Embedded System Diploma Online

Lab 2 Report

Name: Ahmed Emad Hassan

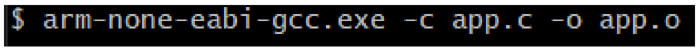
Table of Contents

Steps

1. Create and implement app.c
2. Create and implement uart.h
3. Create and implement uart.c
4. Create and implement startup.s
5. Convert all c files (.c) to object files (.o)
6. Write linker script (.ld)
7. Apply linker script to all object files (.o) to get (.elf) file
8. Convert (.elf) file that contains debugging information to (.bin) file that doesn't contain debugging information
9. Run the binary (.bin) on the board using Qemu simulator

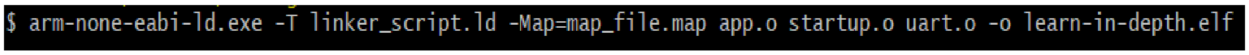
Commands

Convert all c files (.c) to object files (.o)



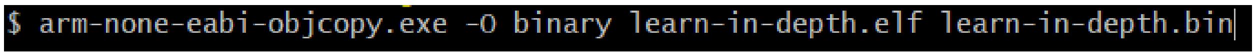
```
$ arm-none-eabi-gcc.exe -c app.c -o app.o
```

Apply linker script to all object files (.o) to get .elf file



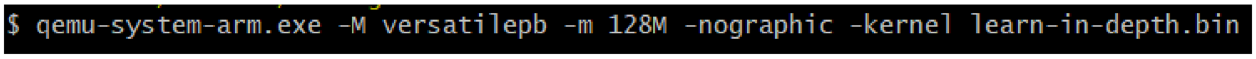
```
$ arm-none-eabi-ld.exe -T linker_script.ld -Map=map_file.map app.o startup.o uart.o -o learn-in-depth.elf
```

Convert (.elf) file to (.bin) file



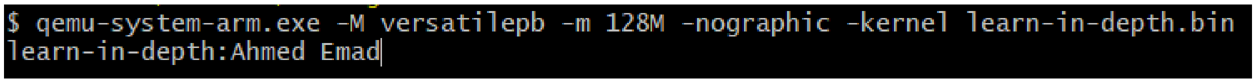
```
$ arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
```

Run the binary (.bin) on the board using Qemu simulator



```
$ qemu-system-arm.exe -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin
```

Output from Qemu



```
$ qemu-system-arm.exe -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin
learn-in-depth:Ahmed Emad
```

Specs

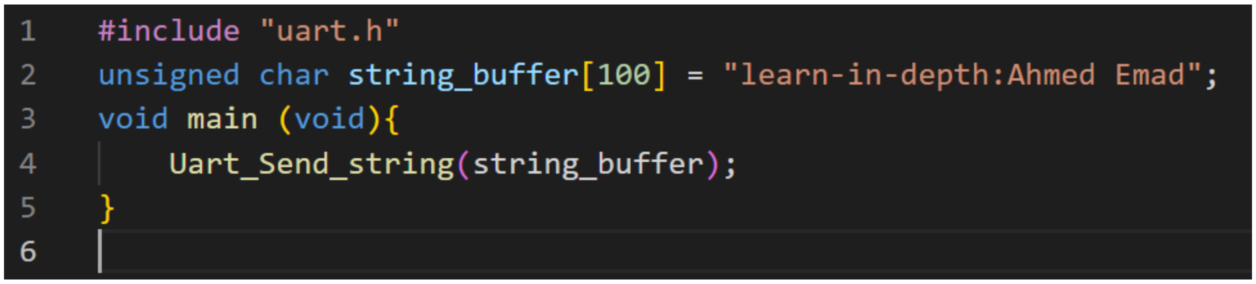
Entry point of processor is: 0x10000

To Activate UART0 you just write on UART0DR register (32bit)

