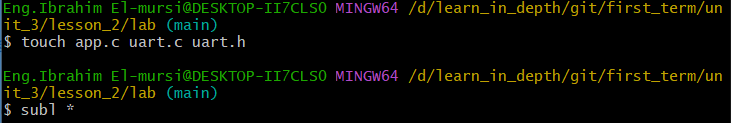
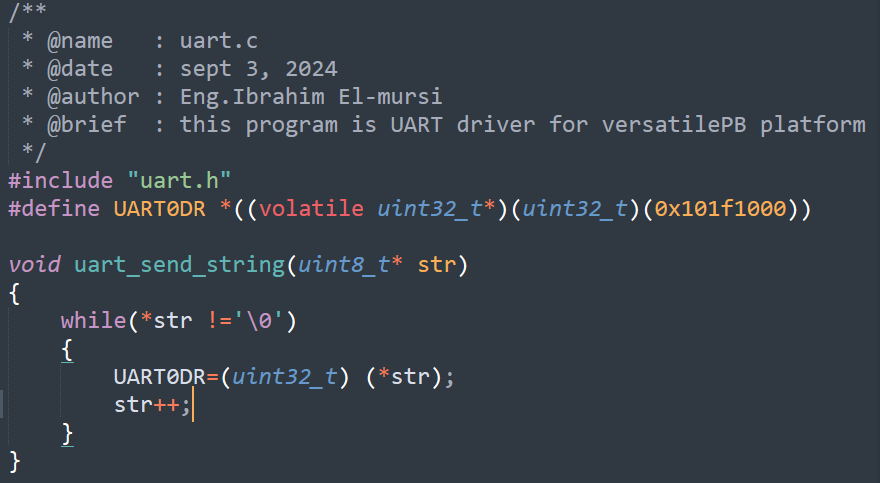
Lab 1 lecture 2 unit 3

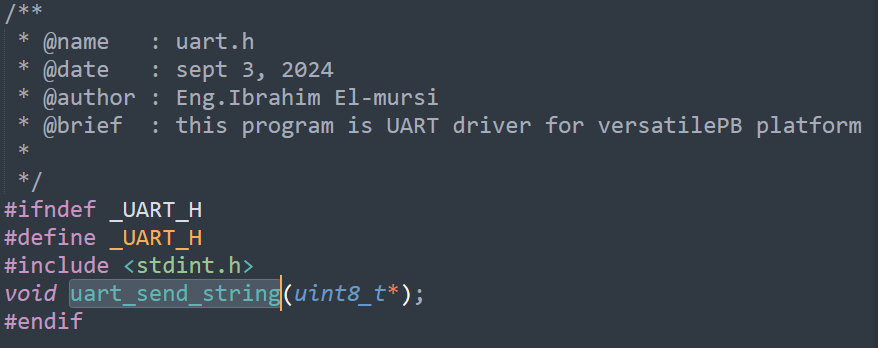
Print string “learn-in-depth:ibrahim” on on QEMU Emulator

