Lab1 Report

Objective1: Write baremetal app to send string through UART and anlayzing the obj files headers.

Objective2: Write custom startup script and linker script.

Objective3: Write a make file to incrementally automate build process.

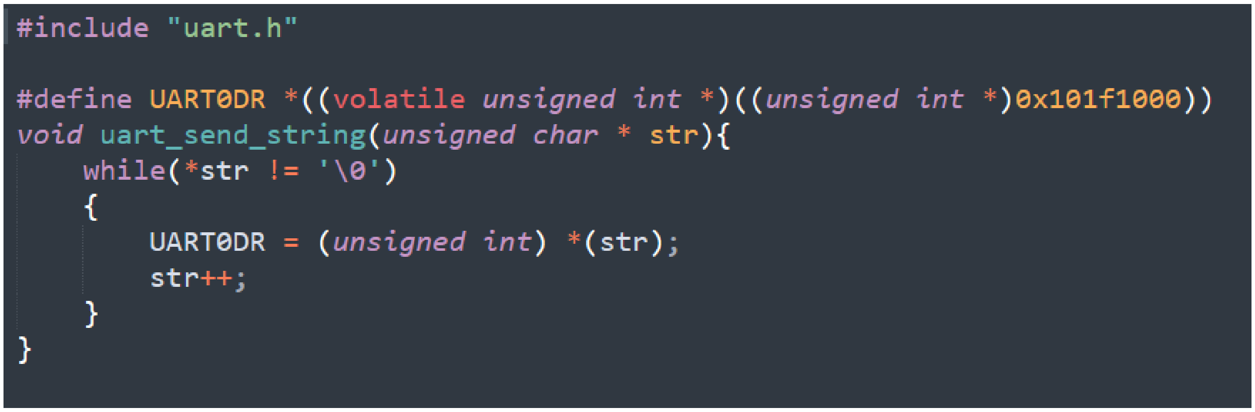
Name: Mohamed Waleed Elsayed Abdelbaset

OBJECTIVE 1

-We are working with a board called “versatilepb”.

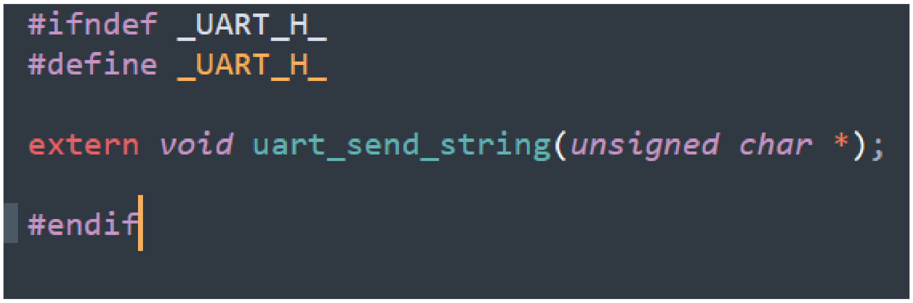
-the UART output data register is located a 0x101f1000 and by default has the FIFO enabeled.

UART.C



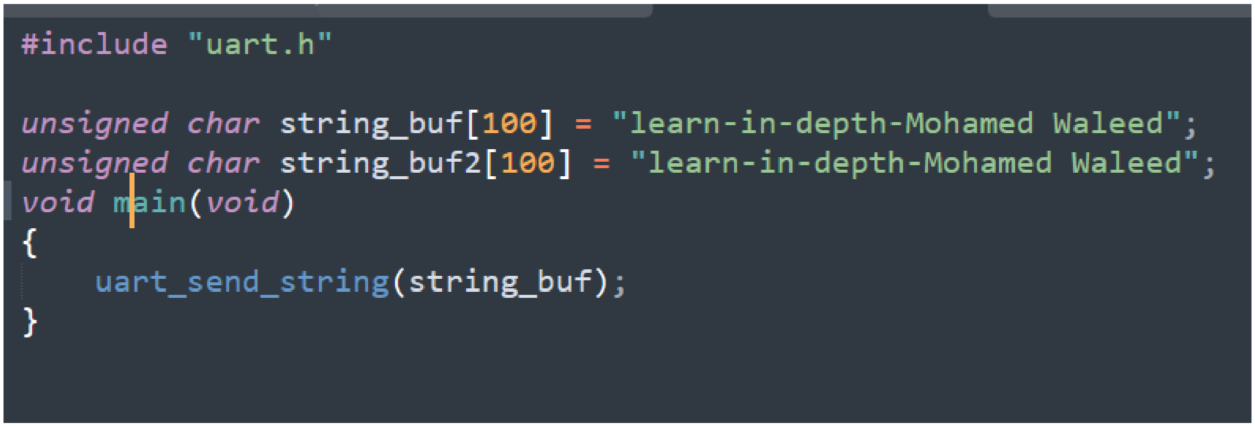
```
#include "uart.h"

#define UART0DR *((volatile unsigned int *)((unsigned int *)0x101f1000))
void uart_send_string(unsigned char * str){
    while(*str != '\0')
    {
        UART0DR = (unsigned int) *(str);
        str++;
    }
}
```