And its address is :0x101f1000

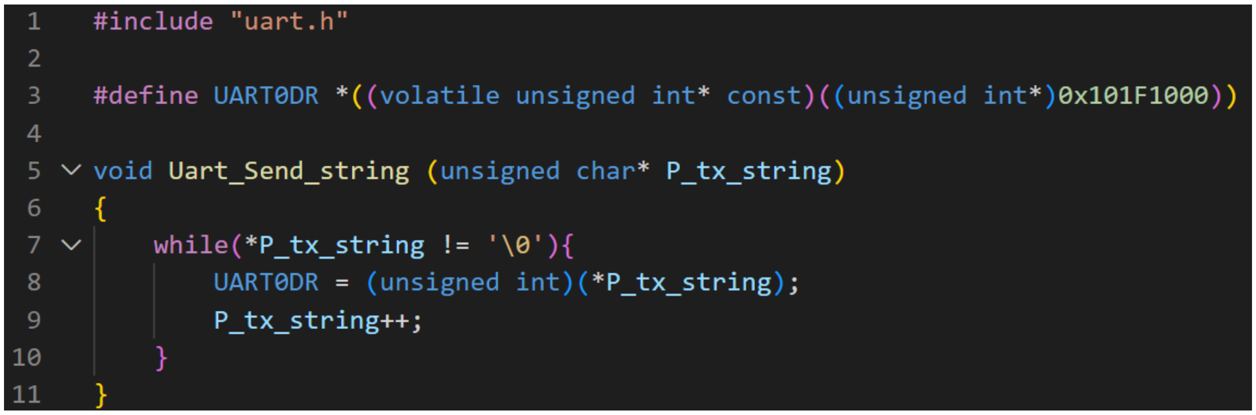
Code

app.c



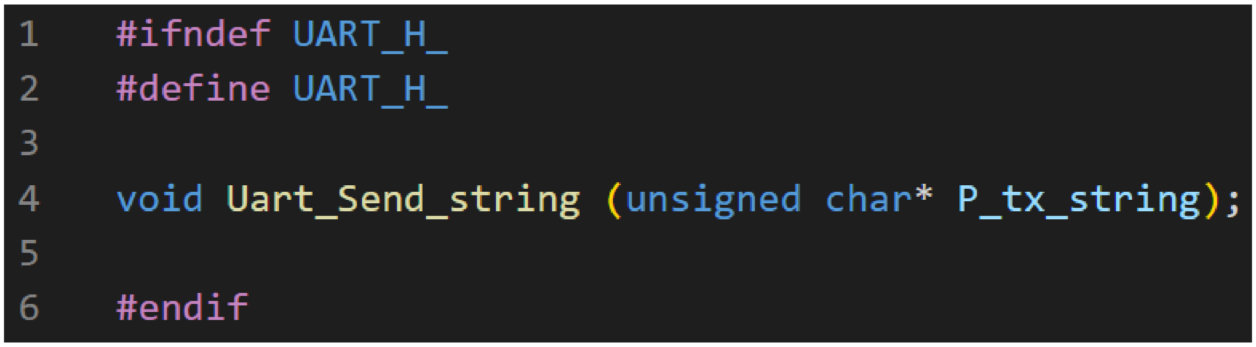
```c
#include "uart.h"
unsigned char string_buffer[100] = "learn-in-depth:Ahmed Emad";
void main (void){
    Uart_Send_string(string_buffer);
}

```

uart.c



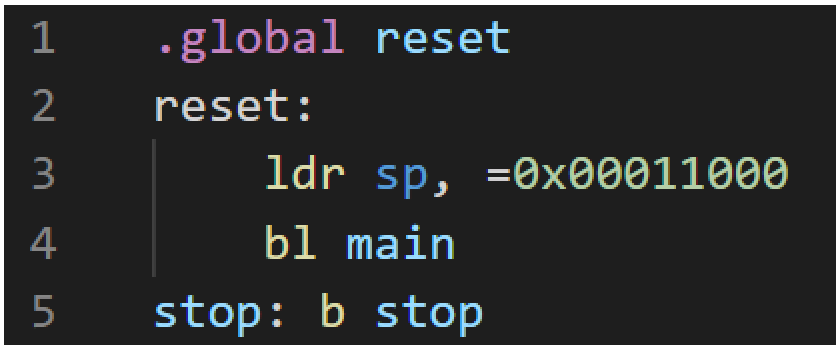
```c
#include "uart.h"

#define UART0DR *((volatile unsigned int* const)((unsigned int*)0x101F1000))

void Uart_Send_string (unsigned char* P_tx_string)
{
    while(*P_tx_string != '\0'){
        UART0DR = (unsigned int)(*P_tx_string);
        P_tx_string++;
    }
}
```

