



Microprocessor Systems

Res. Assist. M. Alpaslan TAVUKCU

tavukcu22@itu.edu.tr

2024



ARM CORTEX M0-PLUS

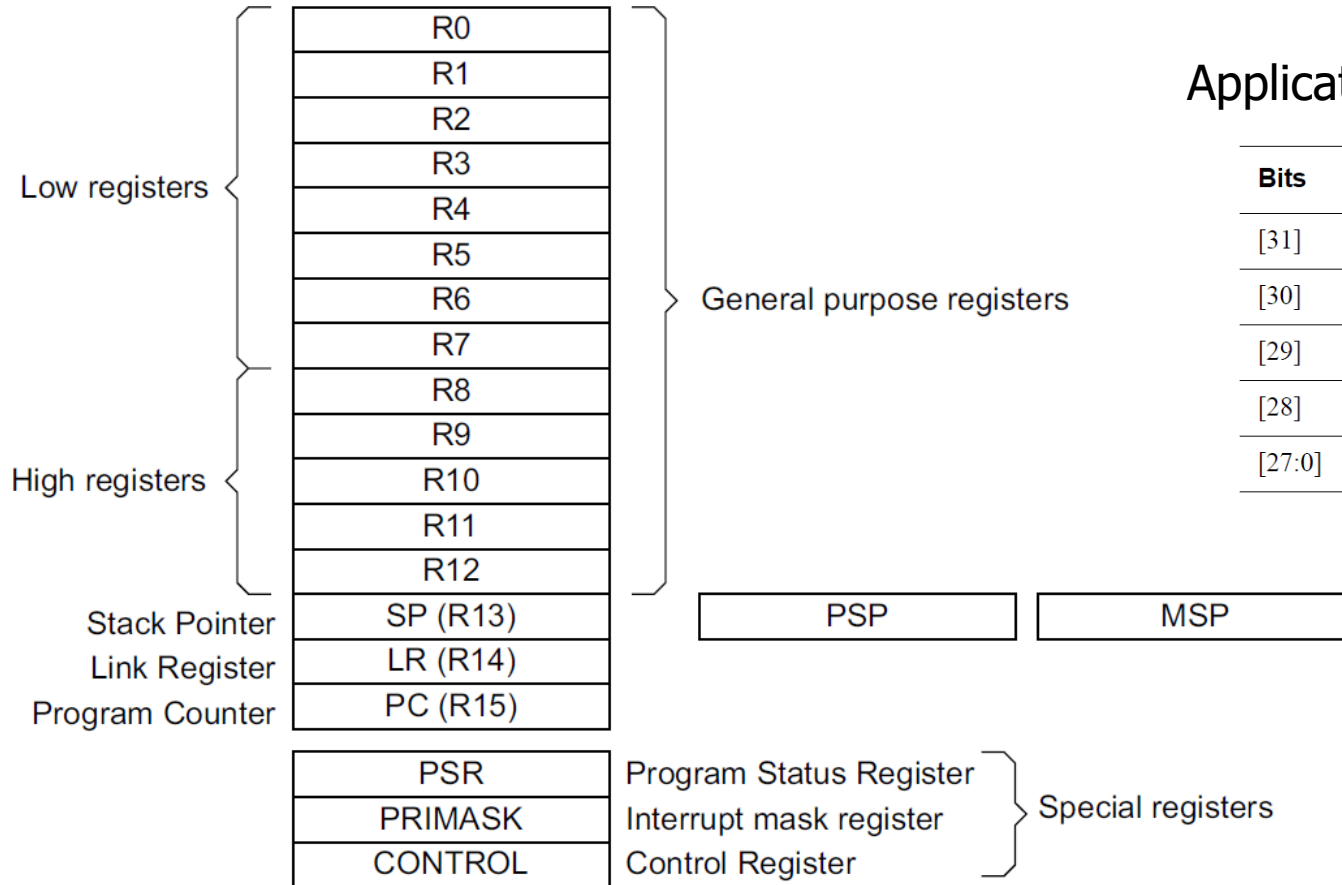


Introduction

- Arm Cortex-M0-plus
 - Armv6-M Architecture
 - 2-stages Pipeline
 - **Thumb** and Thumb-2 ISA Support
 - Non-maskable Interrupt (NMI) + 1 to 32 physical interrupts
- Keil μ Vision IDE
 - Programming Languages: C/C++ and Assembly
 - Onboard Arm Simulator
 - Debugging, Logic Analyzer, Source Browser

Core Registers

The processor core registers are:



Memory

Address range	Memory region	Memory type ^a	XN ^a	Description
0x00000000-0x1FFFFFFF	Code	Normal	-	Executable region for program code. You can also put data here.
0x20000000-0x3FFFFFFF	SRAM	Normal	-	Executable region for data. You can also put code here.
0x40000000-0x5FFFFFFF	Peripheral	Device	XN	External device memory.
0x60000000-0x9FFFFFFF	External RAM	Normal	-	Executable region for data.
0xA0000000-0xDFFFFFFF	External device	Device	XN	External device memory.
0xE0000000-0xE0FFFFFF	Private Peripheral Bus	Strongly-ordered	XN	This region includes the NVIC, System timer, and System Control Block. Only word accesses can be used in this region.
0xE0100000-0xFFFFFFFF	Device	Device	XN	Implementation-specific.

Memory

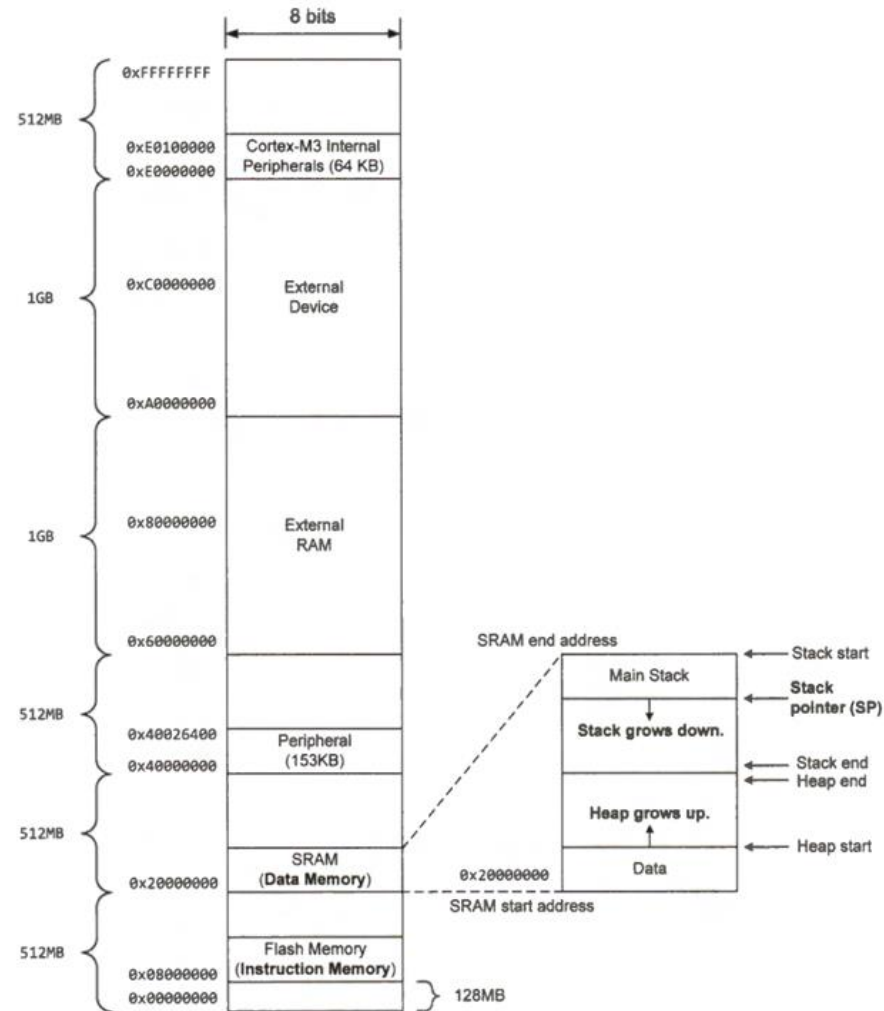


Figure 1-9. Example memory map of a 4GB memory space. The instruction memory, the data memory, and all peripherals share the same memory address space. The memory map is fixed and contains unused region.

