CMPE-250
Assembly and Embedded Programming
Spring 2021
Laboratory Exercise One
Keil MDK-ARM Microcontroller Development Kit Tutorial

This exercise introduces the Keil® MDK-ARM Microcontroller Development Kit, which runs on Microsoft® Windows™. The objective of this exercise is familiarization with the Keil MDK-ARM programming environment (creating projects, entering assembly language programs, assembling, simulating, and debugging). An assembly language program for the ARM Cortex-M0+ is entered and simulated in Keil MDK-ARM, and various questions about the environment explore important features of assembly language programming with Keil MDK-ARM.

Prelab Work

(No prelab work is required for this exercise.)

Procedure

These steps are intended for a course lab workstation. Additional steps may be required for a new Keil MDK-ARM installation, and are indicated by the 💻 computer icon.

Assembling Code Using Keil MDK-ARM

1. Start Keil MDK-ARM's µVision® IDE (integrated development environment).
   Click the start menu icon at the left of the Windows task bar, then then scroll through the list of programs to select Keil µVision5. The Keil µVision IDE will then open, as shown in Figure 1.

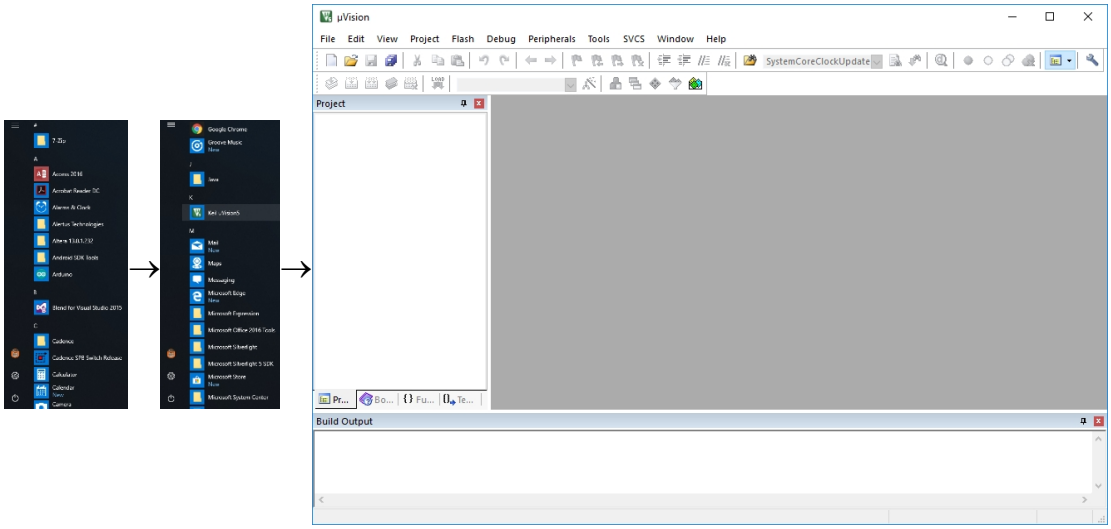


Figure 1. Opening Keil MDK-ARM's µVision IDE

2. Create a new project.
From the Keil µVision IDE main menu, select Project→New µVision Project … to launch the Create New Project window, shown in Figure 2, which is a standard Windows Explorer-style file save dialog box.

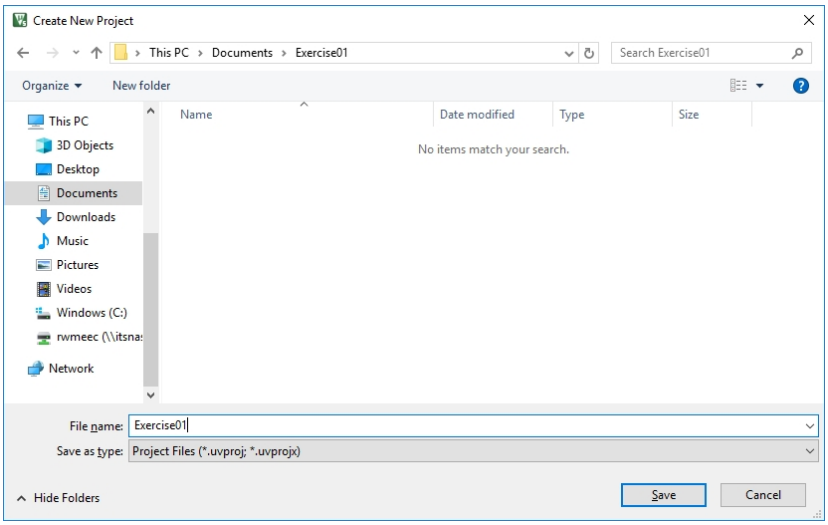


Figure 2. Create New Project Window

a. Browse to a location where you want to keep project files.
   ☞ Suggestion: somewhere in your Documents folder.
   💣 Caution: files on the lab workstations are not permanent and are not accessible from other workstations. Although using local files on your lab workstation is recommended while you are working in lab for speed and file stability, before logging out of your workstation you need to save all of your work in a personal external location, (e.g., myCourses locker, drop box, external USB drive, etc.).
b. Use New Folder to create a folder for your project, and name it *Exercise01*.
c. Double-click the new *Exercise01* folder to open it.
d. In the File name field at the bottom, type *Exercise01*.
e. Press the Save button to create your project. Keil µVision IDE will then launch the Select Device for Target 'Target 1' … window, shown in Figure 3.

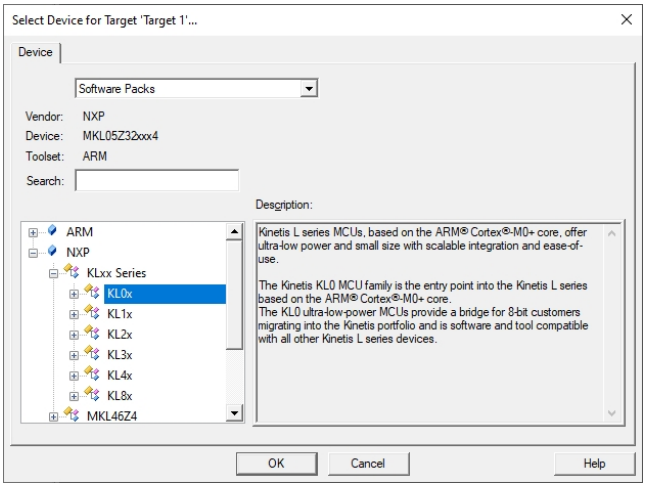


Figure 3. Select Device for Target Window

f. To set the target as the NXP MKL05Z32VFM4 microcontroller on the NXP FRDM-KL05Z Freedom development platform, in the left pane below the Search field, successively click ⊞ next to each of the following categories to select MKL05Z32xxx4, as shown in Figure 4.

NXP → KLxx Series → KL0x → MKL05Z32xxx4.

(Alternatively, type *MKL05Z32* in the Search field to return MKL05Z32xxx4.)

💣 Caution: If there is more than one MKL05Z32 device, make sure to choose the one indicated above from the KLxx Series software pack. There may be newer packs for this device, but they will not be compatible with later hardware exercises.

💻 Note: For work outside of lab on a new or different installation of Keil MDK-ARM, the KLxx Series software pack may not be installed. If so, press the Cancel button followed by the OK button on the resulting pop-up window (shown in Figure 5) to cancel the new project creation, follow the steps in the **Installing Keil Software Packs** section on p. 21, and then return to procedure step 2 to create a new project.

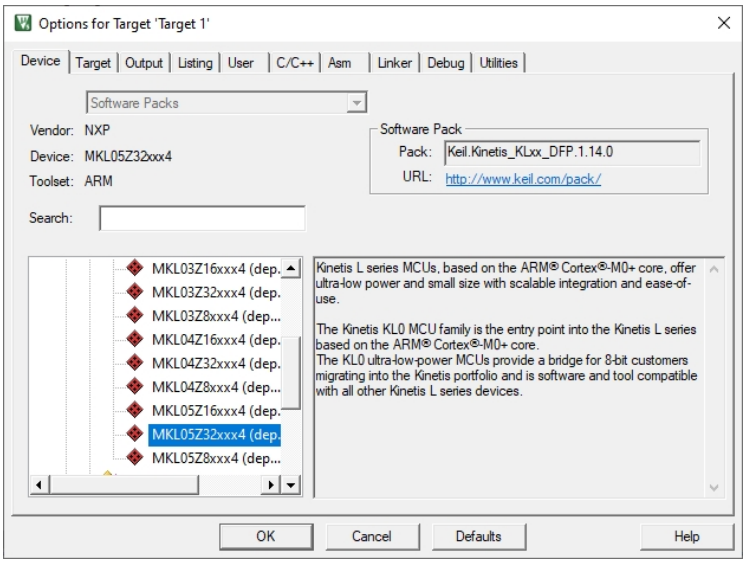


Figure 4. Select Device for Target MKL05Z32VFM4

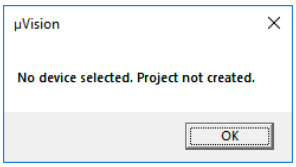


Figure 5. Cancel create new project pop-up window

g. Click the OK button to finish creating the Exercise01 project.

h. When Keil µVision IDE launches the Manage Run-Time Environment window, shown in Figure 6, where Keil ARM-MDK software components can be added to the project, click the OK button to continue without adding any software components.

