Lab1 on ARM VerstilePB of unit3 Lesson2

In this lab we will create a Bare metal Software to send a "Learn-in-depth: <eng: Osama>" using UART

First screen showing how to create file by using command "touch + file name" and display bin libraries of arm-toolchain and the command *"export PATH= ../ARM/bin/:$PATH"* to find (arm-none-eabi-) easily.

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Labs
$ touch app.c uart.c uart.h

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ ls
 Assignment/  'New folder'/   app.c   uart.c   uart.h

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ ls ../ARM/bin
arm-none-eabi-addr2line.exe*   arm-none-eabi-gcc-nm.exe*       arm-none-eabi-ld.exe*
arm-none-eabi-ar.exe*          arm-none-eabi-gcc-ranlib.exe*   arm-none-eabi-nm.exe*
arm-none-eabi-as.exe*          arm-none-eabi-gcc.exe*          arm-none-eabi-objcopy.exe*
arm-none-eabi-c++.exe*         arm-none-eabi-gcov-dump.exe*    arm-none-eabi-objdump.exe*
arm-none-eabi-c++filt.exe*     arm-none-eabi-gcov-tool.exe*    arm-none-eabi-ranlib.exe*
arm-none-eabi-cpp.exe*         arm-none-eabi-gcov.exe*         arm-none-eabi-readelf.exe*
arm-none-eabi-elfedit.exe*     arm-none-eabi-gdb-py.exe*       arm-none-eabi-size.exe*
arm-none-eabi-g++.exe*         arm-none-eabi-gdb.exe*          arm-none-eabi-strings.exe*
arm-none-eabi-gcc-7.2.1.exe*   arm-none-eabi-gprof.exe*        arm-none-eabi-strip.exe*
arm-none-eabi-gcc-ar.exe*      arm-none-eabi-ld.bfd.exe*       gccvar.bat

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ export PATH=../ARM/bin/:$ PATH

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ arm-none-eabi-
```

Uart.h

D:\Embedded Diploma\Units\Uint_3\Labs\uart.h - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

uart.h | uart.c | app.c

```c


#ifndef _UART_H_
#define _UART_H_

typedef unsigned char   uint8_t;
typedef unsigned int    uint32_t;

void Uart_Send_string(uint8_t *P_txt_str);

#endif   /* _UART_H_ */
```

Line 7, Column 34 Tab Size: 4 C++

Lab1 on ARM VerstilePB of unit3 Lesson2

Uart.c

D:\Embedded Diploma\Units\Uint_3\Labs\uart.c - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

uart.h | uart.c | app.c

```c

#include "uart.h"

#define UART0DR *((volatile uint8_t*)((uint8_t*)0x100f1000))

void Uart_Send_string(uint8_t *P_txt_str)
{
    while(*P_txt_str != '\0')
    {
        UART0DR = (uint32_t) (*P_txt_str);
        P_txt_str++;  //next character
    }
}
```

Line 1, Column 1 Tab Size: 4 C

App.c

D:\Embedded Diploma\Units\Uint_3\Labs\app.c • - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

uart.h | uart.c | app.c

```c


#include "uart.h"

uint8_t str_buffer[100] = "Learn-in-Depth<Eng:Osama>";
void main()
{
    Uart_Send_string(str_buffer);   /*VerstilePB physical Board*/
}

```

Line 5, Column 1 Tab Size: 4 C

Lab1 on ARM VerstilePB of unit3 Lesson2

Using GNU ARM-Cross-toolchine"arm-none-eabi-gcc.exe" to create (uart.o and app.o) by using –mcpu=arm926ej-s

'-g' → (to find all debugs sections) now we have two Relocatable files (app.o & uart.o).

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Labs

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ export PATH=../ARM/bin/:$PATH

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ arm-none-eabi-gcc.exe -c -g -I -mcpu=arm926ej-s uart.c -o uart.o

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ arm-none-eabi-gcc.exe -c -g -I -mcpu=arm926ej-s app.c -o app.o

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ |
```

To display app.o file we will use this command "*arm-none-eabi-objdump.exe -h app.o*"
'-h' → (header Sections).

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Labs
$ 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         00000024  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000064  00000000  00000000  00000058  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000bc  2**0
                  ALLOC
  3 .debug_info   00000071  00000000  00000000  000000bc  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 00000065  00000000  00000000  0000012d  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_aranges 00000020  00000000  00000000  00000192  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  6 .debug_line   0000003f  00000000  00000000  000001b2  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_str    000000ab  00000000  00000000  000001f1  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .comment      0000007f  00000000  00000000  0000029c  2**0
                  CONTENTS, READONLY
  9 .debug_frame  00000034  00000000  00000000  0000031c  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 10 .ARM.attributes 00000030  00000000  00000000  00000350  2**0
                  CONTENTS, READONLY

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Sections without debug using of app.o file“`arm-none-eabi-gcc.exe -c -I -mcpu=arm926ej-s app.c -o app.o`”

```
Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ arm-none-eabi-gcc.exe -c -I -mcpu=arm926ej-s app.c -o app.o

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ 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         00000024  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000064  00000000  00000000  00000058  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000bc  2**0
                  ALLOC
  3 .comment      0000007f  00000000  00000000  000000bc  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000030  00000000  00000000  0000013b  2**0
                  CONTENTS, READONLY

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ |
```

Display (.rodata) by adding Const

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Labs
Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ arm-none-eabi-gcc.exe -c -mcpu=arm926ej-s app.c -o app.o

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ 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         0000001c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000064  00000000  00000000  00000050  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b4  2**0
                  ALLOC
  3 .rodata       00000064  00000000  00000000  000000b4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      0000007f  00000000  00000000  00000118  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 00000032  00000000  00000000  00000197  2**0
                  CONTENTS, READONLY

