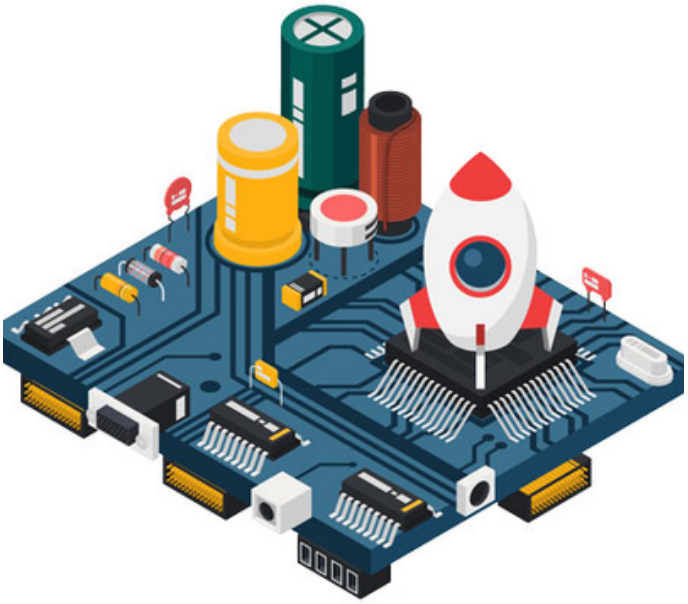
REPORT

LAB2



PREPARED BY

Mohamed kadry Hussien

LAB_2

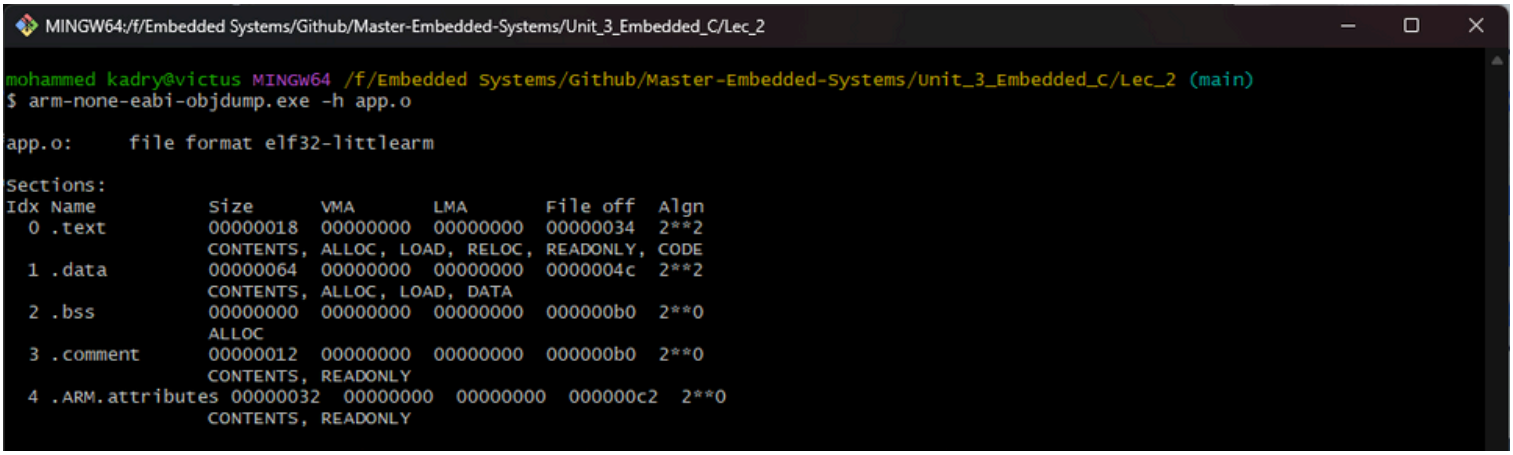
- Physical Board : VersatilePB
- Processor : Arm926ej-s