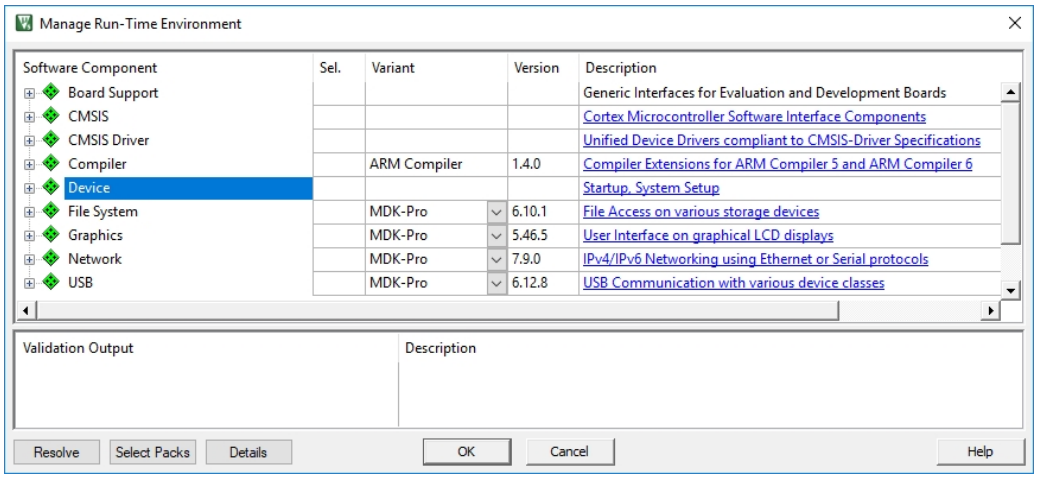


Figure 6. Manage Run-Time Environment

i. The project then opens in Keil µVision IDE, as shown in Figure 7.

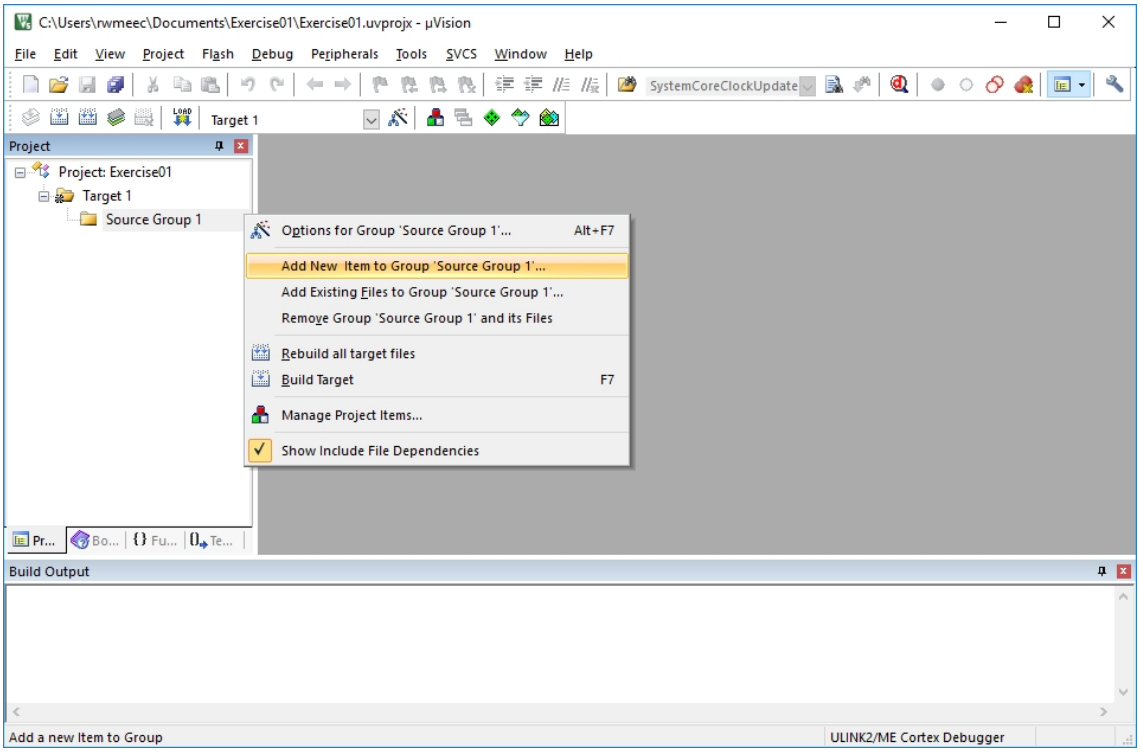


Figure 7. Adding New Item to Project Open in µVision IDE

3. Add an assembly source file to the project.
   a. In the Project pane along the left side of the µVision IDE window, click ⊞ next to Target 1 to show Source Group 1, as shown in Figure 7.
   b. Right-click on Source Group 1, and in the resulting drop-down menu select Add New Item to Group 'Source Group 1' … to open the Add New Item to Group 'Source Group 1' window, as shown in Figure 8.

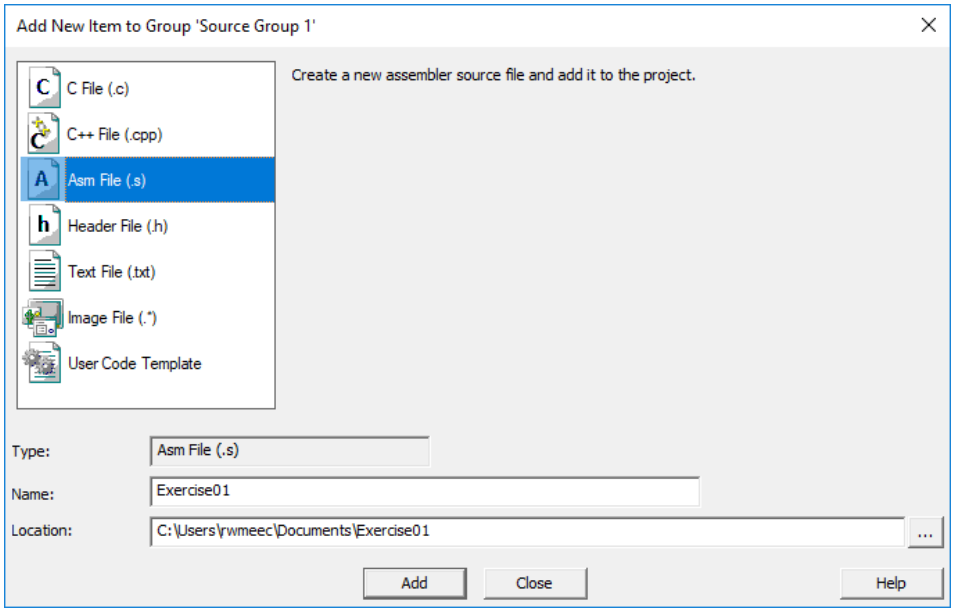


Figure 8. Adding New Assembly Source File to Project

c. In the top left pane, click Asm File (.s) to change the Type field to ASM File (.s).
d. In the Name field, type *Exercise01*.
e. Click the Add button to create Exercise01.s. Exercise01.s is then opened in the editor pane on the right side of the µVision IDE window, as shown in Figure 9.
   ☞ Note: Figure 8 also indicates the most commonly used toolbar icons for simulation with µVision IDE.

