BIRZEIT UNIVERSITY

Electrical and Computer Engineering

Computer Design Lab – ENCS4110

Introduction to ARM Assembly Language and Keil uVision5

Objectives

```
1.    Introduce some of the ARM architecture to students.

2.    Begin to use the lab tool - Keil uVision 5.

3.    The students will create a project and write
      an ARM assembly language program based on a simulated target.
```

The ARM (Advanced RISC Machine) architecture is introduced in the class (also see **http://www.arm.com**.) Keil MDK-ARM is a complete software development toolkit for ARM processor-based microcontrollers. Keil uVision5 will be used in the lab. The ARM Cortex-M3 processor will be examined with the STM32VLDISCOVERY board. The following is some important information for you.

Important Information

1. In the lab room Masri207, computers are running the operating system Windows 10 Pro, and ARM Software Microcontroller Development Kit Version 5.21a (Keil uVision5) is installed.

2. To install it in your home computer, you can download the following files:
   ~ftp/pub/class/301/ftp/uVision5/MDK521a.EXE
   ~ftp/pub/class/301/ftp/uVision5/Keil.STM32F1xx_DFP.2.1.0.pack

   Here is the Link to Keil Tools.

3. To know more about Keil, visit http://www.keil.com/

4. To see STM32VLDISCOVERY board, visit STM32VLDISCOVERY Board.

5. To see The Cortex-M3 Instruction Set, visit Cortex-M3 Devices Generic User Guide.

```
6.      To see more references of Cortex-M3, visit the following:
```
Cortex-M3 Technical Reference Manual - ARM Information Center
Cortex-M3 programming manual

Create an ARM Assembly Language Program

To create an assembly language program, you need to use a text editor such as **NotePad** in Microsoft Windows environment. There is a text edit in the Keil uVision5 for you to use too. The file name must have a **.s** at the end. Let's look at the following program called **FirstArm.s** on a PC. The file **FirstArm.s** contains the source code of the program to load registers and demonstrate a few other operations. We will use Keil uVision5 to create a project and execute this program so that you can get a feel of how Keil uVision5 works.

```
;The semicolon is used to lead an inline documentation.
;This is the first ARM Assembly language program you see in the lab.
;This program skeleton was from Dave Duguid and Trevor Douglas in summer 2013.
;When you write your program, you could have your info at the top document block.
;For Example:  Your Name, Student Number, what the program is for, and what it does
etc.

;;; Directives
          PRESERVE8
          THUMB

; Vector Table Mapped to Address 0 at Reset
; Linker requires __Vectors to be exported

          AREA    RESET, DATA, READONLY
          EXPORT  __Vectors

__Vectors
        DCD  0x20001000     ; stack pointer value when stack is empty
                     ;The processor uses a full descending stack.
                     ;This means the stack pointer holds the address of the last
                     ;stacked item in memory. When the processor pushes a new item
                     ;onto the stack, it decrements the stack pointer and then
                     ;writes the item to the new memory location.

          DCD  Reset_Handler  ; reset vector

          ALIGN

; The program
; Linker requires Reset_Handler
```

```
          AREA    MYCODE, CODE, READONLY

        ENTRY
        EXPORT Reset_Handler


Reset_Handler
;;;;;;;;;;User Code Starts from the next line;;;;;;;;;;;;

        MOV R0, #12

STOP
        ADD R0, R0, #4
        B  STOP

          END;End of the program
```

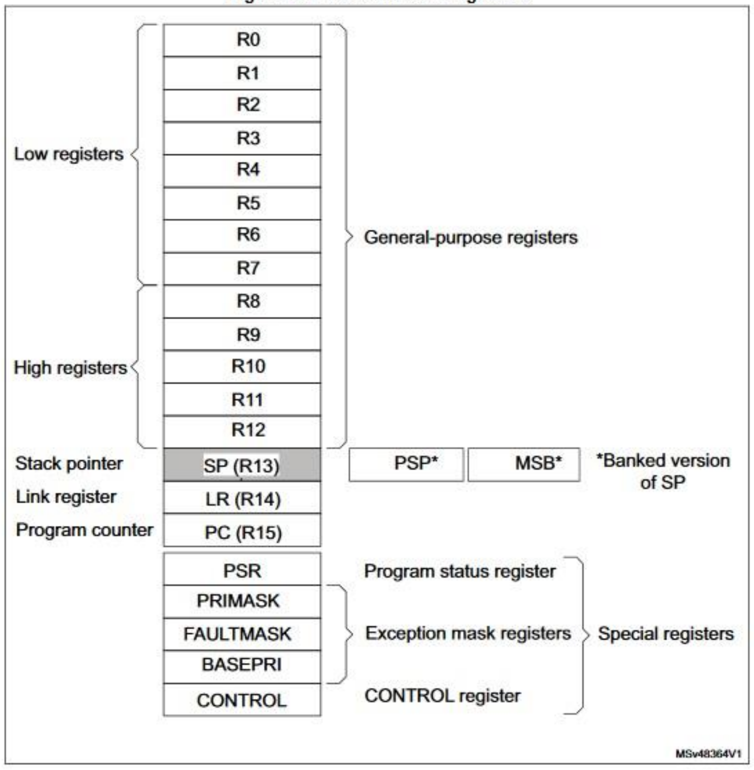
References:

1. A complete list of DIRECTIVES from ARM Information Center
2. Cortex-M3 Devices Generic User Guide
3. Cortex-M3 Programming Manual

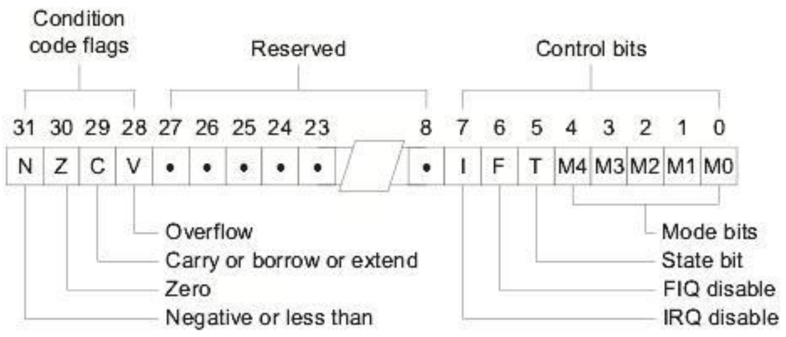
ARM Cortex-M3 Core Registers

2.1.3 Core registers

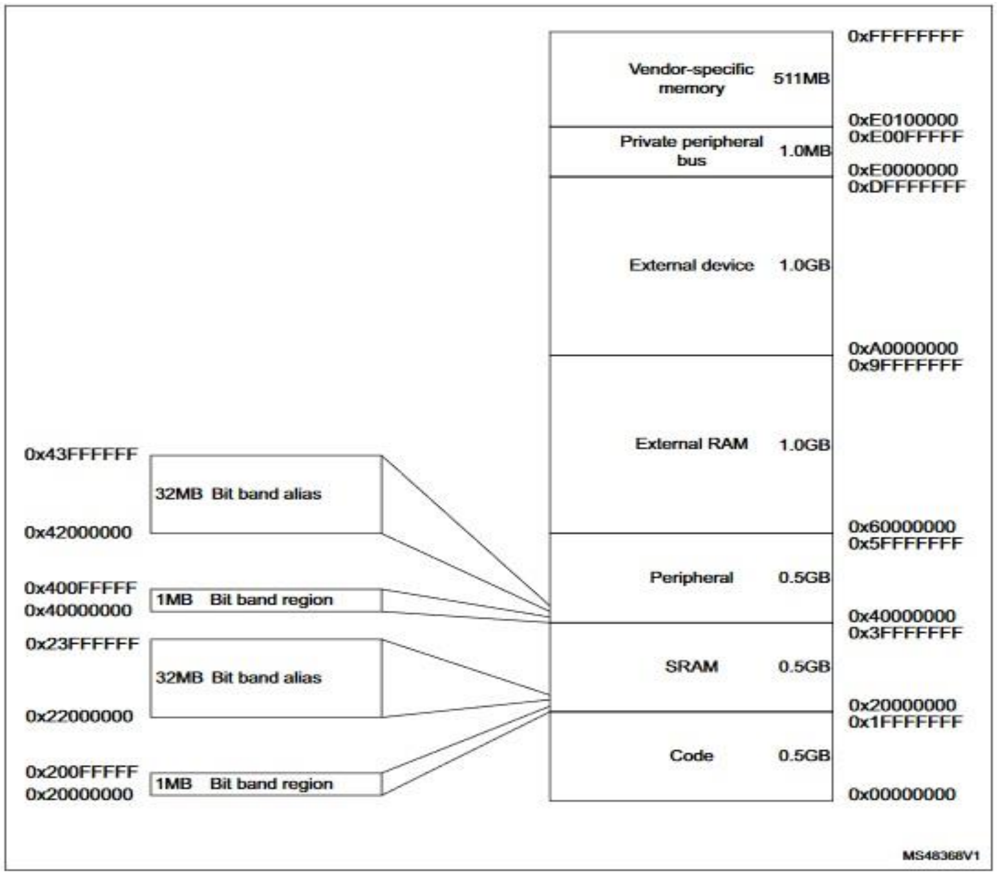
Figure 2. Processor core registers



Here is the Program Status Register Format:



ARM Cortex-M3 Memory Map



STM32F100xB Memory Map

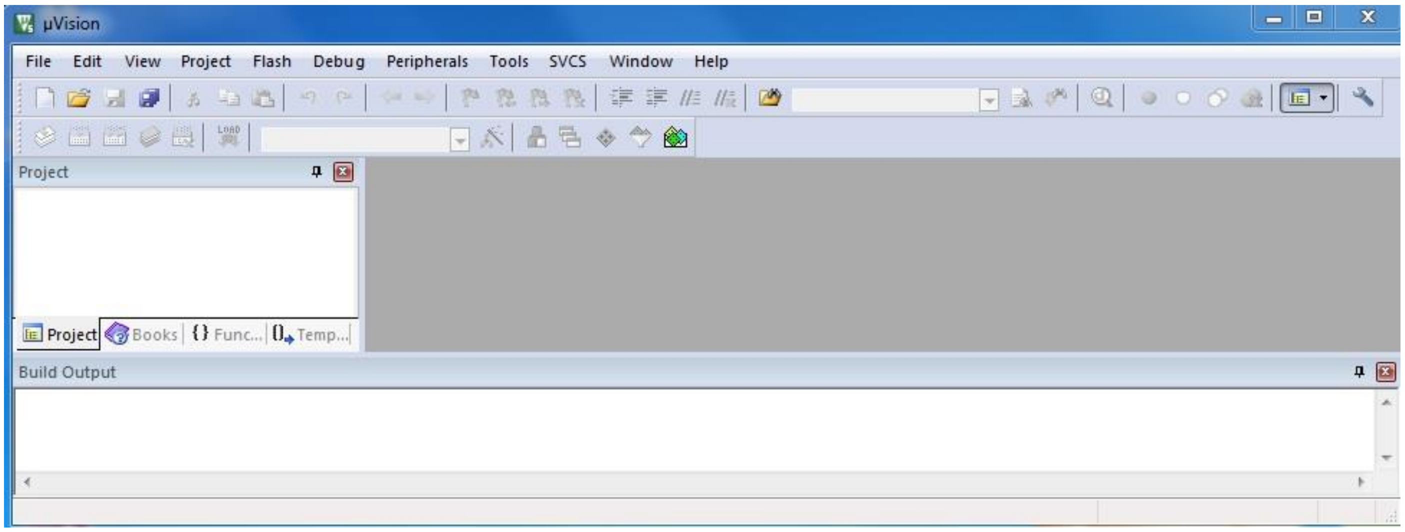
STM32F100xB Memory Map

STM32F100RB Datasheet

Start up Keil uVision5

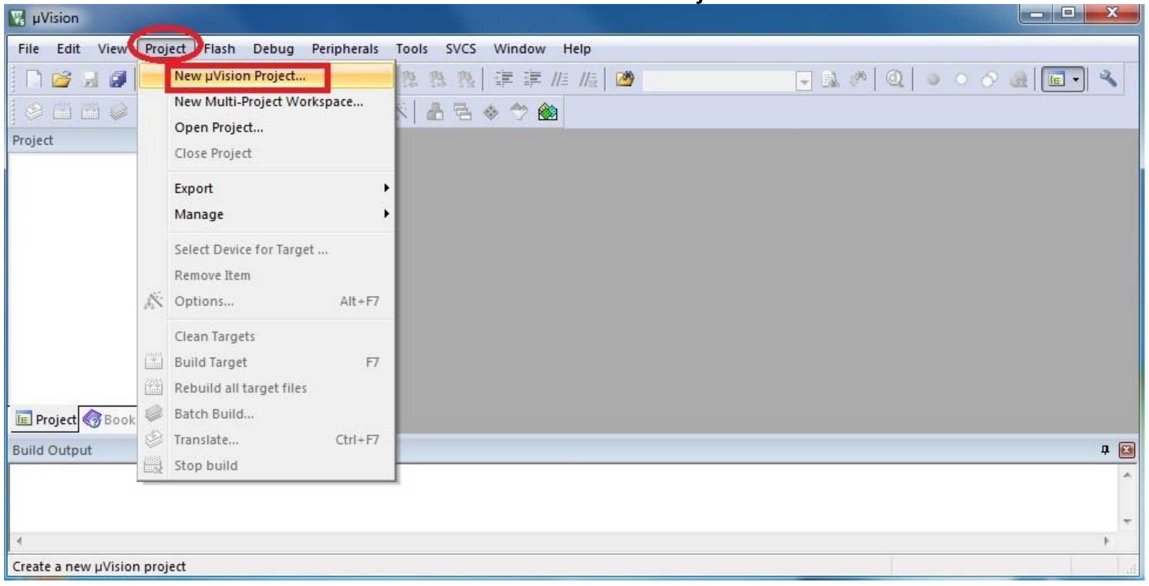
Before you start up, you are recommended that you create a folder to hold all your project files.
For example: you can have a folder "FirstARM-Project" ready before hand.

You can start up **uVision5** by clicking on the icon Keil uVision5 from the desktop or from the "Start" menu or "All Programs" on a lab PC.
The following screen is what you will see.

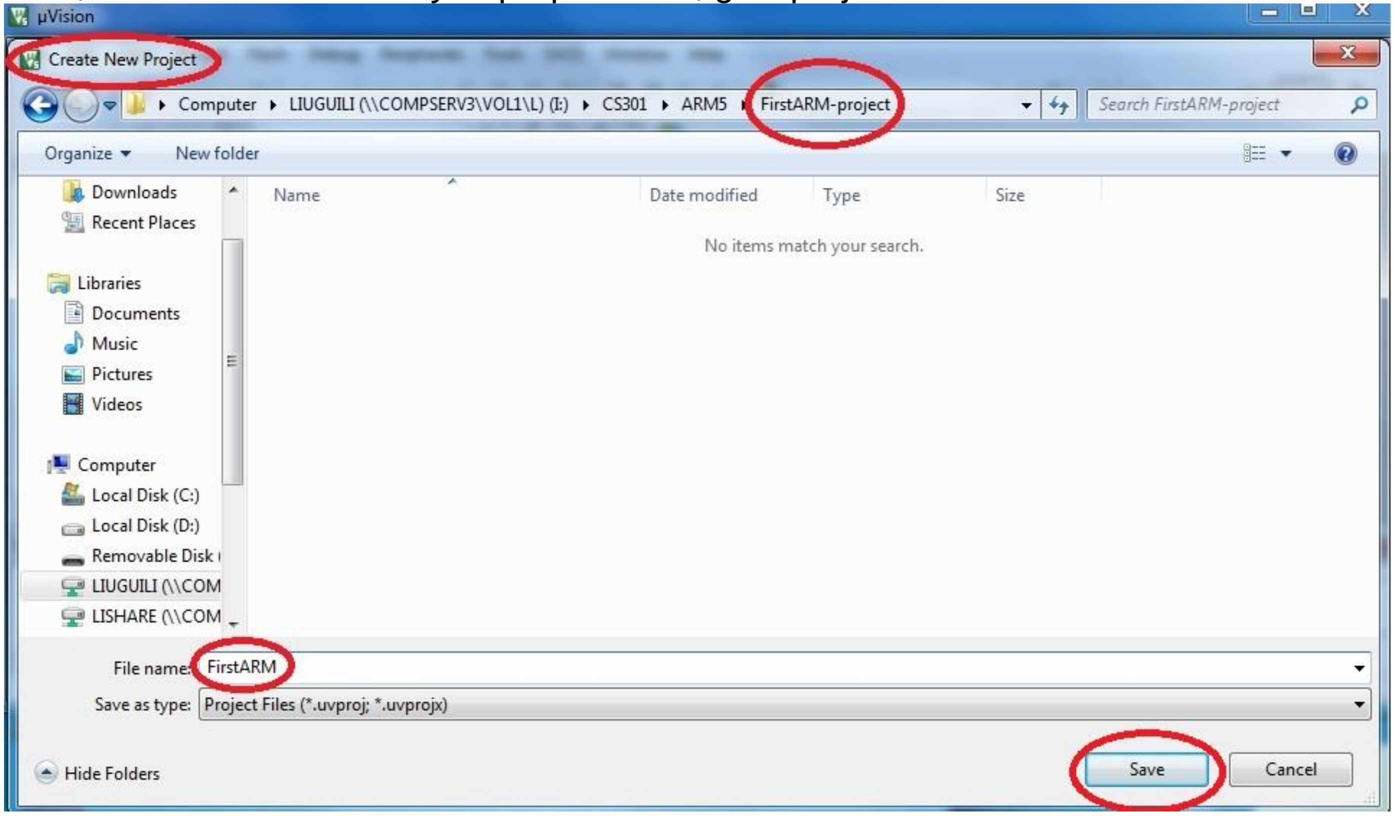

µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Project Books Func... Temp...
Build Output

Create a project

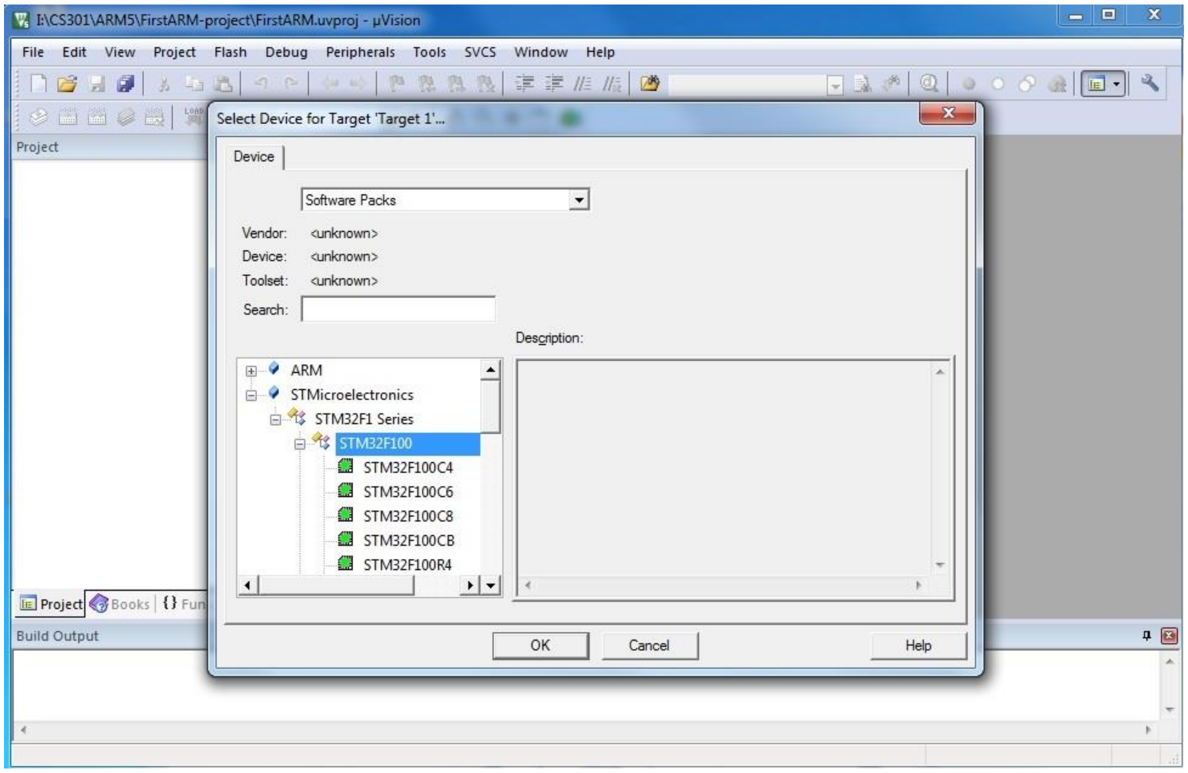
Let's create our first ARM **uVision5** project now. To create a project, click on the "Project" menu from the **uVision5** screen and select "New uVision Project...".



