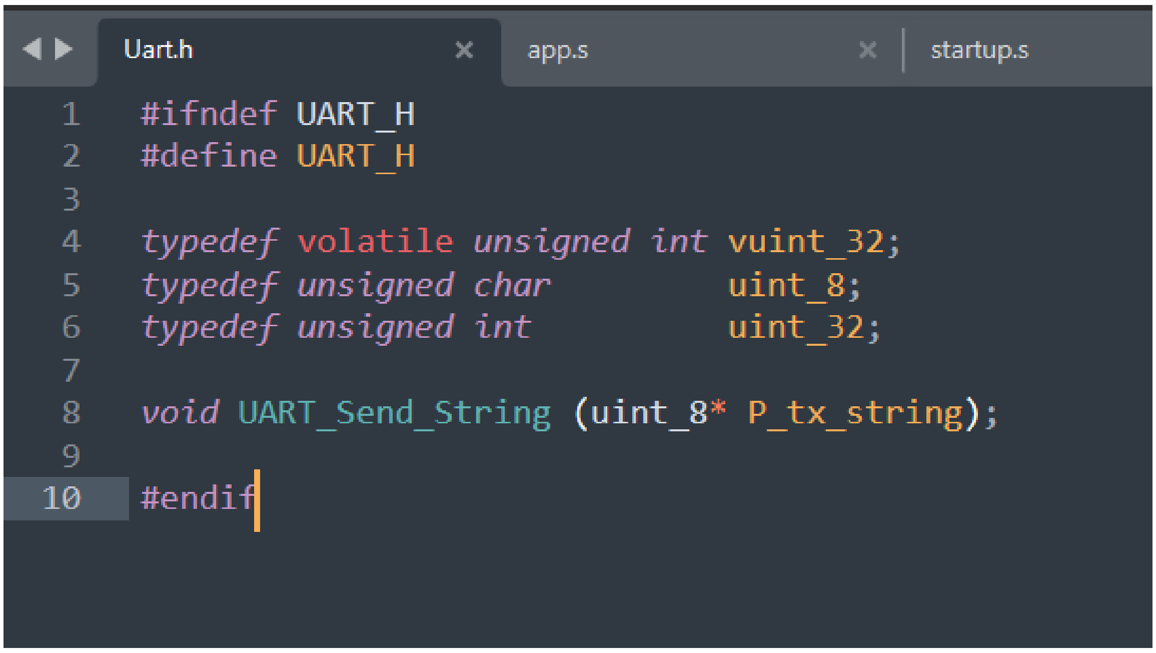
- **uart.c:**

UARTDR register used to transmit data when writing on it, First serial port in particular UART0, the address where the UART0 is mapped: 0x101f1000.


File Edit Selection Find View Goto Tools Project Preferences Help
App.c
Uart.c
Uart.h
app.s

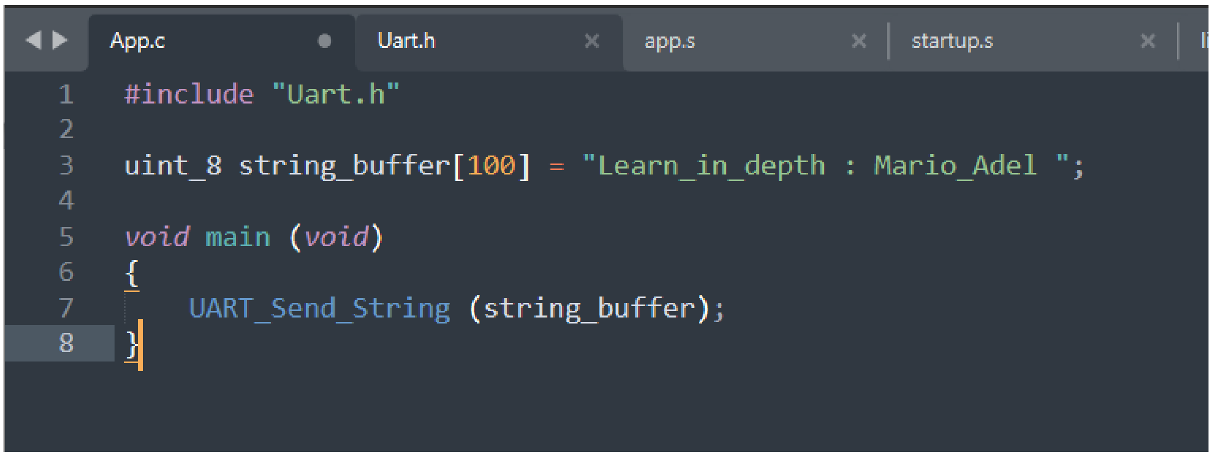
```c
#include "Uart.h"

//UART Registers

#define UART0DR_BASE 0x101f1000
#define UART0DR *((vuint_32* const)((vuint_32*)UART0DR_BASE))

//UART Function

void UART_Send_String (uint_8* P_tx_string)
{
	while(*P_tx_string != '\0')
	{
	UART0DR = (uint_32)(*P_tx_string);
	P_tx_string++;
	}
}
```

