Contents

9.	JTAG and Multi-ICE	
	9.1. 實驗目的	9-1
	9.2. 實驗原理	9-1
	9.2.1. About AXD	9-1
	9.2.2. Multi-ICE	
	9.3. 引導實驗	
	9.3.1. 實驗步驟	
	9.4. 實驗要求	
	9.5. 問題與討論	9-21
	9.6. 參考文件及網頁	9-21

9. JTAG and Multil-CE

9.1. 實驗目的

In this Lab the debugger target is Multi-ICE unit and an ARM Integrator board. You should have set up the hardware and the software of Multi-ICE unit and target board. The debugging skills you learn is the same as Lab3 except you do the debugging tasks with Multi-ICE. You will learn how to start-up the Multi-ICE server and debug program.

9.2. 實驗原理

9.2.1. About AXD

Debugger introduction:

A debugger is software that enables you to make use of a debug agent in order to examine and control the execution of software running on a debug target. ARM support two methods to do this work. One is AXD, the ARM eXtended Debugger. The other is armsd, the ARM Symbolic Debugger. They have the same function to debug the selected target. But AXD with GUI interface, is much easier to use it.

The debugger issues instructions that can:

- Load software into memory on the target
- Start and stop execution of that software
- Display the contents of memory, registers, and variables
- Enable you to change stored values.



Figure 1 Debugger-Target Interface

Debug target:

Debug target can be classified as software and hardware target. In software simulation, the expected behavior of the product is simulated by software. Even though you might run this software on the same computer as the debugger, it is useful to think of the target as a separate piece of hardware. They support ARMulator as the software simulation tool.

In hardware simulation, the interfaces can be classified as Multi-ICE and Angel types. This means that debugger scope and control the hardware device though Multi-ICE (ARM support) or Real-Monitor (Angel support) interface, adding logic, and probe.

In our course, we suggest Multi-ICE as the basic interface connected between hardware and host computer.

Remote debug interface:

The Remote Debug Interface (RDI) is an ARM standard procedural interface between a debugger and the debug agent (see Figure 1-1 on page 1-6).

- RDI gives the debugger a uniform way to communicate with:
 - a debug agent running on the host (for example, ARMulator)
 - a debug monitor running on ARM-based hardware accessed through a communication link (for example, Angel)
 - a debug agent controlling an ARM processor through hardware debug support (for example, Multi-ICE).



Target system with ARM-based ASIC and other components

Figure 2 Debugging connection

9.2.2. Multi-ICE

What is Multi-ICE?

An interface unit that connects the parallel port of a workstation to the JTAG interfaces of an ASIC that includes debug and EmbeddedICE capability

Function of Multi-ICE:

- 1. Debug extensions to the ARM core
- 2. The EmbeddedICE logic
- 3. The ICE extension unit

The debug extensions consist of a number of scan chains around the processor core and some additional signals that are used to control the behavior of the core for debug purposes:

- BREAKPT: enables external hardware to halt processor execution for debug purposes. active high
- DBGRQ: is a level-sensitive input that causes the CPU to enter debug state when the current instruction has completed.
- DBGACK: is an output from the CPU that goes high when the core is in debug state.

The EmbeddedICE logic is the integrated onchip logic that provides JTAG debug support for ARM core, which can be accessed through the TAP controller on the ARM core using the JTAG interface.

The ICE Extension Unit(IEU) is a logic block that can be added to the EmbeddedICE logic when a processor is fabricated.



Figure 3 Multi-ICE connection

Feature of Multi-ICE:

For the real time debug, a system may have many different devices operating at the same time. If they all connect to just one host computer to control and to trace these devices, it is impossible. So they separate this work into many computer, and then use an application so called "port mapper" to synchronize this signal though network connectivity.



Figure 4 Debug and Timing synchronization from network connection

9.3. 引導**實**驗

9.3.1. 實驗步驟

The following instructions are based on the demonstration program that runs the Dhrystone test software. For details of the Dhrystone test program, please refer to the readme.txt file and the various source files in its subdirectory. (e.g., C:\Program Files\ARM\ ADSv1_1\Examples\dhryansi\)

Starting the Multi-ICE server

To start the Multi-ICE server

- 1. Ensure that:
 - The Multi-ICE interface unit is plugged into the workstation
 - The Multi-ICE interface unit is plugged into the target JTAG connector
 - The target is powered up
 - The green power light on the interface unit is glowing brightly
- Select Start -> Program -> ARM Multi-ICE v2.2 -> Multi-ICE Server. The software displays the Multi-ICE server window, shown in Figure 1. The portmap application might also be started and minimized, depending on the host computer configuration.