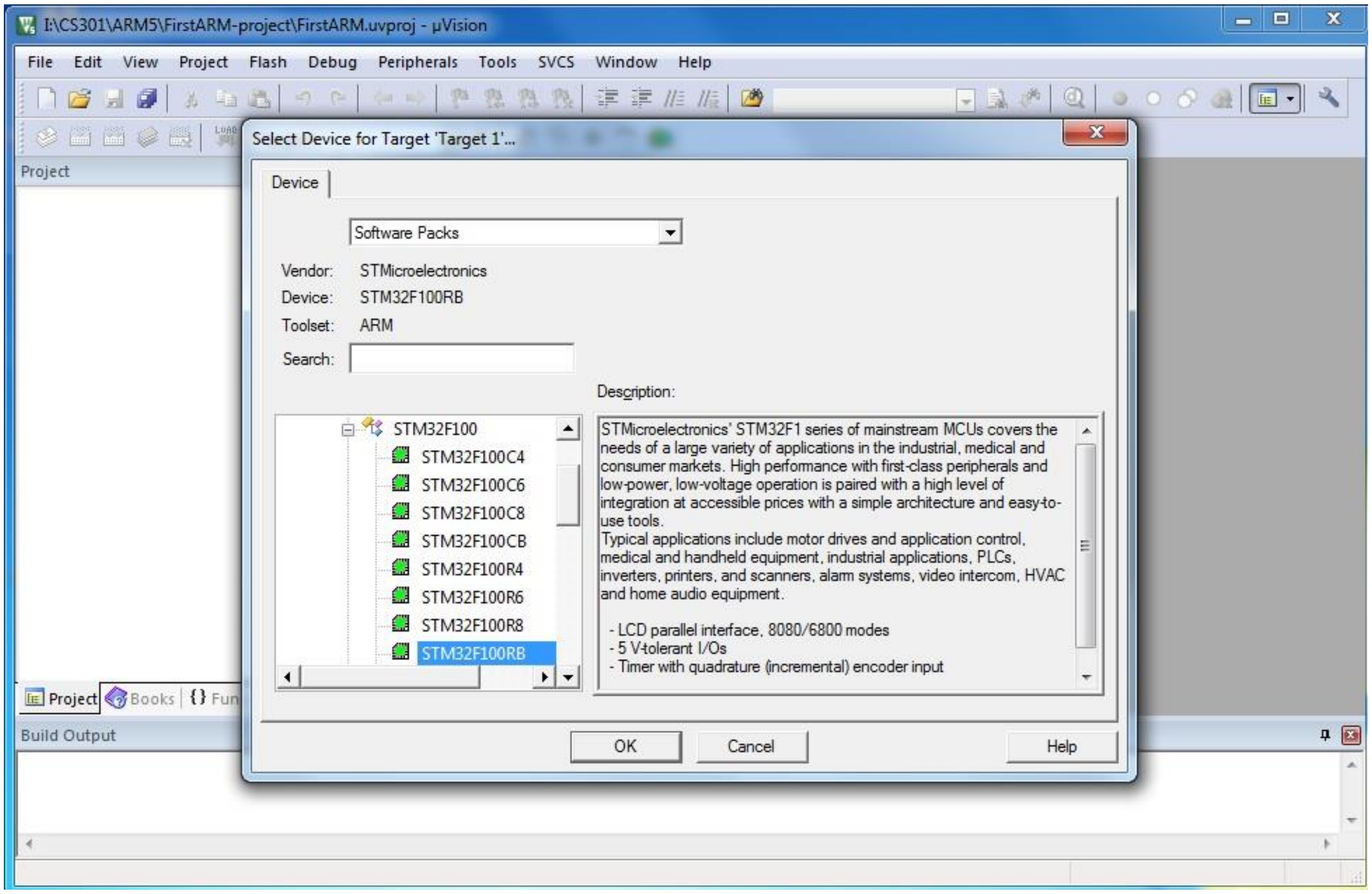
Then, select the folder that you prepared for, give project a name and save it.



From the "Select Device for Target" window, select "STMicroelectronics" and then "STM32F1 Series".



click on "+" beside "STM32F100" and then select "STM32F100RB" and click on "OK".


I:\CS301\ARM5\FirstARM-project\FirstARM.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Build Output
Select Device for Target 'Target 1'...
Device
OK
Cancel
Help

Software Packs

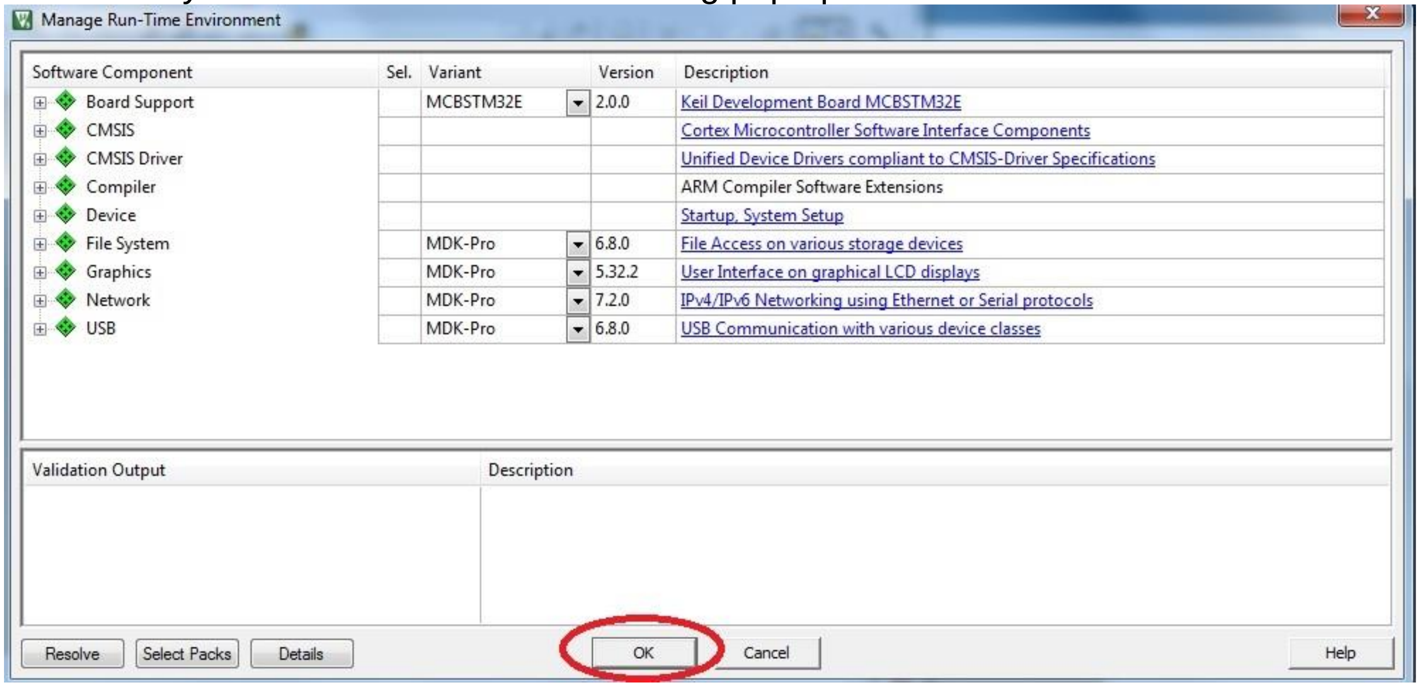
Vendor: STMicroelectronics
Device: STM32F100RB
Toolset: ARM
Search:

STM32F100
STM32F100C4
STM32F100C6
STM32F100C8
STM32F100CB
STM32F100R4
STM32F100R6
STM32F100R8
STM32F100RB

Description:
STMicroelectronics' STM32F1 series of mainstream MCUs covers the needs of a large variety of applications in the industrial, medical and consumer markets. High performance with first-class peripherals and low-power, low-voltage operation is paired with a high level of integration at accessible prices with a simple architecture and easy-to-use tools.
Typical applications include motor drives and application control, medical and handheld equipment, industrial applications, PLCs, inverters, printers, and scanners, alarm systems, video intercom, HVAC and home audio equipment.

- LCD parallel interface, 8080/6800 modes
- 5 V-tolerant I/Os
- Timer with quadrature (incremental) encoder input

Make sure you click on "OK" for the following pop up window.

