Embedded C - Lab 1 Report

Name: Ahmed Ashraf Ahmed Ibrahim Mohamed

Email: ahmed.zahran.aaz@gmail.com

1 Source files

Source Code 1.1: app.c

```c
#include "uart.h"
#include "PLATFORM_TYPES.h"

uint8_t string_buffer[100] = "Learn-in-depth: Ahmed Ashraf";
uint8_t const string_buffer2[100] = "Hello";

void main (void)
{
  uart_send_string(string_buffer);
}
```

Source Code 1.2: uart.c

```c
#include "uart.h"
#include "PLATFORM_TYPES.h"

#define UART0DR *((vuint32_t* const) ((uint32_t*) 0x101f1000))

void uart_send_string (uint8_t* P_tx_string)
{
  while (*P_tx_string != '\0')
  {
    UART0DR = (uint32_t) (*P_tx_string);
    P_tx_string++;
  }
}
```

Source Code 1.3: uart.h

```c
#ifndef _UART_H_
#define _UART_H_

#include "PLATFORM_TYPES.h"

```

```
void uart_send_string (uint8_t* P_tx_string);

#endif
```

Source Code 1.4: PLATFORM$_TYPES.h$

```
#ifndef PLATFORM_TYPES_H_
#define PLATFORM_TYPES_H_
#include <stdbool.h>
#include <stdint.h>



#define CPU_TYPE            CPU_TYPE_32
#define CPU_BIT_ORDER       MSB_FIRST
#define CPU_BYTE_ORDER      HIGH_BYTE_FIRST

#ifndef FALSE
#define FALSE (boolean)false
#endif // !FALSE

#ifndef TRUE
#define TRUE (boolean)true
#endif // !FALSE


typedef volatile int8_t             vint8_t;
typedef volatile uint8_t            vuint8_t;

typedef volatile int16_t            vint16_t ;
typedef volatile uint16_t           vuint16_t ;


typedef volatile int32_t            vint32_t  ;
typedef volatile uint32_t           vuint32_t  ;


typedef volatile int64_t            vint64_t  ;
```

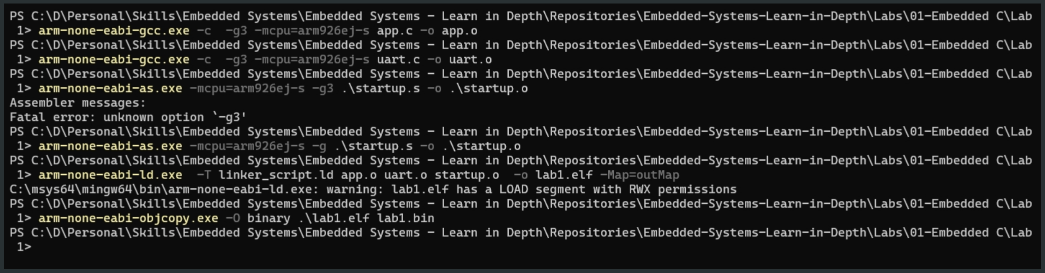
```c
typedef volatile uint64_t         vuint64_t   ;

#endif // !PLATFORM_TYPES_H_
```

Source Code 1.5: startup.s

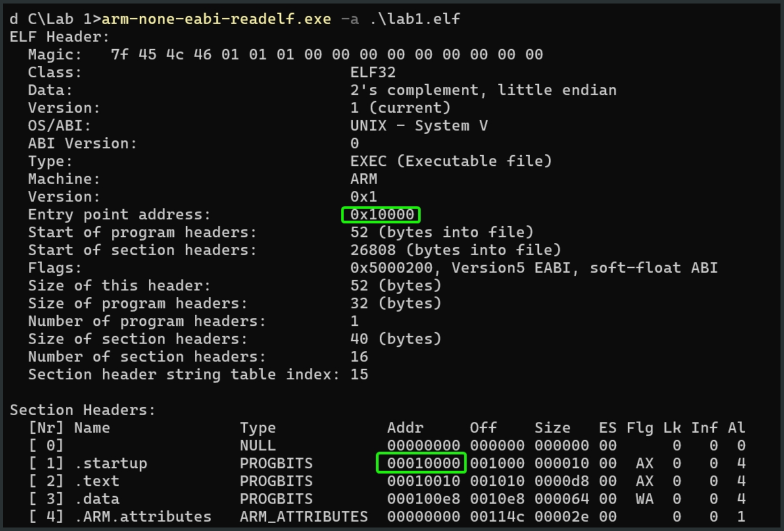
```asm
.globl reset
reset:
  ldr sp, = stack_top
  bl main
stop : b stop
```