4. Enter assembly source code.
   a. Open the course's assembly language program template for simulation from the myCourses lecture section.
      Content → Resources → ProgramTemplateSimulation.s
      ☞ If you download the program template rather than having myCourses open it in the web browser, make sure to use a plain text editor, (e.g., Notepad++), to open the template, as with any assembly language source file.
      💣 All assembly language source files posted on myCourses have had the .txt extension added so that myCourses will work with them as text documents. If you want to download them and work directly with them in Keil MDK-ARM, you will need to remove the .txt extension. Alternatively, the procedure that follows explains copying the contents of these text files into Keil MDK-ARM assembly files.
   b. Copy the contents of the template into the Exercise01.s editor pane.
      ☞ Note the line spacing and indentation of an assembly language source file.
   c. Click the Save icon (or select menu File→Save) to save the program.

d. Edit Exercise01.s to make the changes shown in Figures 10 through 12.
   i. Edit the first line, (TTL, which stands for title), to match the title shown in Figure 10.
   ii. Edit the program description in the first two lines of the program's comment header to match the program description shown in Figure 10.
   iii. Replace the program header information within <> marks shown in Figure 10 with your correct author identification information: name, date, and lab section.

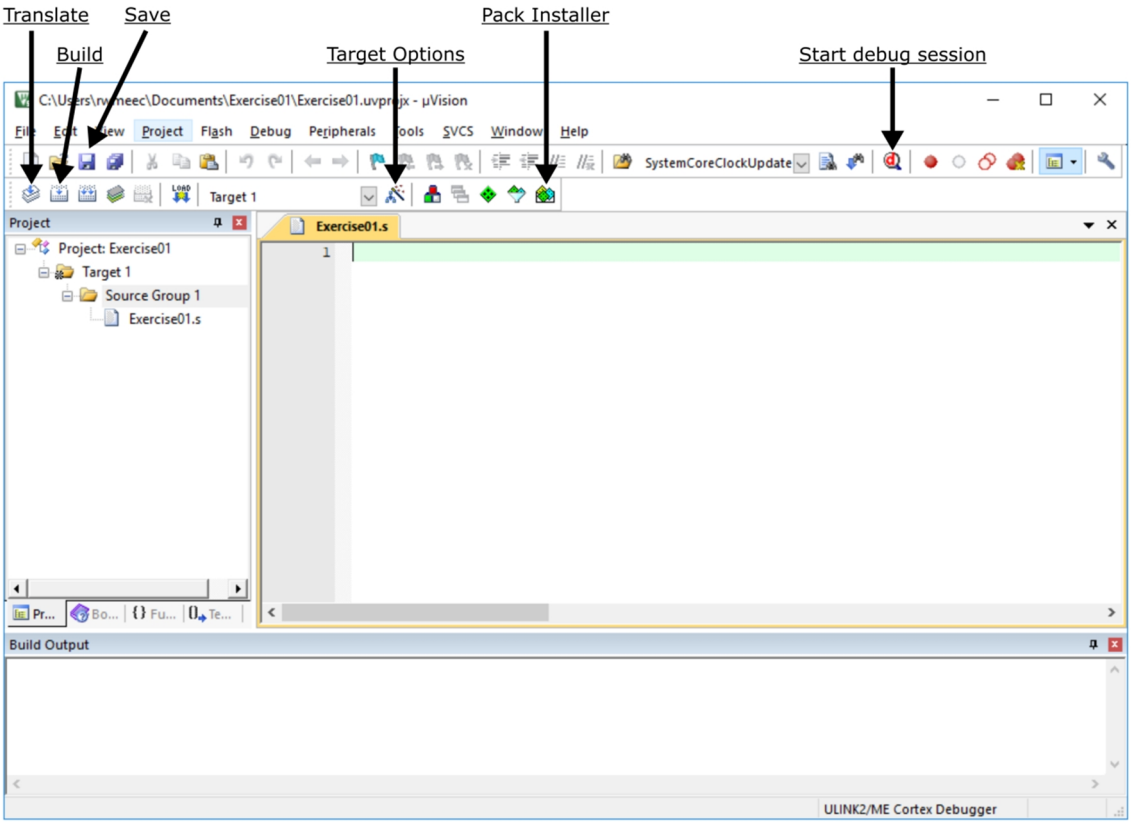


Figure 9. Source File in µVision IDE Editor

```
                TTL CMPE-250 Lab Exercise One
;*****************************************************************************
;Descriptive comment header goes here.
;(What does the program do?)
;Name:  <Your name here>
;Date:  <Date completed here>
;Class:  CMPE-250
;Section:  <Your lab section, day, and time here>
;-----------------------------------------------------------------
;Keil Simulator Template for KL05
;R. W. Melton
;January 21, 2021
;*****************************************************************************
```

Figure 10. Exercise One Assembly Source Code Header

iv. Type the main program code shown in Figure 11 between the "begin main program code" and "end main program code" comment lines. Make sure your typed code uses the same column format and spacing as Figure 11.

```
;*******************************************************************************
;Program
;Linker requires Reset_Handler
            AREA    MyCode,CODE,READONLY
            ENTRY
            EXPORT  Reset_Handler
Reset_Handler  PROC {}
main
;-------------------------------------------------------------
;Initialize registers R0-R12
            BL      RegInit
;>>>>> begin main program code <<<<<
;Mask interrupts
            CPSID I
MainLoop                            ;do {
            NOP                     ;   "no operation"
            LDR   R2,=ConstData     ;   ConstPtr = &ConstData
            MOVS  R3,#0x02          ;   Counter = 2
Loop                                ;   do {
            LDR   R1,[R2, #0]       ;     R1 = ConstPtr[0]
            LDR   R0,=VarData       ;     VarPtr = &VarData[0]
            STR   R1,[R0, #0]       ;     VarPtr[0] = ConstPtr[0]
            LDR   R1,[R2, #4]       ;     R1 = ConstPtr[1]
            STR   R1,[R0, #4]       ;     VarPtr[1] = ConstPtr[1]
            ADDS  R2,R2,#8          ;     ConstPtr = &(ConstPtr[2])
            SUBS  R3,R3,#1          ;     Counter--
            BNE   Loop              ;   } while (Counter != 0)
            NOP                     ;   "no operation"
            B     MainLoop          ;} forever
;>>>>>   end main program code <<<<<
;Stay here
            B       .
            ENDP
;-------------------------------------------------------------
```

Figure 11. Exercise One Assembly Source Code Main Program

v. Type the constants shown in Figure 12 between the “begin constants here” and “end constants here” comment lines.
vi. Type the variable shown in Figure 12 between the “begin variables here” and “end variables here” comment lines.

```
;*****************************************************************
;Constants
            AREA    MyConst,DATA,READONLY
;>>>>> begin constants here <<<<<
ConstData   DCD   0x0000000A,0x0000000B,0x00000010,10
;>>>>>   end constants here <<<<<
;*****************************************************************
            AREA    |.ARM.__at_0x1FFFFC00|,DATA,READWRITE,ALIGN=3
            EXPORT  __initial_sp
;Allocate system stack
            IF      :LNOT::DEF:SSTACK_SIZE
SSTACK_SIZE EQU     0x00000100
            ENDIF
Stack_Mem   SPACE   SSTACK_SIZE
__initial_sp
;*****************************************************************
;Variables
            AREA    MyData,DATA,READWRITE
;>>>>> begin variables here <<<<<
VarData     SPACE   8
;>>>>>   end variables here <<<<<
            END
```

Figure 12. Exercise One Assembly Source Code Constants and Variables

e. Click the Save icon (or select menu File→Save) to save the program.

5. Edit target options for the project.
   ☞ Note: during assembly, a listing file will be produced, which has columns added to the left of the assembly source. Because the number of characters on a line increases in the listing file, it is best viewed in landscape mode. The settings below produce a listing file suitable for printing in landscape mode from a text editor, such as Notepad++.
   a. On the second toolbar row, click the Target Options icon (or select menu Project→Options for Target 'Target1' …) to launch the Options for Target 'Target1' window, as shown in Figure 13.