Instruction Set

Operation	Description	Assembler	Cycles
Move	8-bit immediate	MOVS Rd, #<imm>	1
	Lo to Lo	MOVS Rd, Rm	1
	Any to Any	MOV Rd, Rm	1
	Any to PC	MOV PC, Rm	3
Add	3-bit immediate	ADDS Rd, Rn, #<imm>	1
	All registers Lo	ADDS Rd, Rn, Rm	1
	Any to Any	ADD Rd, Rd, Rm	1
	Any to PC	ADD PC, PC, Rm	3
	8-bit immediate	ADDS Rd, Rd, #<imm>	1
	With carry	ADCS Rd, Rd, Rm	1
	Immediate to SP	ADD SP, SP, #<imm>	1
	Form address from SP	ADD Rd, SP, #<imm>	1
	Form address from PC	ADR Rd, <label>	1
Subtract	Lo and Lo	SUBS Rd, Rn, Rm	1
	3-bit immediate	SUBS Rd, Rn, #<imm>	1
	8-bit immediate	SUBS Rd, Rd, #<imm>	1
	With carry	SBCS Rd, Rd, Rm	1
	Negate	RSBS Rd, Rn, #0	1
Multiply	Multiply	MULS Rd, Rm, Rd	1 or 32 ^a

Operation	Description	Assembler	Cycles
Compare	Compare	CMP Rn, Rm	1
	Negative	CMN Rn, Rm	1
	Immediate	CMP Rn, #<imm>	1
Logical	AND	ANDS Rd, Rd, Rm	1
	Exclusive OR	EORS Rd, Rd, Rm	1
	OR	ORRS Rd, Rd, Rm	1
	Bit clear	BICS Rd, Rd, Rm	1
	Move NOT	MVNS Rd, Rm	1
	AND test	TST Rn, Rm	1
Shift	Logical shift left by immediate	LSLS Rd, Rm, #<shift>	1
	Logical shift left by register	LSLS Rd, Rd, Rs	1
	Logical shift right by immediate	LSRS Rd, Rm, #<shift>	1
	Logical shift right by register	LSRS Rd, Rd, Rs	1
	Arithmetic shift right	ASRS Rd, Rm, #<shift>	1
	Arithmetic shift right by register	ASRS Rd, Rd, Rs	1
Rotate	Rotate right by register	RORS Rd, Rd, Rs	1

Instruction Set

Operation	Description	Assembler	Cycles
Load	Word, immediate offset	LDR Rd, [Rn, #<imm>]	2
	Halfword, immediate offset	LDRH Rd, [Rn, #<imm>]	2
	Byte, immediate offset	LDRB Rd, [Rn, #<imm>]	2
	Word, register offset	LDR Rd, [Rn, Rm]	2
	Halfword, register offset	LDRH Rd, [Rn, Rm]	2
	Signed halfword, register offset	LDRSH Rd, [Rn, Rm]	2
	Signed byte, register offset	LDRSB Rd, [Rn, Rm]	2
	PC-relative	LDR Rd, <label>	2
	SP-relative	LDR Rd, [SP, #<imm>]	2
	Multiple, excluding base	LDM Rn!, {<loreglist>}	1+N ^b
	Multiple, including base	LDM Rn, {<loreglist>}	1+N ^b
Store	Word, immediate offset	STR Rd, [Rn, #<imm>]	2
	Halfword, immediate offset	STRH Rd, [Rn, #<imm>]	2
	Byte, immediate offset	STRB Rd, [Rn, #<imm>]	2
	Word, register offset	STR Rd, [Rn, Rm]	2
	Halfword, register offset	STRH Rd, [Rn, Rm]	2
	Byte, register offset	STRB Rd, [Rn, Rm]	2
	SP-relative	STR Rd, [SP, #<imm>]	2
	Multiple	STM Rn!, {<loreglist>}	1+N ^b

Operation	Description	Assembler	Cycles
Push	Push	PUSH {<loreglist>}	1+N ^b
	Push with link register	PUSH {<loreglist>, LR}	1+N ^b
Pop	Pop	POP {<loreglist>}	1+N ^b
	Pop and return	POP {<loreglist>, PC}	4+N ^c
Branch	Conditional	B<cc> <label>	1 or 3 ^d
	Unconditional	B <label>	3
	With link	BL <label>	4
	With exchange	BX Rm	3
	With link and exchange	BLX Rm	3
Extend	Signed halfword to word	SXTH Rd, Rm	1
	Signed byte to word	SXTB Rd, Rm	1
	Unsigned byte	UXTB Rd, Rm	1
Reverse	Bytes in word	REV Rd, Rm	1
	Bytes in both halfwords	REV16 Rd, Rm	1
	Signed bottom half word	REVSH Rd, Rm	1
State change	Supervisor Call	SVC #<imm>	- ^e
	Disable interrupts	CPSID i	1
	Enable interrupts	CPSIE i	1
	Read special register	MRS Rd, <specreg>	4
	Write special register	MSR <specreg>, Rn	4
	Breakpoint	BKPT #<imm>	- ^e

Instruction Set

Operation	Description	Assembler	Cycles
Hint	Send event	SEV	1
	Wait for event	WFE	2f
	Wait for interrupt	WFI	2f
	Yield	YIELD ^g	1
	No operation	NOP	1
Barriers	Instruction synchronization	ISB	4
	Data memory	DMB	4
	Data synchronization	DSB	4

IMPORTANT!

Literals can be expressed as:

- Decimal numbers, for example `123`.
- Hexadecimal numbers, for example `0x7B`.
- Numbers in any base from `2` to `9`, for example `5_204` is a number in base `5`.
- Floating point numbers, for example `123.4`.
- Boolean values `{TRUE}` or `{FALSE}`.
- Single character values enclosed by single quotes, for example `'w'`.
- Strings enclosed in double quotes, for example `"This is a string"`.