Figure 5Unconfigured Multi-ICE server window

- 3. If a dialog box appears informing you that the Multi-ICE hardware cannot be found, click on OK and recheck the items listed in step 1.
- Configure the server. This can usually be done using the Autoconfiguration command. Select File -> Auto-configure and wait until the server has examined the target. If the configuration works, the screen looks similar to Figure 2.
- 5. Set AXD Configure Target. Open AXD, and select **Options->Configure Target**, and to choose Multi-ICE as target. If Multi-ICE is not listed in the **Target Environment**, we must add it the **Target Environment**.

Target	Open			2)	Help
ADP ARMIS: ABMUU	Look in doce	MuriHCE	<u> </u>		
•	source system Multi-ICE.r	Sie Floppy (Ac) Cl Program Files Am Am			Add amove nfigure
in ABMu	File parte:	Removable Disk (D.) MuriHCE dl	-	<u>O</u> pen	ive Ar
	Files of <u>type</u> :	DLLs(".d)	•	Cancel	and the

6.

📽 ARM - Multi-KE Server	
Elle View Ena Coatrol Computery Settings Holp	
F G2 0 7	
Auto-detected TAP Configuration	
TAP 0 (X) ARWEDT	
100	
Repotting Wilti-ICE hardware Resetting Wilti-ICE hardware	
in the second seco	
	lapat bits 🛄 1 🛄 2

Figure 6 Multi-ICE server window configured for an ARM920T

-----Note------

You can select **Settings** -> **Start-up Options** and click **Auto-Configure** in the **Start-up Configuration box**. This option automatically creates a configuration files naming all devices found as described in Automatic device configuration when you start Multi-ICE server next time.

Perform a variety of debugging tasks

- 1. Start CodeWarrior IDE.
- 2. Select **File** > **New** to create a new project.
 - (1) Select **ARM Executable Image** under the Project stationary.
 - (2) Click the **Set...** button next to the Location field.
 - (3) Navigate to the directory where you want to save the project and enter a project name, for example My_Project. This lab uses this directory C:\My Documents\ARMLAB\<your id no.>\My_Project



Figure 7 New dialog

- 3. Adding source files to the project.
 - (1) Copy files dhry.h, dhry_1.c, and dhry_2.c to your My_Project directory.
 - (2) Select Project > Add Files...
 - (3) Navigate to your My_Project directory and Shift-click on dhry_1.c and dhry_2.c.
 - (4) Click **Open**. Then Add all files to targets.

Add Files		×
Add files to targets:		
Targets		
🔽 DebugRel		*
🔽 Release		
🔽 Debug		
		*
20	F	1
	OK	Cancel

Figure 8 Add files to target dialog box

- 4. Configuring the project build targets
 - (1) Click the DebugRel Settings button. A DebugRel Settings window appears.

		Debug Rol Settings		TX
Delogfiel Delogfiel Tage Tage Tage Tage Tage Tage Tage Tage Tage Tage	(10) x (2	Coday Sof Edition Construction Constructi	Trapellettug Trapellettug Teart Rese Dataglid Linker JJSH Licker Im Te tables Flow Im Non-tables Flow Im Te tables flow Im	<u>One</u> One
Didugtini Didugtini Taire Didugtini Taire Didugtini Taire Taire Taire		Evalue Orthug No Northug No Northug No Northug No Northug Also Also Also Also Also Also Compile Compile Also Compile	Te blan Fire 2 Res blan Fire 2 Res blan Fire 2 Peret For the prostance may obtain pda Pereylining	Open One Baperthe

Figure 9 DebugRel Settings

- (2) Click the **ARM C Compiler** entry in the Target Settings Panels.
- (3) Click Preprocessor tab. Type MSC_CLOCK into the text field beneath the List of #DEFINEs and click Add to define the MSC_CLOCK. The Equivalent Command Line text box displays the result.
- (4) Click **Apply** to apply your changes.