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ |
```

```
D:\Embedded Diploma\Units\Uint_3\Labs\app.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
uart.h   uart.c   app.c

#include "uart.h"

uint8_t str_buffer[100] = "Learn-in-Depth<Eng:Osama>";

uint8_t const str_buffer2[100] = "Learn-in-Depth<Eng:Osama>";

void main()
{
    Uart_Send_string(str_buffer);   /*VerstilePB physical Board*/
}

Line 7, Column 26        Tab Size: 4        C
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Display sections of **uart.o** file

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Labs
Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ 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         00000054  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000088  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000088  2**0
                  ALLOC
  3 .debug_info   00000074  00000000  00000000  00000088  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 0000005e  00000000  00000000  000000fc  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_aranges 00000020  00000000  00000000  0000015a  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  6 .debug_line   00000043  00000000  00000000  0000017a  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_str    000000c0  00000000  00000000  000001bd  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .comment      0000007f  00000000  00000000  0000027d  2**0
                  CONTENTS, READONLY
  9 .debug_frame  00000030  00000000  00000000  000002fc  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 10 .ARM.attributes 00000030  00000000  00000000  0000032c  2**0
                  CONTENTS, READONLY

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Generate the disassembly file from the bin“`arm-none-eabi-objdump.exe -D app.o`”

D:\Embedded Diploma\Units\Uint_3\Labs\app.s - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

uart.h | uart.c | app.c | app.s

```
app.o:     file format elf32-littlearm


Disassembly of section .text:

00000000 <main>:
   0:-e92d4800   push   {fp, lr}
   4:-e28db004   add    fp, sp, #4
   8:-e59f0008   ldr    r0, [pc, #8]    ; 18 <main+0x18>
   c:-ebfffffe   bl     0 <Uart_Send_string>
  10:-e1a00000   nop                    ; (mov r0, r0)
  14:-e8bd8800   pop    {fp, pc}
  18:-00000000   andeq  r0, r0, r0

Disassembly of section .data:
```

```
Disassembly of section .text:

00000000 <main>:
   0:   e92d4800        push    {fp, lr}
   4:   e28db004        add     fp, sp, #4
   8:   e59f0008        ldr     r0, [pc, #8]    ; 18 <main+0x18>
   c:   ebfffffe        bl      0 <Uart_Send_string>
  10:   e1a00000        nop                     ; (mov r0, r0)
  14:   e8bd8800        pop     {fp, pc}
  18:   00000000        andeq   r0, r0, r0

Disassembly of section .data:

00000000 <str_buffer>:
   0:   7261654c        rsbvc   r6, r1, #76, 10 ; 0x13000000
   4:   6e692d6e        cdpvs   13, 6, cr2, cr9, cr14, {3}
   8:   7065442d        rsbvc   r4, r5, sp, lsr #8
   c:   453c6874        ldrmi   r6, [ip, #-2164]!       ; 0xfffff78c
  10:   4f3a676e        svcmi   0x003a676e
  14:   616d6173        smcvs   54803   ; 0xd613
  18:   0000003e        andeq   r0, r0, lr, lsr r0
        ...

Disassembly of section .rodata:

00000000 <str_buffer2>:
   0:   7261654c        rsbvc   r6, r1, #76, 10 ; 0x13000000
   4:   6e692d6e        cdpvs   13, 6, cr2, cr9, cr14, {3}
   8:   7065442d        rsbvc   r4, r5, sp, lsr #8
   c:   453c6874        ldrmi   r6, [ip, #-2164]!       ; 0xfffff78c
  10:   4f3a676e        svcmi   0x003a676e
  14:   616d6173        smcvs   54803   ; 0xd613
  18:   0000003e        andeq   r0, r0, lr, lsr r0
        ...
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Startup file using address 0x00011000 this address point to at the top of the stack.

Startup.s

D:\Embedded Diploma\Units\Uint_3\Labs\startup.ld • - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

uart.h | uart.c | app.c | app.s | startup.ld

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

Line 2, Column 1 Tab Size: 4 Plain Text

Using these commands to display startup.o "`$ arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o statup.o && $ arm-none-eabi-objdump.exe -h startup.o`"

```
Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o startup.o
startup.s: Assembler messages:
startup.s: Warning: end of file not at end of a line; newline inserted

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ 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
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Using Linker

In this picture we put all debus sections in debug section

D:\Embedded Diploma\Units\Uint_3\Labs\linker_script.ld - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

uart.h | uart.c | app.c | app.s | startup.s.ld | star

```
ENTRY(reset)

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

SECTIONS
{
    .startup :
    {
      *(.text)
    }
    .text :
    {
      *(.text)
    }
    .rodata :
    {
      *(.rodata)
    }
    .bss :
    {
      *(.bss)
    }
    .debug :
    {
      *(.debug_info .debug_abbrev .debug_line .debug_str .debug_aranges .debug_frame .comment)
    }
}
```

And using this command "`$ arm-none-eabi-objdump.exe -h learn-in-depth.elf`" to show output

MINGW64:/d/Embedded Diploma/Units/Uint_3/Labs

```
Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ arm-none-eabi-ld.exe -T linker_script.ld app.o uart.o startup.o -o learn-in-depth.elf

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ 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      0000007c  00000000  00000000  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .rodata       00000064  0000007c  0000007c  0001007c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .data         00000064  000000e0  000000e0  000100e0  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 00000030  00000000  00000000  00010144  2**0
                  CONTENTS, READONLY
  4 .debug        000002a4  00000000  00000000  00010174  2**2
                  CONTENTS, READONLY, DEBUGGING

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ |
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Linker_script.ld file → adding memory to sections

D:\Embedded Diploma\Units\Uint_3\Labs\linker_script.ld - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

linker_script.ld | uart.h | uart.c | app.c