BRANCH CONDITION CODES

cond	Mnemonic extension	Meaning	Condition flags
0000	EQ	Equal	$Z == 1$
0001	NE	Not equal	$Z == 0$
0010	CS ^a	Carry set	$C == 1$
0011	CC ^b	Carry clear	$C == 0$
0100	MI	Minus, negative	$N == 1$
0101	PL	Plus, positive or zero	$N == 0$
0110	VS	Overflow	$V == 1$
0111	VC	No overflow	$V == 0$
1000	HI	Unsigned higher	$C == 1$ and $Z == 0$
1001	LS	Unsigned lower or same	$C == 0$ or $Z == 1$
1010	GE	Signed greater than or equal	$N == V$
1011	LT	Signed less than	$N != V$
1100	GT	Signed greater than	$Z == 0$ and $N == V$
1101	LE	Signed less than or equal	$Z == 1$ or $N != V$
1110 ^c	None (AL) ^d	Always (unconditional)	Any

a. HS (unsigned higher or same) is a synonym for CS.

b. LO (unsigned lower) is a synonym for CC.

c. This value is never encoded in any ARMv6-M Thumb instruction.

d. AL is an optional mnemonic extension for always.

EX. 01: MOVE & ADD

```
AREA example, CODE, READONLY
    ENTRY
    ALIGN
__main FUNCTION
    EXPORT __main
    MOVS    R0, #10                ; Set up parameters
    MOVS    R1, #3
    ADD     R0, R0, R1             ; R0 = R0 + R1

stop    B    stop                 ; Branch stop

    ENDFUNC
    END
```

EX. 02: SUBROUTINES

```
AREA example, CODE, READONLY
    ENTRY
    ALIGN
__main FUNCTION
    EXPORT __main
    MOVS    R0, #10                ; Set up parameters
    MOVS    R1, #3
    BL      doadd                 ; Call subroutine

stop    B      stop                ; Branch stop

doadd   ADD     R0, R0, R1          ; Subroutine code
        BX     LR                 ; Return from subroutine

        ENDFUNC                  ; Finish function
        END                      ; Finish assembly file
```



EX. 03: GCD

```
int gcd(int a, int b)
{
    while (a != b)
    {
        if (a > b)
            a = a - b;
        else
            b = b - a;
    }
    return a;
}
```

EX. 03: GCD

```
AREA example, CODE, READONLY      ; Declare new area
    ENTRY                          ; Declare as entry point
    ALIGN                          ; Ensures that __main addresses the following instruction

__main FUNCTION                    ; Enable Debug
    EXPORT __main                  ; Make __main as global to access from startup file
    MOVS    R0, #48                ; Set up parameters (example values)
    MOVS    R1, #18
    BL      doGCD                  ; Call GCD subroutine

stop    B      stop                ; Branch stop

doGCD   CMP     R0, R1              ; Compare a and b (R0 and R1)
        BEQ     endGCD             ; If they are equal, branch to endGCD
        BGT     subtractA          ; If a > b, branch to subtractA
        B       subtractB          ; Else, branch to subtractB

subtractA SUBS    R0, R0, R1         ; a = a - b
        B       doGCD             ; Repeat the loop

subtractB SUBS    R1, R1, R0         ; b = b - a
        B       doGCD             ; Repeat the loop

endGCD   MOV     R0, R0             ; Move the result (GCD) into R0 for returning
        BX      LR                ; Return from subroutine

        ENDFUNC                   ; Finish function
    END                           ; Finish assembly file
```

Directives Reference

- Directives are instructions used by the **assembler** to help automate the assembly process and to improve program readability.
- Important directive for us:
- **AREA:** instructs the assembler to assemble a new code or data section.
 - Example Usage:
 - **AREA Example, CODE, READONLY**
 - Example-> name of section
 - Code-> Contains machine instructions.
 - READONLY-> Indicates that this section must not be written to.

Directives Reference

- **ALIGN:** aligns the current location to a specified boundary by padding with zeros or NOP instructions.

- Example usage:

```
ALIGN                ; ensures that subroutine1 addresses
subroutine1          ; the following instruction.
MOV r5, #0x5
```

- **DCB:** allocates one or more bytes of memory, and defines the initial runtime contents of the memory.
- **DCW - DCWU:** works as DCB for 2 bytes data.
- **DCD and DCDU:** works as DCB for 4 bytes data.
- **DCQ - DCQU:** works as DCB for 8 bytes data.

```
C_string    DCB    "C_string",0
data        DCW    -225,2*number    ; number must already be defined
data1       DCD    1,5,20
data        DCQ    -225,2_101
```

Directives Reference

- **EQU:** gives a symbolic name to a numeric constant, a register-relative value or a PC-relative value.

```
value1    EQU 2
value2    EQU 0x2215, CODE32
value3    EQU 0x214456, DATA
```

- **FUNCTION or PROC:** marks the start of a function. PROC is a synonym for FUNCTION. They enable the debug.
- **ENDFUNC or ENDP:** marks the end of a function. ENDP is a synonym for ENDFUNC.
- **END:** informs the assembler that it has reached the end of a source file.
- **EXPORT or GLOBAL:** declares a symbol that can be used by the linker to resolve symbol references in separate object and library files.



Directives Reference

- **IMPORT and EXTERN:** provide the assembler with a name that is not defined in the current assembly.
- **ENTRY:** declares an entry point to a program.
- **THUMB:** instructs the assembler to interpret subsequent instructions as Thumb instructions, using the UAL syntax.



LDR

- LDR Rd, =const
- LDR Rd, =label

The LDR pseudo-instruction generates the most efficient single instruction for a specific constant:

- If the constant can be constructed with a single MOV instruction, the assembler generates the appropriate instruction.
- If the constant cannot be constructed with a single MOV instruction, the assembler: places the value in a literal pool (a portion of memory embedded in the code to hold constant values) generates an LDR instruction with a program-relative address that reads the constant from the literal pool.