1. Create our app.c , UART.c and UART.h files and open it using editor “e.g. Sublime”

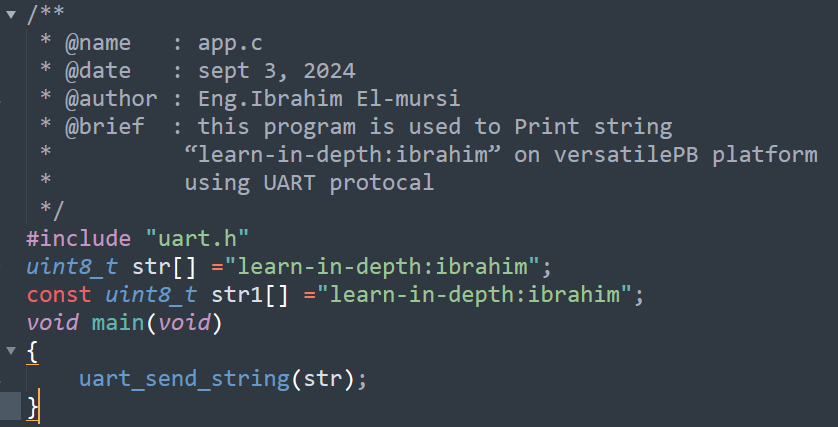


1. Write our UART driver

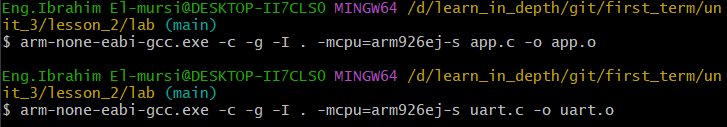




1. Write our app.c program



1. Generate our object files (relocatable files) app.o , uart.o
2. With debug sections

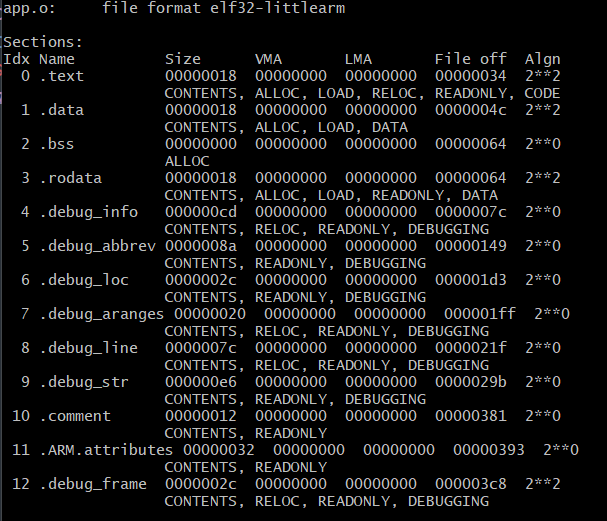
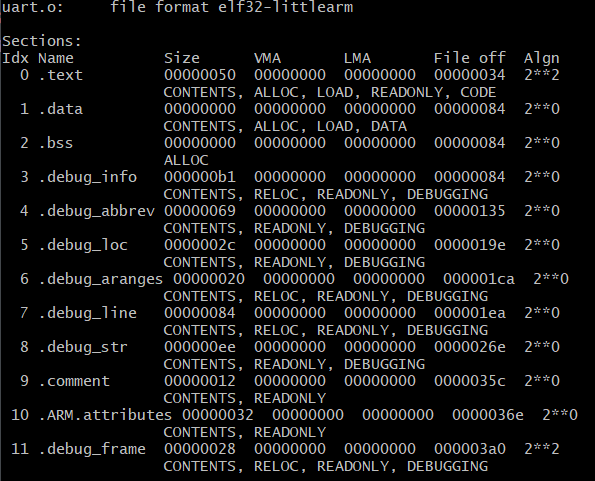


1. Without debug sections
2. Navigate the object files using arm binary utilities
3. With debug sections

Command:

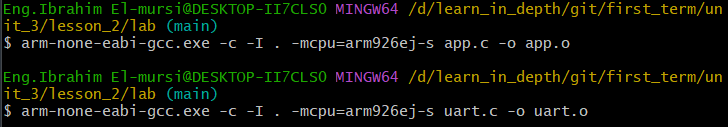


Output:

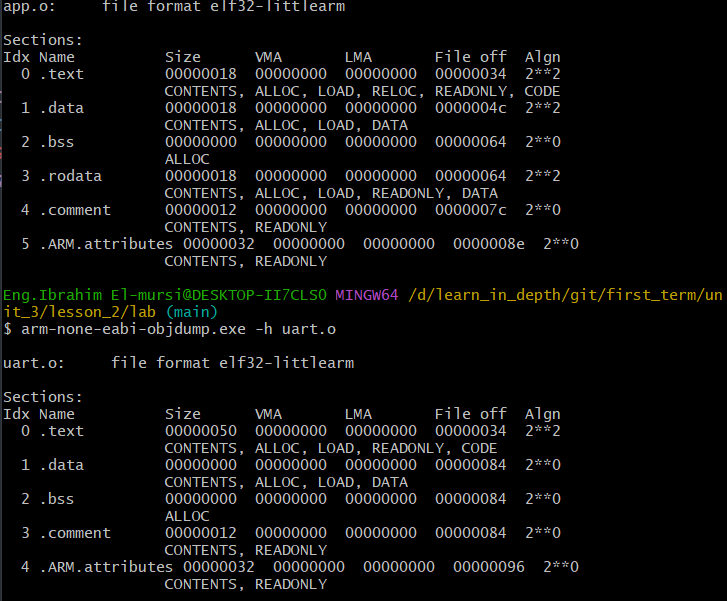
 

1. Without debug sections

Command:

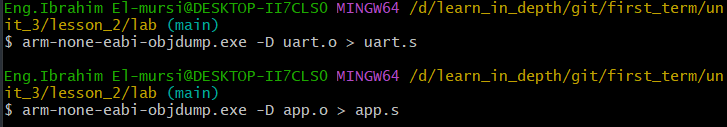


Output:

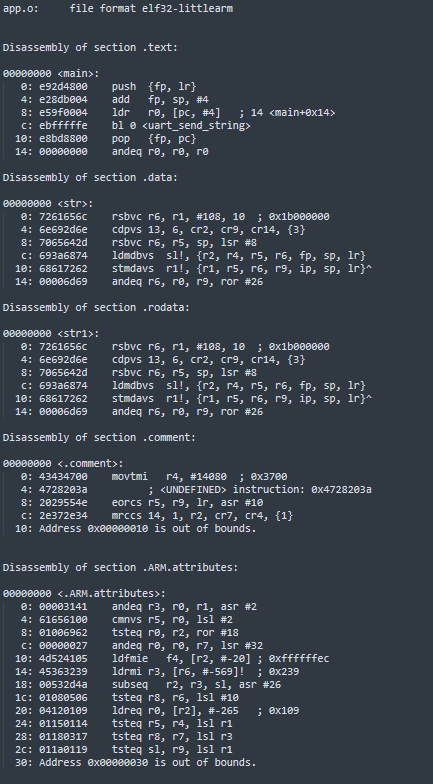
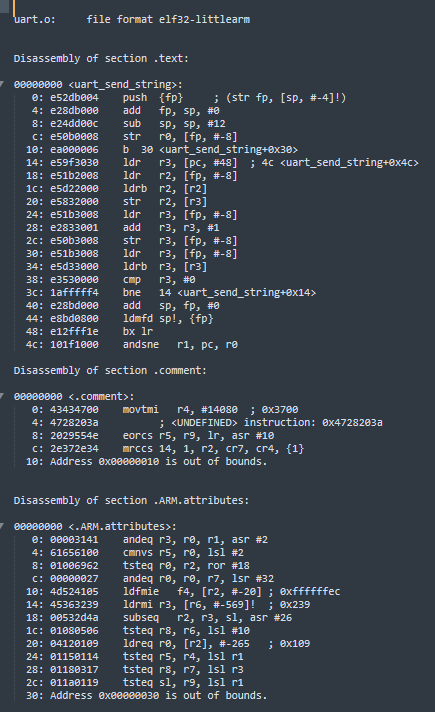


1. Generate assemply file using arm binary utilities

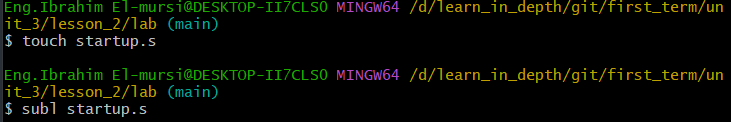
Command:



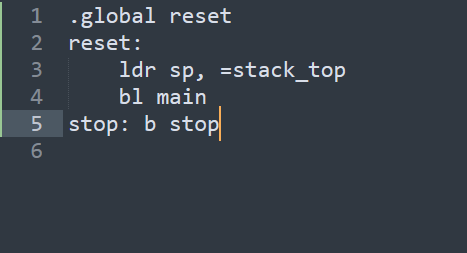
Output:

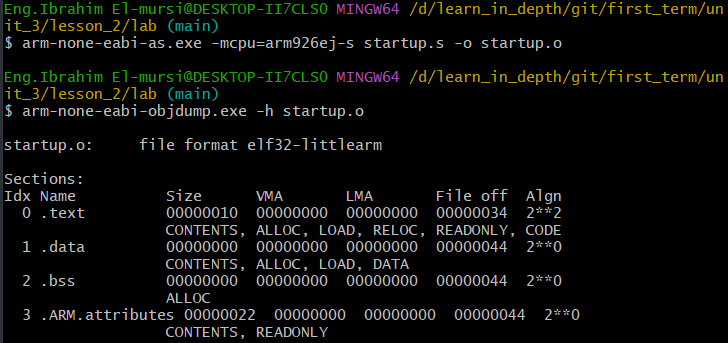
1. Create startup.s file



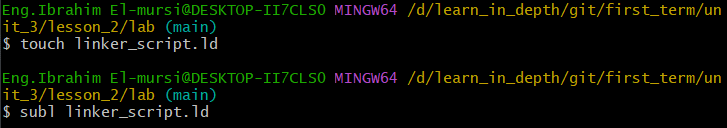
1. Write our startup code



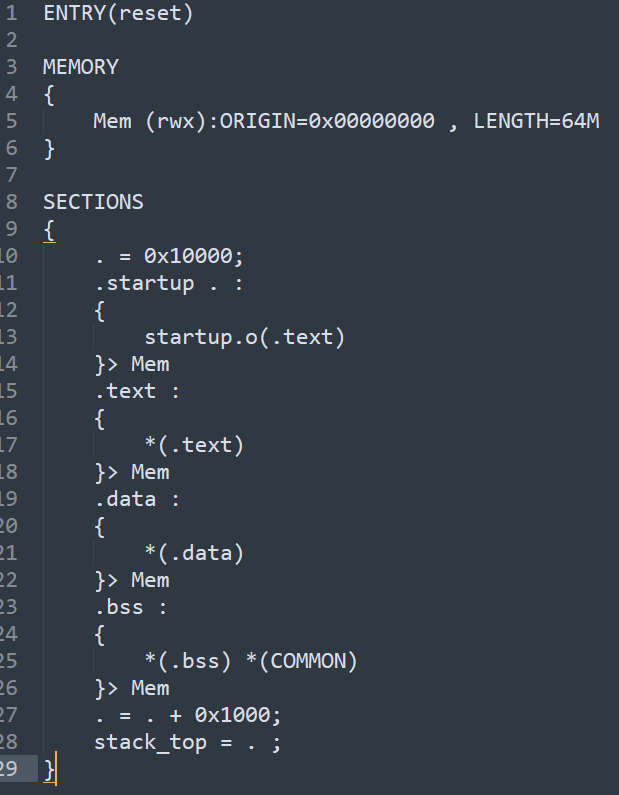
1. Generate startup.o file and analyze it using arm binary utilities



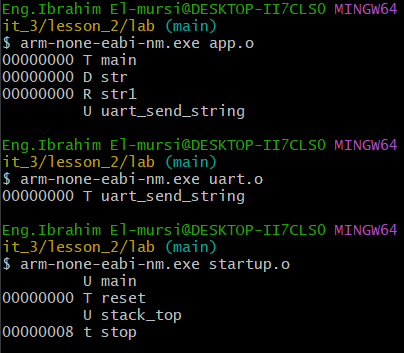
1. Create linker\_script.ld file



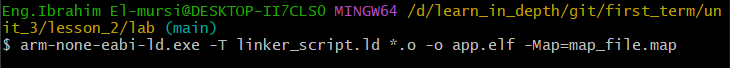
1. Write our linker script and define stack\_top symbol



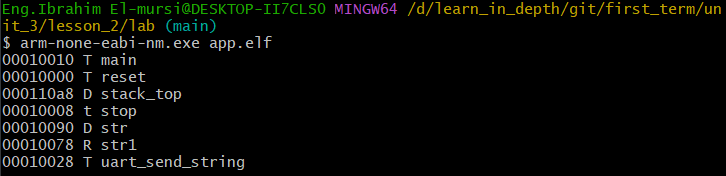
1. Read symbols for each object file before linking



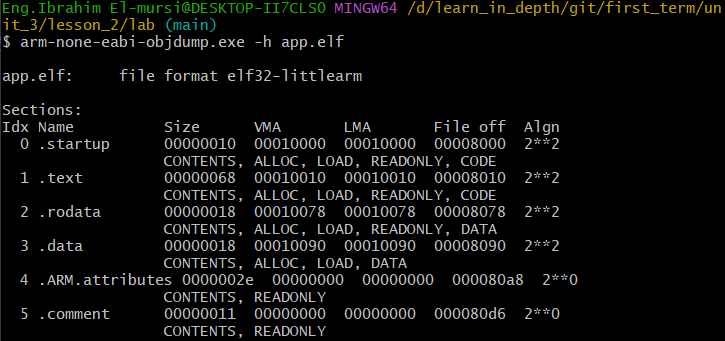
1. Linking all files, generate executable file and map file



1. Read symbols for executable file after linking



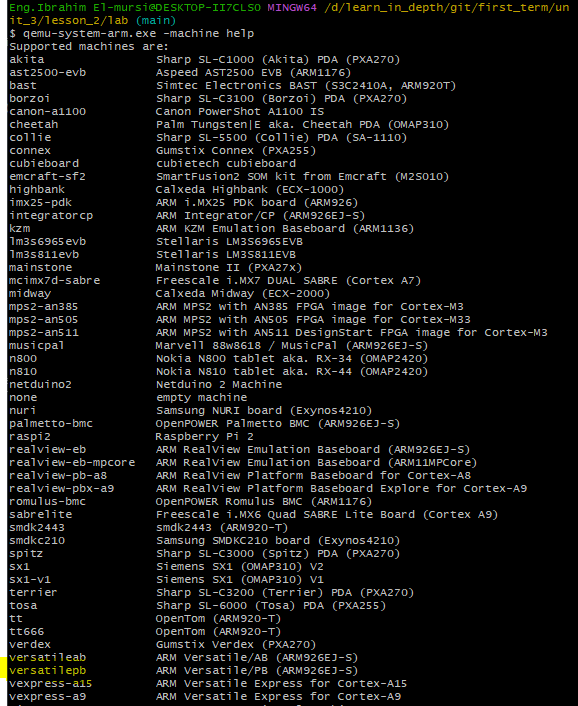
1. Symbols in executable file



1. Generate binary file



1. Check if QEMU support this machine



1. Run the app.bin on the QEMU emulator