➢ **uart.h:**



```c
#ifndef UART_H
#define UART_H

typedef volatile unsigned int vuint_32;
typedef unsigned char         uint_8;
typedef unsigned int          uint_32;

void UART_Send_String (uint_8* P_tx_string);

#endif
```

➢ **app.c:**



```c
#include "Uart.h"

uint_8 string_buffer[100] = "Learn_in_depth : Mario_Adel ";

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

➢ **Generate .o objects files: (Relocatable Binary)**

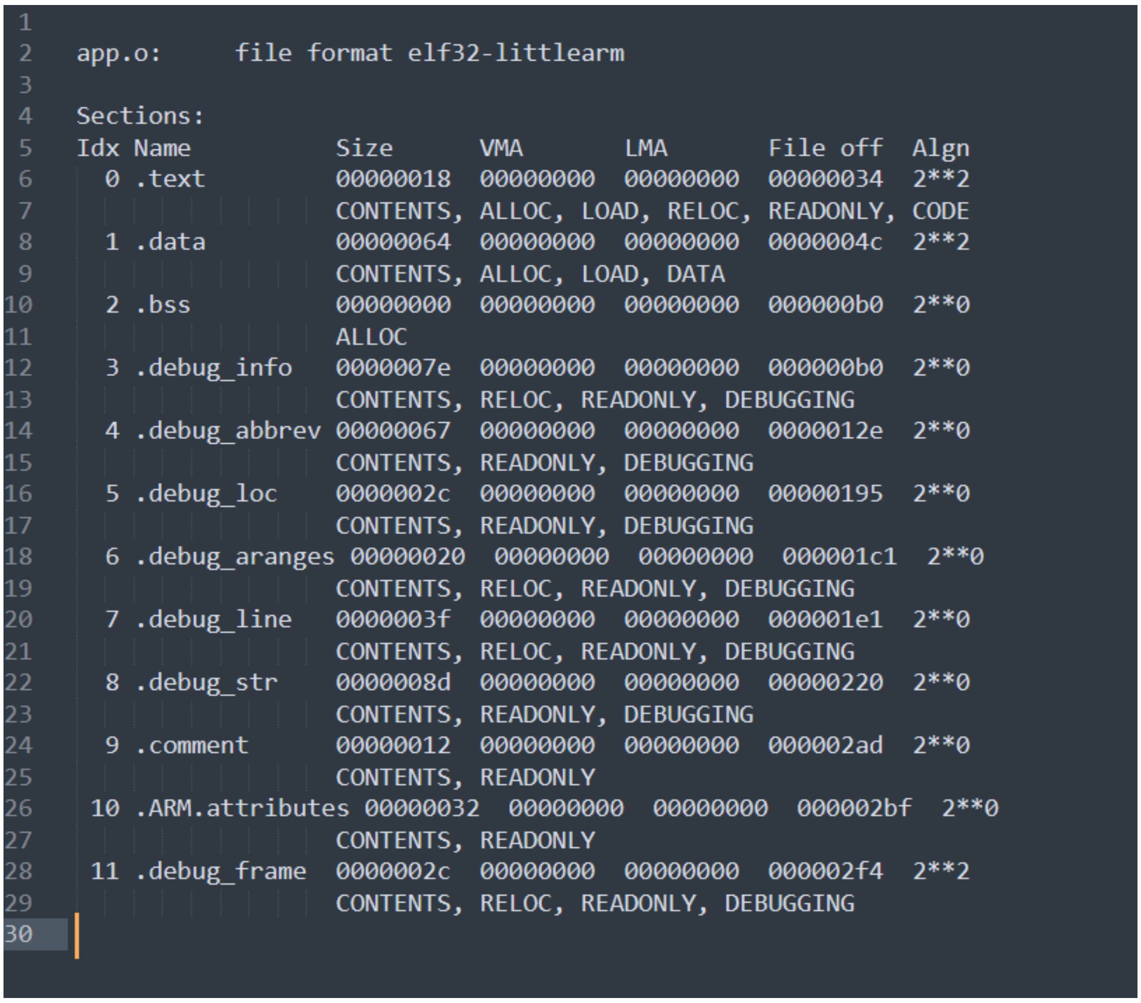
Relocatable Binary: it is a machine code has a virtual address not SoC physical address the physical addresses will located by the linker.

Using arm tool chain by terminal command:

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

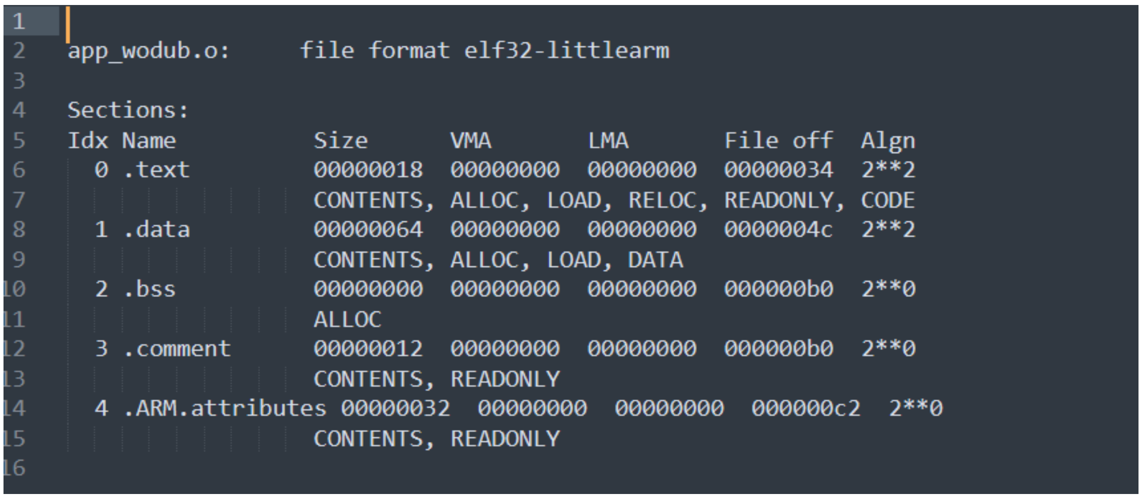
➢ **Sections for .obj files:**

- **app.c:** **(with debug)**



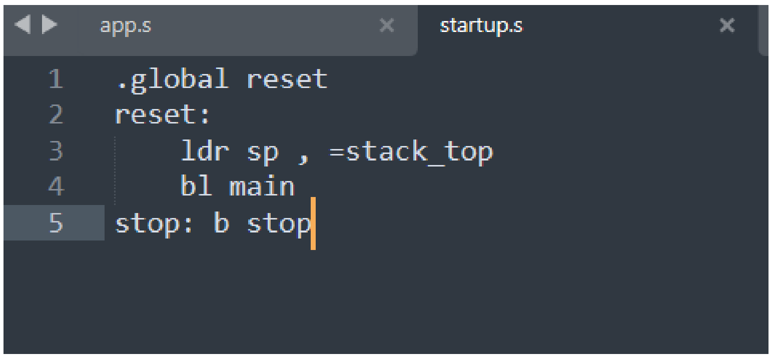
```