5.2v

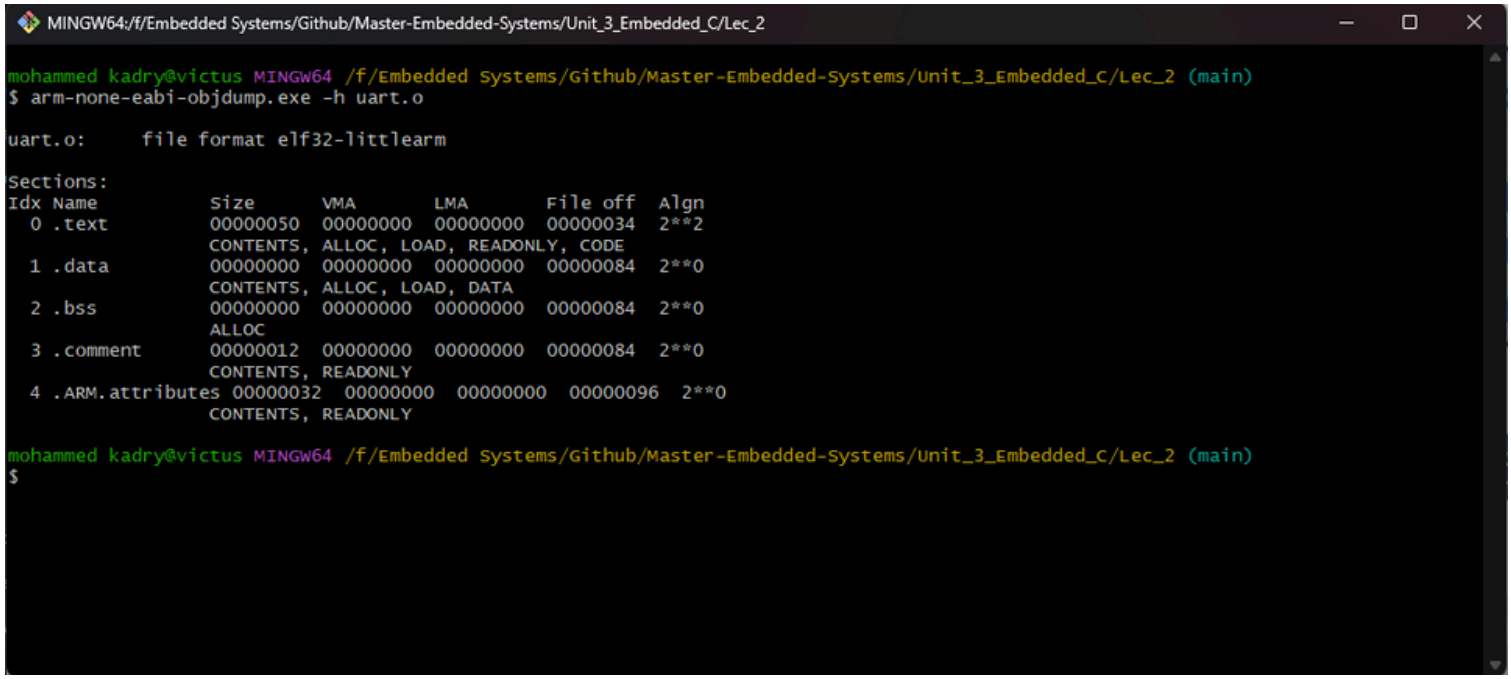
1-Writing source files, getting object files (without debugging)

2-analysis object files of (app.c and uart.c)

-app.o



-uart.o



3-Writing startup code, getting object file and analyzing it.

```
MINGW64:/f/Embedded Systems/Github/Master-Embedded-Systems/Unit_3_Embedded_C/Lec_2

mohammed kadry@victus MINGW64 /f/Embedded Systems/Github/Master-Embedded-Systems/Uni
t_3_Embedded_C/Lec_2 (main)
$ 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
```

4-analysis object file of startup file

```
$ 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
```

5-Writing the linker script, linking all objects, getting the elf file

```
MINGW64:/f/Embedded Systems/Github/Master-Embedded-Systems/Unit_3_Embedded_C/Lec_2
bedded_C/Lec_2 (main)
$ arm-none-eabi-ld.exe -T linker_script.ld startup.o app.o uart.o -o Mohamed_Kadry.elf -Ma
p=Map_File.map
```