```
SECTIONS
{
    . = 0x10000;
    .startup . :
    {
       startup.o(.text)
    }> Mem
    .text :
    {
      *(.text) *(.rodata)
    }> Mem
    .data :
    {
      *(.data)
    }> Mem
    .bss :
    {
      *(.bss) *(COMMON)
    }> Mem
    . = . + 0x1000 ;
    stack_top = . ;
}
```

Output of learn-in-depth.elf file "`$ arm-none-eabi-objdump.exe -h learn-in-depth.elf`"

MINGW64:/d/Embedded Diploma/Units/Uint_3/Labs

```
Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ arm-none-eabi-ld.exe  -T linker_script.ld app.o uart.o startup.o -o learn-in-d

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ 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  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         000000d4  00010010  00010010  00010010  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  000100e4  000100e4  000100e4  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 0000002e  00000000  00000000  00010148  2**0
                  CONTENTS, READONLY
  4 .comment      0000007e  00000000  00000000  00010176  2**0
                  CONTENTS, READONLY

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ |
```

Lab1 on ARM VerstilePB of unit3 Lesson2

To read the symbols we should use nm cross tool chain Using this command "`$ arm-none-eabi-nm.exe app.o`" to display app.o symbols

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Labs

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ arm-none-eabi-nm.exe app.o
00000000 T main
00000000 D str_buffer
00000000 R str_buffer2
         U Uart_Send_string

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ |
```

Using this command "`$ arm-none-eabi-nm.exe startup.o`" to display startup.o symbols

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Labs

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ arm-none-eabi-nm.exe startup.o
         U main
00000000 T reset
         U stack_top
00000008 t stop

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ |
```

Using this command "`$ arm-none-eabi-nm.exe learn-in-depth.elf`" to display Learn-in-depth.elf symbols

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Labs
Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ arm-none-eabi-nm.exe learn-in-depth.elf
00010010 T main
00010000 T reset
00011148 D stack_top
00010008 t stop
000100e4 D str_buffer
00010080 T str_buffer2
0001002c T Uart_Send_string

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Labs
$ |
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Using this command “ `$ arm-none-eabi-ld.exe  -T linker_script.ld -Map=output.map app.o uart.o startup.o -o learn-in-depth.elf` ” to display .MAP file

D:\Embedded Diploma\Units\Uint_3\Labs\output.map - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

output.map | linker_script.ld | uart.h

```
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       0xd4
 *(.text)
 .text           0x00010010       0x1c app.o
                 0x00010010                 main
 .text           0x0001002c       0x54 uart.o
                 0x0001002c                 Uart_Send_string
 *(.rodata)
 .rodata         0x00010080       0x64 app.o
                 0x00010080                 str_buffer2

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

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

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

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

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

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

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

Line 15, Column 44

Lab1 on ARM VerstilePB of unit3 Lesson2

To generate hex or bin file using this command "`$ arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin`"

D:\Embedded Diploma\Units\Uint_3\Labs\learn-in-depth.bin - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

output.map | learn-in-depth.bin | linker_script.ld | uart.h

```
04d0 9fe5 0100 00eb feff ffea 4811 0100
0048 2de9 04b0 8de2 0800 9fe5 0200 00eb
0000 a0e1 0088 bde8 e400 0100 04b0 2de5
00b0 8de2 0cd0 4de2 0800 0be5 0600 00ea
3420 9fe5 0830 1be5 0030 d3e5 0030 c2e5
0830 1be5 0130 83e2 0830 0be5 0830 1be5
0030 d3e5 0000 53e3 f4ff ff1a 0000 a0e1
00d0 8be2 04b0 9de4 1eff 2fe1 0010 0f10
4c65 6172 6e2d 696e 2d44 6570 7468 3c45
6e67 3a4f 7361 6d61 3e00 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 4c65 6172 6e2d 696e 2d44 6570
7468 3c45 6e67 3a4f 7361 6d61 3e00 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Finally to run the program in the QEMU Simulator ("VersatilePB physical Board") using this command "`qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin`"

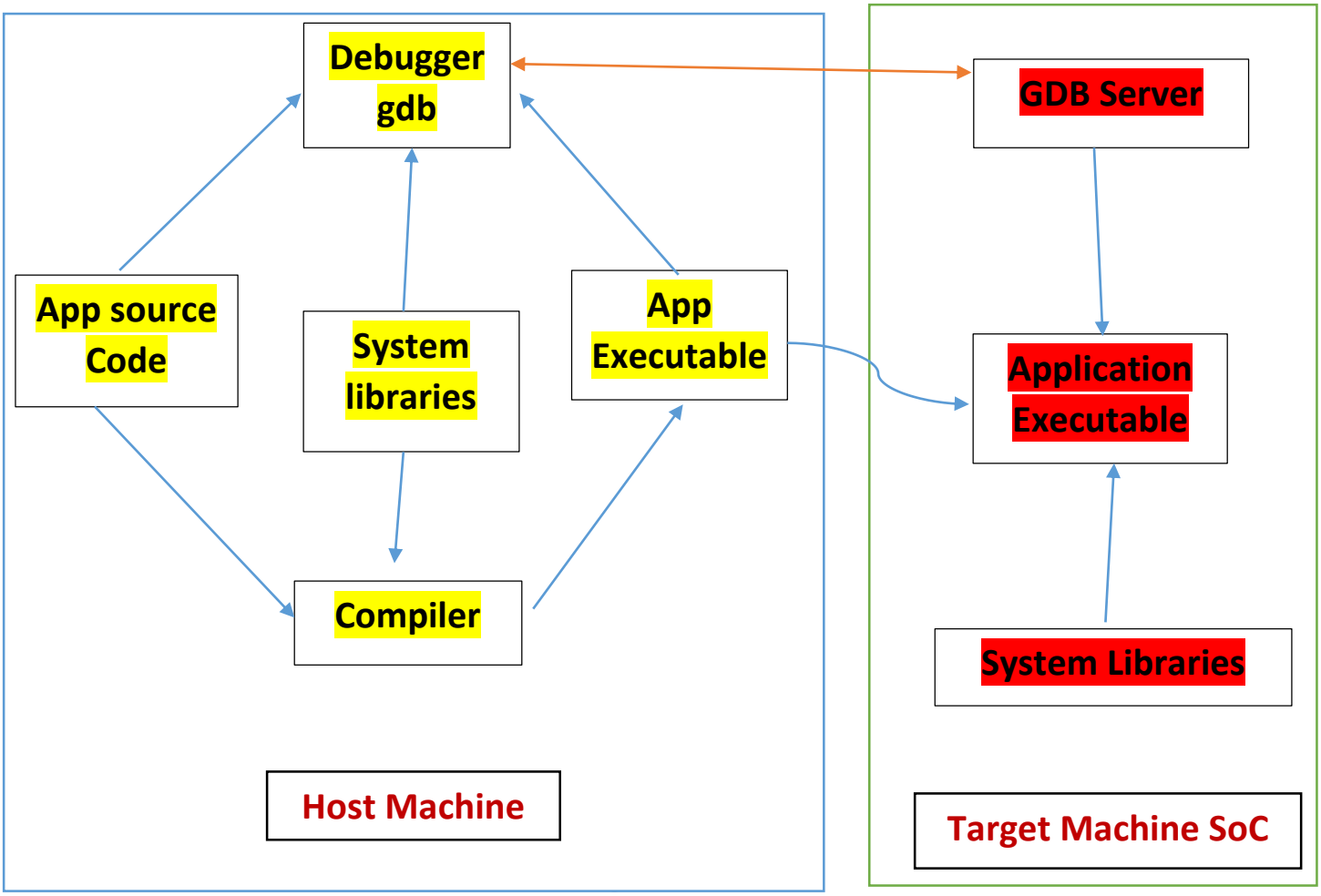
```
MINGW64:/d/Osama/Embedded Diploma/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
    29: 0001005c    24 FUNC    GLOBAL DEFAULT    2 main
    30: 00010074   100 OBJECT  GLOBAL DEFAULT    3 string_buffer