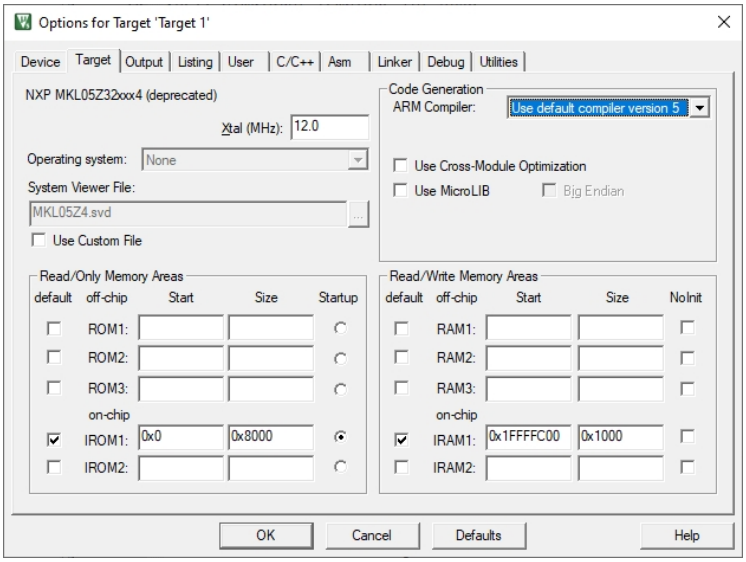


Figure 13. Setting Options for Target Compiler

   b. Click on the Target tab along the top of the window to get the options shown in Figure 13.
   c. At the top right, under the tabs, in the Code Generation options, use the pull-down dialog to the right of ARM Compiler to select Use default compiler version 5, as shown in Figure 13.
   d. Click on the Listing tab along the top of the window to get the options shown in Figure 14.
   e. Change Page Width to *120*.
   f. Change Page Length to *49*.
   g. Click the OK to save the options and close the options window.

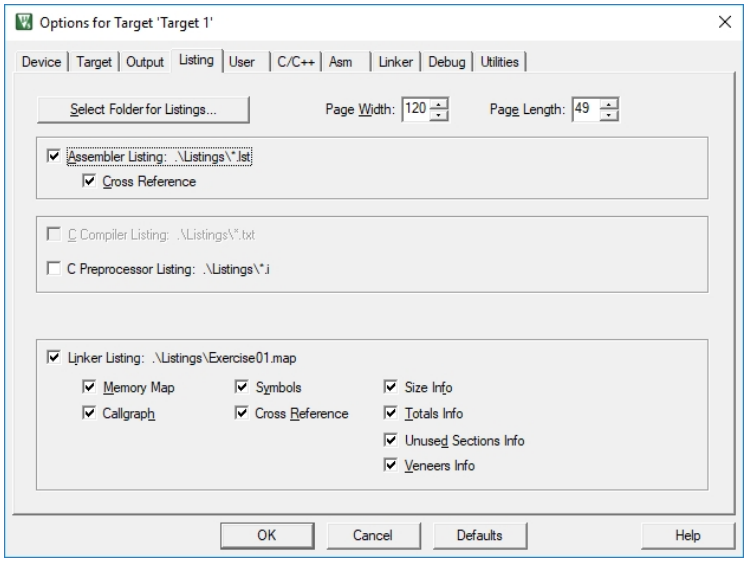


Figure 14. Setting Options for Listing File

6. Assemble Exercise01.s.
   a. On the second toolbar row, click the Translate icon (or select menu Project→Translate <path>\Exercise01.s) to assemble the source code
   b. In the Build Output pane at the bottom of the window, the text assembling Exercise01.s will appear.
   c. If the next text output is "Exercise01.s" - 0 Error(s), 0 Warning(s), as in Figure 15, assembly was successful. Otherwise, any errors must be corrected before assembly can complete. Successful assembly produces an object file in the project's Objects folder and a listing file in the project's Listings folder.

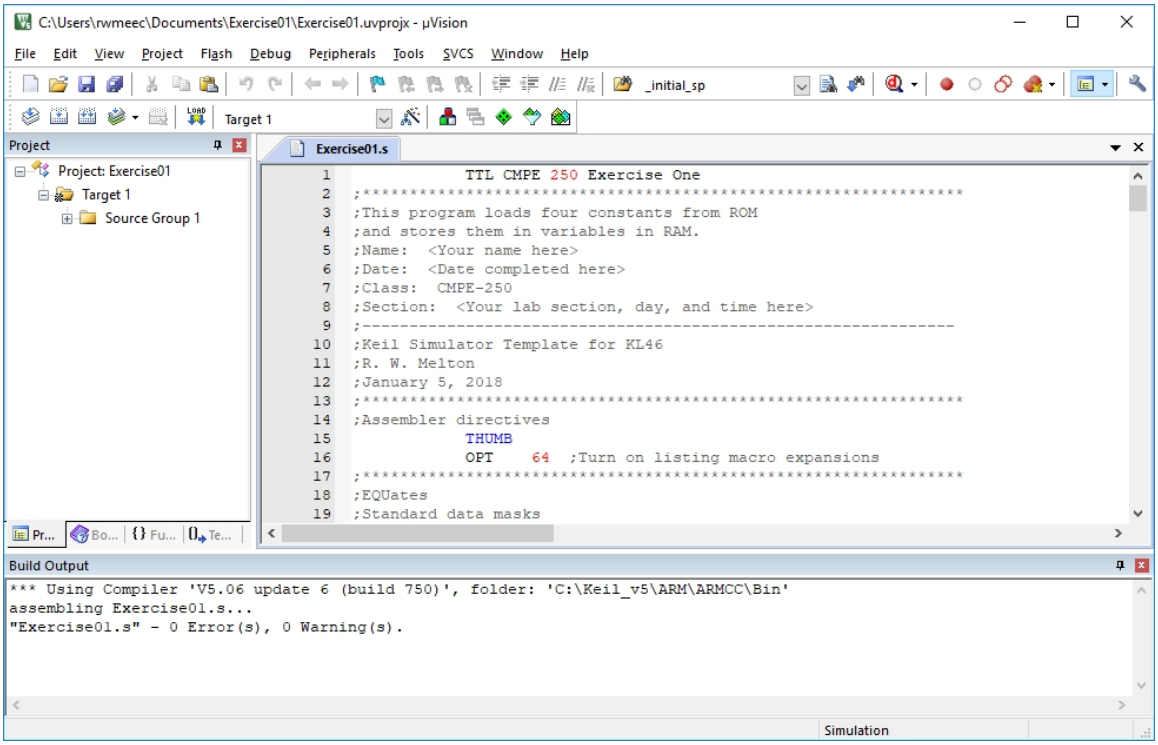


Figure 15. Project after Assembly

7. View the listing file: exercise01.lst.
   a. From the µVision IDE main menu, select File→Open to launch the Open File window, as shown in Figure 16.
   b. Double-click the Listings folder to open it.
   c. Use the pull-down menu to the right of the File name field to select Listing (*.lst; *.m*; *.cod).

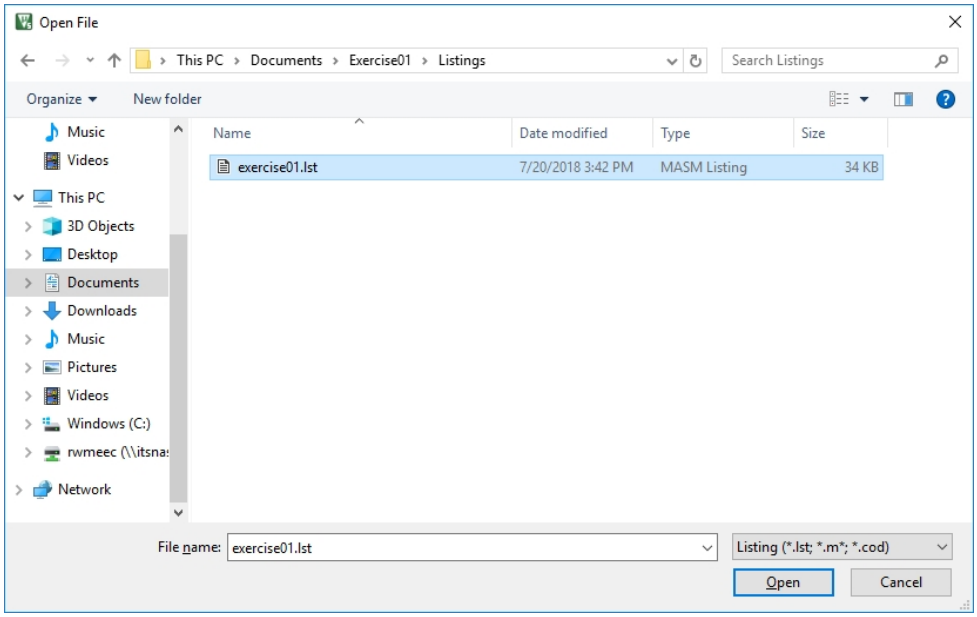


Figure 16. Opening Listing File

d. Select exercise01.lst, and then click Open. (Note: if Windows does not show file extensions, look for exercise01 with Type MASM Listing.)