6-analysis elf file

```
MINGW64:/f/Embedded Systems/Github/Master-Embedded-Systems/Unit_3_Embedded_C/Lec_2
bedded_C/Lec_2 (main)
$ arm-none-eabi-objdump.exe -h Mohamed_Kadry.elf

Mohamed_Kadry.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .startup      00000010  00001000  00001000  00001000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         00000068  00001010  00001010  00001010  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  00001078  00001078  00001078  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 0000002e  00000000  00000000  000010dc  2**0
                  CONTENTS, READONLY
  4 .comment      00000011  00000000  00000000  0000110a  2**0
                  CONTENTS, READONLY
```

7-Getting the binary file of elf file

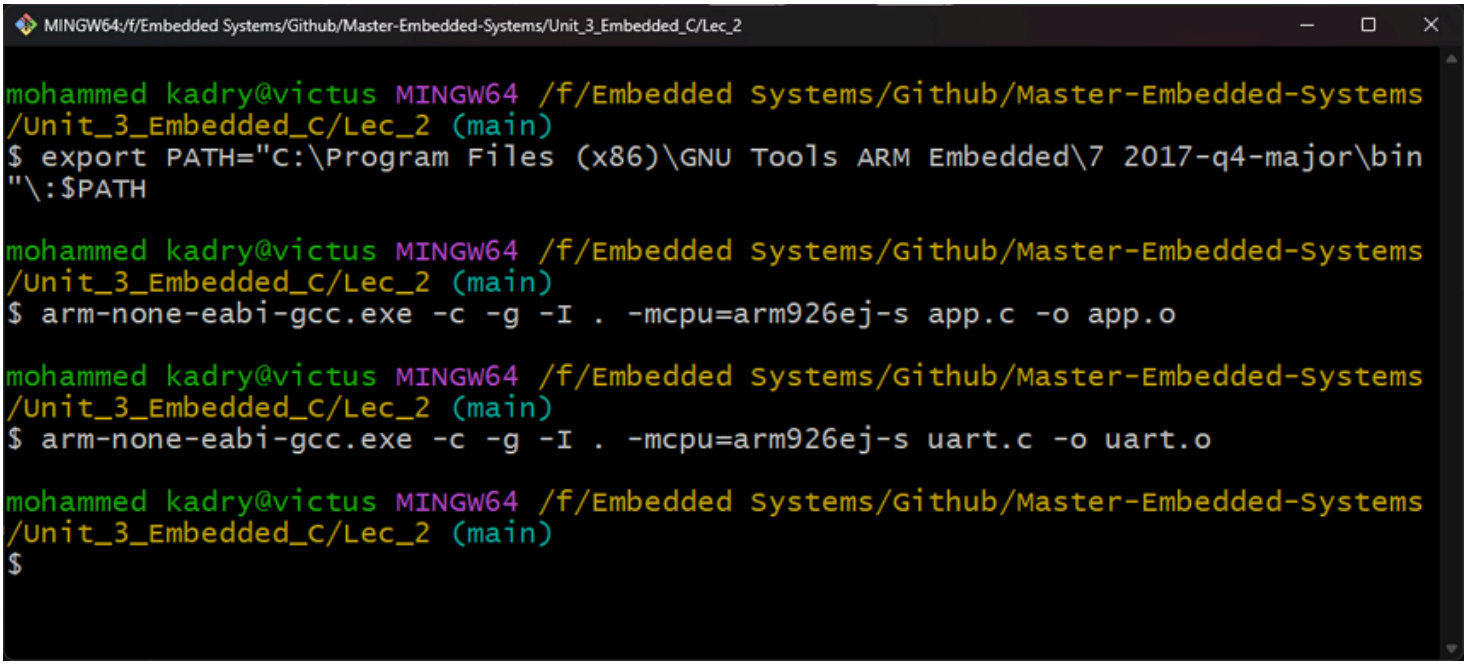
```
mohammed kadry@victus MINGW64 /f/Embedded Systems/Github/Master-Embedded-Systems/Unit_3_Embedded_C/Lec_2 (main)
$ arm-none-eabi-objcopy.exe -O binary Mohamed_Kadry.elf Mohamed_Kadry.bin
```