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 .debug_info   0000007e  00000000  00000000  000000b0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 00000067  00000000  00000000  0000012e  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000002c  00000000  00000000  00000195  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  000001c1  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   0000003f  00000000  00000000  000001e1  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    0000008d  00000000  00000000  00000220  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  000002ad  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000032  00000000  00000000  000002bf  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000002c  00000000  00000000  000002f4  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING

```

- **app.c:** **(without debug)**



```

app_wodub.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

```

➢ **Write Startup assembly code: startup.s**

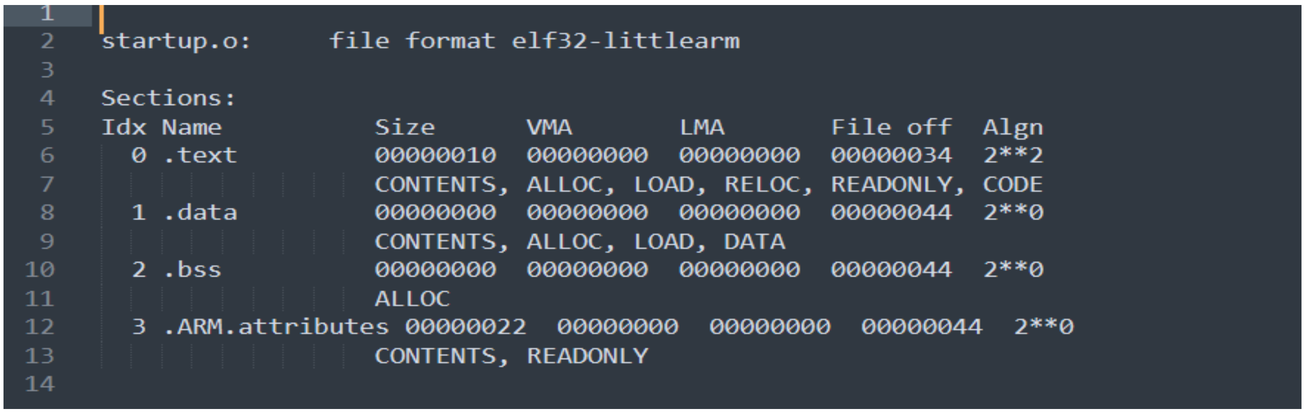


app.s

startup.s

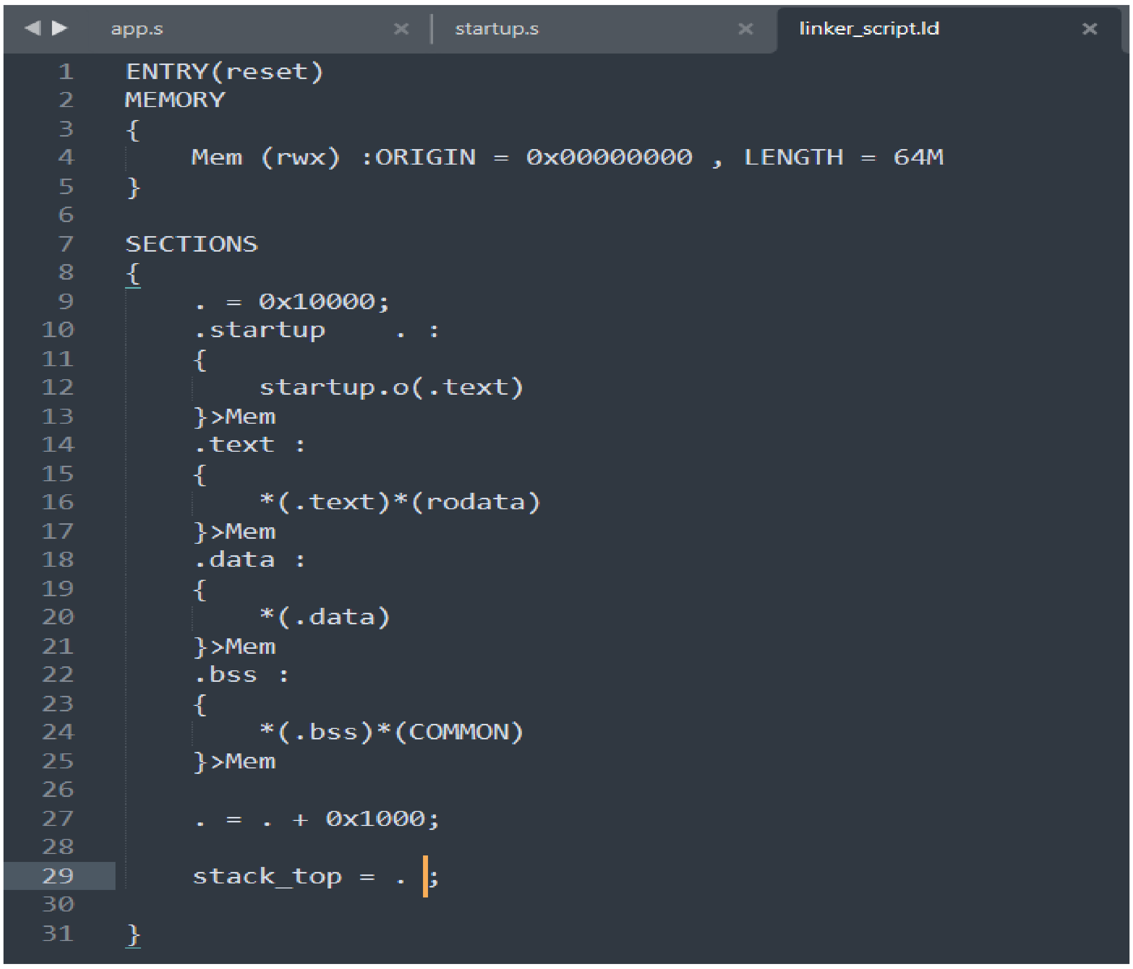
```
.global reset
reset:
	ldr sp , =stack_top
	bl main
stop: b stop
```

- **Sratrup.o Sections:**