Figure 10 MSC_CLOCK setting

(5) Click on the **Debug/Opt**. Select the **For time**. Click **Apply** to apply your settings.

JTAG and Multi-ICE

🖪 DebugRel Settings		<u>? ×</u>
Target Settings Panels	ARM C Compiler	
□ Target □ Target Settings □ Access Paths	S Warnings Errors	Debug/Opt Preprocessor Code Gen E
Build Extras Runtime Settings File Mappings Source Trees ARM Target	∍ generation :or symbols 1line functions	
 Language Settings ARM Assembler ARM C Compiler ARM C++ Compiler Thumb C Compiler Thumb C++ Compil 	bug view) view, good code) iew, best code)	Optimization Criterion © Eor space © For time ←
ARM Linker ARM fromELF Editor		
	Factory Settings H	Revert Import Panel Export Panel
		OK Cancel Apply

Figure 11 ARM C Compiler Panel

- (6) Click the **Current Target** drop-down list and select **Release** and **debug** form.
- (7) Apply the steps you followed above to define MSC_CLOCK the Release and debug form build target.
- 5. Building the project
 - (1) Hit the **Make** button to compile and link the project.
 - (2) A compiling and linking status windows would appear to indicate making progress
 - (3) After finishing compiling and linking, a result message windows would appear. Check for errors and warnings

Correct Television

Mathiana		These bears the terms of these terms	
Debugfiel Debugfiel Take [Sald Defer] Teapre		artis di a jabila Liado Report Brazil 2004 742 8 264 Film Librer Brazil	
r Tá Maji Aquis Aquis Aqui		Date: No Date: AP Date: 22 Date: Desca Date: 0 22 S 2010 1207 Name Date: 0 2010 1 20 Date: 2010 Date: 2010 1 20 Date: 2010 1 20 Date: 2010 Date: 2010 1 20 Date:	2
7 Be	0 1 2	Stead Rid Jacobie + Million + Million - Millio	A

Figure 12 Make the Project

-itisi

6. Debugging the project

(1) Hit the **Debug** button to call the AXD debugger to debug and run images built from the CodeWarrior IDE. A blue arrow indicates the current execution point.

		6 cm		-
		HALF & CALL OF A LOSS TO THE REAL	THE DESCRIPTION OF A DE	I STRILLAND
My Project and DelegRel I Reprint I I Last Tober Tearrin The Rep. 20 Rep. 10 Rep. 10			Anticipy (d_1, d_2, d_3) anticipy (d_1, d_2), d_3) anticipy (d_2, d_2), d_3) anticipy (d_2, d_3), d_4) (d_3, d_3), d_4) (d_4, d_3), d_4) (d_3, d_3), d_4) (d_4, d_3), d_4) (d_3, d_3), d_4) (d_4, d_3), d_4) (d_4, d_4), d_4), d_4), d_4) (d_4, d_4) (d_4, d_4), d_4), d_4), d_4), d_4) (d_4, d_4) (d_4, d_4), d_4), d_4), d_4), d_4), d_4) (d_4, d_4), d_4), d_4) (d_4, d_4), d_4), d_4), d_4), d_4), d_4), d_4) (d_4, d_4), d_4), d_4), d_4), d_4), d_4) (d_4, d_4), d_4	100
1184	at 185 2	ADDINGS PARTING	tel di.Ki adma di.ri.ri	1

Figure 13 Debug the Project

(2) Select **Execute** > **Go**. Execution stops at the beginning of the function main(), where a breakpoint is set by default.



Figure 14 Breakpoint set inside loop

(3) Select **Execute** > **Go**. You are prompted, in the Console processor view, form the number of runs through the benchmark that you want performed. Enter 8000.



Figure 15 Input benchmark

- (4) Select **File** > **Reload Current Image** to repeat the execution of the program.
- 7. Setting a breakpoint
 - (1) Select File > Reload current Image.
 - (2) Select **Execute** > **Go** to reach the first breakpoint, set by default at the beginning of the function main().
 - (3) Right-click on line 150 to position the cursor there and display the pop-up menu and select **Toggle Breakpoint**. You have set a second breakpoint.