EX: LDR

```
AREA example, CODE, READONLY      ; Declare new area
    ENTRY                          ; Declare as entry point
    ALIGN                          ; Ensures __main addresses the following instruction

__main    FUNCTION                  ; Enable Debug
    EXPORT __main                   ; Make __main as global to access from startup file
    LDR R0, =var1                   ; Load the address of 'var1' into r0
    LDR R1, [R0]                    ; Load the value at the
                                    ; address in r0 into r1 (0x12345678)

    LDR R2, var1

stop      B      stop               ; Branch stop

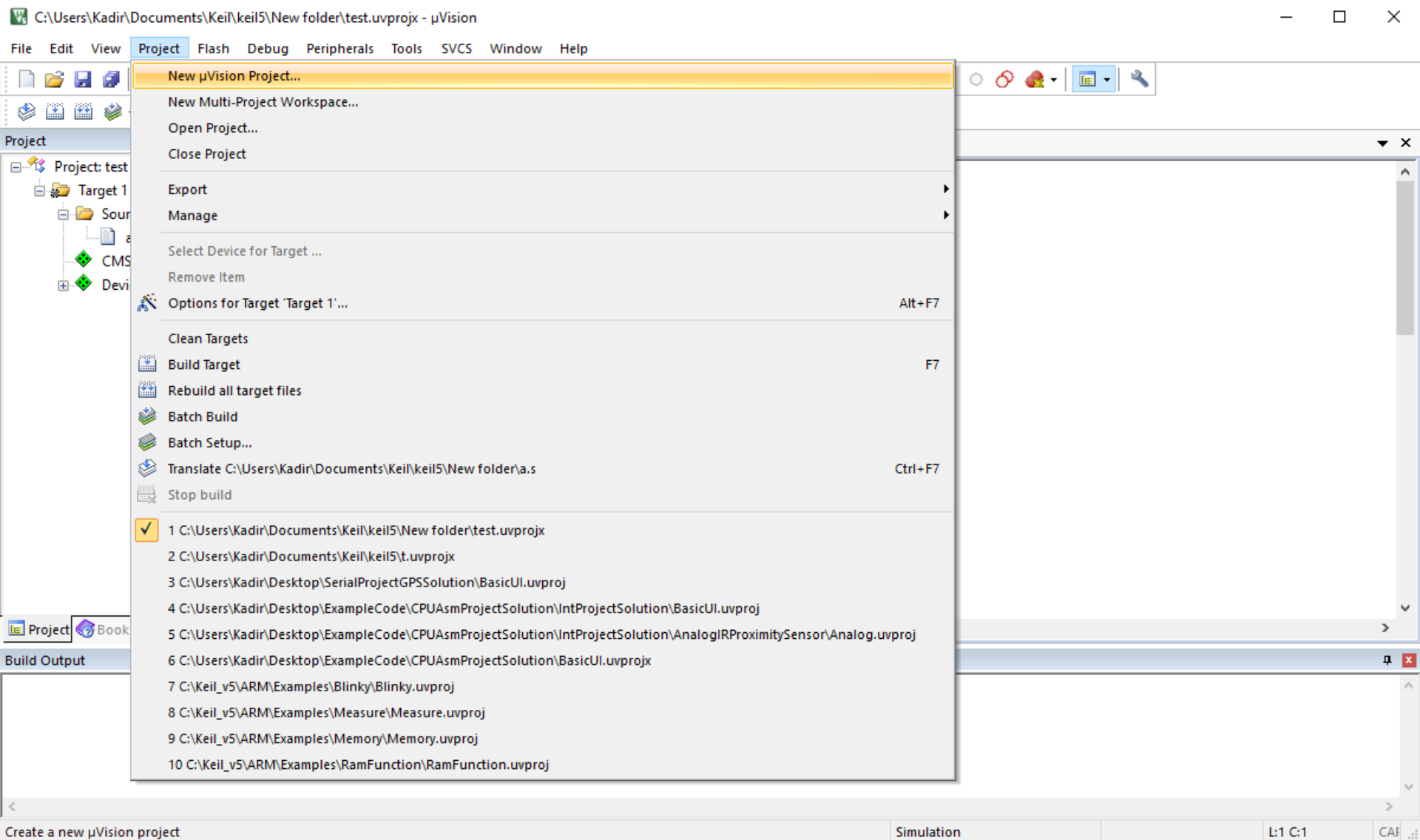
var1      DCD 0x12345678             ; Define a 32-bit constant in memory

    ENDFUNC                         ; Finish function
    END                             ; Finish assembly file
```