UART.H



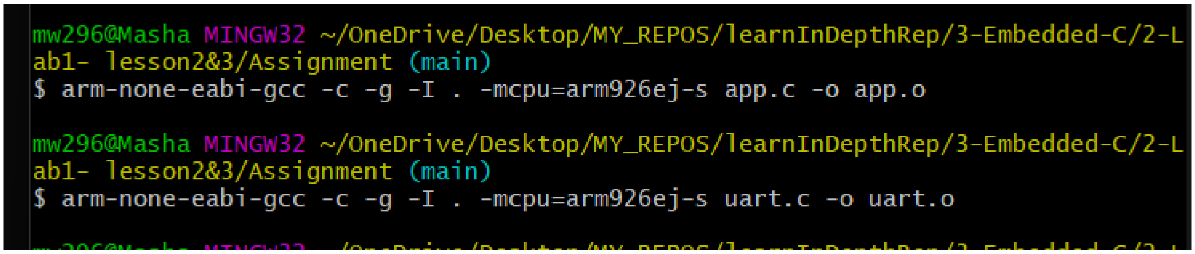
```
#ifndef _UART_H_
#define _UART_H_

extern void uart_send_string(unsigned char *);

#endif
```

APP.C



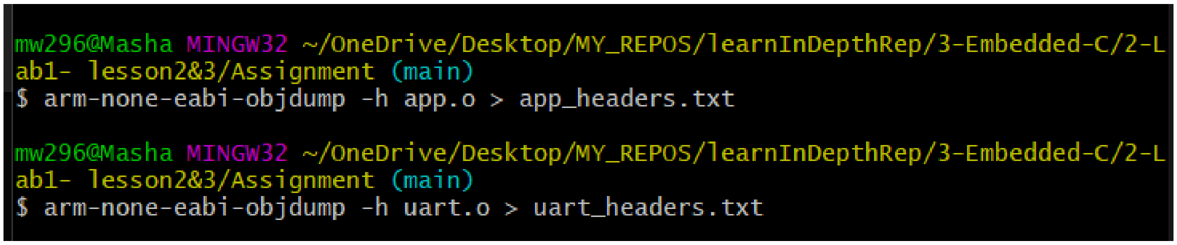
```
#include "uart.h"

unsigned char string_buf[100] = "learn-in-depth-Mohamed Waleed";
unsigned char string_buf2[100] = "learn-in-depth-Mohamed Waleed";
void main(void)
{
    uart_send_string(string_buf);
}
```

After compiling these file without linking them



```
mw296@Masha MINGW32 ~/OneDrive/Desktop/MY_REPOS/learnInDepthRep/3-Embedded-C/2-L
ab1- lesson2&3/Assignment (main)
$ arm-none-eabi-gcc -c -g -I . -mcpu=arm926ej-s app.c -o app.o

mw296@Masha MINGW32 ~/OneDrive/Desktop/MY_REPOS/learnInDepthRep/3-Embedded-C/2-L
ab1- lesson2&3/Assignment (main)
$ arm-none-eabi-gcc -c -g -I . -mcpu=arm926ej-s uart.c -o uart.o
```

We can navigate the relocatable binary files with objdump bin utility



```
mw296@Masha MINGW32 ~/OneDrive/Desktop/MY_REPOS/learnInDepthRep/3-Embedded-C/2-L
ab1- lesson2&3/Assignment (main)
$ arm-none-eabi-objdump -h app.o > app_headers.txt

mw296@Masha MINGW32 ~/OneDrive/Desktop/MY_REPOS/learnInDepthRep/3-Embedded-C/2-L
ab1- lesson2&3/Assignment (main)
$ arm-none-eabi-objdump -h uart.o > uart_headers.txt
```