No version information found in this file.
Attribute Section: aeabi
File Attributes
  Tag_CPU_name: "ARM926EJ-S"
  Tag_CPU_arch: v5TEJ
  Tag_ARM_ISA_use: Yes
  Tag_THUMB_ISA_use: Thumb-1
  Tag_ABI_PCS_wchar_t: 4
  Tag_ABI_FP_denormal: Needed
  Tag_ABI_FP_exceptions: Needed
  Tag_ABI_FP_number_model: IEEE 754
  Tag_ABI_align8_needed: Yes
  Tag_ABI_enum_size: small

IT@DESKTOP-2C4F9RG MINGW64 /d/Osama/Embedded Diploma/Embedded Diploma/Units/Uint_3/Lesson_2/Labs (master)
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin
learn-in-depth:<osama

IT@DESKTOP-2C4F9RG MINGW64 /d/Osama/Embedded Diploma/Embedded Diploma/Units/Uint_3/Lesson_2/Labs (master)
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin
learn-in-depth:<osama
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Now we should add –g option to enable built-in debugging support (which gdb needs):

By using these commands we execute .bin and .elf files

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ arm-none-eabi-gcc.exe -c -I . -g -mcpu=arm926ej-s app.c -o app.o

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ arm-none-eabi-gcc.exe -c -I . -g -mcpu=arm926ej-s uart.c -o uart.o

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ arm-none-eabi-as.exe  -g -mcpu=arm926ej-s startup.s -o startup.o

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ arm-none-eabi-ld.exe -T linker_script.ld startup.o app.o uart.o -o learn-in-depth.elf

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$
```



Lab1 on ARM VerstilePB of unit3 Lesson2

We now should using TCP Connection, to run the emulator by using –s –S options and using this command "`$../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -s -S -kernel learn-in-depth.elf`".

```
    28: 00010080   100 OBJECT  GLOBAL DEFAULT    3 string_buffer

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.elf
learn-in-depth:<osama

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -s -S -kernel learn-in-depth.elf
|
```

After that we will using another terminal to using GNU Debugger, and using "`(gdb) target remote localhost:1234`" to remote with the target machine.

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ arm-none-eabi-gdb.exe learn-in-depth.elf
GNU gdb (GNU Tools for Arm Embedded Processors 7-2017-q4-major) 8.0.50.20171128-git
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=arm-none-eabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from learn-in-depth.elf...done.
(gdb) target remote localhost:1324
localhost:1324: No connection could be made because the target machine actively refused it.
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
reset () at startup.s:3
3               ldr sp, = stack_top
(gdb) |
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Using "`(gdb) l`" to find the file where we stand on it

```
(gdb) l
1       .global reset
2       reset:
3               ldr sp, = stack_top
4               bl main
5       stop: b stop
(gdb) |
```

"`(gdb) display/3i $pc`" using this command to know where the PC refer to what?

```
(gdb) display /3i $pc
1: x/3i $pc
=> 0x10000 <reset>:     ldr     sp, [pc, #4]    ; 0x1000c <stop+4>
   0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      b       0x10008 <stop>
(gdb) |
```

To put breakpoints on main function using this command "`(gdb) b main`" or by suing it`s address "`(gdb) b * 0x10010`"

```
(gdb) b main
Breakpoint 1 at 0x10018: file app.c, line 8.
(gdb) b * 0x10010
Breakpoint 2 at 0x10010: file app.c, line 7.
(gdb) |
```

To make the processor walk step by step using this command "`(gdb) si`"

```
(gdb) si
reset () at startup.s:4
4               bl main
1: x/3i $pc
=> 0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      b       0x10008 <stop>
   0x1000c <stop+4>:    andeq   r1, r1, r8, asr #2
(gdb) |
```

Using c to continue "`(gdb) c`" it will continue and stopped at the last breakpoint we put and it will be the main function.