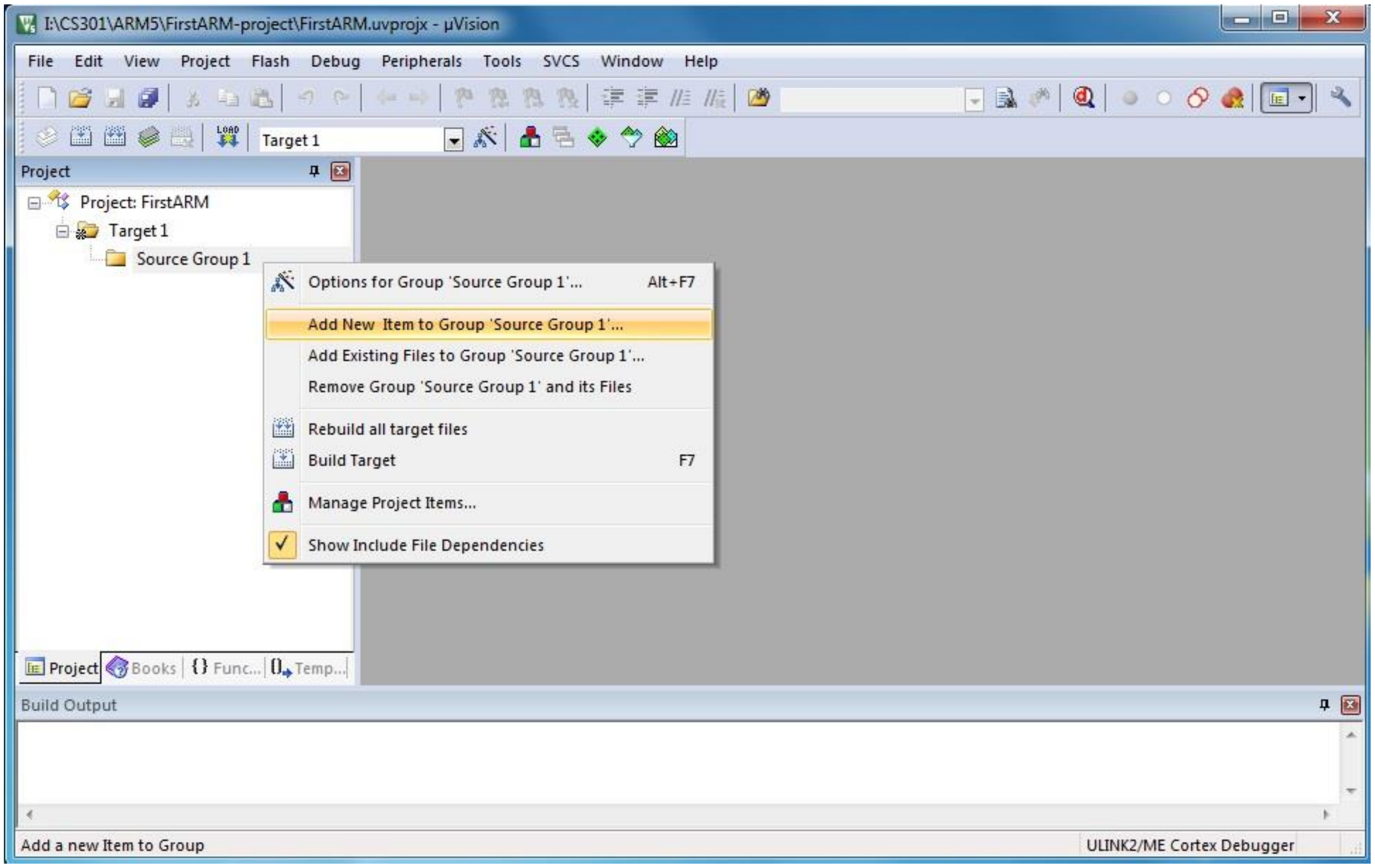


Software Component	Sel.	Variant	Version	Description
Board Support		MCBSTM32E	2.0.0	Keil Development Board MCBSTM32E
CMSIS				Cortex Microcontroller Software Interface Components
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
Compiler				ARM Compiler Software Extensions
Device				Startup, System Setup
File System		MDK-Pro	6.8.0	File Access on various storage devices
Graphics		MDK-Pro	5.32.2	User Interface on graphical LCD displays
Network		MDK-Pro	7.2.0	IPv4/IPv6 Networking using Ethernet or Serial protocols
USB		MDK-Pro	6.8.0	USB Communication with various device classes

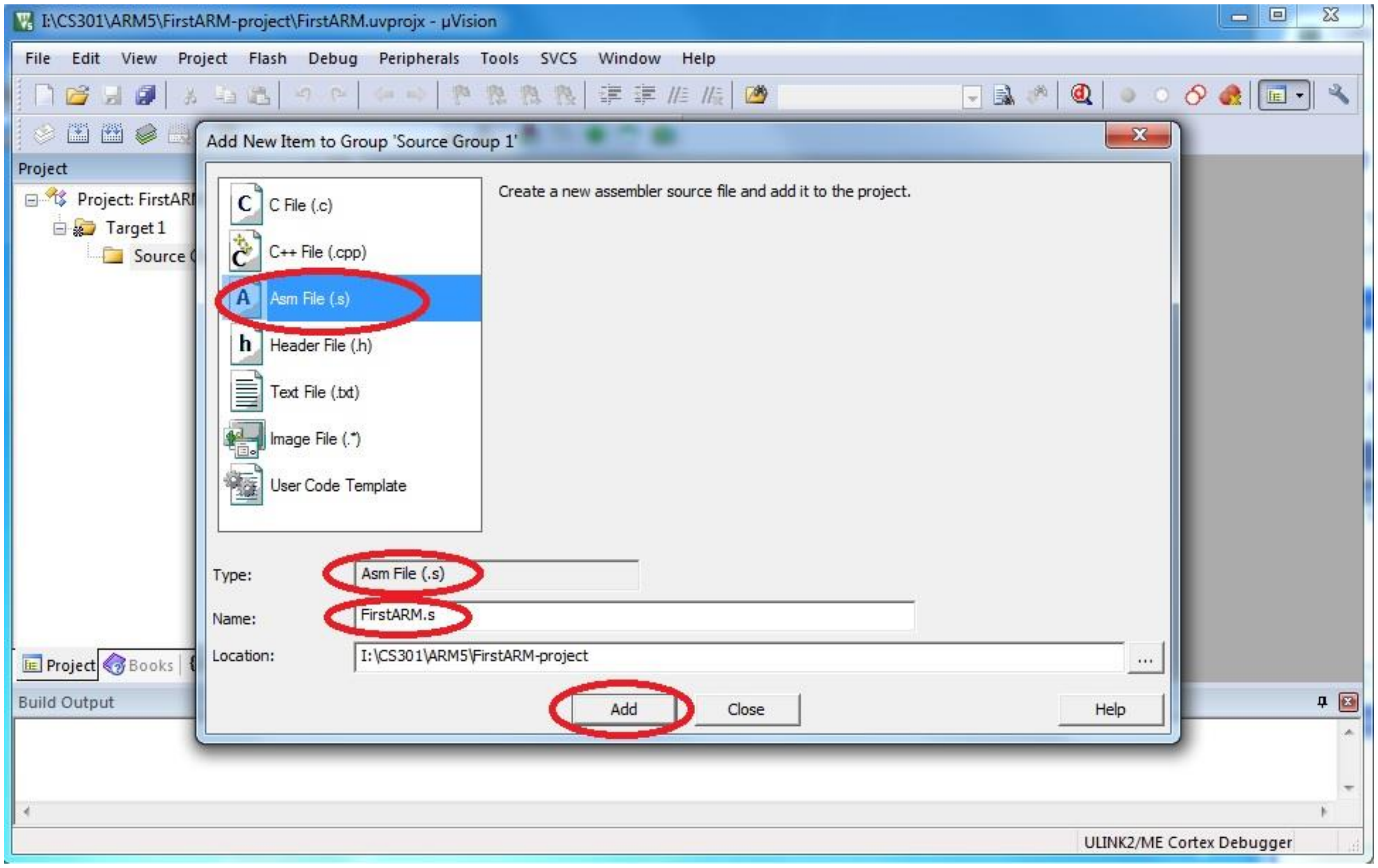
Validation Output	Description

Create Source File and Add Source File to the Project

Right click on "Source Group 1" and then select "Add New Item to Group 'Source Group 1'...".


I:\CS301\ARM5\FirstARM-project\FirstARM.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Project: FirstARM
Target 1
Source Group 1
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Remove Group 'Source Group 1' and its Files
Rebuild all target files
Build Target F7
Manage Project Items...
Show Include File Dependencies
Project Books Func... Temp...
Build Output
Add a new Item to Group
ULINK2/ME Cortex Debugger

You will see the following window and make the suggested selections to proceed.


I:\CS301\ARM5\FirstARM-project\FirstARM.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Add New Item to Group 'Source Group 1'
C File (.c)
C++ File (.cpp)
Asm File (.s)
Header File (.h)
Text File (.txt)
Image File (.*)
User Code Template
Add
Close
Help
Project: FirstARM
Target 1
Source Group 1
Build Output
ULINK2/ME Cortex Debugger

