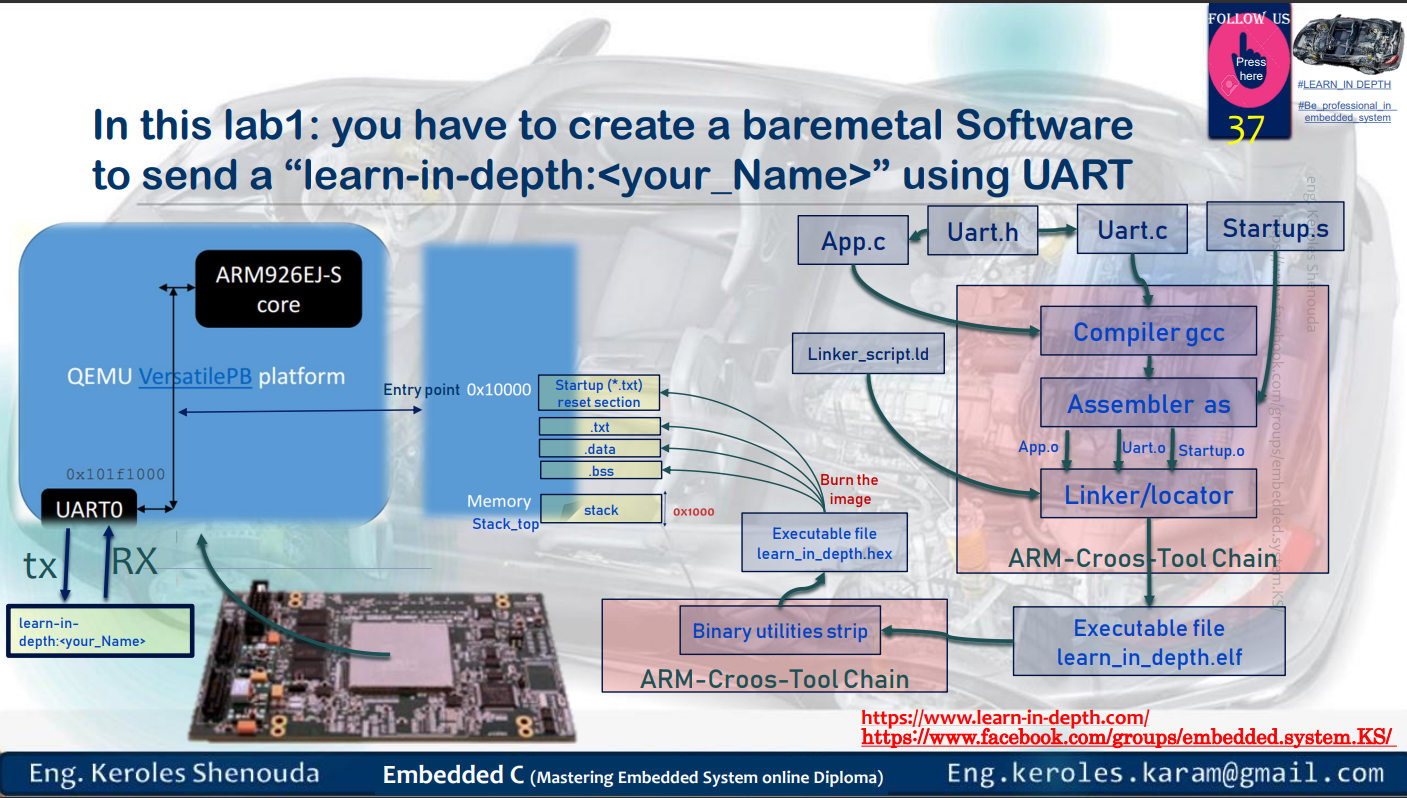
***Saeed Mabrouk Saeed El-Shaikh***

**A simple application of executing a bare metal application on ARM VersatilePB**

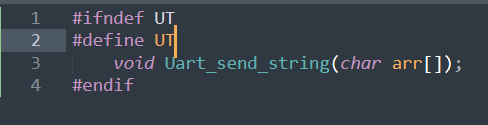
* The program is to send “learn-in-depth: Saeed Fares” to the terminal through the UART0 peripheral
* I will achieve that by going on some steps

