UART Example

1. Write the UART module and the application

a. UART.h

```
#ifndef _UART_H_

#define _UART_H_


//UART APIS
void Uart_Send_String(unsigned char *P_tx_string);

#endif
```

b. UART.c

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

void Uart_Send_String(unsigned char *P_tx_string){

    while (*P_tx_string != '\0')
    {
        UARTODR = (unsigned int)(*P_tx_string);
        P_tx_string++;
    }
}
```

c. App.c

```
#include "uart.h"

//to check content " arm-none-eabi-objdump.exe -h app.o"
unsigned char string_buffer[100] = "Learn-in-depth:<John Magdy William> ~ 23/7/2022";
unsigned char const ro_data[50] = "read only data /'rodata/' ";
unsigned char unint_data[50];

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

2. Compile the files using GNU ARM-Cross-toolchain “arm-none-eabi-gcc.exe” that we downloaded (after adding it to the path variables)

```
arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s app.c  -o app.o
```

```
arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s Uart.c -o Uart.o
```

```
-c          Compile and assemble, but do not link
```

arm926ej-s is the board that we will use

3. Navigate the .obj files (relocatable images)

```
$ arm-none-eabi-objdump.exe --help
Usage: D:\courses\new_diploma\Diploma online\EmbeddedC\labs\unit3-lesson2\ARM\bin
ion(s)> <file(s)>
 Display information from object <file(s)>.
 At least one of the following switches must be given:
  -a, --archive-headers    Display archive header information
  -f, --file-headers       Display the contents of the overall file header
  -p, --private-headers    Display object format specific file header contents
  -P, --private=OPT,OPT... Display object format specific contents
  -h, --[section-]headers  Display the contents of the section headers
  -x, --all-headers        Display the contents of all headers
  -d, --disassemble        Display assembler contents of executable sections
  -D, --disassemble-all    Display assembler contents of all sections
  -S, --source             Intermix source code with disassembly
  -s, --full-contents      Display the full contents of all sections requested
  -g, --debugging          Display debug information in object file
  -e, --debugging-tags     Display debug information using ctags style
  -G, --stabs              Display (in raw form) any STABS info in the file
  -W[lLiaprmfFsoRtUuTgAckK] or
  --dwarf[=rawline,=decodedline,=info,=abbrev,=pubnames,=aranges,=macro,=frames,
```

```
arm-none-eabi-objdump.exe -D app.o > app.s
```

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

4. Startup code

a. Write Startup.s

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

b. Compile

```
arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o startup.o
```

c. Analyze it

```
arm-none-eabi-objdump.exe -h startup.o
```

5. Linker Script

```
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 memory is 4 kb*/
    stack_top = . ;
}
```

6. Link all the objects together

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

```
arm-none-eabi-objdump.exe learn-in-depth-john.elf -h
```

7. Run the program in the QEMU Simulator

```
qemu-system-arm -M versatilepb -m 128M -nographic -kernel
learn-in-depth-john.elf
```

```
johnm@John MINGW64 /d/University/Courses/Embedded Diplo
ma/Embedded_Systems_Diploma/Unit_3/Lesson 2 - UART Exam
ple (master)
$  qemu-system-arm -M versatilepb -m 128M -nographic -k
ernel learn-in-depth-john.elf
Learn-in-depth:<John Magdy William> ~ 23/7/2022
```

8. Creating the Makefile

```
#copyright : John

#name of compiler
CC=arm-none-eabi-

#repeated flags in the build
CFLAGS= -g -mcpu=arm926ej-s
LIBS=
INCS= -I .

SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
As = $(wildcard *.s)
AsOBJ = $(As:.s=.o)
Project_name=Learn-in-depth-john

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

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

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

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

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

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

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