```
   0x1000c <stop+4>:    andeq   r1, r1, r8, asr #2
(gdb) c
Continuing.

Breakpoint 2, main () at app.c:7
7       {
1: x/3i $pc
=> 0x10010 <main>:      push    {r11, lr}
   0x10014 <main+4>:    add     r11, sp, #4
   0x10018 <main+8>:    ldr     r0, [pc, #8]    ; 0x10028 <main+24>
(gdb) |
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Continue again to show the first c code.

```
(gdb) c
Continuing.

Breakpoint 1, main () at app.c:8
8               Uart_Send_string(string_buffer);
1: x/3i $pc
=> 0x10018 <main+8>:    ldr     r0, [pc, #8]    ; 0x10028 <main+24>
   0x1001c <main+12>:   bl      0x1002c <Uart_Send_string>
   0x10020 <main+16>:   nop                     ; (mov r0, r0)
(gdb) l
3       unsigned char string_buffer[100] = "learn-in-depth:<osama" ;
4       unsigned char string_buffer1[100] = "learn-in-depth:<osama" ;
5
6       void main()
7       {
8               Uart_Send_string(string_buffer);
9       }(gdb) |
```

To print the second element from your string "`(gdb) print string_buffer[1]`" and to print all the statement "`(gdb) print string_buffer`"

```
Line number 10 out of range; app.c has 9 lines.
(gdb) print string_buffer[1]
$1 = 101 'e'
(gdb) print string_buffer
$2 = "learn-in-depth:<osama", '\000' <repeats 78 times>
(gdb) |
```

To know all the breakpoints "`(gdb) info breakpoints`"

```
(gdb) info breakpoints
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x00010018 in main at app.c:8
        breakpoint already hit 1 time
2       breakpoint     keep y   0x00010010 in main at app.c:7
        breakpoint already hit 1 time
(gdb) |
```

Make a breakpoint at Using_String_buffer and using where command to know where we are

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
Continuing.

Breakpoint 3, Uart_Send_string (p_tx_string=0x10080 <string_buffer> "learn-in-depth:<osama")
    at uart.c:7
7               while(*p_tx_string != '\0')
1: x/3i $pc
=> 0x1003c <Uart_Send_string+16>:       b       0x1005c <Uart_Send_string+48>
   0x10040 <Uart_Send_string+20>:       ldr     r3, [r11, #-8]
   0x10044 <Uart_Send_string+24>:       ldrb    r2, [r3]
(gdb) l
2
3       #define UARTODR *((volatile unsigned int* const)((unsigned int *)0x101f1000))
4
5       void Uart_Send_string (unsigned char *p_tx_string)
6       {
7               while(*p_tx_string != '\0')
8               {
9                       UARTODR = (unsigned int) (*p_tx_string);
10                      p_tx_string++;
11              }
(gdb) where
#0  Uart_Send_string (p_tx_string=0x10080 <string_buffer> "learn-in-depth:<osama")
    at uart.c:7
#1  0x00010020 in main () at app.c:8
(gdb) |
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Put "`(gdb) s`" more time to write the string on the target machine

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ export PATH=../ARM/bin/:$PATH

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.elf
learn-in-depth:<osama

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -s -S -kernel learn-in-depth.elf
learn-
```

```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
=> 0x1005c <Uart_Send_string+48>:      ldr    r3, [r11, #-8]
   0x10060 <Uart_Send_string+52>:      ldrb   r3, [r3]
   0x10064 <Uart_Send_string+56>:      cmp    r3, #0
(gdb) s
9                   UARTODR = (unsigned int) (*p_tx_string);
1: x/3i $pc
=> 0x10040 <Uart_Send_string+20>:      ldr    r3, [r11, #-8]
   0x10044 <Uart_Send_string+24>:      ldrb   r2, [r3]
   0x10048 <Uart_Send_string+28>:
    ldr r3, [pc, #44]   ; 0x1007c <Uart_Send_string+80>
(gdb) s
10                  p_tx_string++;
1: x/3i $pc
=> 0x10050 <Uart_Send_string+36>:      ldr    r3, [r11, #-8]
   0x10054 <Uart_Send_string+40>:      add    r3, r3, #1
   0x10058 <Uart_Send_string+44>:      str    r3, [r11, #-8]
(gdb)
```

Now we can using this command "`(gdb) b app.c:8`" to just write c to send letter on the target machine screen.

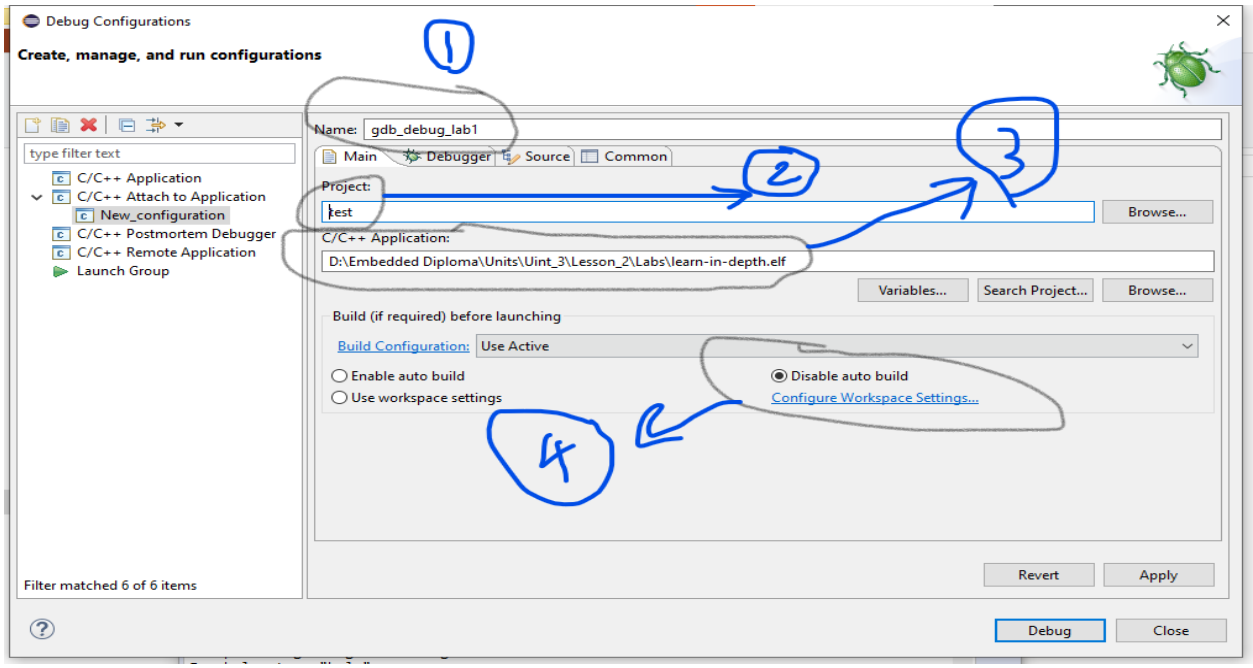
```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ export PATH=../ARM/bin/:$PATH