Understanding Assembler Listing Files

8. Examine the listing file and answer the following questions. Note that listing file has four columns: 1) the assembly source code line number, 2) the corresponding memory offset within an AREA, 3) the memory contents in hex for that address, and 4) the original assembly source code.

Q1. What are the four words (in hexadecimal) assembled for `ConstData` by the assembler directive `DCD`?

Q2. In `ConstData`, why does `DCD` produce different word values for `0x00000010` and `10`?

Q3. How much storage and what memory contents are shown for `VarData` in the listing file?

Q4. At what offset within the `MyData RAM` AREA does `VarData` begin?

Q5. What opcode (in hexadecimal) is generated for the `NOP` instruction? _____

Q6. How many bytes of machine code are generated for each assembly instruction?

Q7. Do assembly language program labels generate any code bytes? Explain why.

Q8. According to the symbol table at the end of the listing file, how many times is `__initial_sp` used (not counting its definition).

9. Show your answers to your lab instructor for signature.

Building Projects Code Using Keil MDK-ARM

10. View the project's target options for mapping the project's AREAs to memory.
    a. On the second toolbar row, click the Target Options icon (or select menu Project→Options for Target 'Target1' …) to launch the Options for Target 'Target1' window, as shown in Figure .

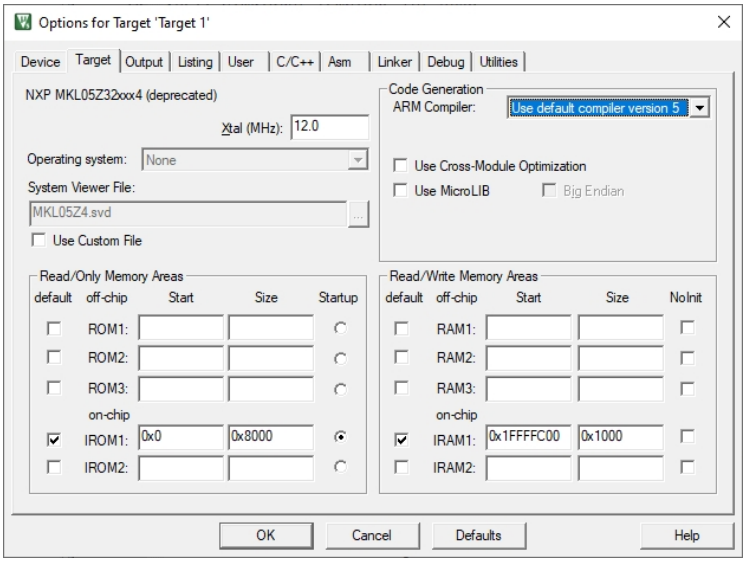


Figure 17. Viewing Options for Target Microcontroller

    b. If the Target tab is not selected, click on the Target tab along the top of the window to get the options shown in Figure .
    c. Verify that the ARM Compiler option in the Code Generation section of the Target tab has Use default compiler version 5 selected.
    d. Answer the following questions about the target memory address options.

       Q9. Where in memory would you expect code and constants? _____

       Q10. Where in memory would you expect variables? _____

    e. Click on the Linker tab along the top of the window to get the options shown in Figure 18.

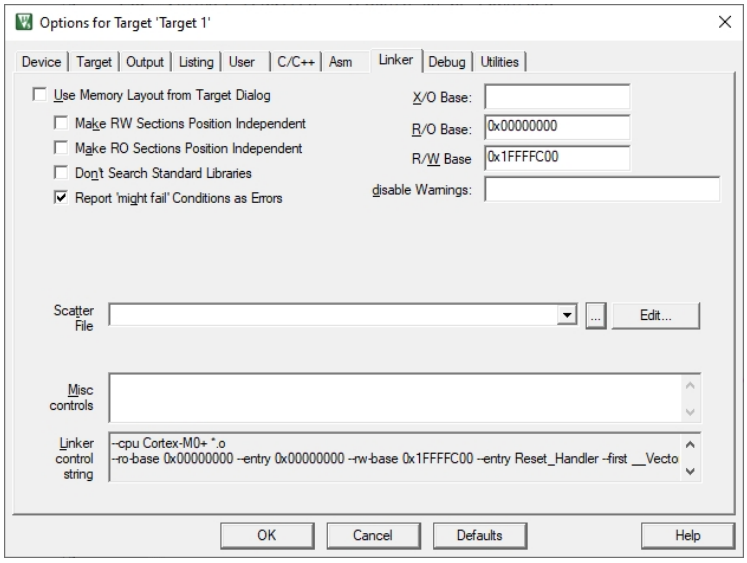


Figure 18. Setting Options for Linker

f. Since the project is a bare metal assembly project that uses no C startup code, the linker options need to be set not to look for C startup objects specified in the default project scatter file.
   i. To be able to edit the scatter file specified for the linker, uncheck the first Linker tab option, Use Memory Layout from Target Dialog.
   ii. To specify not to use the default scatter file, select all of the text in the Scatter File field, and delete it.

g. Click the OK to save the options and close the options window.

11. Build Exercise01.

Building a project consists of linking all of the assembled code. In the previous assembly step, the assembler produced an object file. Now that object file must be linked using linker parameters (specified in the project's target options) to produce an executable file. The executable file consists of memory contents, (i.e., images), to be loaded into specific areas of memory, (as specified in the project's target options).

a. On the second toolbar row, click the Build icon (or select menu Project→Build target) to build the project.
   ☞ If the source file had not already been assembled, the build operation would first assemble the source. Thus, it is not necessary to click the Translate icon before building the project.

b. In the Build Output pane at the bottom of the window, the text Build target 'Target 1' will appear followed on the next line by linking . . . .

c. If, as in Figure 19, the next to last line of text in the Build Output pane shows ".\Objects\Exercise01.axf" – 0 Error(s), 0 Warning(s), the build was successful. Otherwise, any errors must be corrected before an executable file is produced.
   ☞ If the listing file is still open in an editor tab and if building the project re-assembled the code to produce a new listing file, there will be a pop-up window asking, "File has been changed outside the editor, reload?" If so, click the Yes button to load the editor window with the updated listing file contents.

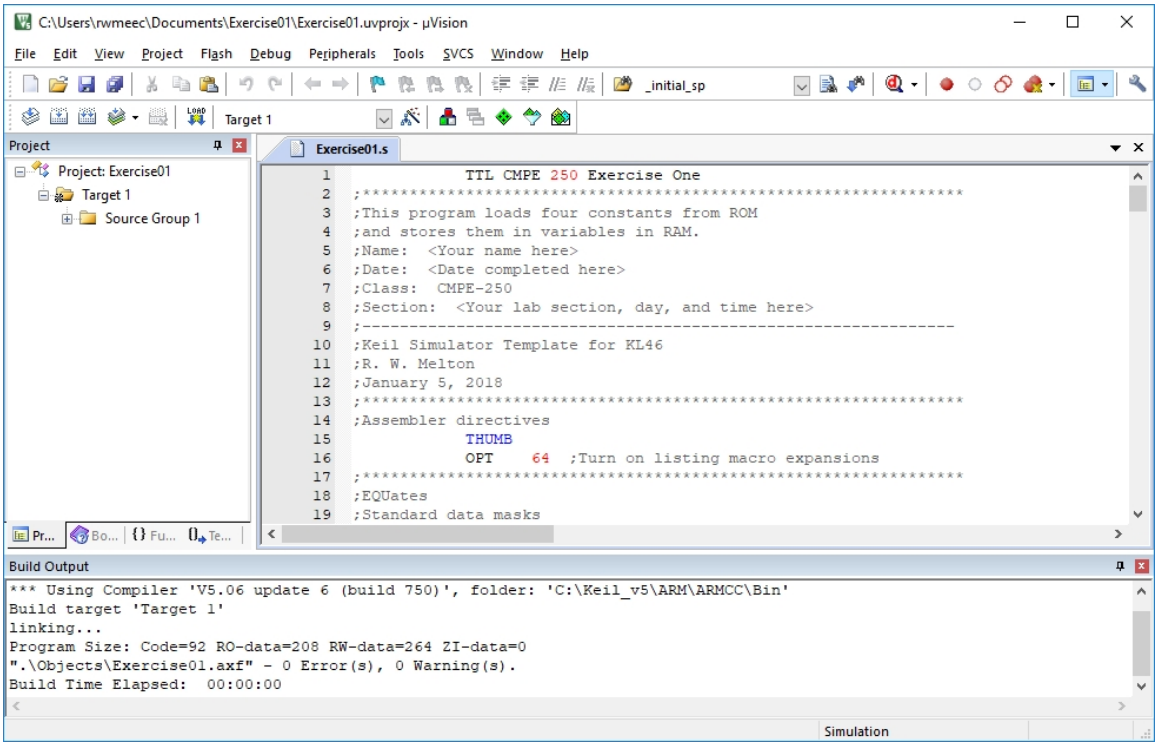


Figure 19. Project after Build

12. View the map file: Exercise01.map.
    a. From the µVision IDE main menu, select File→Open to launch the Open File window, as shown in Figure 20.
    b. Select Exercise01.map, and then click Open. (Note: if Windows does not show file extensions, look for Exercise01 with Type Linker Address Map.)