A CONTRACT OF	
140	pediu time - cime (itoud .) n)t
1.41	Tendir
142	#srder REC_CLOCK
143	Begin_Time = clock();
144	fendir
345	
146	for (Run_Index = 1) Run_Index <= Number_Of_Runs; ++Run_Inde
147	
148	
149	proc 5() /
0 150	Peoc_6(1)
151	/* Ch 1 Glob == 'A', Ch 2 Glob == 'B', Bool Glob == tru
152	Int 1 Loc = 2)
153	Int_2_Loc = 3;
154	stropy (Btr 2 Loc, "DHEYSTONE PROGRAM, 2'ND STRING");
155	Enum Loc = Ident 2;
156	Bool Glob # / Func 2 (Str 1 Loc, Str 2 Loc);
157	/* Baol Glob == 1 */
158	while (Int 1 Loc < Int 2 Loc) /* loop body executed once
159	
7.60	Tet 3 Tes = 6 8 Tet 1 Tes , Tet 2 Test

Figure 16 Set Breakpoint

(4) Select **System Views** > **Breakpoints**. The breakpoint pane is displayed.

State	Processor	Position	Count	Condition	Size	Action	
	ARM920T_0	My_Project.avf : dhry_1.c : 78 [0x000083E4]	0/1		Auto	Break	
	ARM920T_0	My_Project.axf : dhry_1.c : 150 [0x000084E8]	0/750		Auto	Break	

Figure 17 Breakpoints pane

- (5) Right-click on the second breakpoint and select **Properties**. Enter 750 in the out of...field in the Condition group.
- (6) Click **OK**.

arpoint Prope	rties		2
Break At Processor C Source Inage Eile Line	AFM920T_0 My_Project and daty_1.c [150	Address Ox000084E8	OK Cancel Help
Condition Count 0 when Status	out of 750	Size	
E Endwar H/W res I	- D	C Ihanb F Automatic	
Action	Text		

Figure 18 Setting breakpoint details

- (7) Press F5 to resume execution, and enter the smaller number of 5000 this time for the number of runs required. Execution stops the 750th time your new breakpoint is reached.
- Select Processor Views > Variables. Click the Local tab and look for the Run_Index variable. Right-click on the variable and select Format > Decimal and the value is now displayed as 750 (decimal).

variable	Value
Ch Index	Variable not used yet (optimization)
Enum_Loc	0x01
Int_1_Loc	0x0000005
Int 2 Loc	Variable not currently used (optimization)
Int 3 Loc	0x0000007
Number Of Runs	0x00000BB8
Run_Index	750
Fstr 1 Loc	[31] "DHRYSTONE PROGRAM, 1'ST STRING"
- NOL A NOV	

Figure 19 Examining the contents of variable

- (9) Press F5 to resume execution, and the value of the Run_Index local variable changes to 1500.
- (10) Close down the Breakpoints system view.

Variable	Value
Ch Index	Variable not used yet (optimization)
E-Enum_Loc	0x01
Int 1 Loc	0x00000005
Int 2 Loc	Variable not currently used (optimization
Int 3 Loc	0x00000007
Number Of Runs	0x00000BB8
Run_Index	1500
B-str_1_Loc	[31] "DHRYSTONE PROGRAM, 1'ST STRING"



- 8. Setting a watchpoint
 - (1) Select File > Reload current Image.
 - (2) Select **Execute** > **Go** to reach the first breakpoint, set by default at the beginning of the function main().
 - (3) Select **Execute > Go** to continue execution.
 - (4) Enter 770 when you are prompted for the number of runs to execute. Execution continues until it reached the breakpoint at line 150 for 750th time. This is the breakpoint you defined in *Setting a breakpoint* in step 7.
 - (5) Select **System Views > Watchpoints**, right-click in the Watchpoints system view, and select **Add** to display the Watchpoint Properties

dialog.

(6) Set Run_Index in the Watch group and set the out of...field in the Condition group to value of 6.