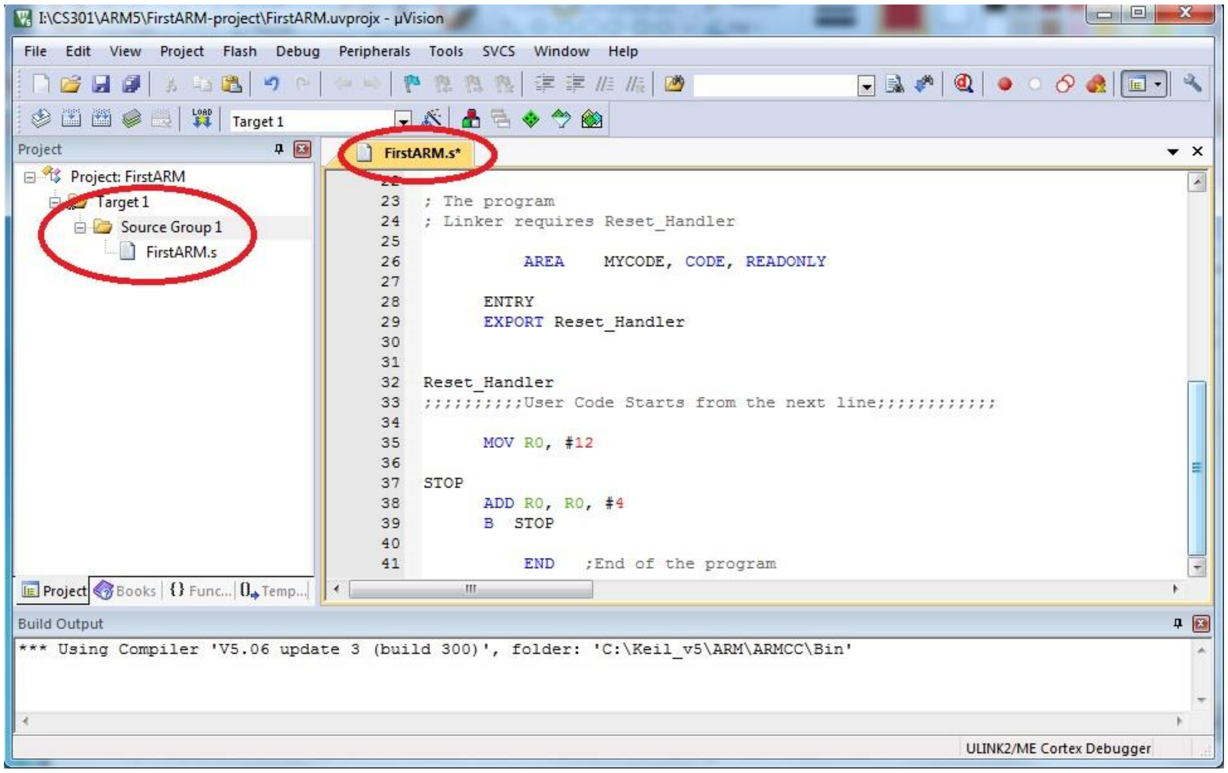
Create a new assembler source file and add it to the project.

Type: Asm File (.s)

Name: FirstARM.s

Location: I:\CS301\ARM5\FirstARM-project

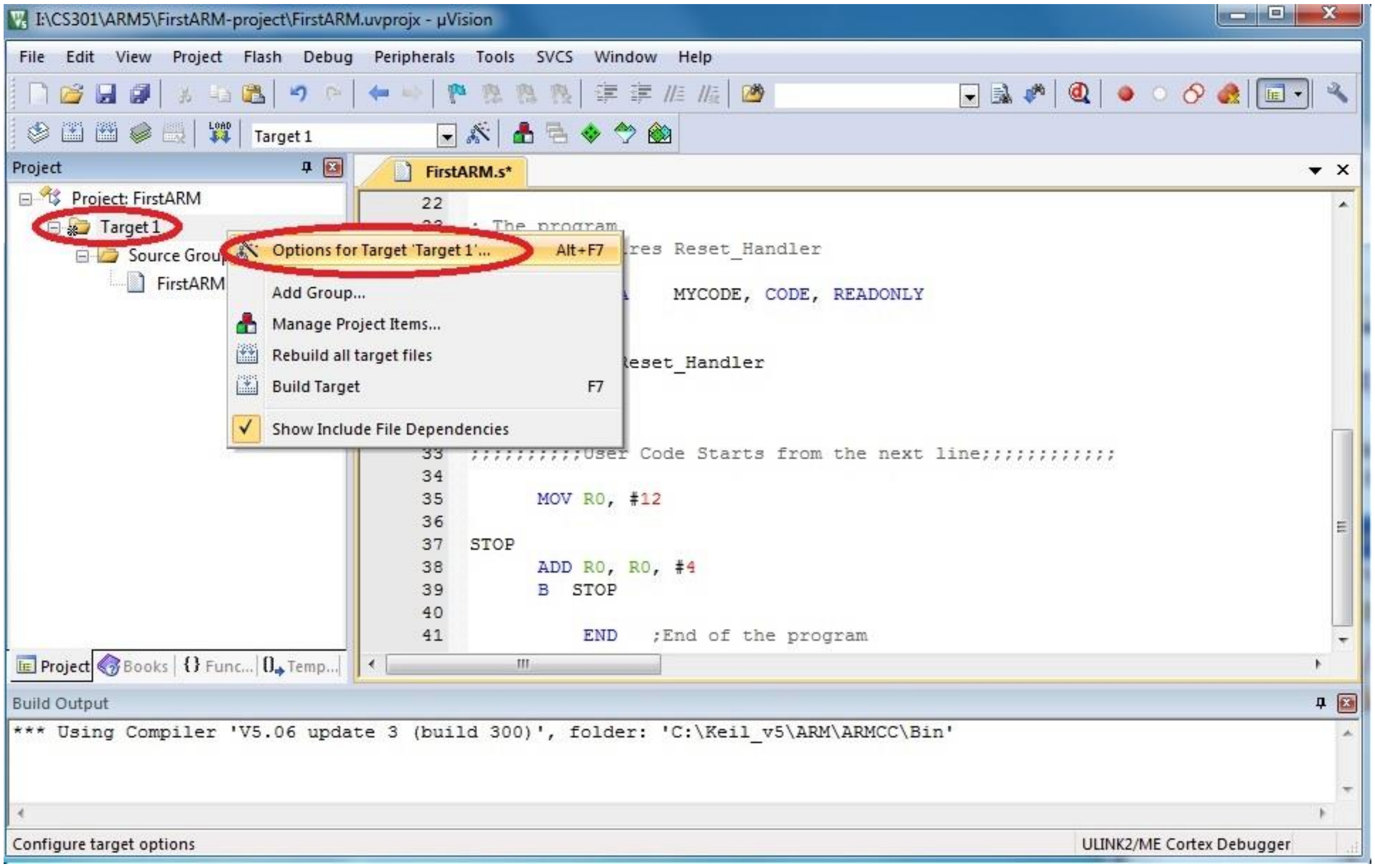
You will see the "FirstARM.s*" text edit window. That is the place you will write your ARM Assembly language program. For a test, you can copy and paste the example program into this window. You can click on the "save" buttom to save your project.


I:\CS301\ARM5\FirstARM-project\FirstARM.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Project: FirstARM
Target 1
Source Group 1
FirstARM.s
FirstARM.s*
Project Books Func... Temp...
Build Output
ULINK2/ME Cortex Debugger

```
; The program
; Linker requires Reset_Handler

        AREA    MYCODE, CODE, READONLY

    ENTRY
    EXPORT Reset_Handler


Reset_Handler
;;;;;;;;;;User Code Starts from the next line;;;;;;;;;;;;

    MOV R0, #12

STOP
    ADD R0, R0, #4
    B  STOP

        END   ;End of the program
```

```
*** Using Compiler 'V5.06 update 3 (build 300)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
```

You can right click on "Target 1" and then select "options for Target 'Target 1'..." the same as the following screen.


I:\CS301\ARM5\FirstARM-project\FirstARM.uvprojx - µVision

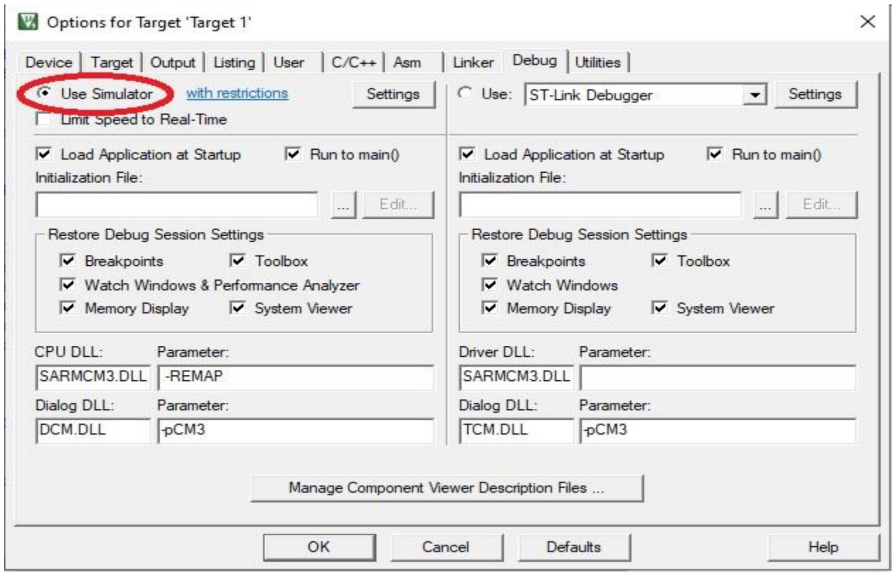
```
; The program
        ...res Reset_Handler

        ...    MYCODE, CODE, READONLY

        ...eset_Handler

;;;;;;;;;;User Code Starts from the next line;;;;;;;;;;;;

        MOV R0, #12

STOP
        ADD R0, R0, #4
        B  STOP

            END   ;End of the program
```

Build Output

```
*** Using Compiler 'V5.06 update 3 (build 300)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
```

Configure target options   ULINK2/ME Cortex Debugger

Please click on "Debug" and then select "Use Simulator".