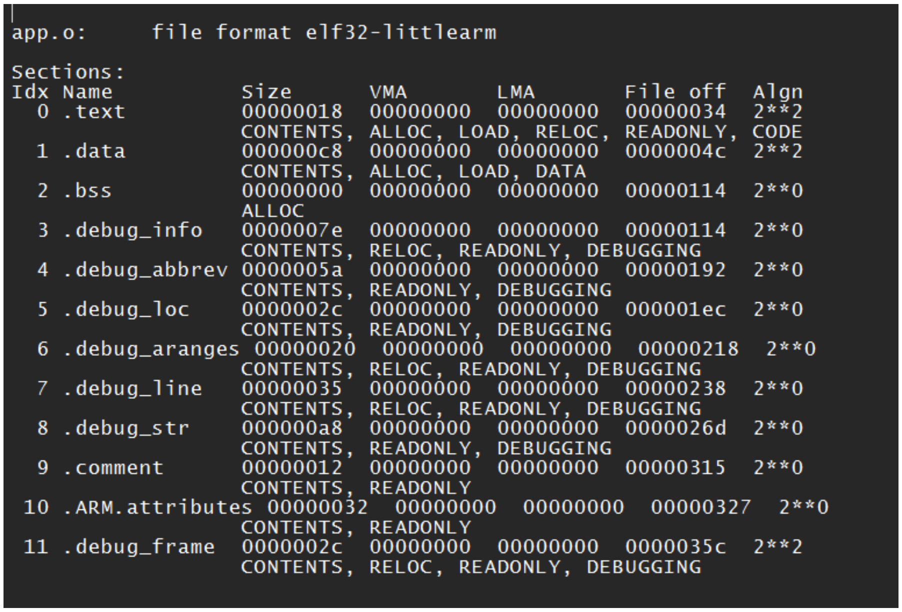
This outputs the sections headers of these objfiles (.data .text .bss .rodata and debug)

.text is for function symbols.

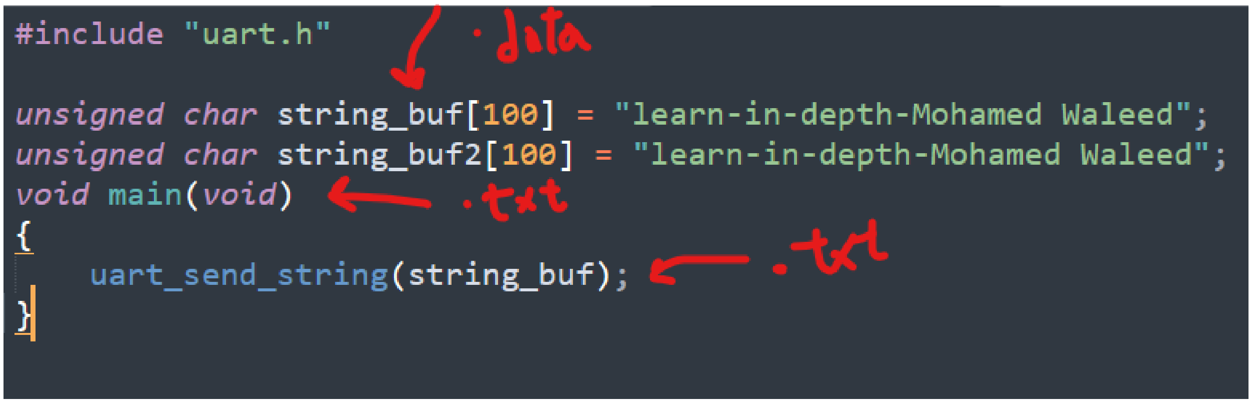
.data is for initialized global and static variables.

.bss is for uninitialized global and static variables.

.rodata is for global const variables.



```
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         000000c8  00000000  00000000  0000004c  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000114  2**0
                  ALLOC
  3 .debug_info   0000007e  00000000  00000000  00000114  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 0000005a  00000000  00000000  00000192  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000002c  00000000  00000000  000001ec  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000218  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000035  00000000  00000000  00000238  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000000a8  00000000  00000000  0000026d  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  00000315  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000032  00000000  00000000  00000327  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000002c  00000000  00000000  0000035c  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

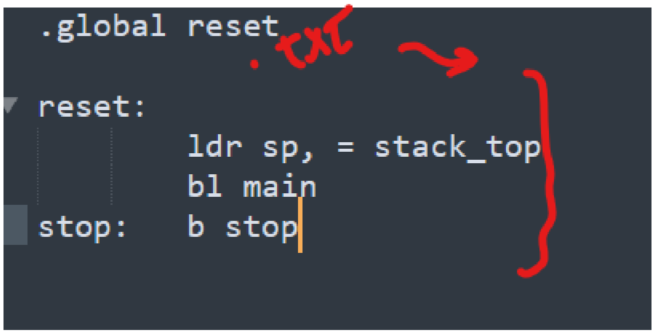