1. Write C codes which achieve this task and compile it to get object files
2. Write startup.s file and feed it to the compiler and get object file
3. Write the linker\_script file to give it the linker
4. Use linker to link these object files and generate the binary file
5. burn the binary file in the VersatilePB to run the code

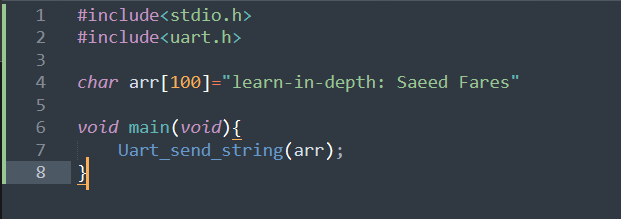


1. ***Write C codes which achieve this task***

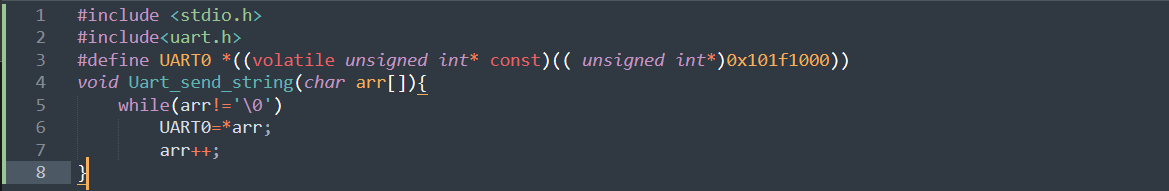
***Uart.h***



***Uart.c***

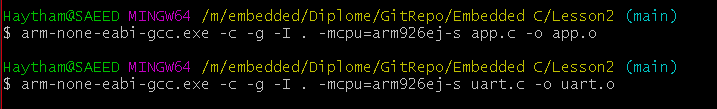


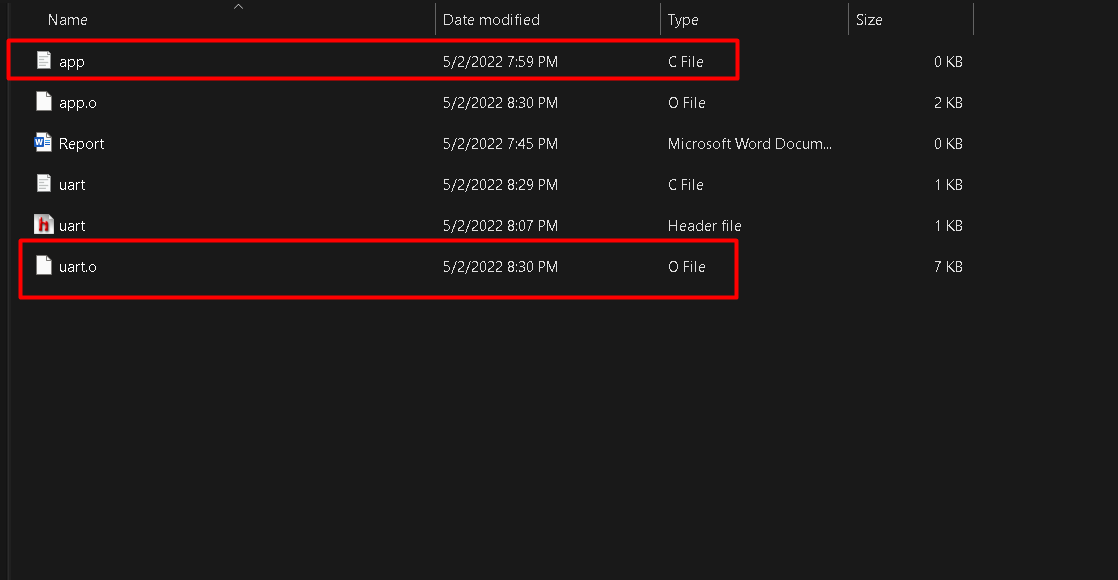