A CONTRACTOR OF			OK
Processor [ARM90	0_T0		Capcel
heat Roa_lo	lex	_	15.1.
Wothing:			Hed
Joaditica.			
Yalas:	1		
Count: 0 out of	6 -		
Const: 0 out of	6 =		
Const: 0 out of others:		-	
Const: 0 out of others:	6 <u></u>		
Const: 10 out of others: 1 ibnus 17 Eachiel	6 the state	e (34	
Const: 0 out of when: F Eachief F Eachief	⁶	e (34. e pro	
Connet: 0 out of orbane Fr Easebled Fr Maniferen H/W ner ID	6 the second sec	• (24. • 10 ta • 20 ta	
Connet: 0 out of others Total T Hardware H/W res ID	State	 € [24. € [254. € 2536 	

Figure 21 Setting a watchpoint

- (7) Select **Processor Views > Variables**. Click the Local tab and look for the Run_Index variable.
- (8) Press F5 to resume execution. Soon the value of the Run_Index local variable changes to 756.

RM9201_0 - Variables										
Local Global Class										
Variable	Value									
Ch_Index	Variable not currently used (optimization)									
E-Enum_Loc	0x01									
Int_1_Loc	0x0000005									
Int_2_Loc	0x0000000D									
Int_3_Loc	0x0000007									
Number Of Runs	0x00000302									
Run_Index	756									
Estr_1_Loc	[31] "DHRYSTONE PROGRAM, 1'ST STRING"									
H-Str 2 Loc	[31] "DHRYSTONE PROGRAM, 2'ND STRING"									

Figure 22 Examining the contents of variable

(9) Delete the Watchpoint and breakpoint you set up for this example, by

right-clicking on its line in the Watchpoints and breakpoints window and select Delete from the pop-up menu, then close down them.

9. Examining the contents of variables

Two methods of examining the contents of variables are described:

(1) Contents of variables (variable processor view):

This method is simpler and shows the contents of the specified variables.

(2) Addresses and contents of variables (watch processor view):

This method shows the addresses of the variables as well as their contents.

10. Contents of variables

- (1) Select File > Reload current Image.
- (2) Select **Execute** > **Go** to reach the first breakpoint, set by default at the beginning of the function main().
- (3) Select **Execute > Go** to continue execution.
- (4) Select Processor Views > Variables
- (5) Select **Properties** > **Dec** and click **OK**. The display is now similar to that shown below.

Local Global Class		
Variable	Value	
Ch_Index	Variable not used yet (optimization)	
ErEnum_Loc	1	
Int_1_Loc	5	
Int_2_Loc	Variable not currently used (optimization)	
Int_3_Loc	7	
Number_Of_Runs	760	
Run_Index	750	
H-Str_1_Loc	[31] "DHRYSTONE PROGRAM, 1'ST STRING"	
W-Str 2 Loc	[31] "DHRYSTONE PROGRAM, 2'ND STRING"	

Figure 23 Examining the contents of variable

11. Addresses and contents of variables

- (1) Select File > Reload current Image.
- (2) Select **Execute** > **Go** to reach the first breakpoint, set by default at the beginning of the function main().
- (3) Select **Execute > Go** to continue execution.
- (4) Enter 760 when you are prompted for the number of runs to execute. Execution continues until it reached the breakpoint at line 150 for 750th time. This is the breakpoint you defined in *Setting a breakpoint*.
- (5) Select **Processor Views** > Watch.
- (6) Right-click in the window, and select **Add Watch** from the pop-up menu.
- (7) Enter the first expression in the expression by typing &Enum_Loc.
- (8) Press the Return key or clock on the **Evaluate** button.
- (9) Enter, in the similar way:

&Int_1_Loc &Int_3_Loc Run_Index

(10) Ensure that **Proc** and **Tab1** are selected, then Click the **Add to View** button and the **Close** button

oression:	Run_Index	Add To View
ocessor:	ARM920T_0	• Evaluate
apression	Value	Close
ARM920	T_0 {} m Loc 0x0007EED0	Halp
⊕ ∬_ +* ⊖ ∬_ *	0x01 1_Loc 0x0007FFD8 0x00000005 3_Loc 0x0007FFD4 0x00000007	View
	adex 0x000002EE	← Tab 1 ← Tab 2 ← Tab 3 ← Tab 4

Figure 24 Specifying variable to watch

(11) The variables you have specified are now displayed in the Watch processor view, and if you expand the lines you can see both the addresses and the contents of the variables.