uart.h



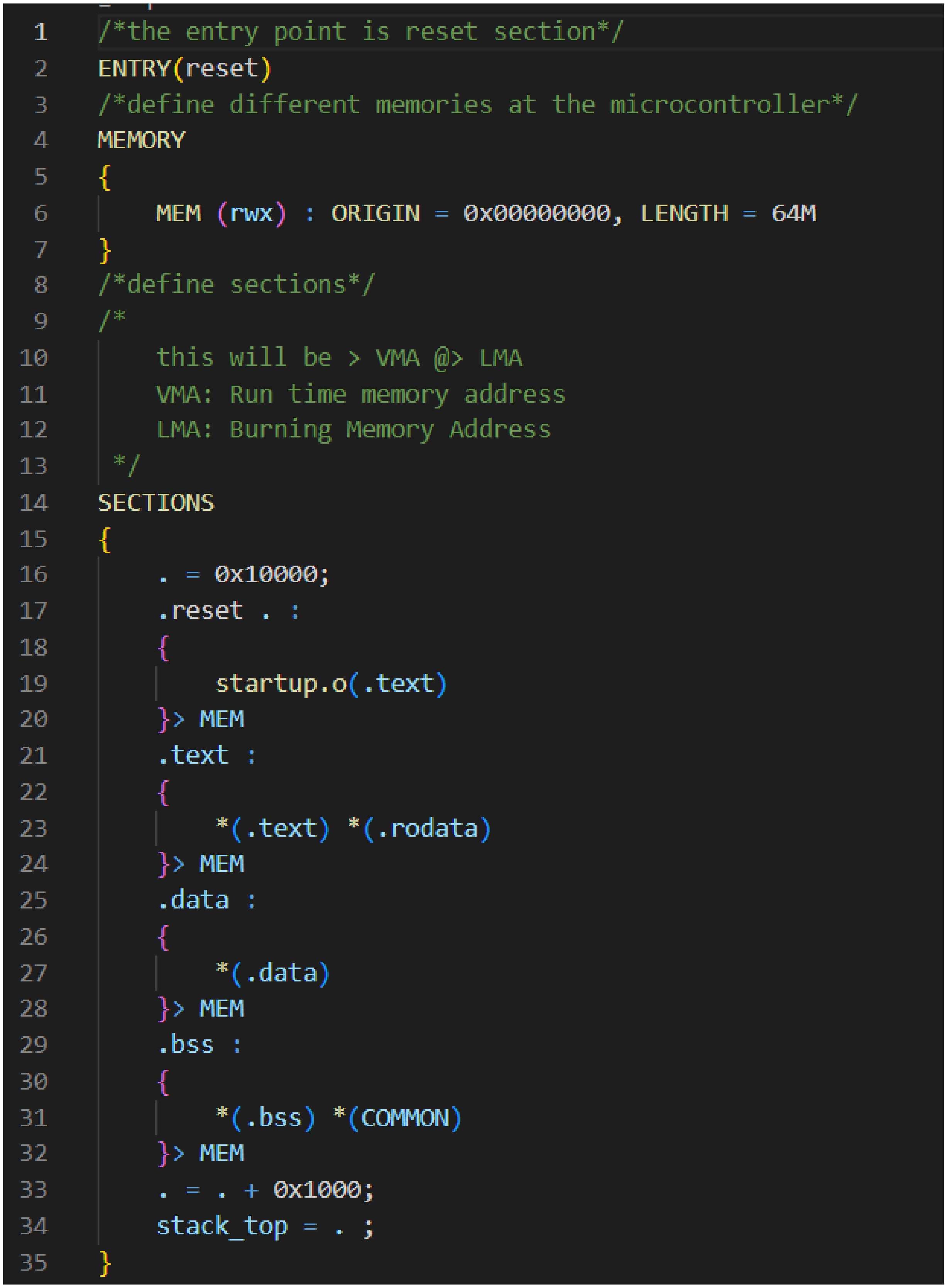
```c
#ifndef UART_H_
#define UART_H_

void Uart_Send_string (unsigned char* P_tx_string);

#endif
```

startup.s



```
.global reset
reset:
    ldr sp, =0x00011000
    bl main
stop: b stop
```