***App.c***



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

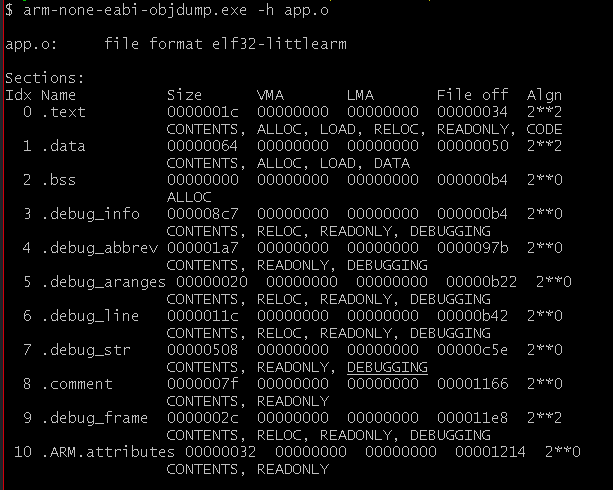
* -c 🡺 compile and assemple only not link
* -g 🡺 generate debug information
* -I 🡺include header
* . 🡺 the header in the same directory I’m in
* -mcpu= arm926ej-s 🡺 machine
* -o 🡺output to file





***Navigate object files***

1. ***App.o***

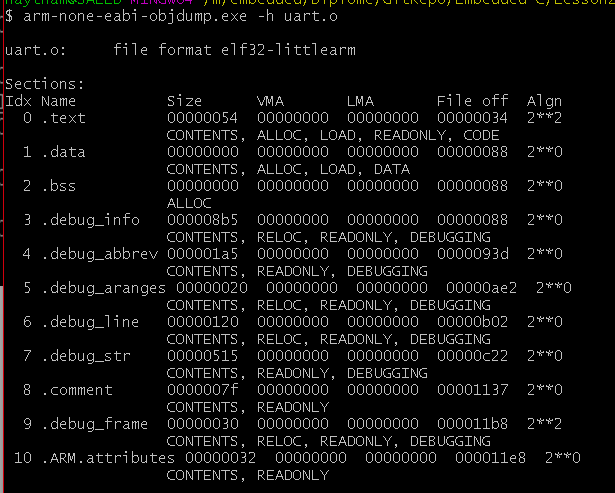


***data section 64 hexa===100 byte which is arr[100]***

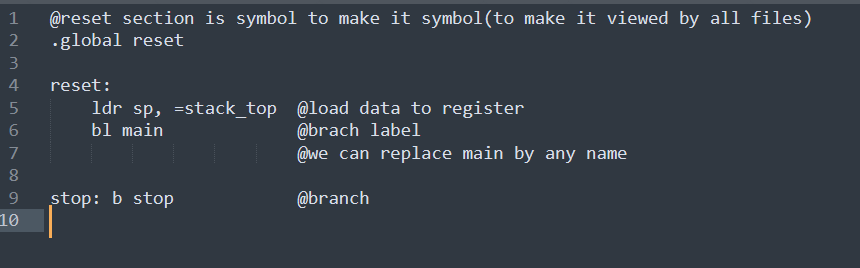
***bss section 0 because we not have uninitialized variables***