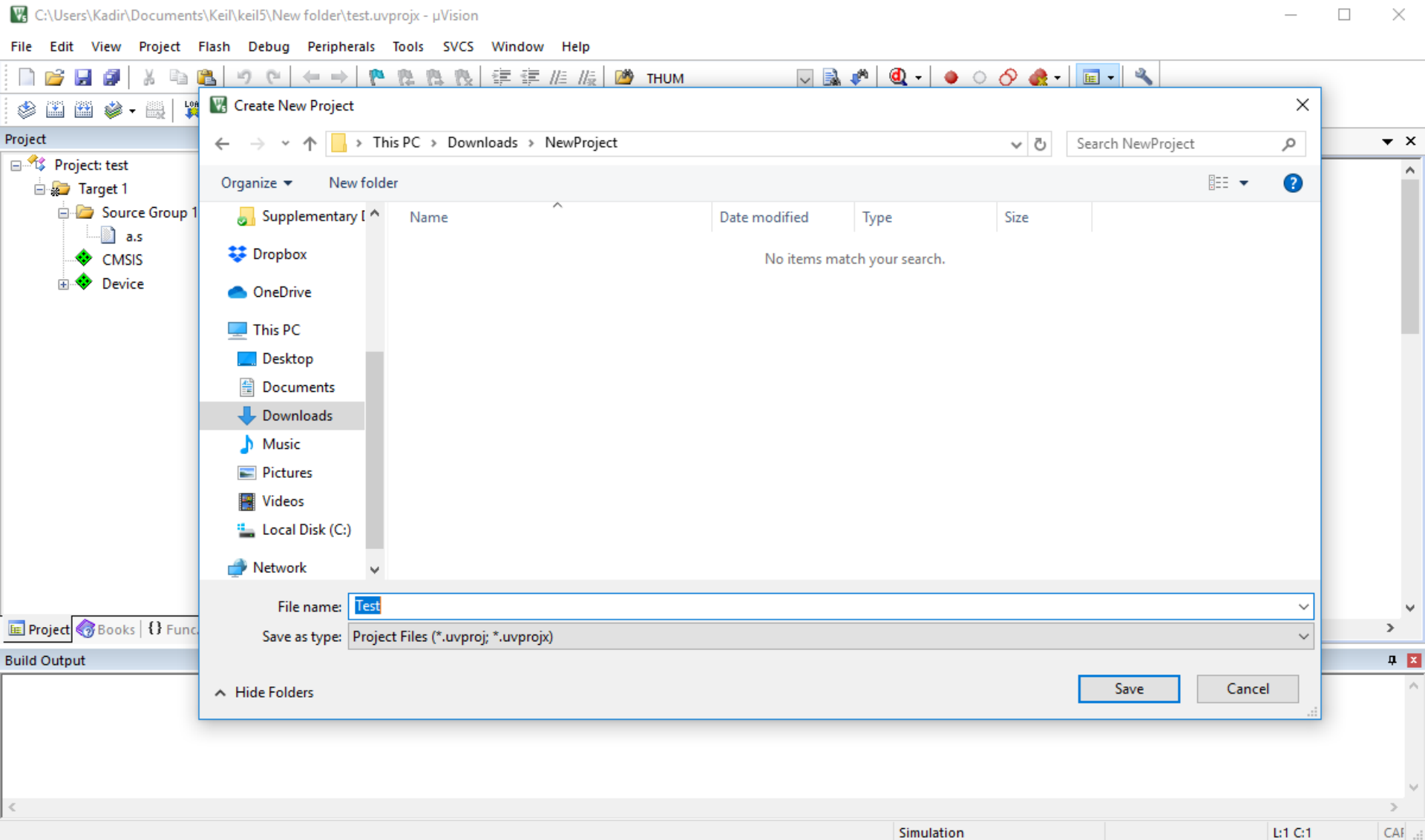


DEMO

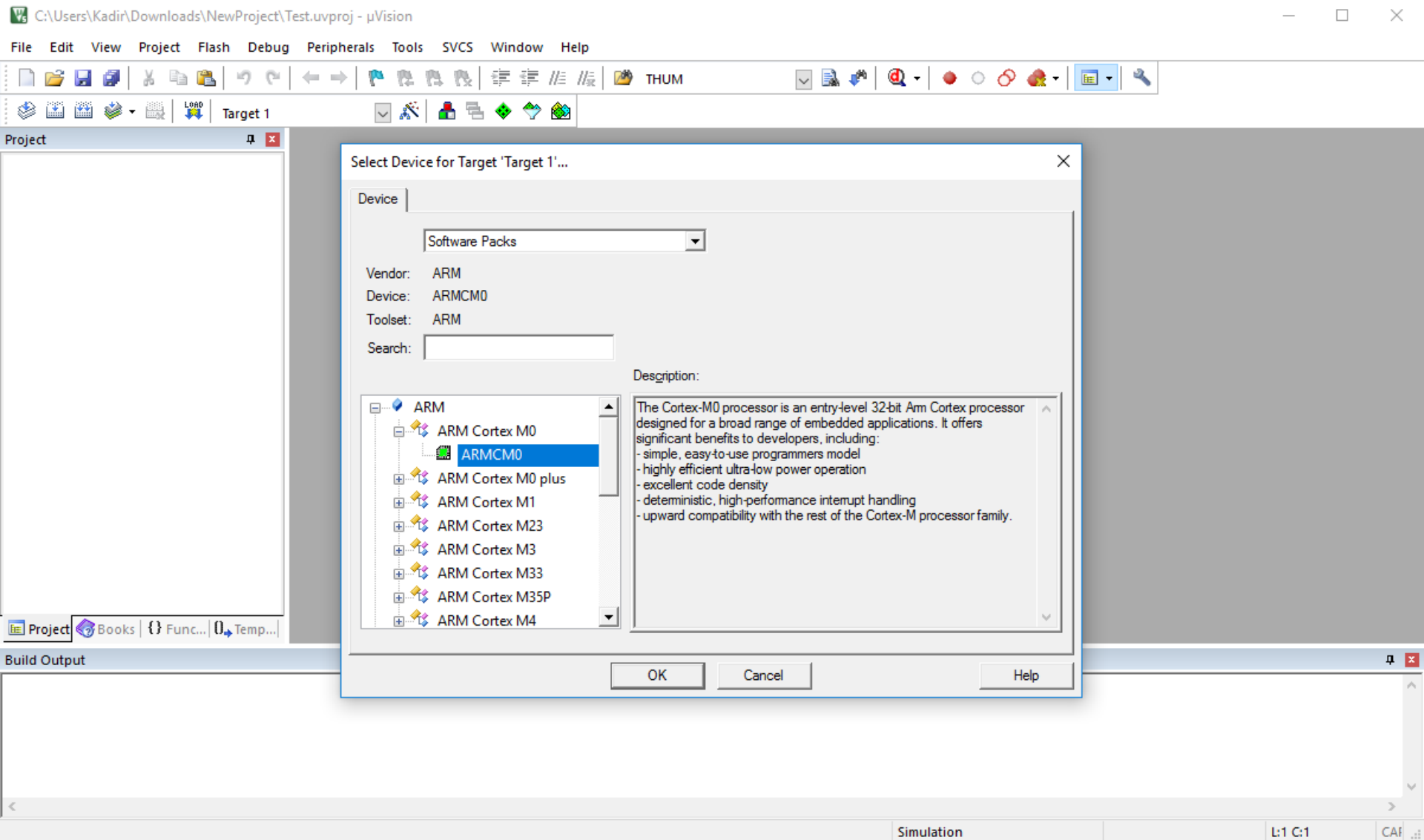
Start New μ Vision Project – Step 1



Start New μ Vision Project – Step 2



Start New μ Vision Project – Step 3



Start New μ Vision Project – Step 4

C:\Users\alpas\Downloads\micro_proc_recitations\defe\defe.uvprojx - μ Vision [Non-Commercial Use License]

File Edit View Project Flash Debug Per... Manage Run-Time Environment

Project Target_1

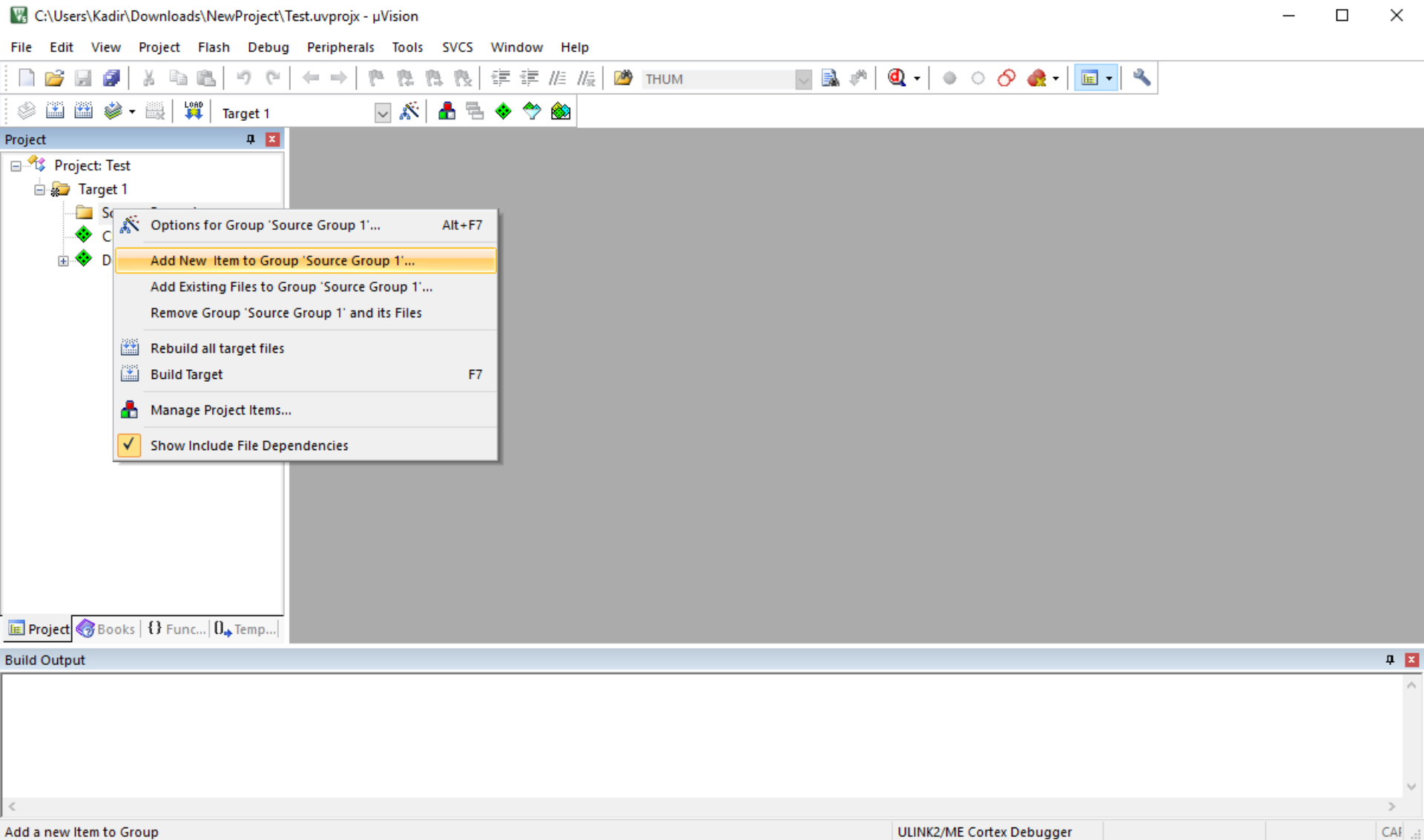
Software Component	Sel.	Variant	Vendor	Version	Description
CMSIS	<input checked="" type="checkbox"/>				Cortex Microcontroller Software Interface Components
CORE	<input checked="" type="checkbox"/>		ARM	6.1.0	CMSIS-CORE for Cortex-M, SC000, SC300, Star-MC1, ARMv8-M, ARMv8.1
DSP	<input type="checkbox"/>	Source	ARM	1.16.2	CMSIS-DSP Library for Cortex-M and Cortex-A
NN Lib	<input type="checkbox"/>		ARM	6.0.0	CMSIS Neural Network(NN) Library
OS Tick (API)	<input type="checkbox"/>			1.0.1	RTOS Kernel system tick timer interface
RTOS2 (API)	<input type="checkbox"/>			2.3.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
CMSIS Driver	<input type="checkbox"/>				Unified Device Drivers compliant to CMSIS-Driver Specifications
CMSIS-Compiler	<input type="checkbox"/>				Compiler Specific Interfaces
CMSIS-View	<input type="checkbox"/>				Debugger visualization of software events and statistics
Device	<input type="checkbox"/>				Startup, System Setup
Startup	<input checked="" type="checkbox"/>	C Startup	ARM	2.2.0	System and Startup for Generic Arm Cortex-M0+ device
File System	<input type="checkbox"/>	MDK-Plus	Keil		File Access on various storage devices
Network	<input type="checkbox"/>	MDK-Plus	Keil		IPv4 Networking using Ethernet or Serial protocols
USB	<input type="checkbox"/>	MDK-Plus	Keil		USB Communication with various device classes