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.elf
learn-in-depth:<osama

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -s -S -kernel learn-in-depth.elf
learn-in-depth:<osama
```

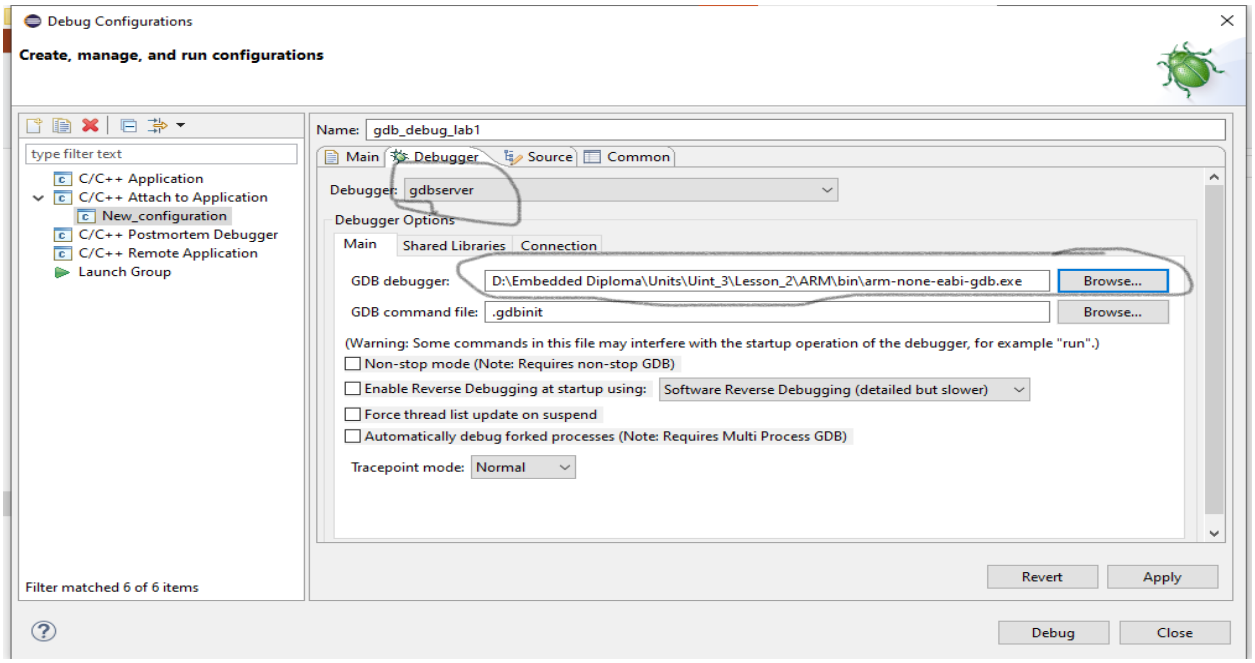
```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
1: x/3i $pc
=> 0x10040 <Uart_Send_string+20>:      ldr    r3, [r11, #-8]
   0x10044 <Uart_Send_string+24>:      ldrb   r2, [r3]
   0x10048 <Uart_Send_string+28>:
    ldr r3, [pc, #44]   ; 0x1007c <Uart_Send_string+80>
(gdb) s
10                  p_tx_string++;
1: x/3i $pc
=> 0x10050 <Uart_Send_string+36>:      ldr    r3, [r11, #-8]
   0x10054 <Uart_Send_string+40>:      add    r3, r3, #1