***all addresses is 0 because onbect files are allocatable files and physical addresses is set during linker stage***

1. ***uart.o***



1. ***Write startup.s file***



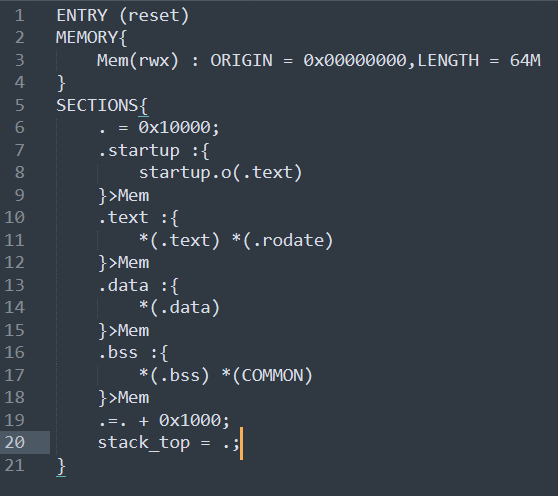
***Compile startup file***



***Navigiation***



1. ***Writing linker script***

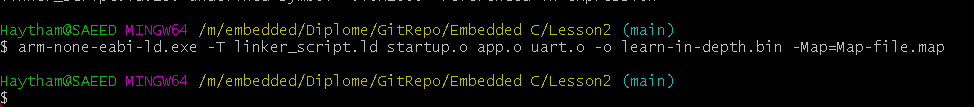


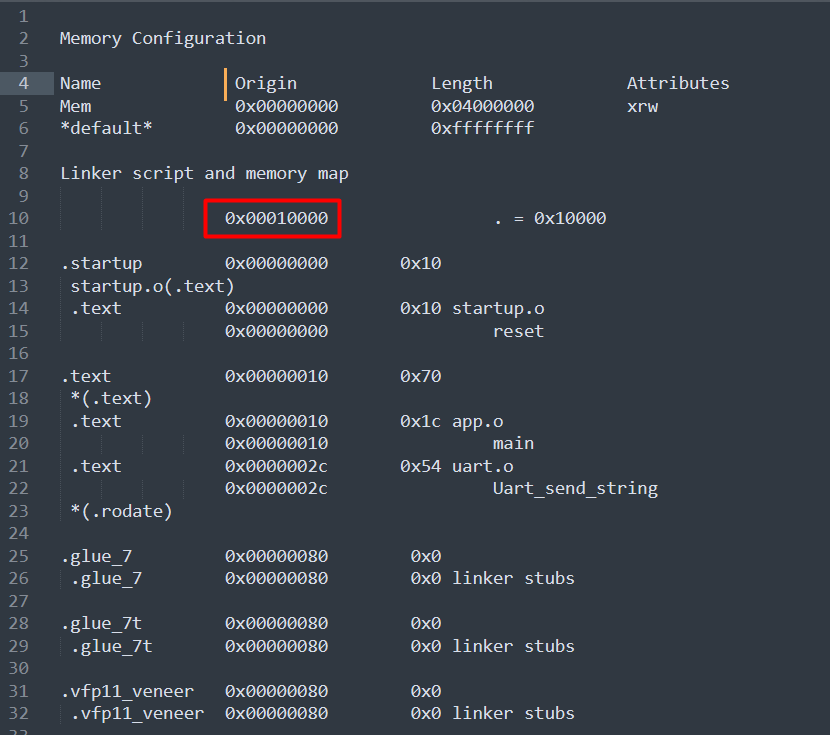
***Checking symbols in object files***

***The addresses is zero because object files are reallocate***



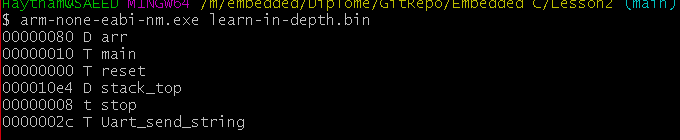
1. ***Use linker to link these object files and generate the binary file***





***In map file it’s shown the startup section(reset) in the entry point 0x00010000***

***Symbols in the .exe file***



***Real addresses are shown here***

1. ***burn the binary file in the VersatilePB to run the code***