8- simulation the application using QEMU

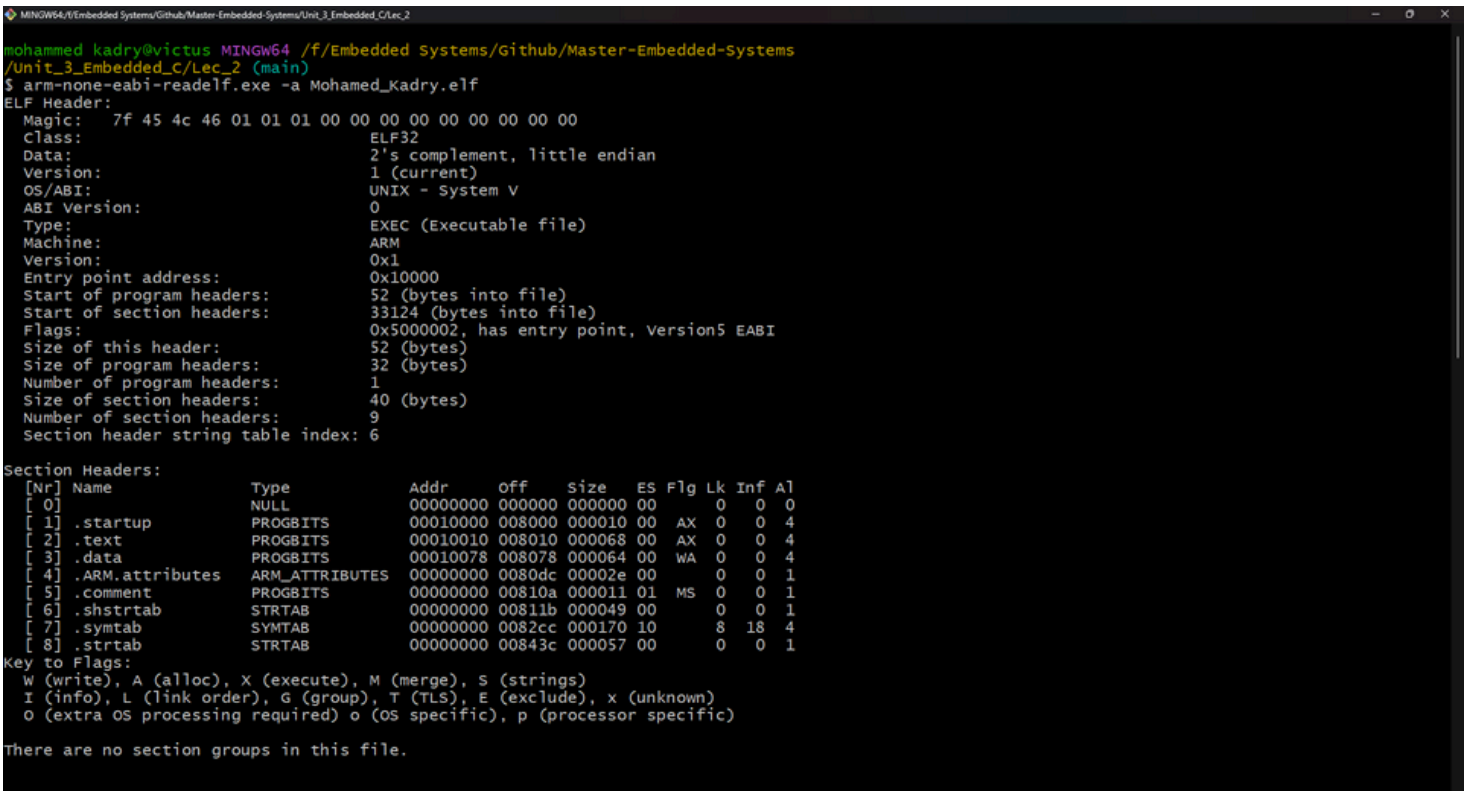
```
mohammed kadry@victus MINGW64 /f/Embedded Systems/Github/Master-Embedded-Systems/Unit_3_Embedded_C/Lec_2 (main)
$ c:/qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel Mohamed_Kadry.bin
Learn-in-Depth:<Mohamed Kadry>
```

Additional Options

-we can also getting object files (with debugging)



-We also can use readelf.exe To make sure about the entry point at address.



-Getting the symbol table for the object files