```
#include "uart.h"

unsigned char string_buf[100] = "learn-in-depth-Mohamed Waleed";
unsigned char string_buf2[100] = "learn-in-depth-Mohamed Waleed";
void main(void)
{
    uart_send_string(string_buf);
}
```

VMA and LMA will later be mapped by linker script (linker counter)

OBJECTIVE 2

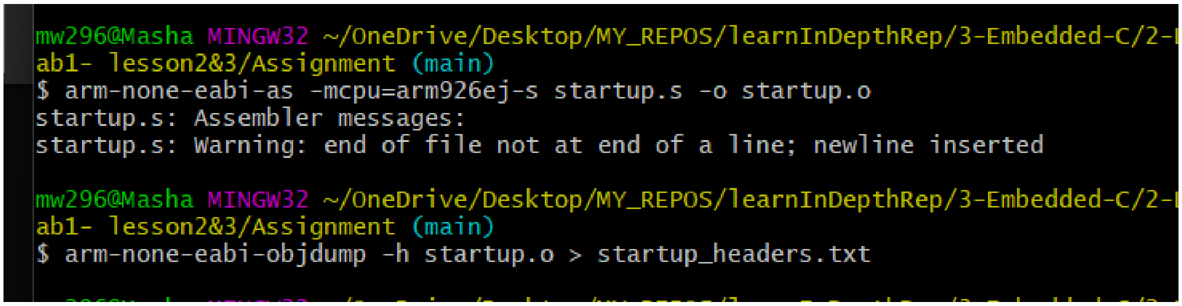
STARTUP FILE



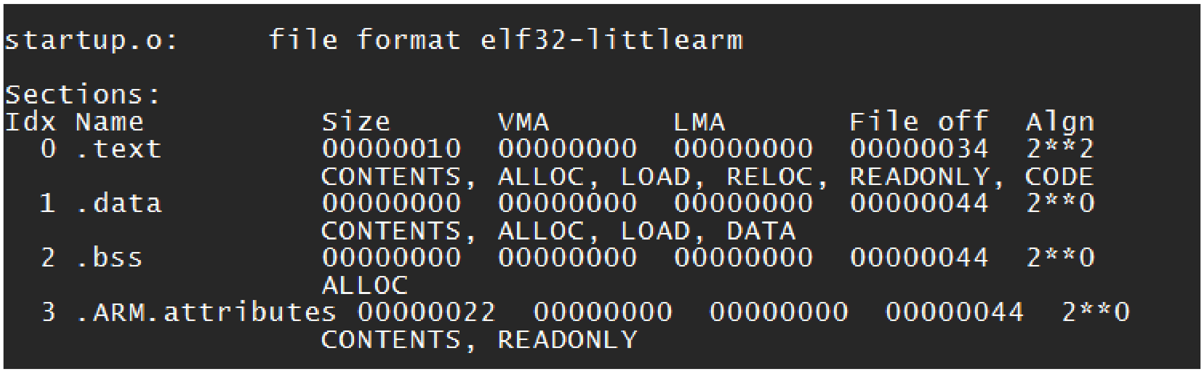
```
 .global reset