Watch	Value	
Hagnum_Loc Ha Baint_1_Loc La Baint_3_Loc La Run_Index	0x0007FFD0 0x01 0x0007FFD8 0x0000005 0x0007FFD4 0x0000007 0x000002EE	

Figure 25Examining contents of registers

- (12) Press F10. The program executes a single instruction and stops.
- (13) Press F10 a few more times. As you execute the program, one instruction at a time, you can see the values of several of the registers change.
- (14) Press F5 to allow the program to complete its execution, the close down the registers processor view.

12. Examining the contents of registers

- (1) Select File > Reload current Image.
- (2) Select **Execute** > **Go** to reach the first breakpoint, set by default at the beginning of the function main().
- (3) Select **Processor** > **Registers**.

ARM/20T 0 - Exemption		2
Register	Value	
@ Current	11	- 12
E-User/System	()	
⊢r∎	0x00D0C8AC	
r1	0x0007FFE0	
===2	0x00000200	
-13	0x0000010	
	0x00011C34	
-25	0x0000AE80	
16	0x0000000	
-e7	0x0000000	
	0x0000000	
- = 9	0x49525453	
-== 1.0	0x00004748	_
r11	0x00000000	
-r12	0x0000002	
-r13	0x20403A10	
	0x04400048	
-pe	0x00008384	
-cper	n3CvqIFt_8VC	
E-FIQ	()	
H-IRQ	6 1	
E-SAG	S 3	
H Abort	()	
E-Undef	()	
B-C#15	1 1	*

Figure 26 Examining contents of registers

- (4) Press F10. The program executes a single instruction and stops.
- (5) Press F10 a few more times. As you execute the program, one instruction at a time, you can see the values of several of the registers change.
- (6) Press F5 to allow the program to complete its execution, the close down the registers processor view.

13. Examining the contents of memory

- (1) Select File > Reload current Image.
- (2) Select **Execute** > **Go** to reach the first breakpoint, set by default at the beginning of the function main().
- (3) Select **Execute > Go** to continue execution.
- (4) Enter 760 when you are prompted for the number of runs to execute.

Execution continues until it reached the breakpoint at line 150 for 750th time. This is the breakpoint you defined in *Setting a breakpoint*.

- (5) Select **Processor Views > Memory**.
- (6) Set the Start address value to 0x07FFFF00.

Address	0	17.1	2	3	4	0.5	6	3				in the		et.	1.10	.f.	ASILL	-
0x87FFFFF00	28	0.5	02	DZ.	78.	79	12	RE	00	50	ZA	02	7F	EA.	TA	1.5	(By *. B. z.	
0x07FFFF10	00	1.0	40	80	28	22	58	88.	0.0	30	#1	0.0	78	#2	88	FD	··· 8	
0x878787820	42	0.0	02	12	0T	DF	79	CA	10	90	0.0	8.2	DF	31	11	DF	B	-
18178787830	00	DD:	56	12	74	22	DIF	HE.	50	04	40	114	TT .	10	78	11		
1x17FFFF40	24	24	40	30	TT.	78	IC.	RE	52	- 0C	40	80	11	78	5.8	12	.58081002.	
1x07FFFFF50	82	OD.	0.0	41	BF	DF	DD.	BZ.	80	02	11	EC.	8.8	78	17	DD	and the second second	
1x17888860	19	01	0.0	24	R.F.	00	27	DE	2A	0.6	40	80	D.A.	10	77	Ab		
1x17555570	80	0.0	CB	10	F.E.	22	CA.	R.E.	20	98	80	20	EF	26	10	£3		
STREETED.	00	60	00	88	114	22	11	28	CU	UE	00	22	78	80	DE	D D	· · · · · · · · · · · · · · · · · · ·	
1x07FFFF90	10	0.0	40	A2 -	TF	71	92	25	00	10	AD .	80	FD.	7.8	17	111		
N87EFFFA0	EA.	0.0	0.0	28	115	22	11	BD.	88	DE	UA:	00	35	IA.	17	111	···· *······ ··· ?···	
1807555550	08	38	80	0.0	TT.	00	BF.	28	40	DE	60	23	11	22	80	38	.0/9.**7	
asi7rfrrc0	81.	0.0	0.0	21	DF	IF.	38	A9	48	DE	02	00	TE .	10	FA.	5.8		

Figure 27 Examining contents of memory

- (7) Press F10. The program executes a single instruction and stops.
- (8) Press F10 a few more times. As you execute the program, one instruction at a time, you can see the values of several of the memory addresses change.
- (9) Press F5 to allow the program to complete its execution, the close down the Memory processor view.

