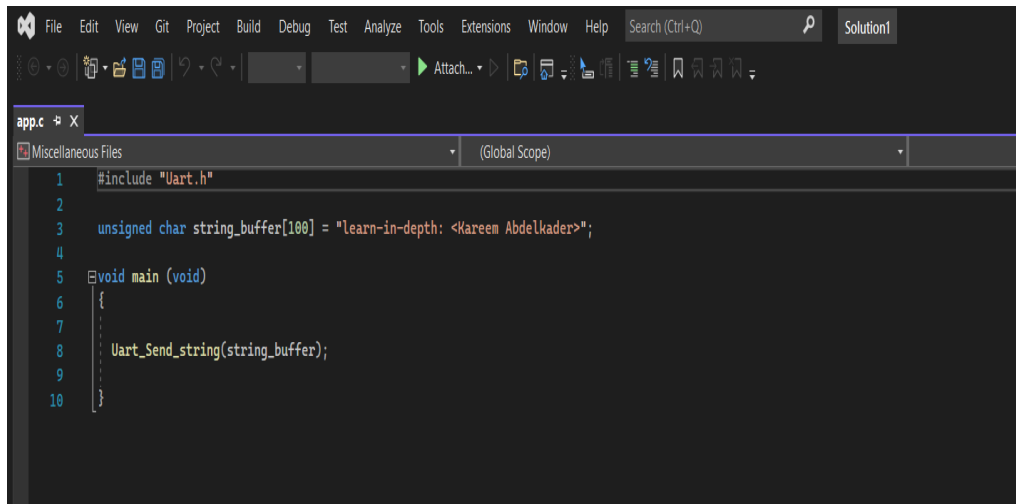


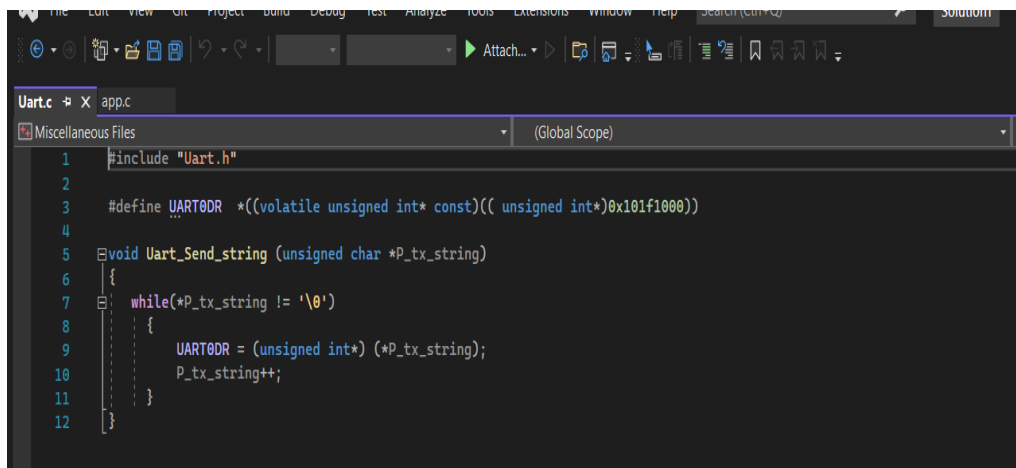
1- The main application:

app.c



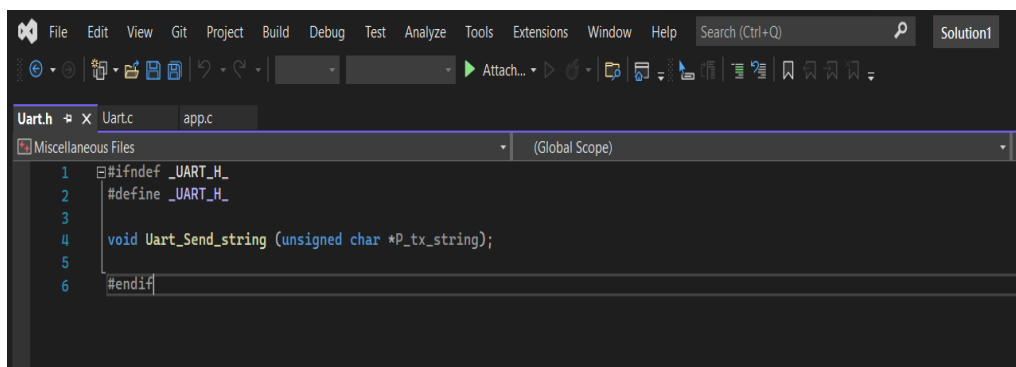
```
1 #include "Uart.h"
2
3 unsigned char string_buffer[100] = "learn-in-depth: <Kareem Abdelkader>";
4
5 void main (void)
6 {
7     Uart_Send_string(string_buffer);
8 }
9
10
```