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

-initializes stack pointer

-jumps to main

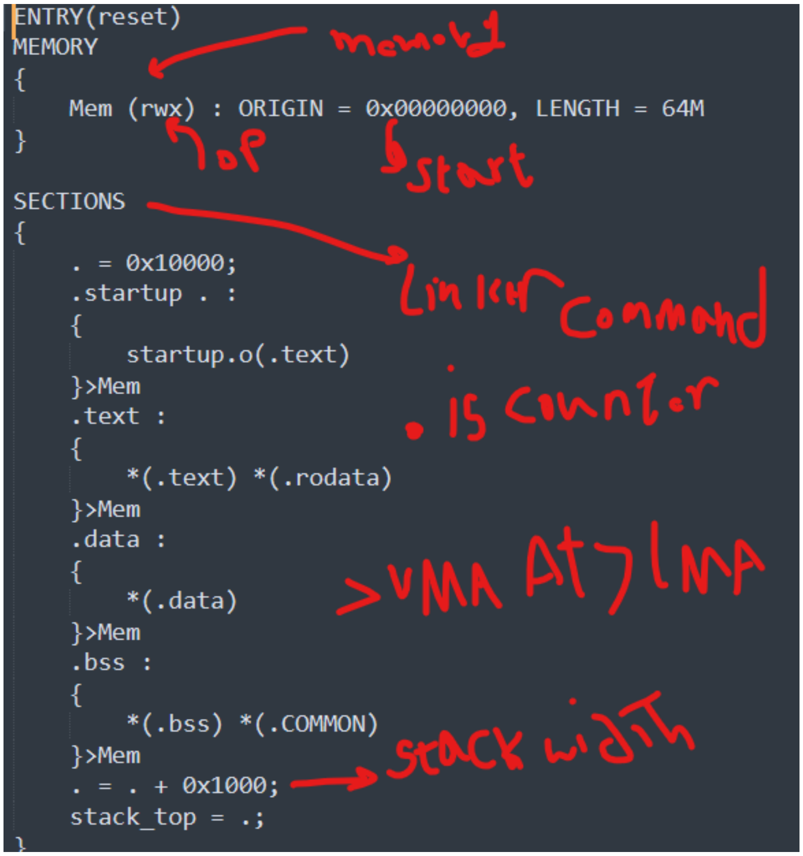


```
mw296@Masha MINGW32 ~/OneDrive/Desktop/MY_REPOS/learnInDepthRep/3-Embedded-C/2-L
ab1- lesson2&3/Assignment (main)
$ arm-none-eabi-as -mcpu=arm926ej-s startup.s -o startup.o
startup.s: Assembler messages:
startup.s: Warning: end of file not at end of a line; newline inserted

mw296@Masha MINGW32 ~/OneDrive/Desktop/MY_REPOS/learnInDepthRep/3-Embedded-C/2-L
ab1- lesson2&3/Assignment (main)
$ arm-none-eabi-objdump -h startup.o > startup_headers.txt
```



```
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
```

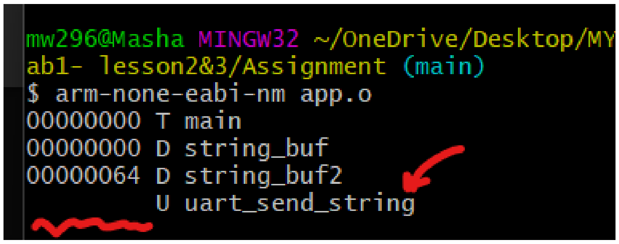
LINKER



```
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 = .;
}
```

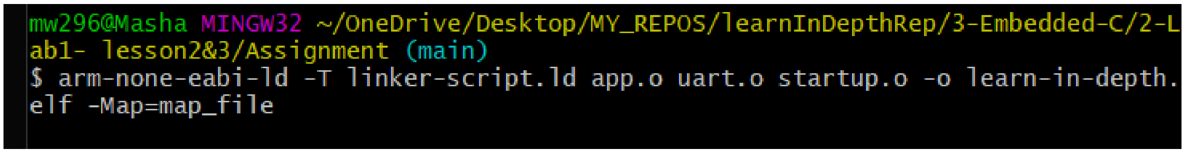
Entry point is at reset in the .text sections header of the startup.o file this is why we start there.

The linker resolves unresolved symbols of the other objfiles:

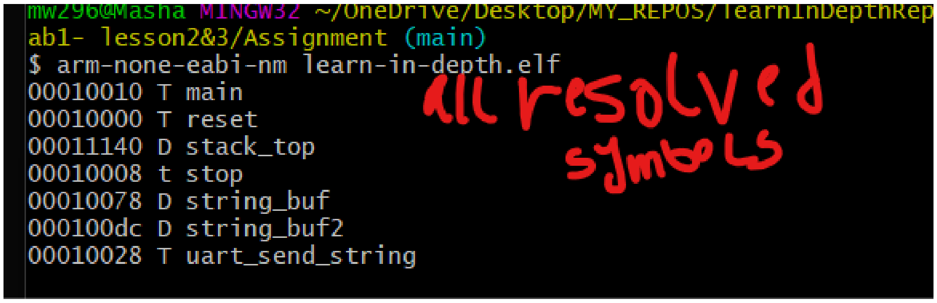


```
mw296@Masha MINGW32 ~/OneDrive/Desktop/MY
ab1- lesson2&3/Assignment (main)
$ arm-none-eabi-nm app.o
00000000 T main
00000000 D string_buf
00000064 D string_buf2
         U uart_send_string
```