14. Locating and changing values and verifying changes

- (1) Select File > Reload current Image.
- (2) Select Execute > Go to reach the first breakpoint, set by default at the beginning of the function main().
- (3) Select **Search** > **Memory**.
- (4) Enter 2'ND in the Search for field, set the In range and to addresses 0x0 and 0xFFFF, and select **ASCII** for Search string type.

Processor:	ARM920T	0] Eind
<u>S</u> earch for:	2'ND	•] Find Next
In range	0x0	to OxFFFF	
Search strin	ng type		Cancel
ΦA	SCII	C Hexadecimal	Help

Figure 28 Searching for a string in memory

- (5) Click the Cancel button to close the Search Memory dialog.
- (6) The four hexadecimal values highlighted are 32 27 4E 44.

Tabl - Hex - Hora	antre l	Stattally	100 (Us	2000			and all a	10.110	Maria	101							
Address	8	1	2	3	4	8	6	3	E-140 P	. 9		l. h		et.		I.E.	ASCI
08860000080	45	2.0	50	32	43	47	52	41	4D	DC.	20	32	27	410	44	20	E PROGRAM Z*ND
0938000000	52	54	52	49	42	47	00	00	00	00	A0	87	14	60	00	85	STRENG
0.A36DIGDEC	68	DO	90	15	03	00	50	83	60	DE	00	.8.8.	.04	50	BD.	122	h P dP *
OB960100m0	68	DO	9D	15	00	10	80	ED	04	DD.	40	80	-04	00	BD.	85	h
addideco.	05	23	AD	#1	04	10	AD .	81.	68	0E	90	83	98	71	17	11.0	· · · · · · · · · · · · · · · · · · ·
033600008600	68	00	9D	85	01	00	50	82	65	0.0	80	85	04	0.0	50	01	h
OB380000m	22	11	FF .	EA.	DB-	10	A.L	E1	10	0.2	91	12	65	20	9D	85	
033600008680	64	318	9D	15	.98	TE	17	ED	50	UE .	28	85	EC.	31	17	0.0	d0
0x000005700	41	50	AD	83	01	00	DB	85	41	DE	50	83	16	80	DD.	3A	A9
0x00008710	42	10	AD	13	05	00	AU.	81	AP.	75	77	10	60	10	DD	85	C
0x00006720	111	00	50	#1	U.D.	00	DO	1.6	0.9	1.0	A0	81	00	00	All	83	
000008730	62	11	EF.	11	85	OF	10	82	0.00	50	80	11	0E	50	AA.	120	Burners Fire Fire
0x00006740	010	00	90	88	0.8	0.0	BA.	20	14	35	AB.	82	0.6	10	All	81	
•																	•

Figure 29 String in memory

- (7) Double-click on the value 32 and type 0x4E and press Return
- (8) Double-click on the value 27 and type "o and press Return
- (9) Double-click on the value 4E and type 46 and press Return
- (10) Double-click on the value 44 and type o62 and press Return