2 Compiling the source files and startup files and creating the binary image



```
PS C:\D\Personal\Skills\Embedded Systems\Embedded Systems - Learn in Depth\Repositories\Embedded-Systems-Learn-in-Depth\Labs\01-Embedded C\Lab
 1> arm-none-eabi-gcc.exe -c  -g3 -mcpu=arm926ej-s app.c -o app.o
PS C:\D\Personal\Skills\Embedded Systems\Embedded Systems - Learn in Depth\Repositories\Embedded-Systems-Learn-in-Depth\Labs\01-Embedded C\Lab
 1> arm-none-eabi-gcc.exe -c  -g3 -mcpu=arm926ej-s uart.c -o uart.o
PS C:\D\Personal\Skills\Embedded Systems\Embedded Systems - Learn in Depth\Repositories\Embedded-Systems-Learn-in-Depth\Labs\01-Embedded C\Lab
 1> arm-none-eabi-as.exe -mcpu=arm926ej-s -g3 .\startup.s -o .\startup.o
Assembler messages:
Fatal error: unknown option `-g3'
PS C:\D\Personal\Skills\Embedded Systems\Embedded Systems - Learn in Depth\Repositories\Embedded-Systems-Learn-in-Depth\Labs\01-Embedded C\Lab
 1> arm-none-eabi-as.exe -mcpu=arm926ej-s -g .\startup.s -o .\startup.o
PS C:\D\Personal\Skills\Embedded Systems\Embedded Systems - Learn in Depth\Repositories\Embedded-Systems-Learn-in-Depth\Labs\01-Embedded C\Lab
 1> arm-none-eabi-ld.exe  -T linker_script.ld app.o uart.o startup.o  -o lab1.elf -Map=outMap
C:\msys64\mingw64\bin\arm-none-eabi-ld.exe: warning: lab1.elf has a LOAD segment with RWX permissions
PS C:\D\Personal\Skills\Embedded Systems\Embedded Systems - Learn in Depth\Repositories\Embedded-Systems-Learn-in-Depth\Labs\01-Embedded C\Lab
 1> arm-none-eabi-objcopy.exe -O binary .\lab1.elf lab1.bin
PS C:\D\Personal\Skills\Embedded Systems\Embedded Systems - Learn in Depth\Repositories\Embedded-Systems-Learn-in-Depth\Labs\01-Embedded C\Lab
 1>
```

3 Analyzing the binary output using `readelf`

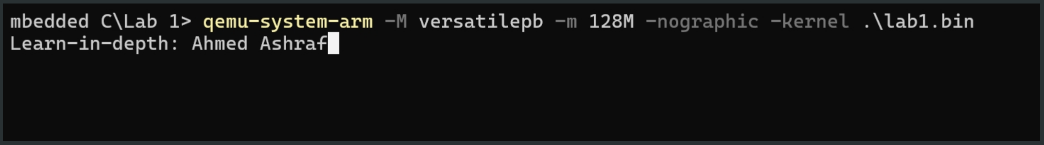


```
d C\Lab 1>arm-none-eabi-readelf.exe -a .\lab1.elf
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x10000
  Start of program headers:          52 (bytes into file)
  Start of section headers:          26808 (bytes into file)
  Flags:                             0x5000200, Version5 EABI, soft-float ABI
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         1
  Size of section headers:           40 (bytes)
  Number of section headers:         16
  Section header string table index: 15

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .startup          PROGBITS        00010000 001000 000010 00  AX  0   0  4
  [ 2] .text             PROGBITS        00010010 001010 0000d8 00  AX  0   0  4
  [ 3] .data             PROGBITS        000100e8 0010e8 000064 00  WA  0   0  4
  [ 4] .ARM.attributes   ARM_ATTRIBUTES  00000000 00114c 00002e 00      0   0  1
```

Figure 3.1: Making sure startup address and entry point address are the same

4 Running the result using qemu



```
mbedded C\Lab 1> qemu-system-arm -M versatilepb -m 128M -nographic -kernel .\lab1.bin
Learn-in-depth: Ahmed Ashraf
```