LINKING AND ANALYZING ELF FILE



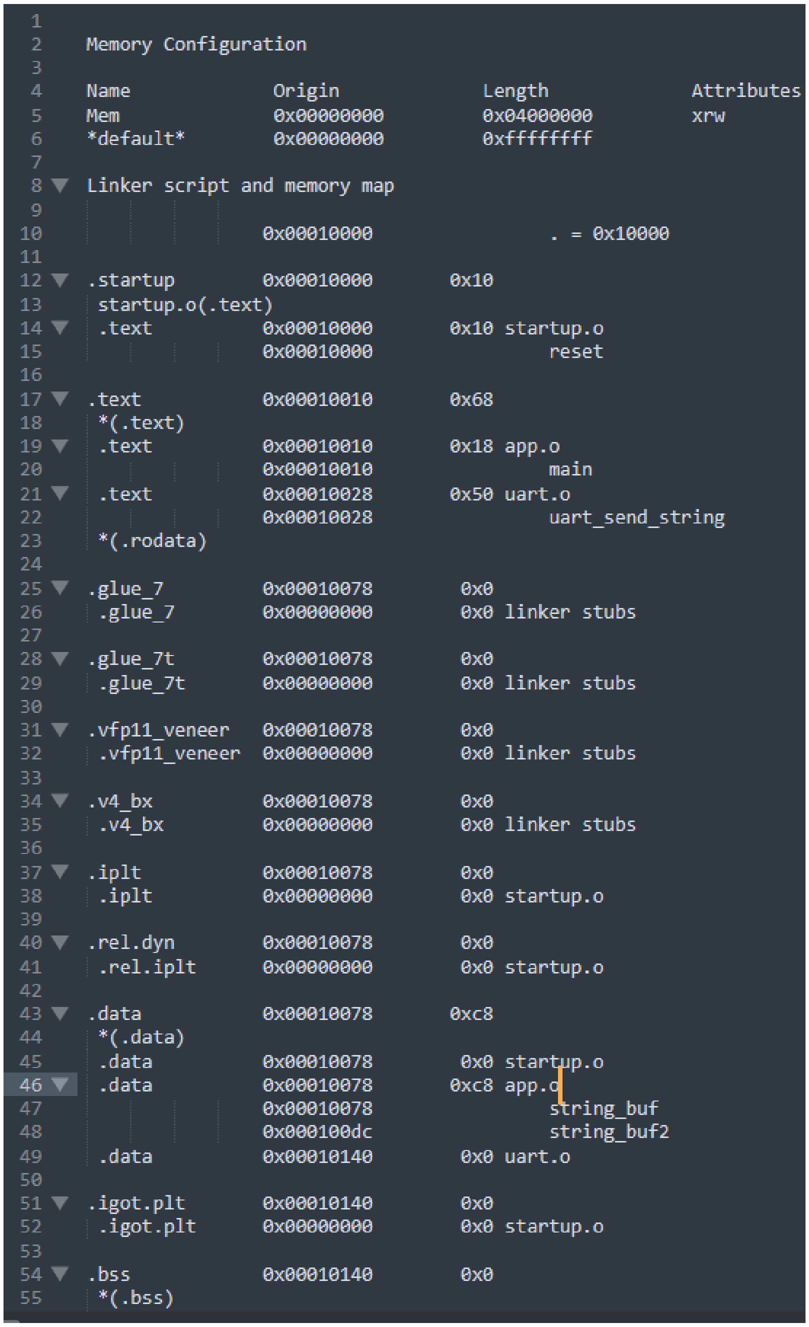
```
mw296@Masha MINGW32 ~/OneDrive/Desktop/MY_REPOS/learnInDepthRep/3-Embedded-C/2-L
ab1- lesson2&3/Assignment (main)
$ arm-none-eabi-ld -T linker-script.ld app.o uart.o startup.o -o learn-in-depth.
elf -Map=map_file
```



```
mw296@Masha MINGW32 ~/OneDrive/Desktop/MY_REPOS/learnInDepthRep
ab1- lesson2&3/Assignment (main)
$ arm-none-eabi-nm learn-in-depth.elf
00010010 T main
00010000 T reset
00011140 D stack_top
00010008 t stop
00010078 D string_buf
000100dc D string_buf2
00010028 T uart_send_string
```