Address 0 =00008680 45 =00008680 68 =00008680 68 =00008680 68 =00008680 68 =00008680 82 =00008680 82	3 20 54 00 20 00	2 50 32 90 90 80 90	52 47 85 85 85 85 85 85 85 85 85 85 85 85 85	4 4 03 00 04	47 47 00 01 10	52 50 50 80 A0	7 41 00 83 80	4D 00 0D 04	2C 00 00 00	20 A0 00 40	48 83 84 80	6F 14 64	11 10 50	* 21 2 2 2	1000	ARCEL E FROGRAM, No. 2 STRING.
#00008688 45 #00008688 88 #00008688 88 #00008688 68 #00008688 68 #00008688 68 #00008688 68 #00008688 68	20 54 00 20 00	50 52 90 90 90	52 47 85 85 85 81 85	4F 4E 03 00 04	47 47 00 01 10	52 00 50 80 M0	41 00 83 80	4D 00 0D 04	2C 00 00 90	20 A0 01 40	48 8.7 AA 8.0	0F 14 64	20 50 80	1111 11 10	1222	E FROGRAM, No.2 STRING.
#00008695 53 #00008685 68 #00008685 68 #00008685 68 #00008685 68 #00008685 82 #00008685 64	54 00 20 00	32 90 90 90	特許的影響	4E 03 00 04	47 00 01 10	80 50 80 30	00 83 80	00 0D 04	00 00 90	A0 00 40	8.3 AA ED	64	50 80	ED ED	85 82	brningdF.
x000086AB HE x000086AB HE x0000866B 05 x000086AB 6E x000086AB F2 x000086AB F3	00 00 20 00	90 90 A0 90	85 85 81	0.8 0.0 0.4	00 01 10	50 80 30	11 11	0D 04	00	00 40	АА, Е 0	64	50	ED ED	82	him Provider.
000006680 68 000006600 05 000006600 68 000006680 82	00 20 00	90 A0 90	85 81 85	00	01 10	68 A0	10	04	.90	4.0	50	44	30	100	10.0	the second second second
1000006500 05 1000006600 68 1000006600 #2	20	A0 90	81	04	10	3.0	111	45.00	24 July 1			1.0	10.10	1 N.M. 1	0.0	111 1 1 1 1 1 1 1 1 1 1 1 W 1 W 1 1 1 1
000006600 68 000006660 #2	0.0	90	100					0.5	9.0	.90	85	98	78	22	8.8	
000008680 82	10.00			01	0.0	80	#2	60	0.0	80	85	0.4	10	50	83	h
000008680 64	6.6	28	BA.	08	2.0	. AG	81	10	0.3	98	85	68	20	-90	85	another and the second second
	30.	90	85	98	3.8	11	KB	08	30	14	K5	80	FE	89	8.8	dl
000008708 41	50	AG	ET	01	0.0	10	85	41	0.0	-54	63	16	10	00	3.8	APA. Press
000000710 40	10	A0	83	05.	0.0	A0	#1	AF	28	10	KE.	60	3.0	DD	8.5	G
000000720 01	an	30	81	0.8	0.0	00	LA.	0.0	3.0	.84	81	0.0	00	80	82	
000000730 42	FE	22	2.5	25	0.0	19	81	O'R.	50	20	80	0.E	10	AA.	22	h
#000009740 OE	0.0	00	24	0E	0.0	EA.	22	14	AD	4.8.	82	0.0	40	DA.	81	J

Figure 30 Changing contents of memory

(11) Press F5 to continue execution, and enter a value of 100 when you are prompted in the Console processor view for the number of runs to perform.

Contole		×
ARM920T_0 - Comola		
Int_1_Loc:	5	*
should be:	5	
Int_2_Loc:	13	
should ber	13	
Int J Loc:	7	
should be:	7	
Enum Loc:	1	
should be:	1	
Str_1_Loc:	DHRYSTONE PROGRAM, 1'87 STRING	
should be:	DERYSTONE PROGRAM 1'ST STRING	
Str_2_Loc:	DERYSTONE PROGRAM, No.2 STRING	
should be:	DERYSTONE PROGRAM, 2'HD STRING	
Microseconds for o	one run through Dhrystone: 95100.0	
Dhrystones per Nec	iond: 10.5	East.
		*
×		2 1

Figure 31 Contents display in the console

-----Note------In this example, the change you made was not permanent, because you did not alter the source code or the executable image stored in a disk file. You altered only the temporary copy of the image in the target memory.

9.4. 實驗要求

Write a lotto program that generates N sets of number. The user can specify:

- 1. Number of the set: N.
- 2. The numbers must be included in these N sets of number
- 3. The numbers must not be included in these N sets of number

Note: Numbers cannot be duplicated within the same set of number. In addition, output numbers within the same set in ascending order.

This exercise lets you familiar with the debugging skills and ARM CodeWarrior Editor when you try to build the program correct.

9.5. 問題與討論

What's different between ARMulator and MultiICE that we do the debugging task.

9.6. 參考文件及網頁

- Multi-ICE [DUI_0048F_MICE2_2_UG]
- AXD and armsd Debuggers Guide [DUI_0066D_AXDDG_2_UG]
- Getting Started Guide [DUI_0064D_GSG_UG]