Options for Target 'Target 1'

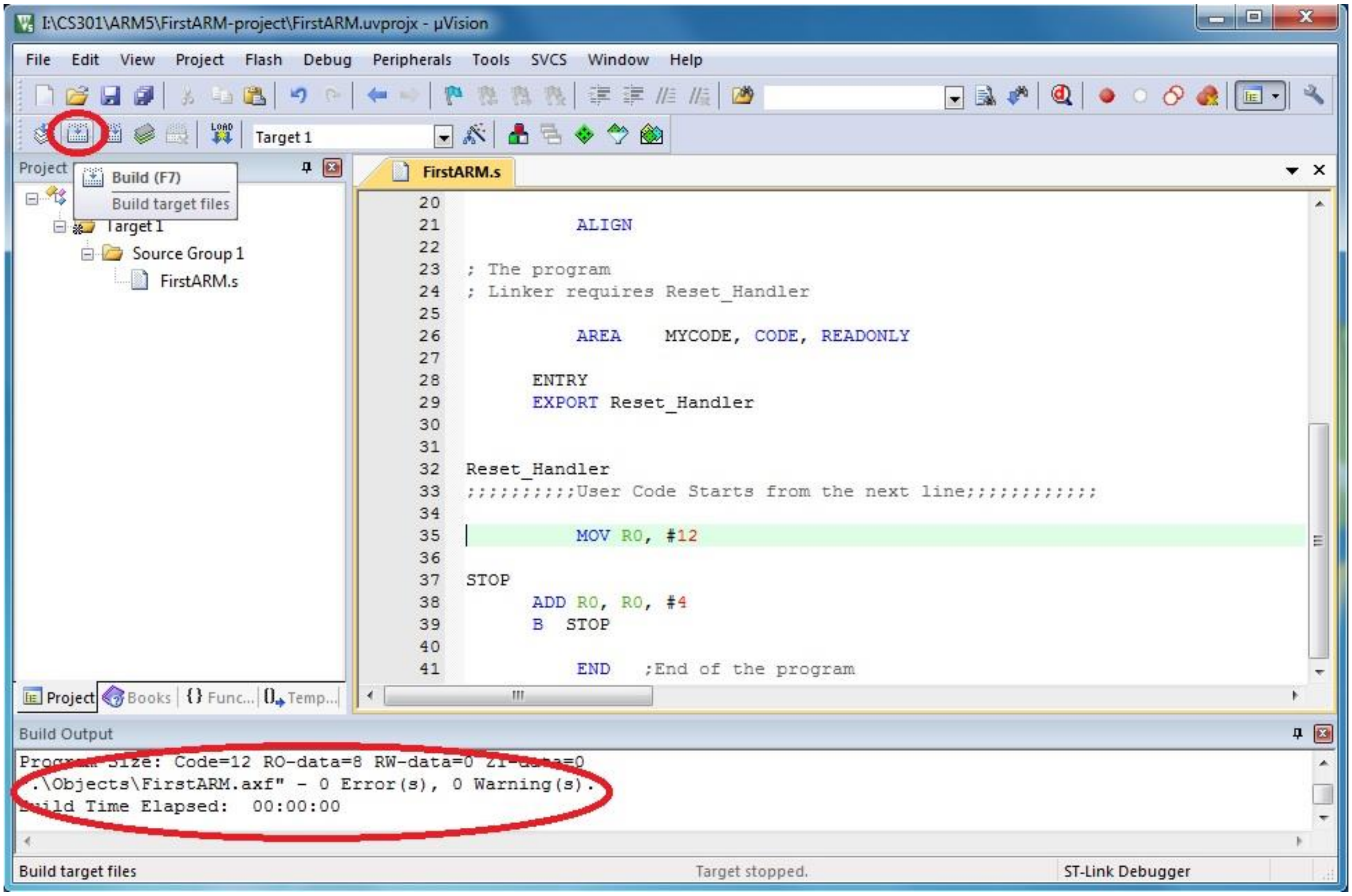
Device Target Output Listing User C/C++ Asm Linker Debug Utilities
Settings
Settings
...
Edit...
Manage Component Viewer Description Files ...
OK Cancel Defaults Help

[✓ Use Simulator] with restrictions
☐ Limit Speed to Real-Time
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows & Performance Analyzer
☑ Memory Display ☑ System Viewer
CPU DLL: SARMCM3.DLL Parameter: -REMAP
Dialog DLL: DCM.DLL Parameter: -pCM3

○ Use: ST-Link Debugger
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows
☑ Memory Display ☑ System Viewer
Driver DLL: SARMCM3.DLL Parameter:
Dialog DLL: TCM.DLL Parameter: -pCM3

Build your project

Click on the "Build" button or from the "Project" menu, you will see the following screen.

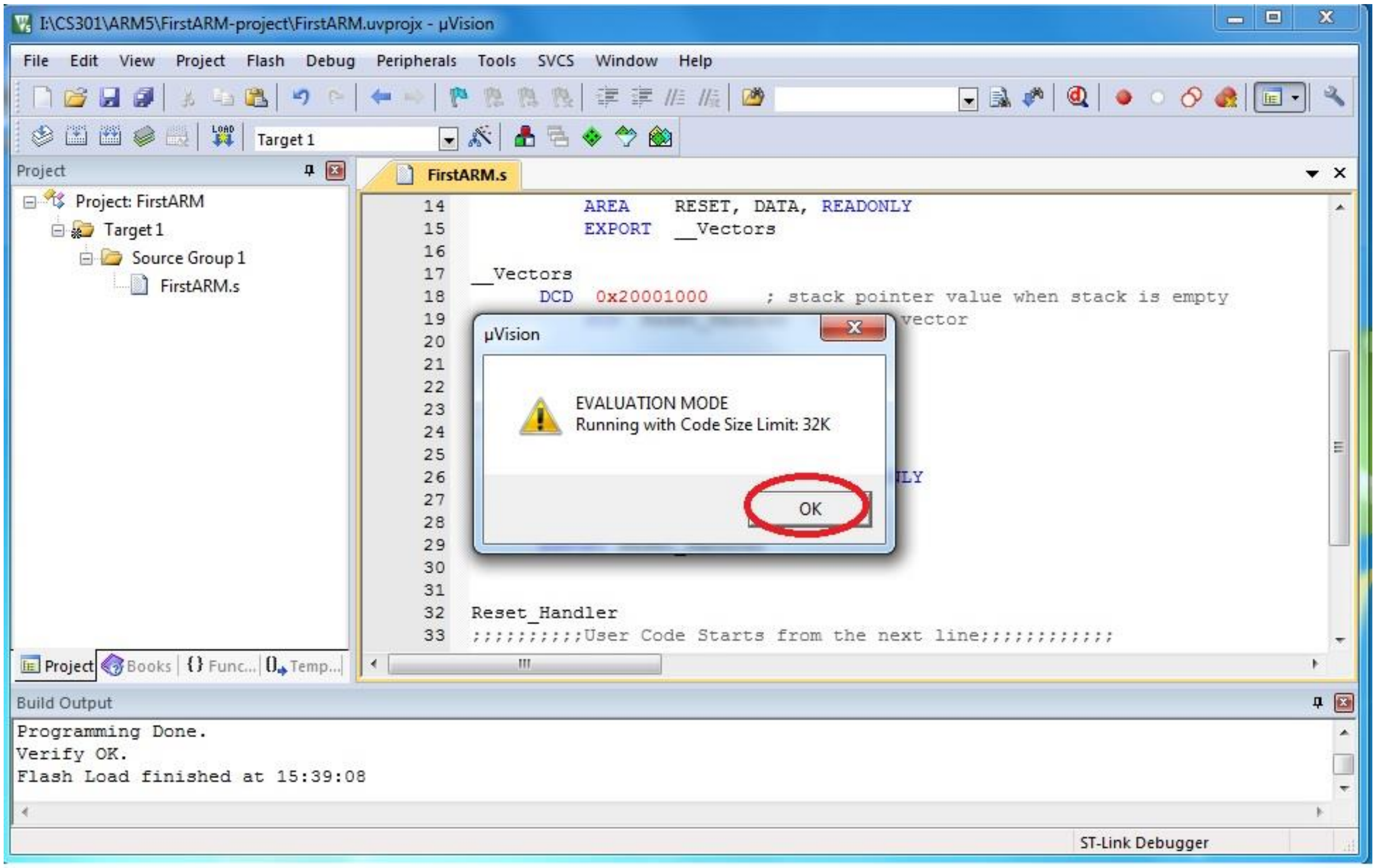

I:\CS301\ARM5\FirstARM-project\FirstARM.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Build (F7)
Build target files
Target 1
Source Group 1
FirstARM.s
Project Books Func... Temp...
Build Output
Build target files
Target stopped.
ST-Link Debugger

```
        ALIGN

; The program
; Linker requires Reset_Handler

        AREA    MYCODE, CODE, READONLY

    ENTRY
    EXPORT Reset_Handler


Reset_Handler
;;;;;;;;;;User Code Starts from the next line;;;;;;;;;;;;

        MOV R0, #12

STOP
    ADD R0, R0, #4
    B  STOP

        END    ;End of the program
```

```
Program Size: Code=12 RO-data=8 RW-data=0 ZI-data=0
".\Objects\FirstARM.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:00
```

Run the program in your project

When the assembler is happy with the program, we can run the program by selecting "Start/Stop Debug Session" from the "Debug" menu or clicking on the debug button.


I:\CS301\ARM5\FirstARM-project\FirstARM.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Project: FirstARM
Target 1
Source Group 1
FirstARM.s
Project Books Func... Temp...
Build Output
ST-Link Debugger

```
        AREA    RESET, DATA, READONLY
        EXPORT  __Vectors

__Vectors
    DCD  0x20001000     ; stack pointer value when stack is empty
                                    vector

Reset_Handler
;;;;;;;;;;User Code Starts from the next line;;;;;;;;;;;;
```

