[Microprocessor Application]
Lab 2: Memory Access

Chester Sungchung Park

SoC Design Lab, Konkuk University

Webpage: http://soclab.konkuk.ac.kr

Outline

- ❑ Creating projects
- ❑ Programming C applications
- ❑ Running C applications
- ❑ Debugging C applications
- ❑ Optimizing C applications

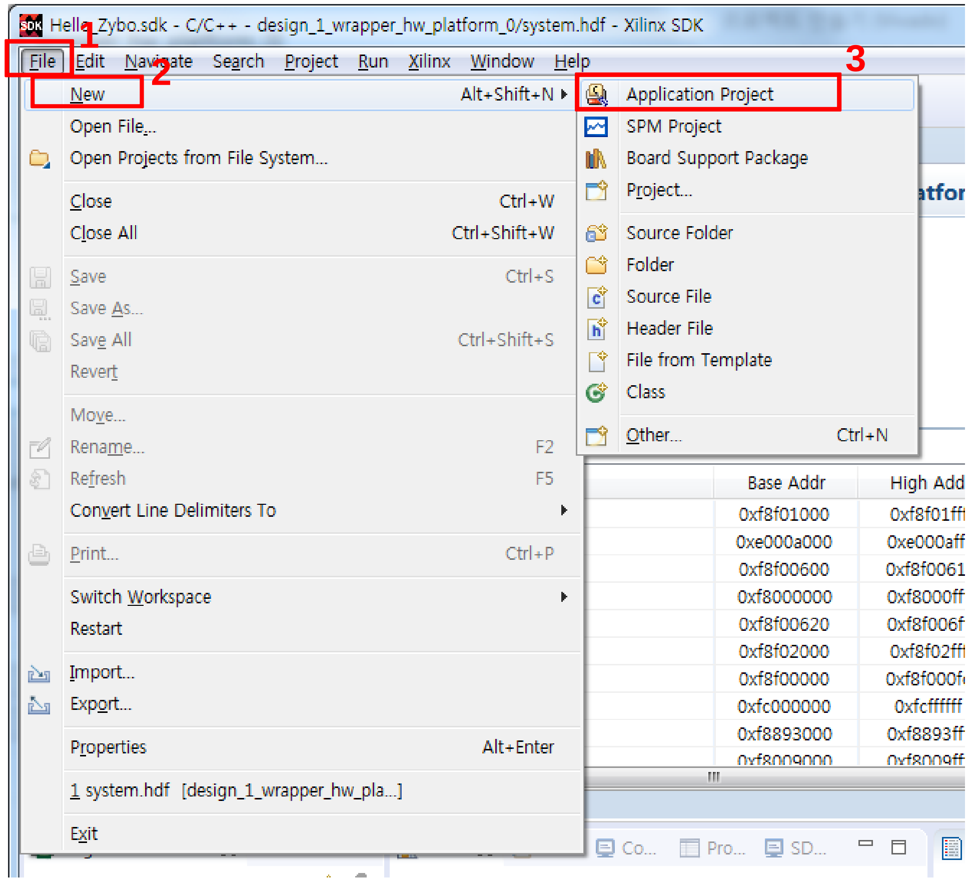
Creating Projects

- Repeat the previous steps
  - Follow pp. 4~25 of the following lab workbook: **Lab_MP2022_1_work.pdf**

Creating C Applications

❑ Create a C application project

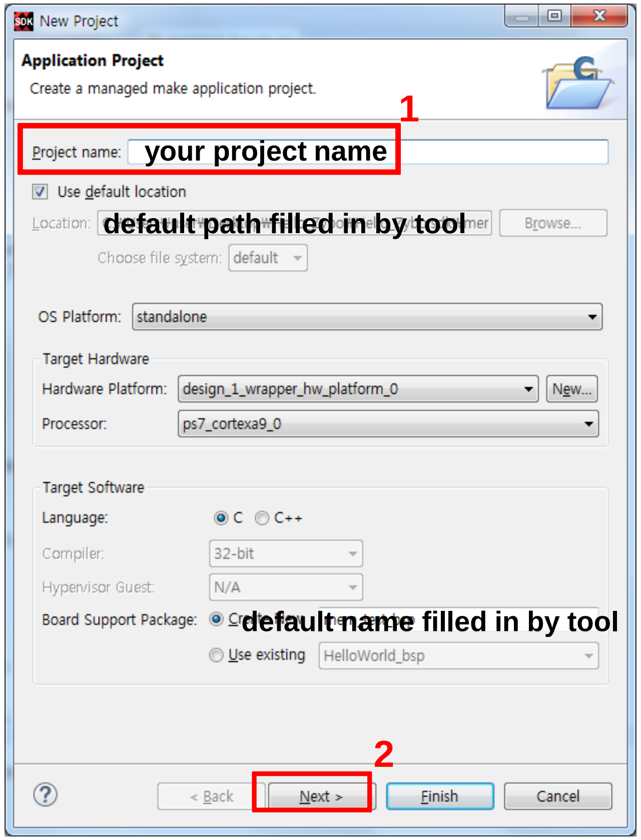
- Click ***'File' > 'New' > 'Application Project'***