Validation Output	Description
-------------------	-------------

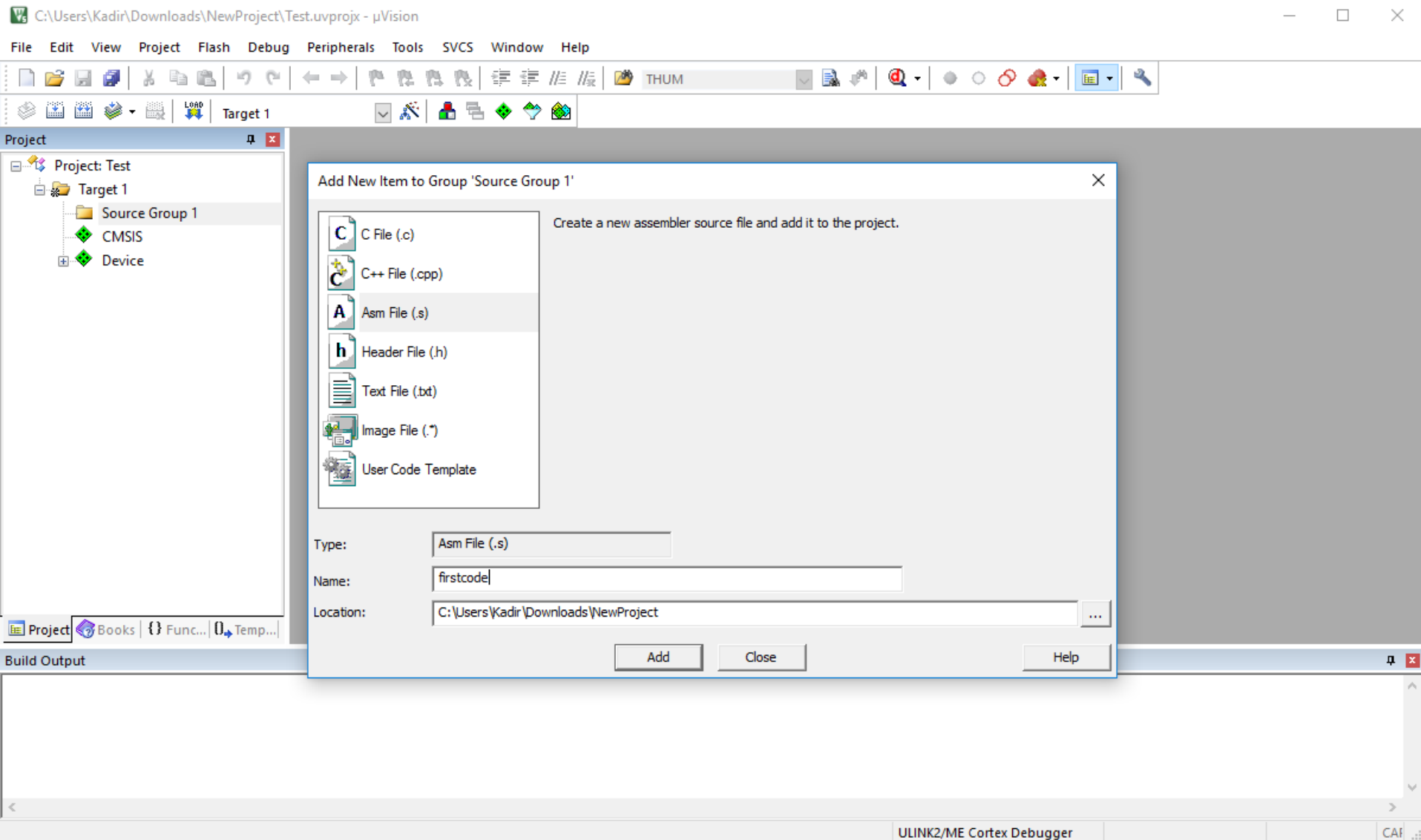
Resolve Select Packs Details OK Cancel Help