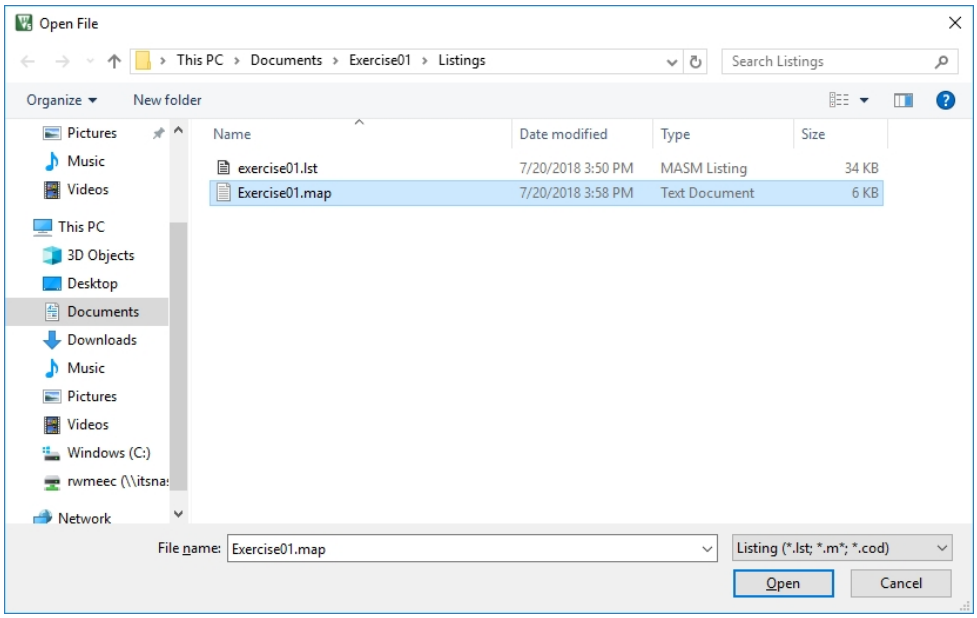


Figure 20. Opening Map File

13. Examine the linker map file and answer the following questions.

Image Symbol Table

Q11. Where do the variables `VarData` start in memory? _____

Q12. How large (in bytes) is the `MyCode` area.? _____

Q13. What number base is used for Size of the symbols? _____

Q14. How much of RAM is used? _____

Memory Map of the image

Q15. What is the starting address of the program? _____

Q16. What number base is used for Size of the regions? _____

Q17. How many bytes are used for the constants in `MyConst`? _____

Q18. What is the starting address of RAM? _____

14. Show your answers to your lab instructor for signature.

Simulation Using µVision Debug

15. Select simulation for debug.
    a. On the second toolbar row, click the Target Options icon (or select menu Project→Options for Target 'Target1' …) to launch the Options for Target 'Target1' window, as shown in Figure 21.
    b. Click on the Debug tab along the top of the window to get the options shown in Figure 21.
    c. At the top left, under the tabs, select Use Simulator.
    d. Click the OK button to close the options window and return to the µVision IDE.

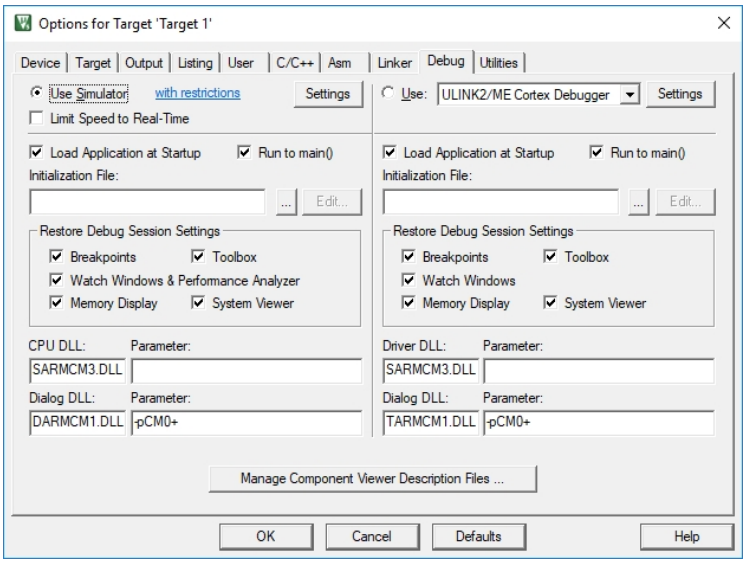


Figure 21. Setting Options for µVision Debug Simulator

16. Start a debug session.

    To start a debug session, on the top toolbar row toward the right of the window, click the Start/Stop Debug Session icon (or select menu Debug→ Start/Stop Debug Session), as shown in Figure 23.

    ☞ Note: Figure 23 also indicates the most commonly used toolbar icons for the µVision IDE debug simulator.

    ☞ Note: Whereas the lab workstations have a professional license for the µVision software, if you download the software for your personal use, you will not have a license, and you will get a pop-up window (shown in Figure 22) at this point that warns the debugger will run in evaluation mode with a code size limit of 32 KB. Just click the OK button in the pop-up window, and then the debug session will start.