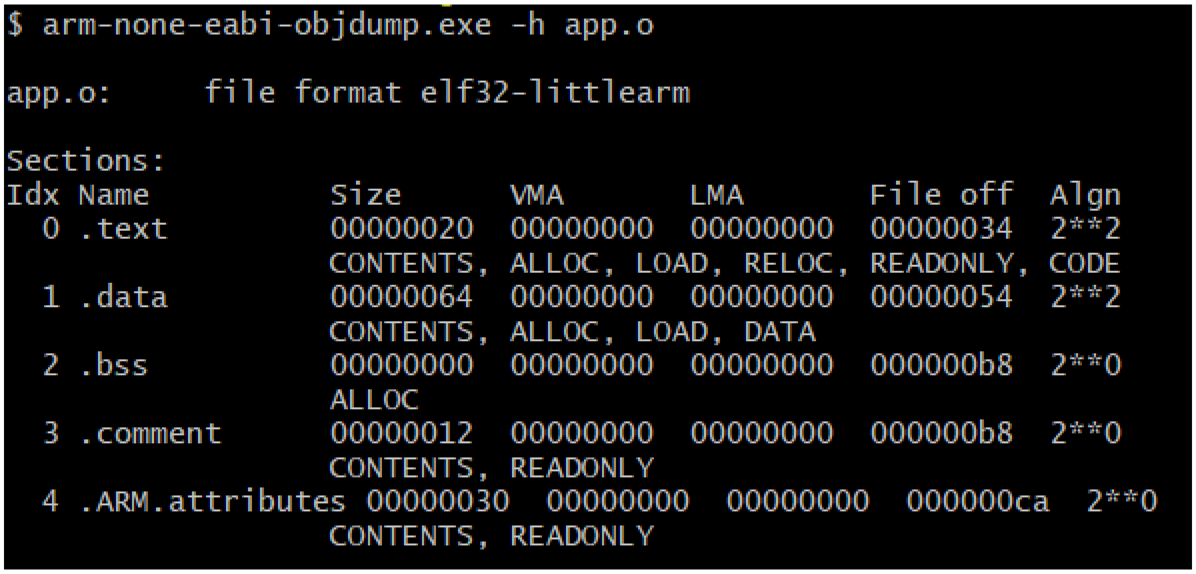
Linker_script.ld



```
/*the entry point is reset section*/
ENTRY(reset)
/*define different memories at the microcontroller*/
MEMORY
{
    MEM (rwx) : ORIGIN = 0x00000000, LENGTH = 64M
}
/*define sections*/
/*
    this will be > VMA @> LMA
    VMA: Run time memory address
    LMA: Burning Memory Address
 */
SECTIONS
{
    . = 0x10000;
    .reset . :
    {
        startup.o(.text)
    }> MEM
    .text :
    {
        *(.text) *(.rodata)
    }> MEM
    .data :
    {
        *(.data)
    }> MEM
    .bss :
    {
        *(.bss) *(COMMON)
    }> MEM
    . = . + 0x1000;
    stack_top = . ;
}
```