uart.c



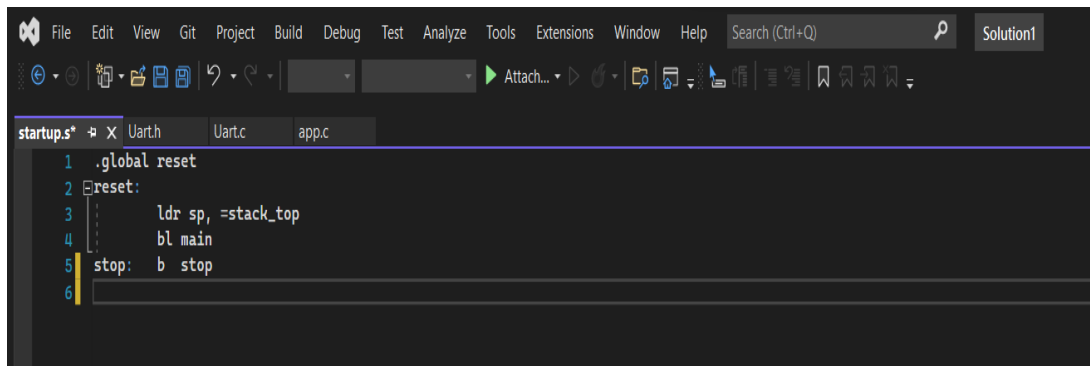
```
1 #include "Uart.h"
2
3 #define UART0DR *((volatile unsigned int* const)(( unsigned int*)0x101f1000))
4
5 void Uart_Send_string (unsigned char *P_tx_string)
6 {
7     while(*P_tx_string != '\0')
8     {
9         UART0DR = (unsigned int*) (*P_tx_string);
10        P_tx_string++;
11    }
12 }
```

uart.h



```
1 #ifndef _UART_H_
2 #define _UART_H_
3
4 void Uart_Send_string (unsigned char *P_tx_string);
5
6 #endif
```

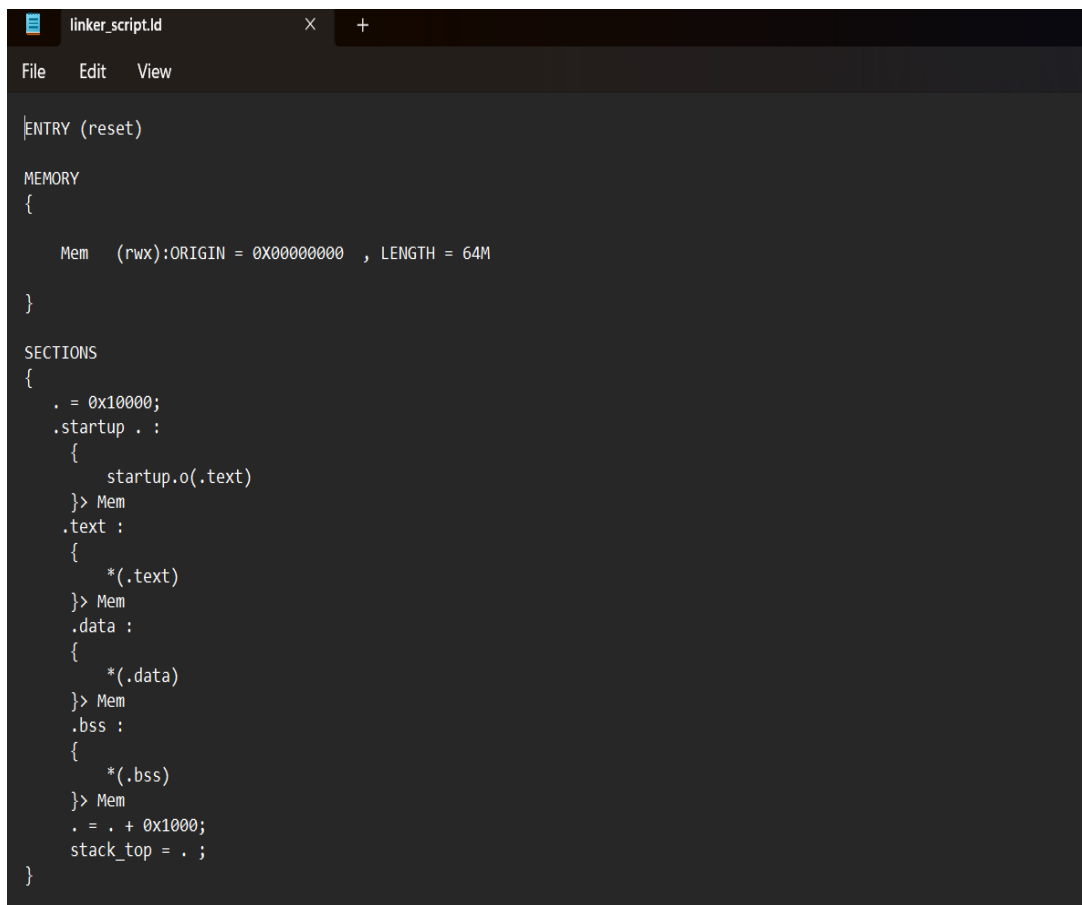
2- Startup of the application:



The screenshot shows an IDE window with the file 'startup.s' open. The code is written in assembly and includes a 'reset' section. The code is as follows:

```
1 .global reset
2 reset:
3     ldr sp, =stack_top
4     bl main
5 stop: b stop
6
```

3- Linker script of the application:



The screenshot shows an IDE window with the file 'linker_script.ld' open. The code is a linker script that defines the entry point, memory layout, and sections. The code is as follows:

```
ENTRY (reset)

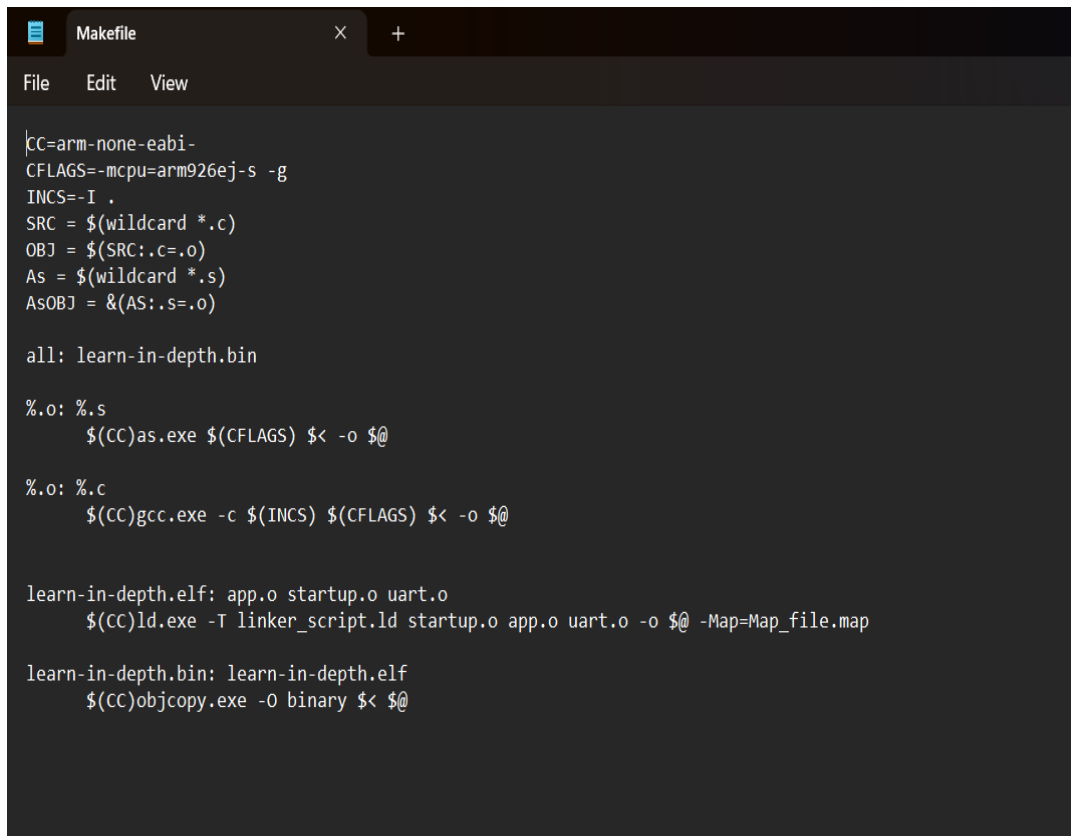
MEMORY
{
    Mem (rwx):ORIGIN = 0x00000000 , LENGTH = 64M
}

SECTIONS
{
    . = 0x10000;
    .startup . :
    {
        startup.o(.text)
    }> Mem
    .text :
    {
        *(.text)
    }> Mem
    .data :
    {
        *(.data)
    }> Mem
    .bss :
    {
        *(.bss)
    }> Mem
    . = . + 0x1000;
    stack_top = . ;
}
```

4- Map file:

Map_file.map			
File Edit View			
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	0x70	
*(.text)			
.text	0x00010010	0x1c	app.o
	0x00010010		main
.text	0x0001002c	0x54	Uart.o
	0x0001002c		Uart_Send_string
.glue_7	0x00010080	0x0	
.glue_7	0x00010080	0x0	linker stubs
.glue_7t	0x00010080	0x0	
.glue_7t	0x00010080	0x0	linker stubs
.vfp11_veneer	0x00010080	0x0	
.vfp11_veneer	0x00010080	0x0	linker stubs
.v4_bx	0x00010080	0x0	
.v4_bx	0x00010080	0x0	linker stubs
.iplt	0x00010080	0x0	
.iplt	0x00010080	0x0	startup.o
.rel.dyn	0x00010080	0x0	
.rel.iplt	0x00010080	0x0	startup.o
.data	0x00010080	0x64	
*(.data)			
.data	0x00010080	0x0	startup.o
.data	0x00010080	0x64	app.o
	0x00010080		string_buffer
.data	0x000100e4	0x0	Uart.o
.igot.plt	0x000100e4	0x0	
.igot.plt	0x000100e4	0x0	startup.o
.bss	0x000100e4	0x0	
*(.bss)			
.bss	0x000100e4	0x0	startup.o
.bss	0x000100e4	0x0	app.o
.bss	0x000100e4	0x0	Uart.o
	0x000110e4	. = (. + 0x1000)	
	0x000110e4	stack_top = .	
LOAD startup.o			
LOAD app.o			
LOAD Uart.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	0x7e	
.comment	0x00000000	0x7e	app.o
		0x7f (size before relaxing)	
.comment	0x0000007e	0x7f	Uart.o

5- Make file:



```
Makefile
File Edit View

CC=arm-none-eabi-
CFLAGS=-mcpu=arm926ej-s -g
INCS=-I .
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
AS = $(wildcard *.s)
ASOBJ = $(AS:.s=.o)

all: learn-in-depth.bin

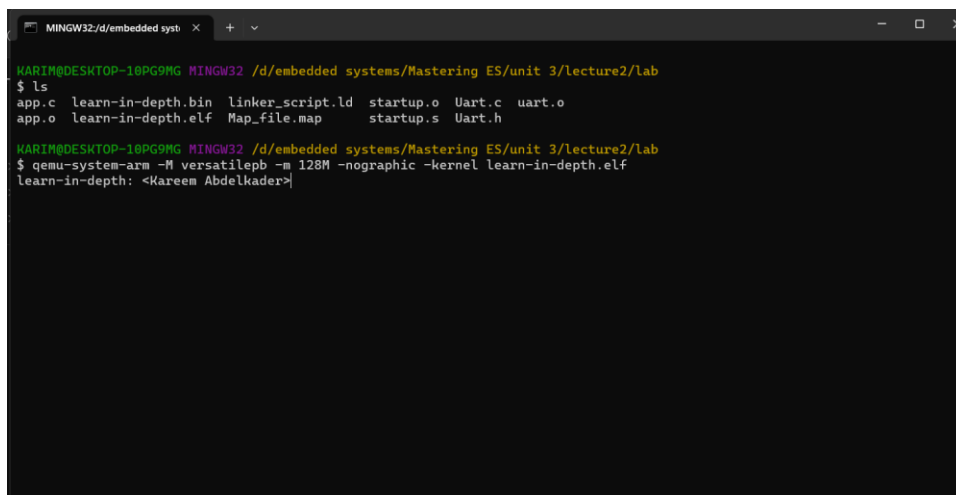
%.o: %.s
    $(CC)as.exe $(CFLAGS) $< -o $@

%.o: %.c
    $(CC)gcc.exe -c $(INCS) $(CFLAGS) $< -o $@

learn-in-depth.elf: app.o startup.o uart.o
    $(CC)ld.exe -T linker_script.ld startup.o app.o uart.o -o $@ -Map=Map_file.map

learn-in-depth.bin: learn-in-depth.elf
    $(CC)objcopy.exe -O binary $< $@
```

6- The final application:



```
MINGW32/d/embedded syst
KARIM@DESKTOP-10PG9MG MINGW32 /d/embedded systems/Mastering ES/unit 3/lecture2/lab
$ ls
app.c  learn-in-depth.bin  linker_script.ld  startup.o  Uart.c  uart.o
app.o  learn-in-depth.elf  Map_file.map     startup.s  Uart.h

KARIM@DESKTOP-10PG9MG MINGW32 /d/embedded systems/Mastering ES/unit 3/lecture2/lab
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.elf
learn-in-depth: <Kareem Abdelkader>
```