   0x10058 <Uart_Send_string+44>:      str    r3, [r11, #-8]
(gdb) b app.c:8
Note: breakpoint 1 also set at pc 0x10018.
Breakpoint 4 at 0x10018: file app.c, line 8.
(gdb) c
Continuing.
```

Lab1 on ARM VerstilePB of unit3 Lesson2

Gdb GUI : eclipse

Firstly open eclipse an on the debug icon click on it and then chose "debug configuration" then chose "C/C++ attach to application" after that follow these two screen



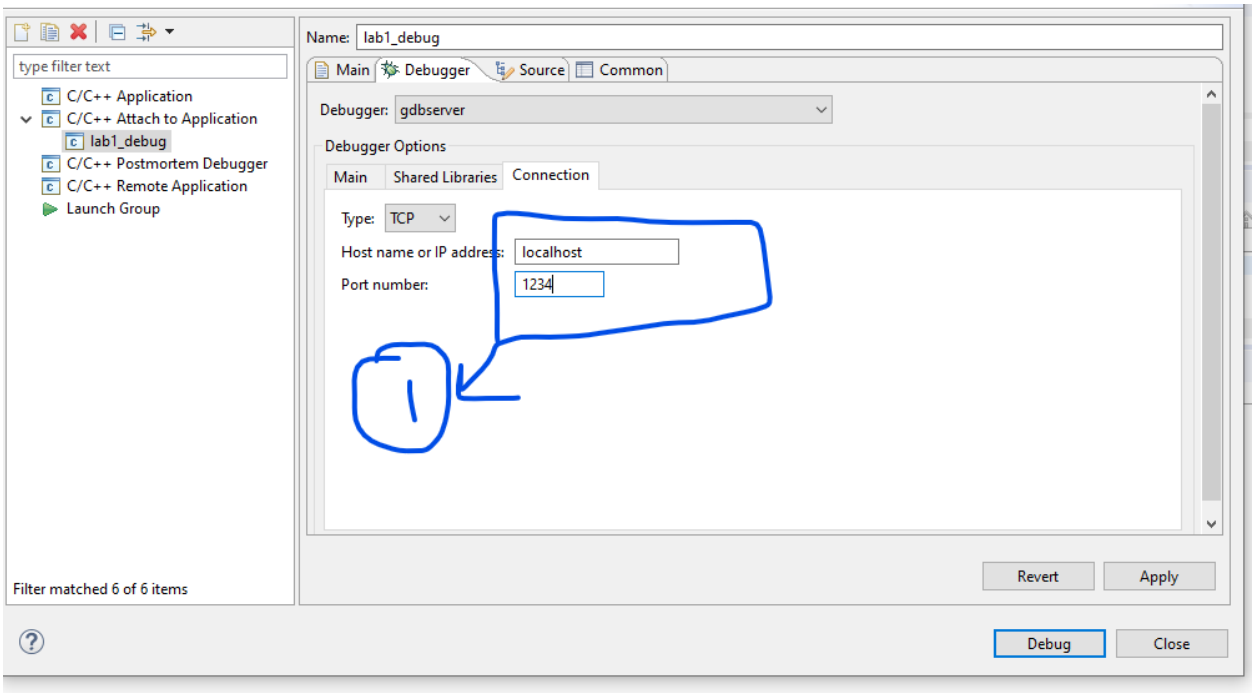


Lab1 on ARM VerstilePB of unit3 Lesson2

Make this terminal ready

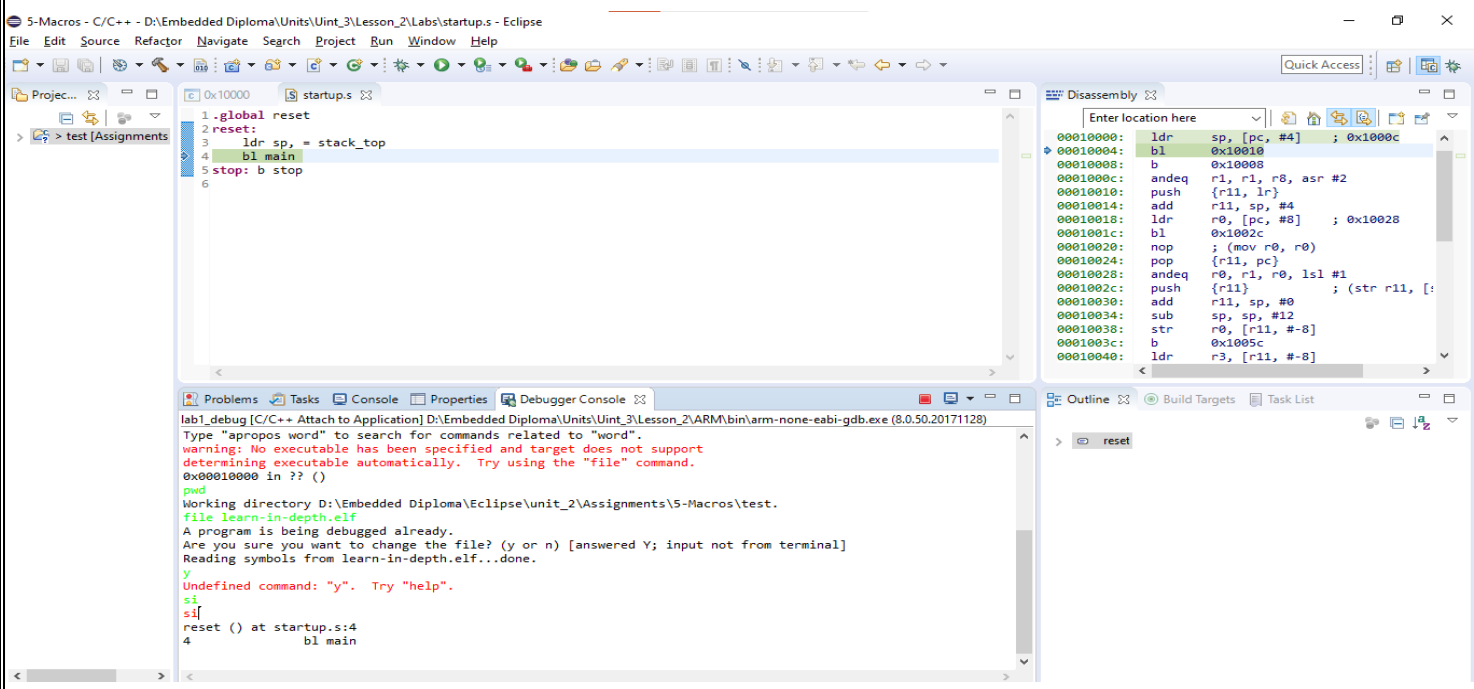
```
MINGW64:/d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ export PATH=../ARM/bin/:$PATH