```

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

```

➢ **Write Linker_script.ld:**



app.s

startup.s

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 = . ;

}
```

- **generate .elf file:**

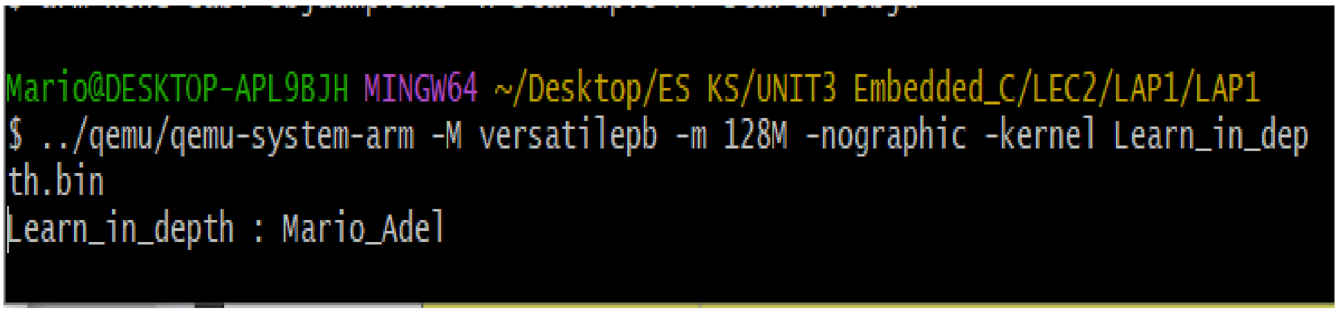
  ```
  $ arm-none-eabi-ld.exe -T linker_script.ld startup.o app.o uart.o -o Learn_in_depth.elf
  ```

- **generate .bin file:**

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

- **Burn .bin binary file on the board using qemu and run it:**

  ```
  $ qemu-system-arm -M versatilepb -m 128M -nographic -kernel Learn_in_depth.bin
  ```



```
Mario@DESKTOP-APL9BJH MINGW64 ~/Desktop/ES KS/UNIT3 Embedded_C/LEC2/LAP1/LAP1
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel Learn_in_dep
th.bin
Learn_in_depth : Mario_Adel
```