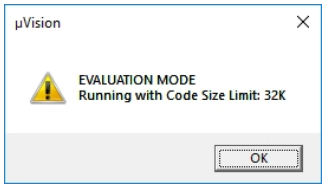


Figure 22. Pop-up window for evaluation version of debugger

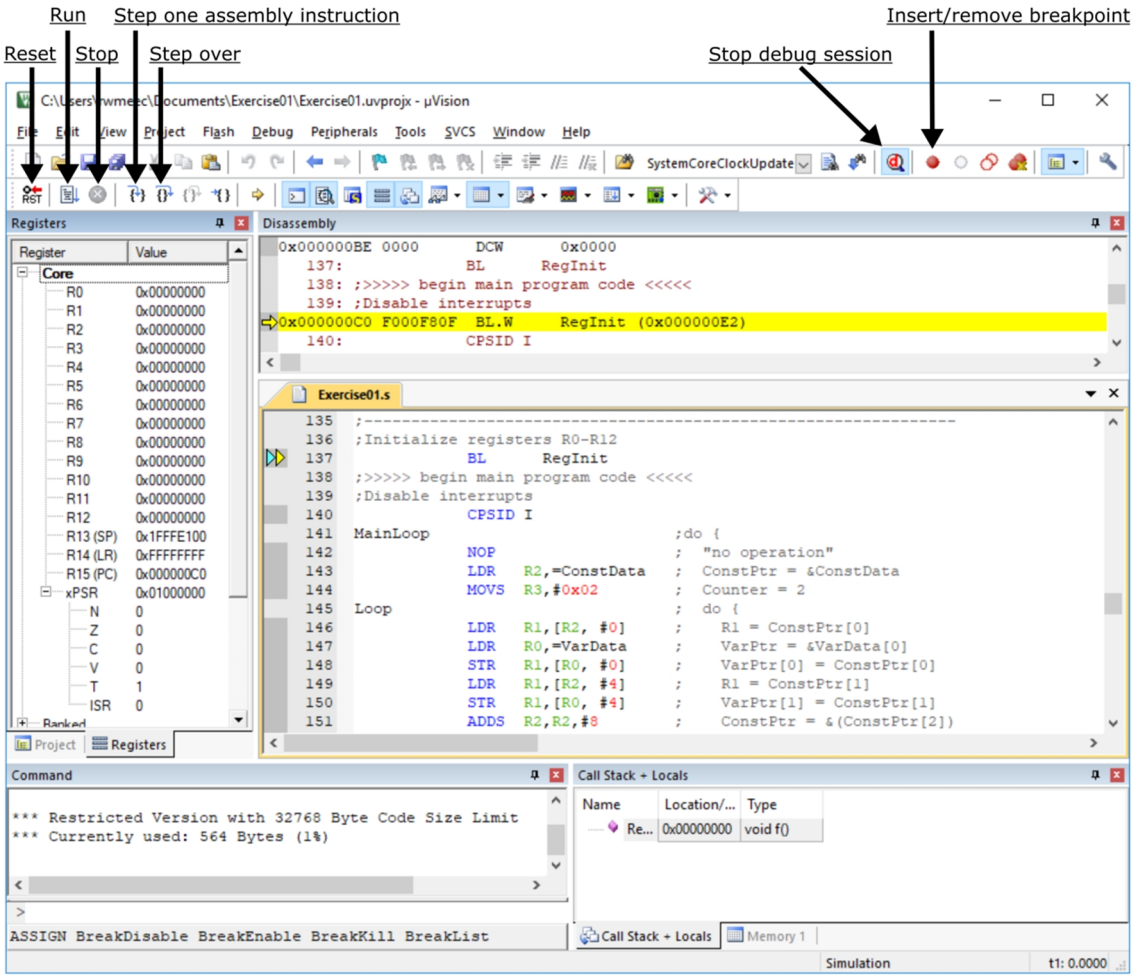


Figure 23. µVision Debug Simulator

17. Explore the debug simulator.
    a. The Register pane at the top left shows the current contents/status of all registers. Expand ⊞ xPSR to see the condition codes: N, Z, C, and V.
    b. The Disassembly pane at the top right shows the machine instructions in code memory.
        ☞ If the programmer used any pseudoinstructions in the source code, the corresponding machine instructions(s) will appear in the Disassembly pane.
    c. The source pane at the center right shows the programmer's original assembly code.
    d. The data pane at the bottom right can be toggled between Call Stack + Locals and Memory 1. For now select Memory 1 and enter *0x1FFFFD00* in the Address field to view the `VarData` variables.
        ☞ Right clicking in this window gives a context menu, which can be used to change how the contents are displayed.
    e. The Command pane is at the bottom left

18. Run the program in the simulator.
    a. In the source pane, position the mouse pointer over the first `NOP` instruction, right click, and select Insert/Remove Breakpoint.
        ☞ Setting a breakpoint at an instruction will cause program execution to stop *before* executing this instruction.
    b. On the second toolbar row, either click the Run button, select menu Debug→Run, or press F5.
        ☞ In the Register pane, any registers that have changed value because of the code executed, (i.e., run), are highlighted.
    c. On the second toolbar row, either click the Step button, select menu Debug→Step, or press F11. Look at what is highlighted in the Register pane to see the effects of executing the instruction.
        ☞ In the Register pane, any registers that have changed value because of the instruction executed, (i.e., stepped), are highlighted.
19. Start a new debug session and track all changes to the registers in Table **1** on p. 26.
    a. On the top toolbar row toward the right of the window, click the Start/Stop Debug Session icon to stop the debug session and return to the µVision IDE.
    b. From the µVision IDE click the Start/Stop Debug Session icon to start a new debug session.
    c. Run to the breakpoint set previously in step 18.a. Record in the table the resulting changes made to the registers listed in Table 1 on p. 26. Note that xPSR will have to be expanded to show N, Z, C, and V.
        ☞ Note that the *register contents are always written using 8 hex digits* to show the value of each of the register's 32 bits.
        ☞ If an instruction does not change the value of a register, leave its cell blank in the table rather than rewriting the same value.
    c. Step through each subsequent program instructions listed in Table 1 on p. 26. For each instruction, record in the table the changes made by the instruction to the registers listed in Table 1. Note that xPSR will have to be expanded to show N, Z, C, and V.
        ☞ In the debugger, *the yellow arrow* to the left of the source code *indicates the next instruction to be executed*. To see the results of executing an instruction, Step through it so that the yellow arrow points to the following instruction.
        ☞ If an instruction does not change the value of a register, leave its cell blank in the table rather than rewriting the same value.
20. Show your completed table to your lab instructor for signature.
21. (Optional) Create and simulate a C project for ARM Cortex-M0+ in Keil MDK-ARM. Follow the instructions in the Further Investigation section, which begins on p. 23.

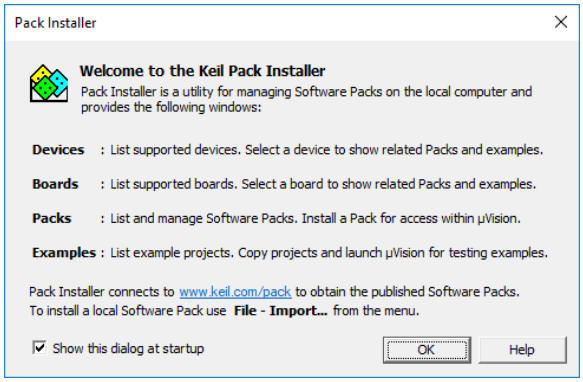
Report

(No report is due for this exercise.)

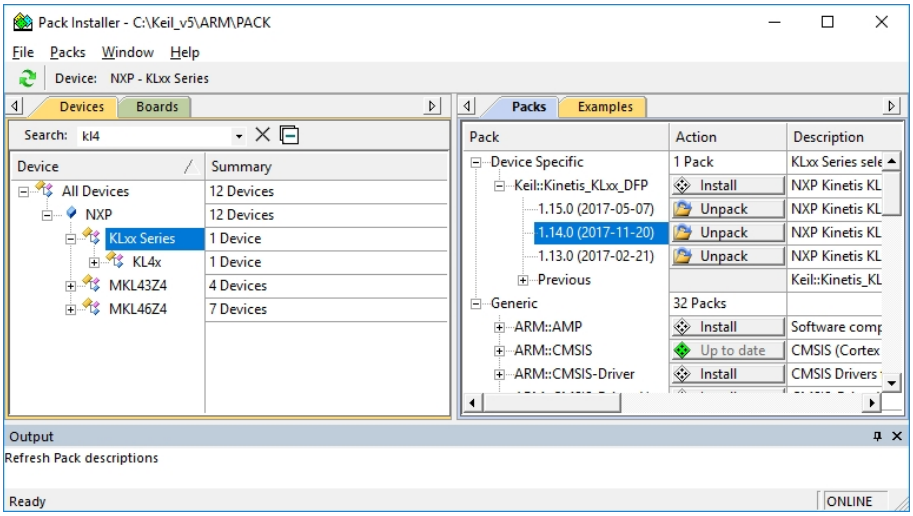
Installing Keil Software Packs

Note: Although this procedure references KL46 devices, the device pack installed by following this procedure is the correct device pack for KL05, KL25, and KL46 devices used in this course during various semesters.

1. At the right of the second toolbar row of Keil µVision IDE, click the Pack Installer icon to launch the Pack Installer window.
2. If the Pack Installer dialog shown below appears, click the OK button to dismiss it.



3. With the Devices tab selected in the left half pane of the Pack Installer window shown below, type *KL4* in the Search field just below the tabs.



4. To find the device packs for the NXP MKL05Z32VFM4 microcontroller on the NXP FRDM-KL05Z Freedom development platform, in the left pane below the Search field, successively click ⊞ next to each of the following categories to select KLxx Series, as shown in Figure 4.

   All Devices → NXP → KLxx Series

☞ If KLxx Series does not appear, change the search box to *KL43*.

5. With Packs selected in the right half pane of the Pack Installer window, successively click ⊞ next to each of the following categories to select Kinetis_KLxx_DFP version 1.14.0.

   Device Specific → Keil::Kinetis_KLxx_DFP → 1.14.0 (2017-11-20)

6. To the right of 1.14.0 (2017-11-20), click on Install or Unpack to install the software pack.

☞ If version 1.15.0 (2017-05-07) shows installed, click on Remove.

💣 Although the KL46 devices are included in the newer MKL46Z4 device pack, that pack will not work for later assignments. Both packs can be installed if desired, but all class projects should use Keil::Kinetis_KLxx_DFP 1.14.0 (2017-11-20).

7. If the Action field for the following packs shows Update or Install, click to install the latest version: 1) ARM::CMSIS; 2) Keil::ARM_Compiler.

💣 Once Keil::Kinetis_KLxx_DFP 1.14.0 (2017-11-20), do not install any subsequent updates to Keil::Kinetis_KLxx_DFP, as the updates will inhibit selecting version 1.14.0 needed for projects in this course. Ignore that it shows as Deprecated.