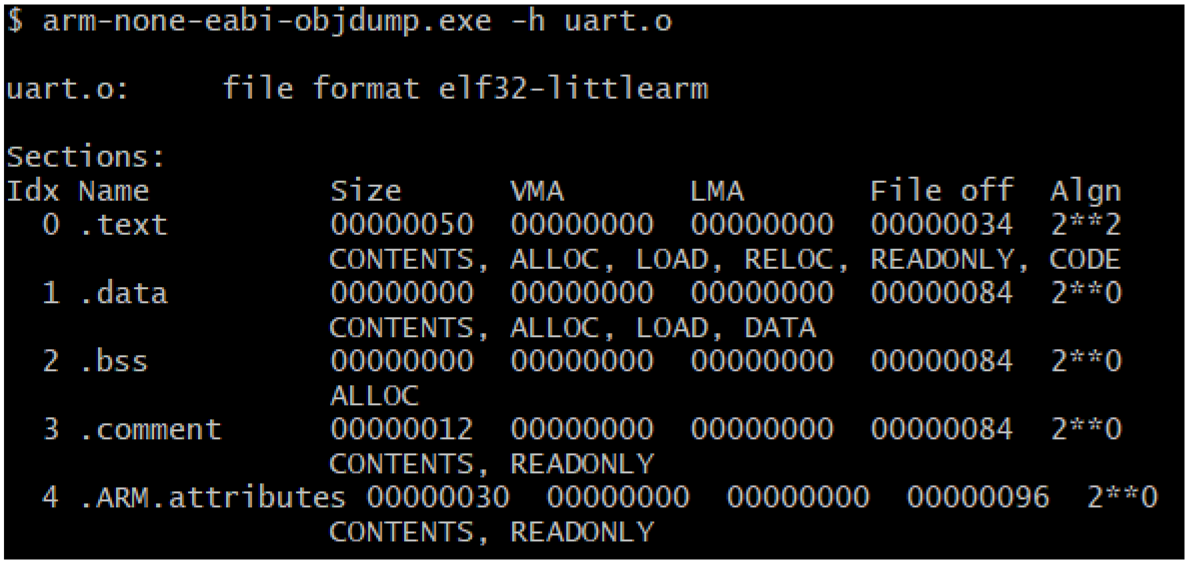
Object files sections

app.o



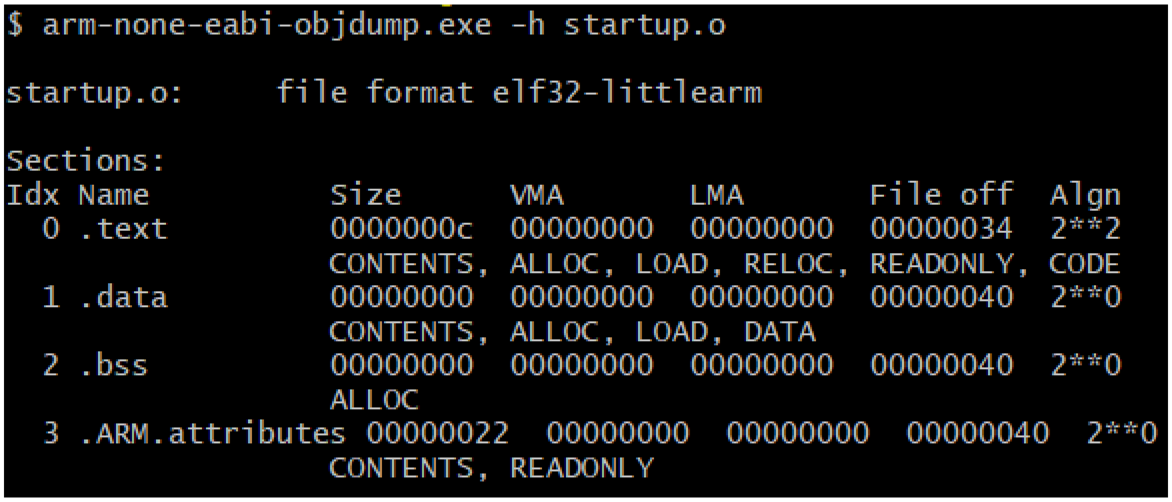
```
$ arm-none-eabi-objdump.exe -h app.o

app.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000020  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000064  00000000  00000000  00000054  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b8  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  000000b8  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000030  00000000  00000000  000000ca  2**0
                  CONTENTS, READONLY
```

uart.o



```
$ arm-none-eabi-objdump.exe -h uart.o

uart.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000050  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000084  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000084  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  00000084  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000030  00000000  00000000  00000096  2**0
                  CONTENTS, READONLY
```

startup.o



```
$ 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         0000000c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000040  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000040  2**0
                  ALLOC
  3 .ARM.attributes 00000022  00000000  00000000  00000040  2**0
                  CONTENTS, READONLY
```