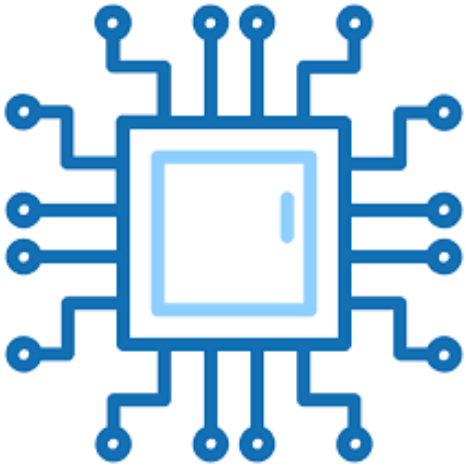


Mastering embedded systems diploma

embedded c lab2 report

made by: Abdelrahman osama

supervisor: Keroles Shenouda

Table of Contents

objectives:

- Writing c code to send a message using Uart on ARM VersatilePB board
- Writing a startup file to ARM VersatilePB board
- Writing linker script file to ARM VersatilePB board
- Writing make file
- Build automation by make file
- Simulation of code on qemu

c code:

1) app.c:

app.c

```c
#include "uart.h"

unsigned char string_buffer[100] = "Learn in depth: Abdelrahman";

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

2) uart.c:

```c
//include "uart.h"
#define  UART0DR     *(volatile unsigned int * const)((unsigned int *) 0x101f1000)

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

3) uart.h:

```c
#ifndef _UART_H_
#define _UART_H_

void uart_send_string (unsigned char* ptr_tx_string);

#endif
```

Startup file:

Here we write a simple startup by:

1) putting reset section at entry point
2) initializing stack pointer
3) branching to main

```
1  .global reset
2  reset :
3      ldr sp, = stack_top
4      bl main
5
6  stop: bl stop
7
```

Linker script file:

Using linker script commands we

1) define the memory:
   - specifications (read or/and write or/and execute)
   - origin (address)
   - length (size)

2) define different sections:
   a. .text section
   b. .data section (for initialized static and global variables)
   c. .bss section (for uninitialized static and global variables)

3) Defining the stack top

```
ENTRY (reset)

MEMORY
{
    mem(rwx): ORIGIN = 0x00000000 , LENGTH = 64M
}

SECTIONS
{
    . = 0x10000 ;

    .startup . :
    {
```

```
11
12      .startup . :
13      {
14          startup.o(.text)
15      } > mem
16
17      .text :
18      {
19          *(.text)
20      } > mem
21
22      .data :
23      {
24          *(.data)
25      } > mem
26
27      .bss :
28      {
29          *(.bss)
30      } > mem
31
32      . = . + 0x1000;
33
34      stack_top = .;
35  }
```

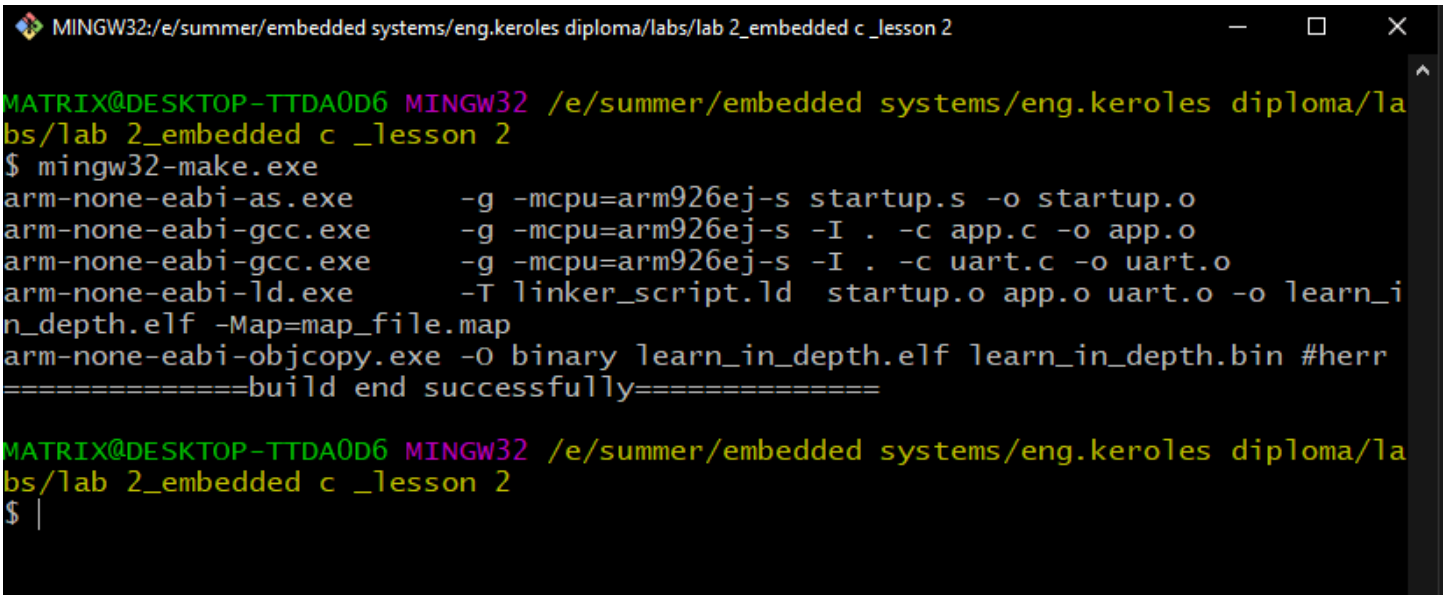
Line 1, Column 1

Make file:

Make file is the file used to automate the build process

```
3   CC=arm-none-eabi-
4   CFLAGS=-g -mcpu=arm926ej-s
5   INSC=-I .
6   LIBS=
7   SRC=$(wildcard *.c)
8   OBJ=$(SRC:.c=.o)
9   AS=$(wildcard *.s)
10  ASOBJ=$(AS:.s=.o)
11  PROJECT_NAME=learn_in_depth
12
13  all : $(PROJECT_NAME).bin
14      @echo "==============build end successfully=============="
15
16  %.o : %.c
17      $(CC)gcc.exe     $(CFLAGS) $(INSC) -c $< -o $@
18
19  %.o : %.s
20      $(CC)as.exe      $(CFLAGS) $< -o $@
21
22  $(PROJECT_NAME).elf : $(ASOBJ) $(OBJ)
23      $(CC)ld.exe      -T linker_script.ld $(LIBS) $(ASOBJ) $(OBJ) -o $@ -Map=map_file.map
24
25
26  $(PROJECT_NAME).bin : $(PROJECT_NAME).elf
27      $(CC)objcopy.exe -O binary $< $@ #herr
28
29  clean_all :
30      rm *.o *.elf *.bin
31      @echo "==============cleaned successfully=============="
32
33  clean :
34      rm *.elf *.bin       #in case there is a change in linker script
35      @echo "==============cleaned successfully=============="
36
37  # two none convention rules added by me
Line 1, Column 1
```

Build process:



Symbols:

1) app.o symbols

```
MATRIX@DESKTOP-TTDA0D6 MINGW32 /e/summer/embedded systems/eng.keroles diplo
mbedded c _unit 3_lesson 2
$ arm-none-eabi-nm.exe app.o
00000000 T main
00000000 D string_buffer
         U uart_send_string

MATRIX@DESKTOP-TTDA0D6 MINGW32 /e/summer/embedded systems/eng.keroles diplo
mbedded c _unit 3_lesson 2
$
```

2) uart.o symbols

```
MATRIX@DESKTOP-TTDA0D6 MINGW32 /e/summer/embedded systems/eng.keroles dip
mbedded c _unit 3_lesson 2
$ arm-none-eabi-nm.exe uart.o
00000000 T uart_send_string
```

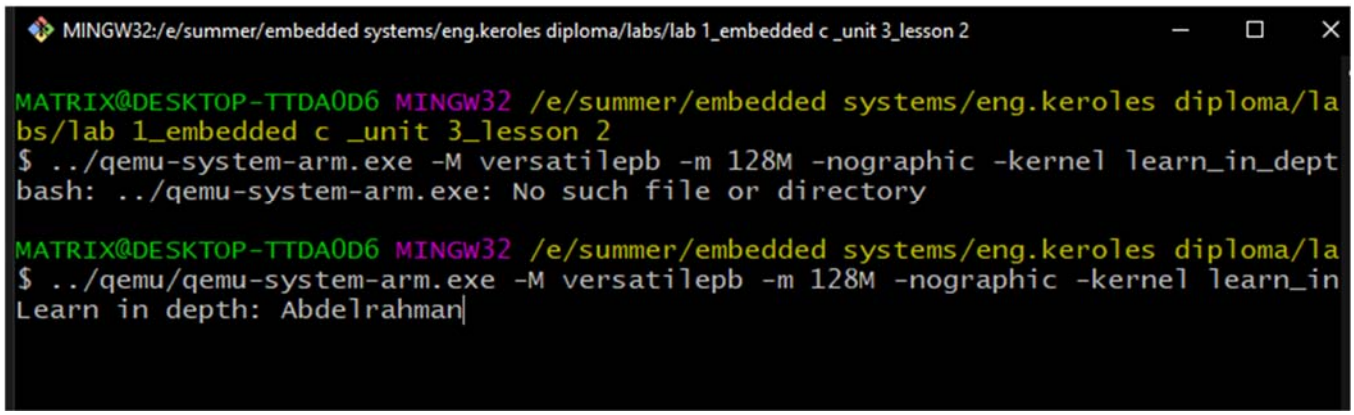
3) startup.o symbols

```
MATRIX@DESKTOP-TTDA0D6 MINGW32 /e/summer/embedded systems/eng.keroles diploma
mbedded c _unit 3_lesson 2
$ arm-none-eabi-nm.exe startup.o
         U main
00000000 T reset
         U stack_top
00000008 t stop
```

4) project symbols

```
MATRIX@DESKTOP-TTDA0D6 MINGW32 /e/summer/embedded systems/eng.keroles diploma
mbedded c _unit 3_lesson 2
$ arm-none-eabi-nm.exe learn_in_depth.elf
00010060 T main
00010000 T reset
000110dc D stack_top
00010008 t stop
00010078 D string_buffer
00010010 T uart_send_string
```

qemu simulation:



map file:

```
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

.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
```

Note on map file:

- in map file, we check that reset section is put at entry point of the processor