Simulation L:69 C:1 CAP NUM SCRL OVR R\W

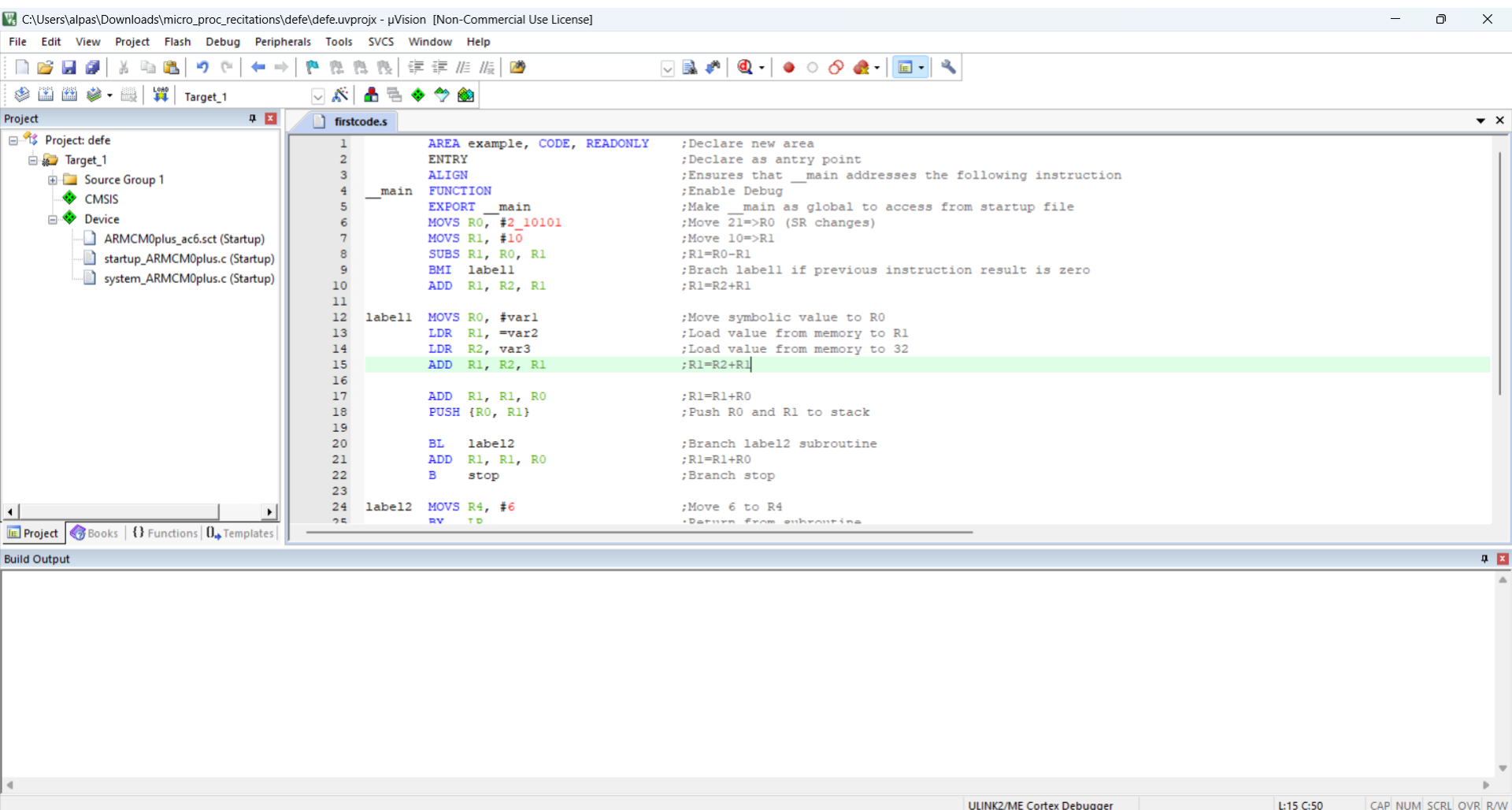
Adding new assembly file – Step 1



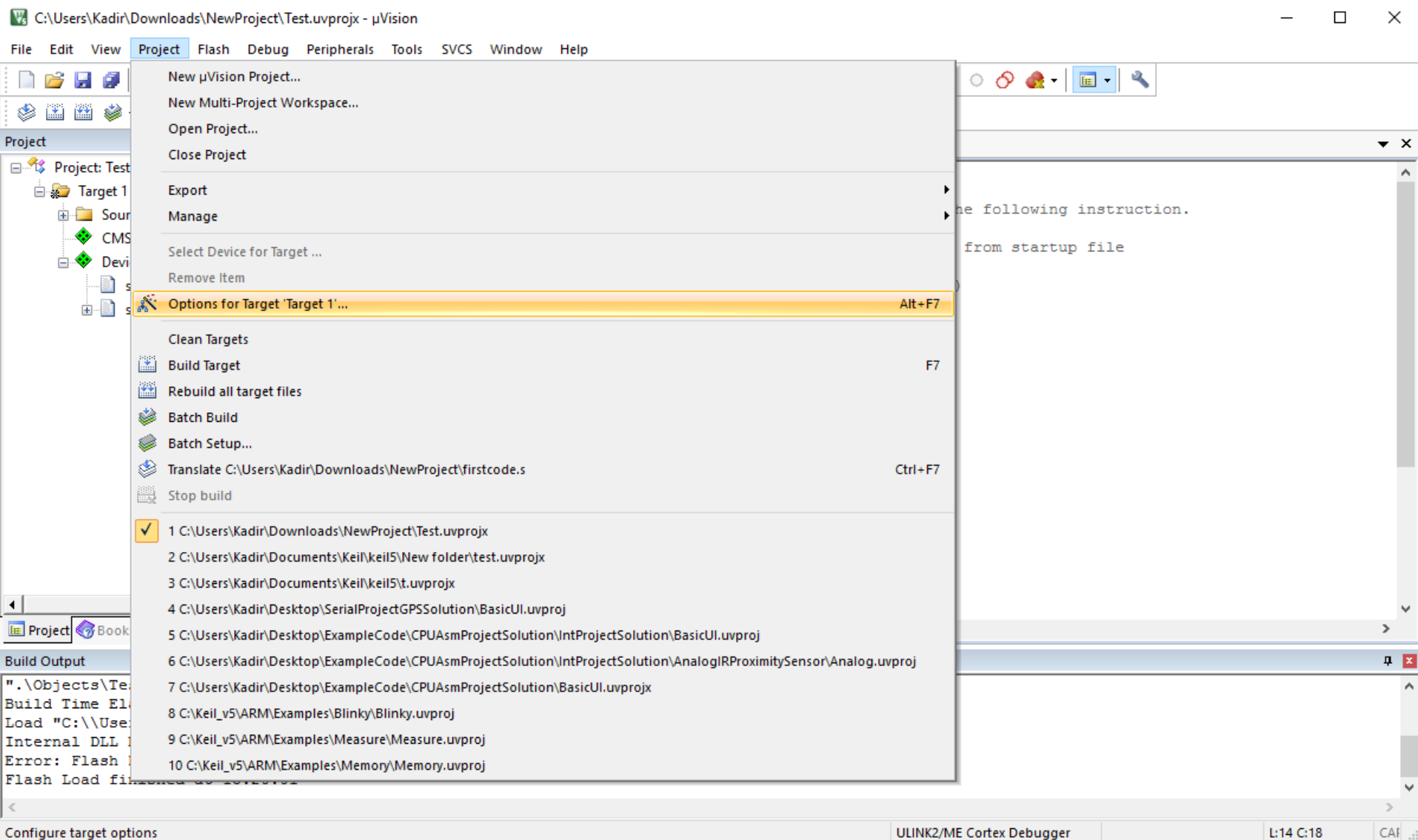
Adding new assembly file – Step 2



Write your code in your assembly file



Configure simulator – Step 1



Configure simulator – Step 2

C:\Users\Kadir\Downloads\NewProject\Test.uvprojx - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Project

Project: Test

Target 1

Source Group 1

CMSIS

Device

startup_ARMCM0.s (Start

system_ARMCM0.c (Start

Options for Target 'Target 1'

Device Target Output Listing User C/C++ Asm Linker **Debug** Utilities

☒ Use Simulator [with restrictions](#) Settings

☐ 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:

Dialog DLL: DARMCM1.DLL Parameter: pCM0

☐ Warn if outdated Executable is loaded

Manage Component Viewer Description Files ...

OK Cancel Defaults Help

Build Output

".\Objects\Test.axf" - 0 Error(s), 0 Warning(s)

Build Time Elapsed: 00:00:00

Load "C:\\Users\\Kadir\\Downloads\\NewProject\\Objects\\Test.axf"

Internal DLL Error

Error: Flash Download failed - Target DLL has been cancelled

Flash Load finished at 15:20:01

ULINK2/ME Cortex Debugger

L:14 C:18

Build your project to run

The screenshot shows the Keil uVision IDE interface. The title bar indicates the project path: C:\Users\Kadir\Downloads\NewProject\Test.uvprojx - uVision. The menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The 'Project' menu is open, displaying various options. The 'Build Target' option is highlighted in yellow, with the keyboard shortcut F7 shown to its right. Below the menu, the 'Project: Test' tree view shows a hierarchy with 'Target 1' and 'Source Files'. The 'Build Output' window at the bottom shows the status 'Build target files' and 'Simulation'. The status bar at the very bottom displays 'L:14 C:18' and 'CAI'.