µVision

EVALUATION MODE
Running with Code Size Limit: 32K

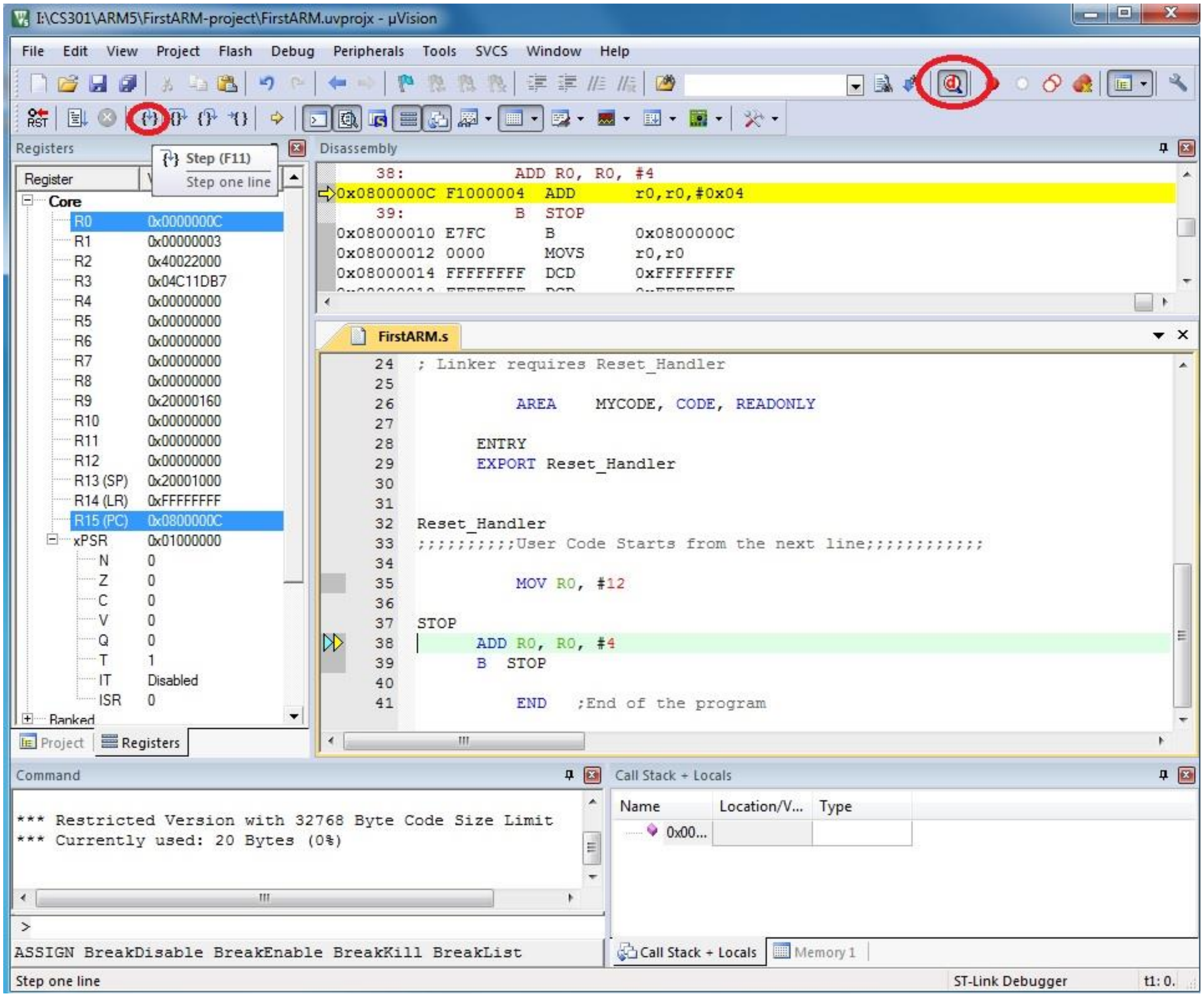
OK

```
Programming Done.
Verify OK.
Flash Load finished at 15:39:08
```

Click on "OK" for the pop up window showing "EVALUATION MODE, Running with Code Size Limit: 32K".

Open your uVision5 to full screen to have a better and complete view. The left hand side window shows you the registers and the right side window shows the program code. There are some other windows open. You may adjust the size of them to see better.

Run the program step by step, you can observe the change of the values in the registers.



Register	Value
Core	
R0	0x0000000C
R1	0x00000003
R2	0x40022000
R3	0x04C11DB7
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x20000160
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x0800000C
xPSR	0x01000000
N	0
Z	0
C	0
V	0
Q	0
T	1
IT	Disabled
ISR	0
Banked	

```
    38:          ADD R0, R0, #4
0x0800000C F1000004  ADD      r0,r0,#0x04
    39:          B  STOP
0x08000010 E7FC      B        0x0800000C
0x08000012 0000      MOVS     r0,r0
0x08000014 FFFFFFFF  DCD      0xFFFFFFFF
```

```
; Linker requires Reset_Handler

        AREA    MYCODE, CODE, READONLY

    ENTRY
    EXPORT Reset_Handler


Reset_Handler
;;;;;;;;;User Code Starts from the next line;;;;;;;;;;;

        MOV R0, #12

STOP
    ADD R0, R0, #4
    B  STOP

        END   ;End of the program
```

```
*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 20 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Location/V...	Type
0x00...		

Click on the "Start/Stop Debug Session" from the "Debug" menu or click on the debug button to stop executing the program.

We will analyze the program and see how it works.

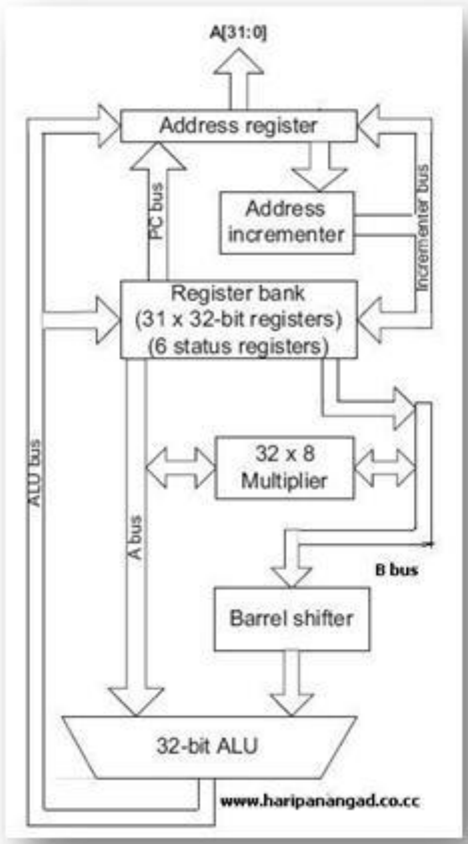
It works with both the simulated target and the real circuit board STM32VLDISCOVERY Board.

We will demonstrate it in the lab for you.

ARM Architecture

ARM processors are mainly used for low-power and low cost applications such as mobile phones, communication modems, automotive engine management systems, and hand-held digital systems.

Here is a diagram of the ARM architecture for your reference.



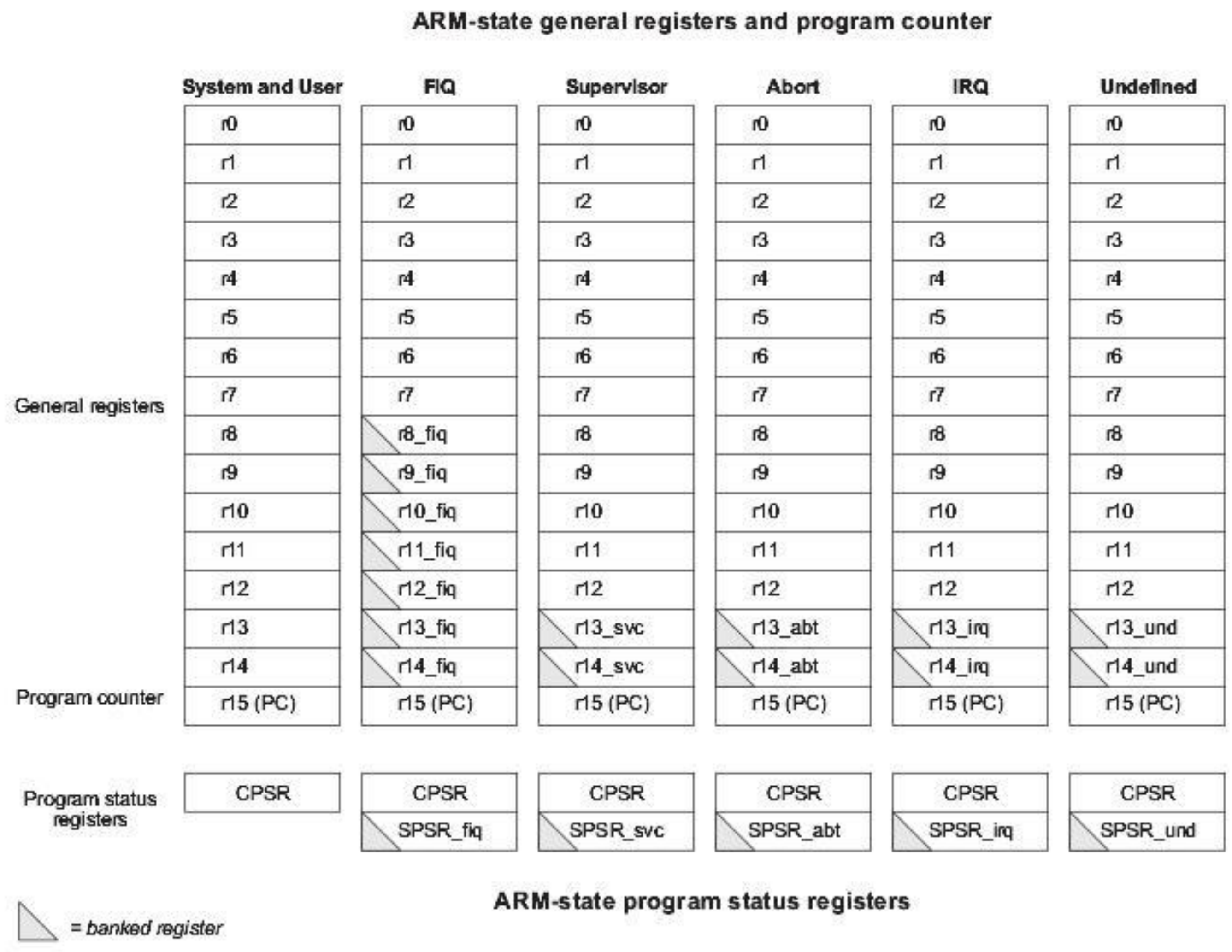
```
    ARM Architecture is an Enhanced RISC Architecture.
    It has large uniform Register file and uses Load Store Architecture.
       i.e. operations operate on registers and not in memory locations.