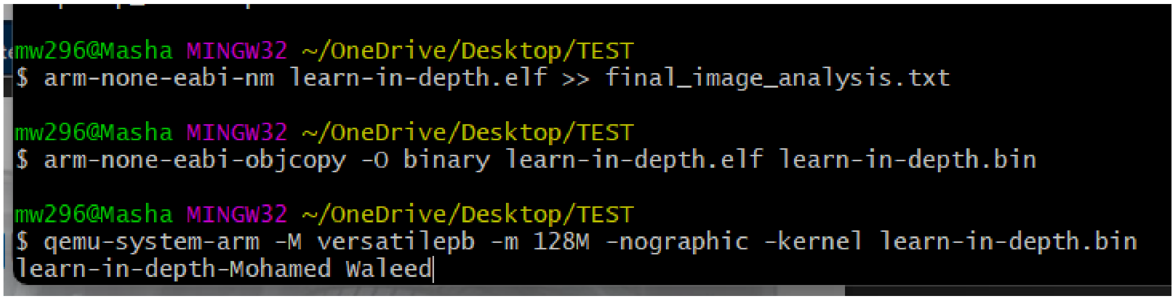
all resolved
symbols

MAPFILE gives the layout of the final image of the software.



```

Memory Configuration

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       0x68
 *(.text)
 .text          0x00010010       0x18 app.o
                0x00010010                main
 .text          0x00010028       0x50 uart.o
                0x00010028                uart_send_string
 *(.rodata)

.glue_7         0x00010078        0x0
 .glue_7        0x00000000        0x0 linker stubs

.glue_7t        0x00010078        0x0
 .glue_7t       0x00000000        0x0 linker stubs

.vfp11_veneer   0x00010078        0x0
 .vfp11_veneer  0x00000000        0x0 linker stubs

.v4_bx          0x00010078        0x0
 .v4_bx         0x00000000        0x0 linker stubs

.iplt           0x00010078        0x0
 .iplt          0x00000000        0x0 startup.o

.rel.dyn        0x00010078        0x0
 .rel.iplt      0x00000000        0x0 startup.o

.data           0x00010078       0xc8
 *(.data)
 .data          0x00010078        0x0 startup.o
 .data          0x00010078       0xc8 app.o
                0x00010078                string_buf
                0x000100dc                string_buf2
 .data          0x00010140        0x0 uart.o

.igot.plt       0x00010140        0x0
 .igot.plt      0x00000000        0x0 startup.o

.bss            0x00010140        0x0
 *(.bss)
```

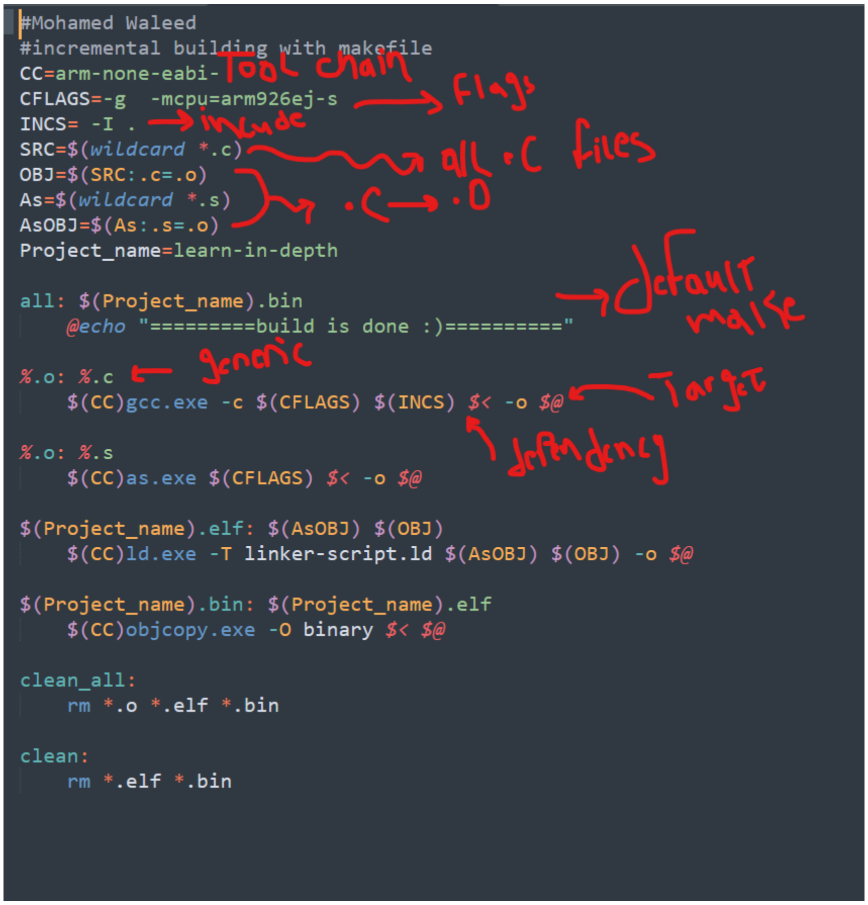
SIMULATION:



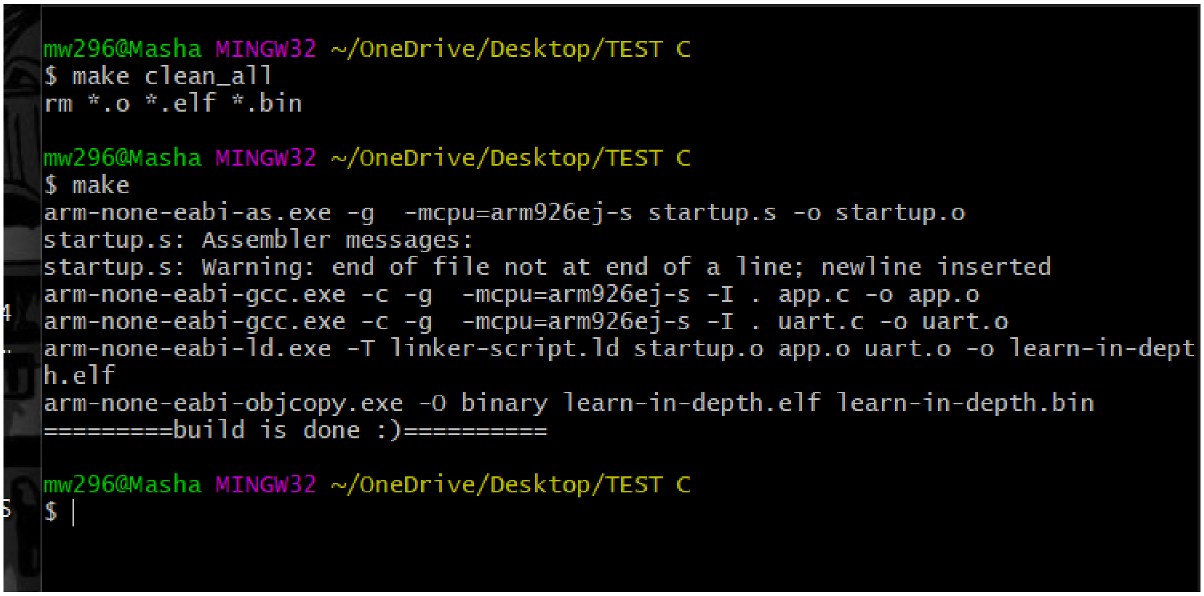
```
mw296@Masha MINGW32 ~/OneDrive/Desktop/TEST
$ arm-none-eabi-nm learn-in-depth.elf >> final_image_analysis.txt

mw296@Masha MINGW32 ~/OneDrive/Desktop/TEST
$ arm-none-eabi-objcopy -O binary learn-in-depth.elf learn-in-depth.bin

mw296@Masha MINGW32 ~/OneDrive/Desktop/TEST
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin
learn-in-depth-Mohamed Waleed
```

OBJECTIVE 3

Make files automate the building process



```
#Mohamed Waleed
#incremental building with makefile
CC=arm-none-eabi-
CFLAGS=-g  -mcpu=arm926ej-s
INCS= -I .
SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)
As=$(wildcard *.s)
AsOBJ=$(As:.s=.o)
Project_name=learn-in-depth

all: $(Project_name).bin
    @echo "=========build is done :)=========="

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

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

$(Project_name).elf: $(AsOBJ) $(OBJ)
    $(CC)ld.exe -T linker-script.ld $(AsOBJ) $(OBJ) -o $@

$(Project_name).bin: $(Project_name).elf
    $(CC)objcopy.exe -O binary $< $@

clean_all:
    rm *.o *.elf *.bin

clean:
    rm *.elf *.bin
```



```
mw296@Masha MINGW32 ~/OneDrive/Desktop/TEST C
$ make clean_all
rm *.o *.elf *.bin

mw296@Masha MINGW32 ~/OneDrive/Desktop/TEST C
$ make
arm-none-eabi-as.exe -g  -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
arm-none-eabi-gcc.exe -c -g  -mcpu=arm926ej-s -I . app.c -o app.o
arm-none-eabi-gcc.exe -c -g  -mcpu=arm926ej-s -I . uart.c -o uart.o
arm-none-eabi-ld.exe -T linker-script.ld startup.o app.o uart.o -o learn-in-dept
h.elf
arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
=========build is done :)==========

mw296@Masha MINGW32 ~/OneDrive/Desktop/TEST C
$ |
```