Further Investigation: C Project for Cortex-M0+ Simulation

1. Create a new project.
   Use the Keil µVision IDE main menu to select Project→New µVision Project … and launch the Create New Project window.
   a. Browse to a location where you want to keep project files.
      ☞ Suggestion: somewhere in your Documents folder.
      💣 Caution: avoid creating multiple projects in the same folder.
   b. Use New Folder to create a folder for your project, and name it *Exercise01_C*.
   c. Double-click the new *Exercise01_C* folder to open it.
   d. In the File name field at the bottom, type *Exercise01_C*.
   e. Press the Save button to create your project. Keil µVision IDE will then launch the Select Device for Target 'Target 1' … window.
   f. To set the target as the ARM Cortex-M0+, which is the processor core on the NXP FRDM-KL46Z Freedom development platform's KL46, in the left pane below the Search field, successively click ⊞ next to each of the following categories to select ARMCM0P.
      ARM → ARM Cortex M0 plus → ARMCM0P
      (Alternatively, type *ARMCM0P* in the Search field.)
   g. Click the OK button to finish creating the Exercise01 project.
   h. When Keil µVision IDE launches the Manage Run-Time Environment window, add the following Keil ARM-MDK software components to the project, as shown in Figure 24.
      i. In the Software Components column, click ⊞ next to CMSIS to show its components, and select Core.
      ii. Click ⊞ next to Device, and select Startup.
      iii. Click the OK button to add the components to the project.

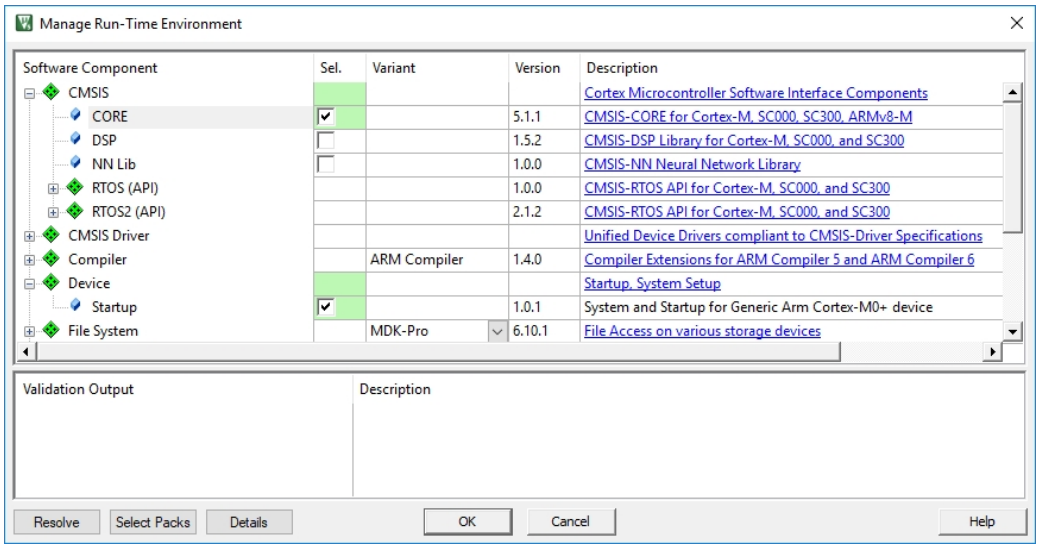


Figure 24. Manage Run-Time Environment for C Simulation

2. Add a C source file to the project.
   a. In the Project pane along the left side of the µVision IDE window, click ⊞ next to Target to show Source Group 1.
   b. Right-click on Source Group 1, and in the resulting drop-down menu select Add New Item to Group 'Source Group 1' … to open the Add New Item to Group 'Source Group 1' window.
   c. In the top left pane, click C File (.c) to change the Type field to C File (.c).
   d. In the Name field, type *Exercise01*.
   e. Click the Add button to create Exercise01.c. Exercise01.c is then opened in the editor pane on the right side of the µVision IDE window.
3. Enter the C source code shown in Figure 25 and click the Save icon (or select menu File→Save) to save the program.

```
/*****************************************************************************/
/* CMPE-250 Lab Exercise One                                                 */
/* This program loads four constants from ROM                                */
/* and stores them in variables in RAM.                                      */
/* Name:  <Your name here>                                                   */
/* Date:  <Date completed here>                                              */
/* Class:  CMPE-250                                                          */
/* Section:  <Your lab section, day, and time here>                          */
/*****************************************************************************/

#define TRUE (1)

const int ConstData[4] = {0x0000000A, 0x0000000B, 0x00000010, 10};
int VarData[2];

int main (void) {
  register const int *ConstPtr;
  register int *VarPtr;
  register int Counter,
               Temp;

  do {
    __asm ("CPSID I");
    __asm ("NOP");

    ConstPtr = ConstData;
    Counter = 2;

    do {
      Temp = ConstPtr [0];
      VarPtr = VarData;
      VarPtr[0] = Temp;
      Temp = ConstPtr [1];
      VarPtr[1] = Temp;
      ConstPtr = &(ConstPtr[2]);
      Counter--;
    } while (Counter);

    __asm ("NOP");
  } while (TRUE);

  return (0);
}
```

Figure 25. Exercise One C Code

4. Edit target options for the project.
   a. On the second toolbar row, click the Target Options icon (or select menu Project→Options for Target 'Target1' …) to launch the Options for Target 'Target1' window, as shown in Figure 26.

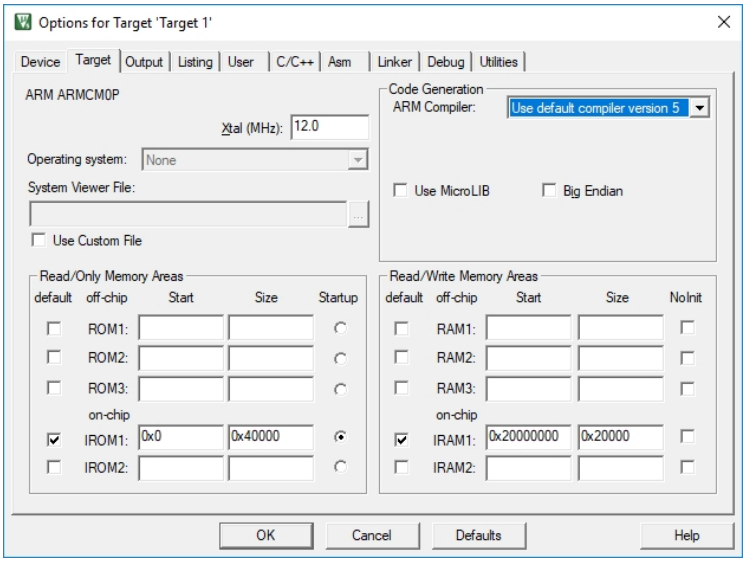


Figure 26. Setting Options for Target Compiler

   b. Click on the Target tab along the top of the window to get the options shown in Figure 26.
   c. At the top right, under the tabs, in the Code Generation options, use the pull-down dialog to the right of ARM Compiler to select Use default compiler version 5, as shown in Figure 26.
   d. Click the OK to save the options and close the options window.
4. Compile and link the project by clicking the Build icon (or selecting menu Project→Build target) to build the project.
5. Select simulation for debug, (as in step 15 of the assembly language procedure).
6. Click the Start/Stop Debug Session icon (or select menu Debug→ Start/Stop Debug Session), to start the debugger.
7. Use the debugger's step, breakpoint, and run functionality to execute the C code. Compare and contrast the behavior with this C project versus your earlier assembly project.

This tutorial was developed for Keil MDK-Arm Version 5.31.0.0 with Keil device pack Kinetis_KLxx_DFP.1.14.0.

Table 1. Register contents (in hex) *after* each program instruction

	Registers								
After **Instruction**	**R0**	**R1**	**R2**	**R3**	**PC R15**	**xPSR**			
						N	**Z**	**C**	**V**
Initial debug state									
After run to breakpoint									
`NOP`									
`LDR R2,=ConstData`									
`MOVS R3,#0x02`									
`LDR R1,[R2,#0]`									
`LDR R0,=VarData`									
`STR R1,[R0,#0]`									
`LDR R1,[R2,#4]`									
`STR R1,[R0,#4]`									
`ADDS R2,R2,#8`									
`SUBS R3,R3,#1`									
`BNE Loop`									
`LDR R1,[R2,#0]`									
`LDR R0,=VarData`									
`STR R1,[R0,#0]`									
`LDR R1,[R2,#4]`									
`STR R1,[R0,#4]`									
`ADDS R2,R2,#8`									
`SUBS R3,R3,#1`									
`BNE Loop`									
`NOP`									
`B MainLoop`									
`NOP`									