    ARM Architecture instructions are of uniform and fixed length.
    It is a 32 bit processor.
    It also has 16 bit variant called THUMB.
       i.e. it can be used as 32 bit and as 16 bit processor.
```

ARM cores are licensed to partners/manufacturers so as to develop and fabricate new microcontrollers around same processor cores. A microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.

The ARM Cortex-M3 microcontroller will be used in the lab with the STM32VLDISCOVERY board. For more information, visit **STM32VLDISCOVERY Board**.

ARM Registers

Here is the Register Organization in ARM State.



ARM-state general registers and program counter

	System and User	FIQ	Supervisor	Abort	IRQ	Undefined
General registers	r0	r0	r0	r0	r0	r0
	r1	r1	r1	r1	r1	r1
	r2	r2	r2	r2	r2	r2
	r3	r3	r3	r3	r3	r3
	r4	r4	r4	r4	r4	r4
	r5	r5	r5	r5	r5	r5
	r6	r6	r6	r6	r6	r6
	r7	r7	r7	r7	r7	r7
	r8	◺ r8_fiq	r8	r8	r8	r8
	r9	◺ r9_fiq	r9	r9	r9	r9
	r10	◺ r10_fiq	r10	r10	r10	r10
	r11	◺ r11_fiq	r11	r11	r11	r11
	r12	◺ r12_fiq	r12	r12	r12	r12
	r13	◺ r13_fiq	◺ r13_svc	◺ r13_abt	◺ r13_irq	◺ r13_und
	r14	◺ r14_fiq	◺ r14_svc	◺ r14_abt	◺ r14_irq	◺ r14_und
Program counter	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)

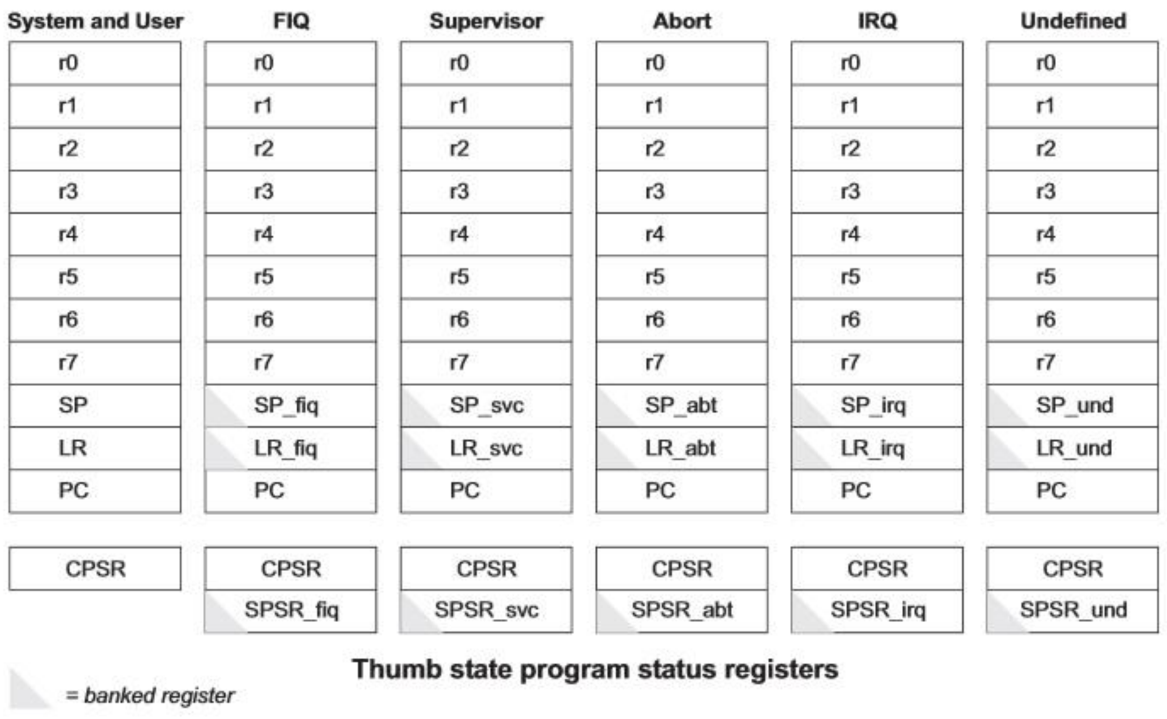
Program status registers	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		◺ SPSR_fiq	◺ SPSR_svc	◺ SPSR_abt	◺ SPSR_irq	◺ SPSR_und

ARM-state program status registers

◺ = *banked register*

Here is the Register Organization in THUMB State.

Thumb state general registers and program counter



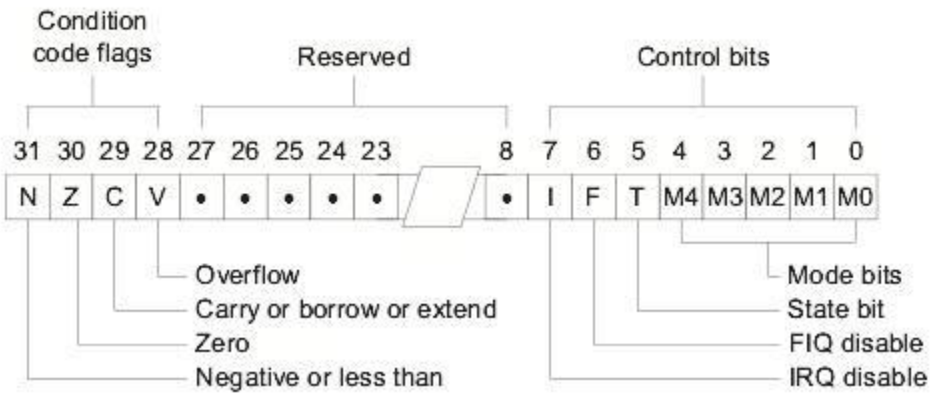
System and User	FIQ	Supervisor	Abort	IRQ	Undefined
r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r2	r2
r3	r3	r3	r3	r3	r3
r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7
SP	◣ SP_fiq	◣ SP_svc	◣ SP_abt	◣ SP_irq	◣ SP_und
LR	◣ LR_fiq	◣ LR_svc	◣ LR_abt	◣ LR_irq	◣ LR_und
PC	PC	PC	PC	PC	PC

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	◣ SPSR_fiq	◣ SPSR_svc	◣ SPSR_abt	◣ SPSR_irq	◣ SPSR_und

Thumb state program status registers

◣ = banked register

Here is the Program Status Register Format:



```
In ARM State, there are 16 general purpose registers;
              one or more status registers are accessible at any one time.

In  THUMB State, there are 8 general purpose registers;
                 PC, SP, LR and CPSR are accessible.

Conditonion code flags in CPSR:
N - Negative or less than flag
Z - Zero flag
C - Carry or bowrrow or extended flag
V - Overflow flag
```

ARM Instructions

Here are a few sample ARM Instructions for you to test out for this lab:

```
MOV     R2, #0x76
; Move the 8-bit Hex number 76 to the low portion of R2

MOV     R2, #0x7654
; Move the 16-bit Hex number 7654 to the low portion of R2

MOVT    R2, #0x7654
; Move the 16-bit Hex number 7654 to the high portion of R2

MOV32   R2, #0x76543210        ; Move the 32-bit Hex number 76543210 to the R2

LDR     R2, = 0x76543210       ; Load R2 with the 32-bit Hex number 76543210

ADD     R1, R2, R3             ; R1 = R2 + R3

ADDS    R1, R2, R3             ; R1 = R2 + R3, and FLAGs are updated

SUB     R1, R2, R3             ; R1 = R2 - R3

SUBS    R1, R2, R3             ; R1 = R2 - R3, and FLAGs are updated

B       LABEL                  ; Branch to LABEL
```

The entire list of the Instructions can be found in the **Cortex-M3 Devices Generic User Guide.**
OR see The Cortex-M3 Instruction Set in **Cortex-M3 Devices Generic User Guide**, in Chapter 3: The Cortex-M3 Instruction Set.

Lab work:

Write your first ARM assembly language program **MyFirstARM.s**.
The program will execute the following instructions. You will run the program step by step, observe and answer the question after each statement.

```
      MOV     R2,     #0x01           ; R2 = ?
      MOV     R3,     #0x02           ; R3 = ?

      ;Other examples to move immediate values
      MOV     R5,     #0x3210         ; R5 = ?

      MOVT    R5,     #0x7654         ; R5 = ?

      MOV32   R6,     #0x87654321     ; R6 = ?

      LDR     R7,     = 0x87654321     ; R7 = ?


      ADD     R1,R2,R3                ; R1 = ?
      MOV32    R3,    #0xFFFFFFFF     ; R3 = ?
      ADDS    R1,R2,R3                ; R1 = ?
                                      ; specify Condition Code updates

      SUBS    R1,R2,R3                ; R1 = ?
                                      ; specify Condition Code updates

      MOV     R4,     #0xFFFFFFFF     ; R4 = ?
      ADD     R1,R2,R4                ; R1 = ?
                               ; How did that operation affect the flags in CPSR?

      ADDS    R1,R2,R4         ; R1 = ?
                               ; Please specify Condition Code updates
                               ; and now what happened to the flags in the CPSR?


      MOV     R2,     #0x00000002     ; R2 = ?
      ADDS    R1,R2,R4                ; R1 = ?
                                      ; again, what happened to the flags?

      MOV     R2,     #0x00000001     ; R2 = ?
      MOV     R3,     #0x00000002     ; R3 = ?
      ADDS    R1,R2,R3                ; R1 = ?
                                      ; Add some small numbers again
                                      ; and check the flags again......

      ; Add numbers that will create an overflow
      MOV      R2,    #0x7FFFFFFF     ; R2 = ?
      MOV     R3,     #0x7FFFFFFF     ; R3 = ?

      ADDS     R1,R2,R3               ; R1 = ?
                                      ; Check the flags in the CPSR?
```

You will hand in the following:

1. The screenshot of the program successfully built in Keil uVision.
2. The source code in the file **MyFirstARM.s** with the answers.