C:\Users\Kadir\Downloads\NewProject\Test.uvprojx - uVision

File Edit View **Project** Flash Debug Peripherals Tools SVCS Window Help

Project: Test

- New uVision Project...
- New Multi-Project Workspace...
- Open Project...
- Close Project
- Export
- Manage
- Select Device for Target ...
- Remove Item
- Options for Target 'Target 1' ... Alt+F7
- Clean Targets
- Build Target F7**
- Rebuild all target files
- Batch Build
- Batch Setup...
- Translate C:\Users\Kadir\Downloads\NewProject\firstcode.s Ctrl+F7
- Stop build
- 1 C:\Users\Kadir\Downloads\NewProject\Test.uvprojx
- 2 C:\Users\Kadir\Documents\Keil\keil5\New folder\test.uvprojx
- 3 C:\Users\Kadir\Documents\Keil\keil5\test.uvprojx
- 4 C:\Users\Kadir\Desktop\SerialProjectGPSolution\BasicUI.uvproj
- 5 C:\Users\Kadir\Desktop\ExampleCode\CPUAsmProjectSolution\IntProjectSolution\BasicUI.uvproj
- 6 C:\Users\Kadir\Desktop\ExampleCode\CPUAsmProjectSolution\IntProjectSolution\AnalogIRProximitySensor\Analog.uvproj
- 7 C:\Users\Kadir\Desktop\ExampleCode\CPUAsmProjectSolution\BasicUI.uvprojx
- 8 C:\Keil_v5\ARM\Examples\Blinky\Blinky.uvproj
- 9 C:\Keil_v5\ARM\Examples\Measure\Measure.uvproj
- 10 C:\Keil_v5\ARM\Examples\Memory\Memory.uvproj

Build Output

Build target files Simulation L:14 C:18 CAI

Start your program in debug mode

The screenshot shows the Keil uVision IDE interface. The 'Debug' menu is open, displaying various options for starting and managing a debug session. The 'Start/Stop Debug Session' option is highlighted at the top of the menu. The background shows the project tree on the left with 'Project: Test' and 'Target 1' selected. The main editor window displays assembly code. At the bottom, the 'Build Output' window shows the compilation results for 'Target 1'.

File Edit View Project Flash **Debug** Peripherals Tools SVCS Window Help

Start/Stop Debug Session Ctrl+F5

- Energy Measurement without Debug
- Reset CPU
- Run F5
- Stop
- Step F11
- Step Over F10
- Step Out Ctrl+F11
- Run to Cursor Line Ctrl+F10
- Show Next Statement
- Breakpoints... Ctrl+B
- Insert/Remove Breakpoint F9
- Enable/Disable Breakpoint Ctrl+F9
- Disable All Breakpoints in current Target
- Kill All Breakpoints in Current Target Ctrl+Shift+F9
- OS Support
- Execution Profiling
- Memory Map...
- Inline Assembly...
- Function Editor (Open Ini File)...

Project: Test

- Target 1
- Source Group 1
- CMSIS
- Device
- startup_ARMCM0
- system_ARMCM0

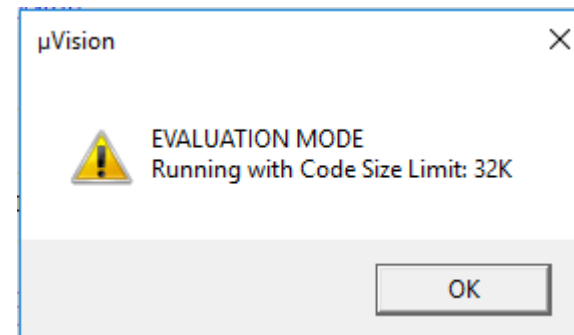
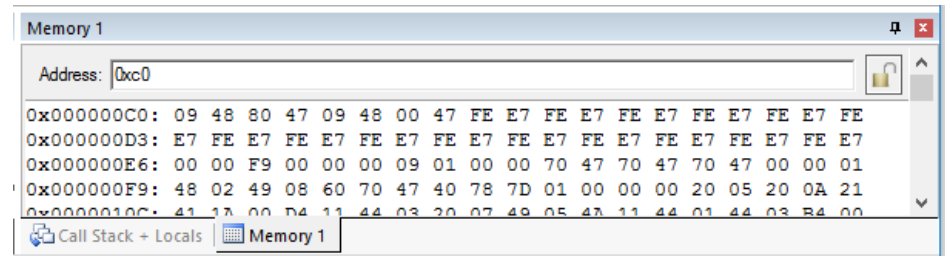
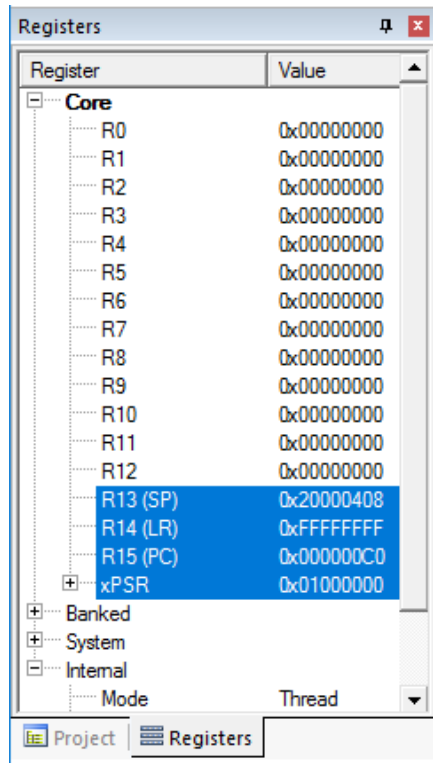
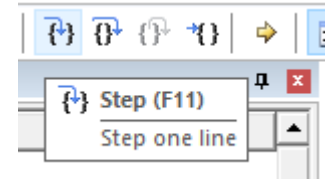
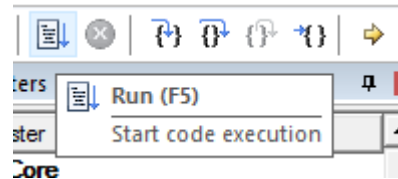
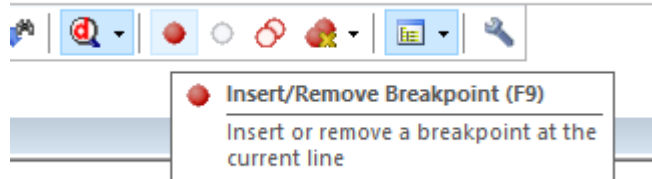
```
only
Declare as entry poin
Ensures that __main addresses the following instruction.
Enable Debug
Make __main as global to access from startup file
Move 5 -> r0 (SR changes)
Move 10 -> r1 (SR do not change)
r1
```

Build Output

Build started: Project: Test
*** Using Compiler 'V5.06 update 6 (build 750)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
Build target 'Target 1'
".\Objects\Test.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:00

Enter or leave a debug session Simulation L:14 C:18

Enjoy your new IDE...



If you get this error at the beginning of debugging, you are probably using free mode. Just click Ok and continue your work 😊



Refences

- <http://www.keil.com/support/man/docs/armasm/>
- <http://infocenter.arm.com>
- <http://www.ece.utep.edu/courses/web3376/Directives.html>

Recommended Refences

- <https://www.davespace.co.uk/arm/introduction-to-arm/>