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.elf
learn-in-depth:<osama

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -s -S -kernel learn-in-depth.elf
learn-in-depth:<osama

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -s -S -kernel learn-in-depth.elf
```

Make 1234 to the localhost as the following screen, then click on apply button and finally click Debug.

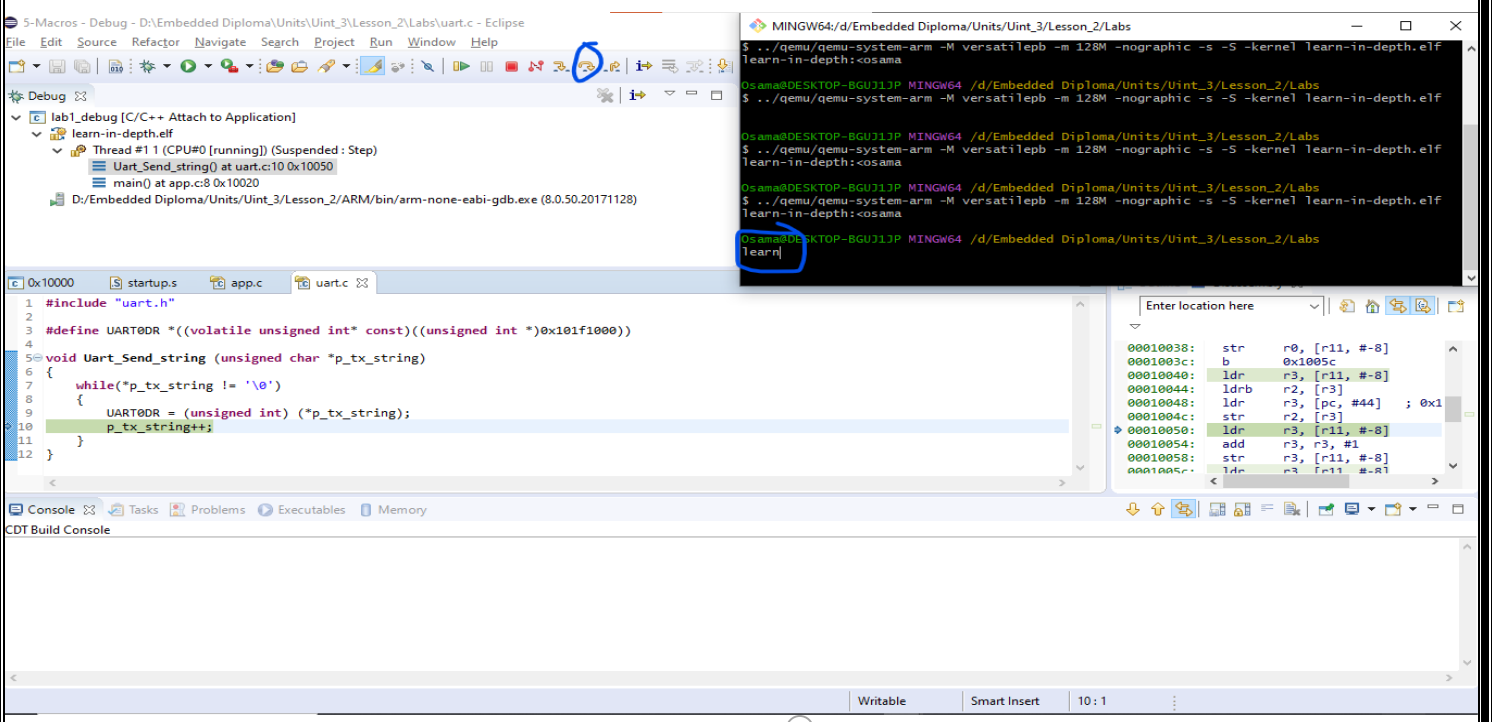


Lab1 on ARM VerstilePB of unit3 Lesson2

Now we should copy the .elf file and put it in app path to debug it.



By clicking to the next step it will be send to the terminal letter by letter till it reach to the last letter in the string



Lab1 on ARM VerstilePB of unit3 Lesson2

Makefile Tutorial

D:\Embedded Diploma\Units\Uint_3\Lesson_2\Labs\makefile - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

linker_script.ld | startup.s | makefile

```makefile
#@copyright : eng<Osama_Khallaf>
CC= arm-none-eabi-
CFLAGS= -g -mcpu=arm926ej-s
INCS = -I .
LIBS=
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
As = $(wildcard *.s)
AsOBJ=$(As:.s=.o)

all: learn-in-depth.bin
	@echo "======Build is Done======"

startup.o: startup.s
	$(CC)as.exe $(CFLAGS) $< -o $@

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

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

learn-in-depth.bin: learn-in-depth.elf
	$(CC)objcopy.exe -O binary $< $@

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

Line 22, Column 1 | Tab Size: 4 | Makefile

Now we can use this command to clear all files "`$ make clean_all`" and this command to restore our files again "`$ make all`"

```
Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ make clean
rm *.elf *.bin

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ make clean_all
rm *.o *.elf *.bin
rm: cannot remove '*.elf': No such file or directory
rm: cannot remove '*.bin': No such file or directory
make: *** [clean_all] Error 1

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ make all
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
arm-none-eabi-as.exe -g -mcpu=arm926ej-s startup.s -o startup.o
arm-none-eabi-ld.exe -T linker_script.ld  app.o uart.o startup.o -o learn-in-depth.elf
arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
======Build is Done======

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Uint_3/Lesson_2/Labs
$ |
```

Lab1 on ARM VerstilePB of unit3 Lesson2