BAREMETAL APPLICATION ON ARM VERSTILEPB REPORT

--First of all, this BareMetal SW without os and running this SW without IDE.

- SO, we need to write and execute every driver

objectives

Create a SW to send a “learn-in-depth<Ahmed>” using uart

tools

-notepad++ or sublime

--cross toolchain to compile like arm-non-eapi toolchain

-- emulator to have versatile board like qemu

Source code

- App.c
- Uart.h
- Uart.c
- Startup.s
- Linker_ script. ld.

Uart.c

```c

#include"uart.h"

#define UART0DR_REG   *((volatile unsigned int*)((unsigned int *)0x101f1000))

void uart_send_string(unsigned char *p_tx_string)
{
    while(*p_tx_string != '\0')
    {
        UART0DR_REG = (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
```

App.c

```c
#include "uart.h"
unsigned char string_buffer[100]="learn-in-depth<ahmed>";
unsigned char const string_buffer1[100]="learn-in-depth<ahmed>";
int main()
{
    uart_send_string(string_buffer);
    return 0;
}
```

Startup.c

```
.global reset

reset:
    ldr sp, =stack_top
    bl main

stop: b stop
```

Linker_script.ld

```
1
2   ENTRY(reset)
3
4   MEMORY
5   {
6       Mem (rwx) : o = 0x00000000 , l = 64M
7   }
8   SECTIONS
9   {
10      . = 0x10000;
11      .startup . :
12      {
13        startup.o(.text)
14      }> Mem
15      .txt :
16      {
17        *(.text)
18      }> Mem
19      .data :
20      {
21       *(.data)
22      }> Mem
23      .bss :
24      {
25        *(.bss)
26      }> Mem
27      . = . + 0x1000;
28      stack_top = .;
29  }
```

Compilation process

Pre-processing and compiling

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

$ arm-none-eabi-gcc.exe -c -I -g -mcpu=arm926ej-s uart.c -o uart.o

Assembling

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

Linking and map_file

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

Binary_file:

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

Run in qemu simulator

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

Sections of relocatable locations

```
$ 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, 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 00000032  00000000  00000000  00000096  2**0
                  CONTENTS, READONLY
```

```
$ 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         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
```

```
$ 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 .rodata       00000064  00000000  00000000  000000b8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      00000012  00000000  00000000  0000011c  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 00000032  00000000  00000000  0000012e  2**0
                  CONTENTS, READONLY
```

Run time addresses

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

learn-in-depth.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .startup      00000010  00010000  00010000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .txt          00000070  00010010  00010010  00008010  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .rodata       00000064  00010080  00010080  00008080  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .data         00000064  000100e4  000100e4  000080e4  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  4 .ARM.attributes 0000002e  00000000  00000000  00008148  2**0
                  CONTENTS, READONLY
  5 .comment      00000011  00000000  00000000  00008176  2**0
                  CONTENTS, READONLY
```

Symbols of elf file

```
$ arm-none-eabi-nm.exe learn-in-depth.elf
00010010 T main
00010000 T reset
00011148 D stack_top
00010008 t stop
000100e4 D string_buffer
00010080 R string_buffer1
00010030 T uart_send_string
```

Disassembly

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

app.o:     file format elf32-littlearm


Disassembly of section .text:

00000000 <main>:
   0:   e92d4800        push    {fp, lr}
   4:   e28db004        add     fp, sp, #4
   8:   e59f000c        ldr     r0, [pc, #12]   ; 1c <main+0x1c>
   c:   ebfffffe        bl      0 <uart_send_string>
  10:   e3a03000        mov     r3, #0
  14:   e1a00003        mov     r0, r3
  18:   e8bd8800        pop     {fp, pc}
  1c:   00000000        andeq   r0, r0, r0

Disassembly of section .data:

00000000 <string_buffer>:
   0:   7261656c        rsbvc   r6, r1, #108, 10        ; 0x1b000000
   4:   6e692d6e        cdpvs   13, 6, cr2, cr9, cr14, {3}
   8:   7065642d        rsbvc   r6, r5, sp, lsr #8
   c:   613c6874        teqvs   ip, r4, ror r8
  10:   64656d68        strbtvs r6, [r5], #-3432        ; 0xd68
  14:   0000003e        andeq   r0, r0, lr, lsr r0
        ...

Disassembly of section .rodata:

00000000 <string_buffer1>:
   0:   7261656c        rsbvc   r6, r1, #108, 10        ; 0x1b000000
   4:   6e692d6e        cdpvs   13, 6, cr2, cr9, cr14, {3}
   8:   7065642d        rsbvc   r6, r5, sp, lsr #8
   c:   613c6874        teqvs   ip, r4, ror r8
  10:   64656d68        strbtvs r6, [r5], #-3432        ; 0xd68
  14:   0000003e        andeq   r0, r0, lr, lsr r0
        ...
```

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

.txt             0x00010010       0x70
 *(.text)
 .text           0x00010010       0x20 app.o
                 0x00010010                main
 .text           0x00010030       0x50 uart.o
                 0x00010030                uart_send_string

.rodata          0x00010080       0x64
 .rodata         0x00010080       0x64 app.o
                 0x00010080                string_buffer1

.glue_7          0x000100e4        0x0
 .glue_7         0x00000000        0x0 linker stubs

.glue_7t         0x000100e4        0x0
 .glue_7t        0x00000000        0x0 linker stubs

.vfp11_veneer    0x000100e4        0x0
 .vfp11_veneer   0x00000000        0x0 linker stubs

.v4_bx           0x000100e4        0x0
 .v4_bx          0x00000000        0x0 linker stubs

.iplt            0x000100e4        0x0
 .iplt           0x00000000        0x0 startup.o

.rel.dyn         0x000100e4        0x0
 .rel.iplt       0x00000000        0x0 startup.o

.data            0x000100e4       0x64
 *(.data)
 .data           0x000100e4        0x0 startup.o
```

Qemu output

```
soft@DESKTOP-VJG2KBC MINGW64 /d/mastering embedded diploama/c programming/unit_3/codes/lec2/lab1
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin
learn-in-depth<ahmed>
```