Creating C Applications

❑ Create a C application project (cont'd)

- Type ***<your project name>*** in the Project name field
- The ***'Board Support Package'*** field can be set up to use an existing BSP or a new BSP can be created based on the project name. (Do not modify)



New Project

Application Project

Create a managed make application project.

1

Project name: your project name

☑ Use default location

Location: default path filled in by tool

Choose file system: default

OS Platform: standalone

Target Hardware

Hardware Platform: design_1_wrapper_hw_platform_0

Processor: ps7_cortexa9_0

Target Software

Language: (●) C ( ) C++

Compiler: 32-bit

Hypervisor Guest: N/A

Board Support Package: (●) Create New: default name filled in by tool

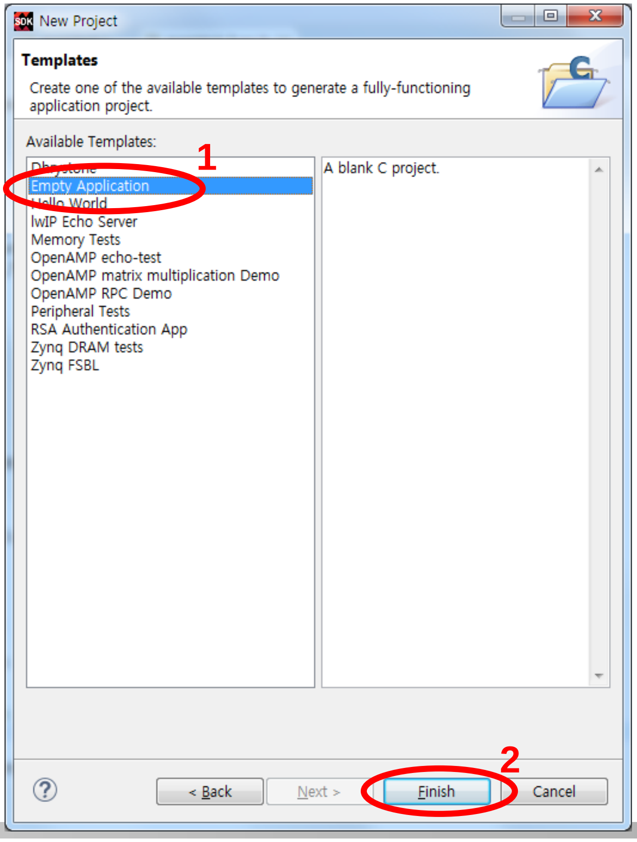
( ) Use existing: HelloWorld_bsp

2

Creating C Applications

❑ Create a C application project (cont'd)

- Select ***'Empty Application'*** from the template list
- Click ***'Finish'***



Templates

Create one of the available templates to generate a fully-functioning application project.

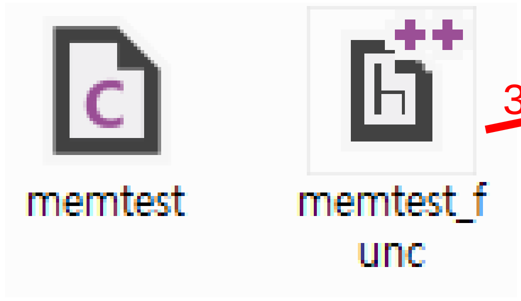
Available Templates:

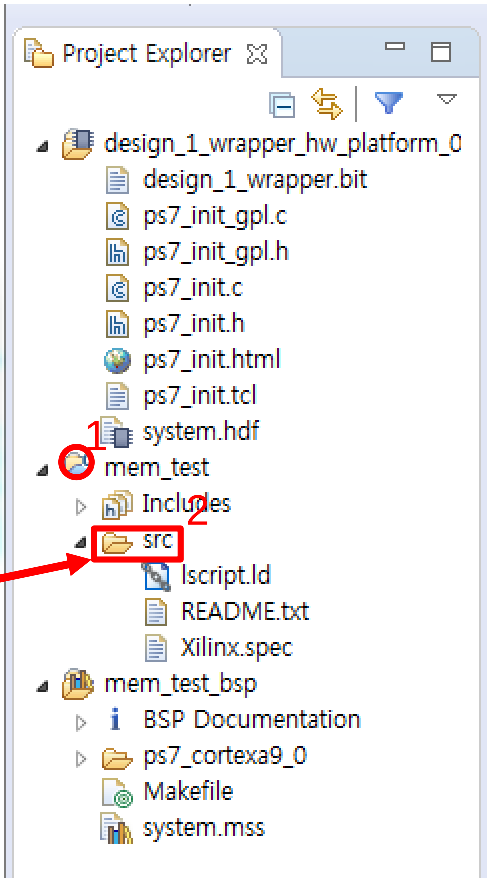
Dhrystone
Empty Application
Hello World
lwIP Echo Server
Memory Tests
OpenAMP echo-test
OpenAMP matrix multiplication Demo
OpenAMP RPC Demo
Peripheral Tests
RSA Authentication App
Zynq DRAM tests
Zynq FSBL

A blank C project.

Creating C Applications

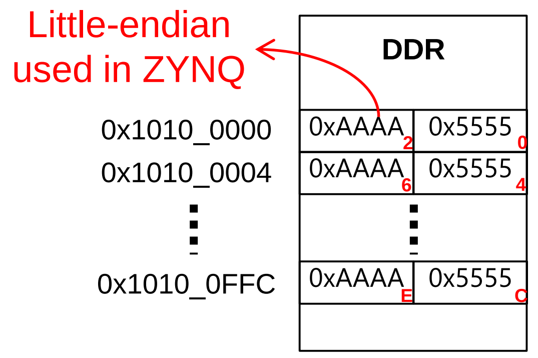
- Add and change source files
  - Unfold your project and select ***'src'*** folder
  - Copy **memtest.c, memtest_func.h** and paste into the ***'src'*** folder
  - Double-click ***'memorytest.c'*** to review the source files

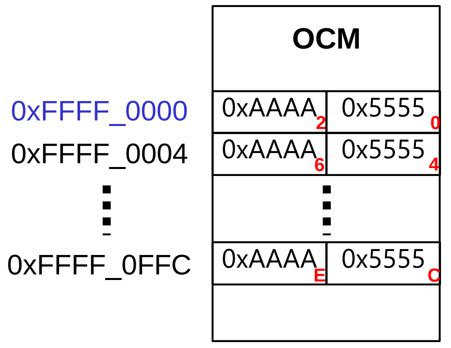


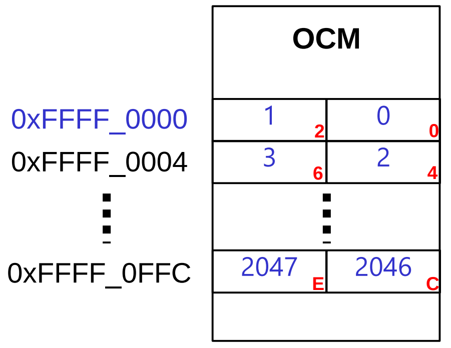


Programming C Applications

- memtest_example( )
  - Access size: 32 bits (word)
  - Pattern: 0xAAAA_5555 (1024 words)
  - Region: 0x1010_0000 ~ 0x1010_0FFC (DDR)
- memtest_0( )
  - Access size: 32 bits (word)
  - Pattern: 0xAAAA_5555 (1024 words)
  - Region: 0xFFFF_0000 ~ 0xFFFF_0FFC (OCM)
- memtest_1( )
  - Access size: 16 bits (halfword)
  - Pattern: 0xAAAA_5555 (1024 words)
  - Region: 0xFFFF_0000 ~ 0xFFFF_0FFC (OCM)
- memtest_2( )
  - Access size: 16 bits (halfword)
  - Pattern: halfword offset (0 ~ 2047)
  - Region: 0xFFFF_0000 ~ 0xFFFF_0FFC (OCM)

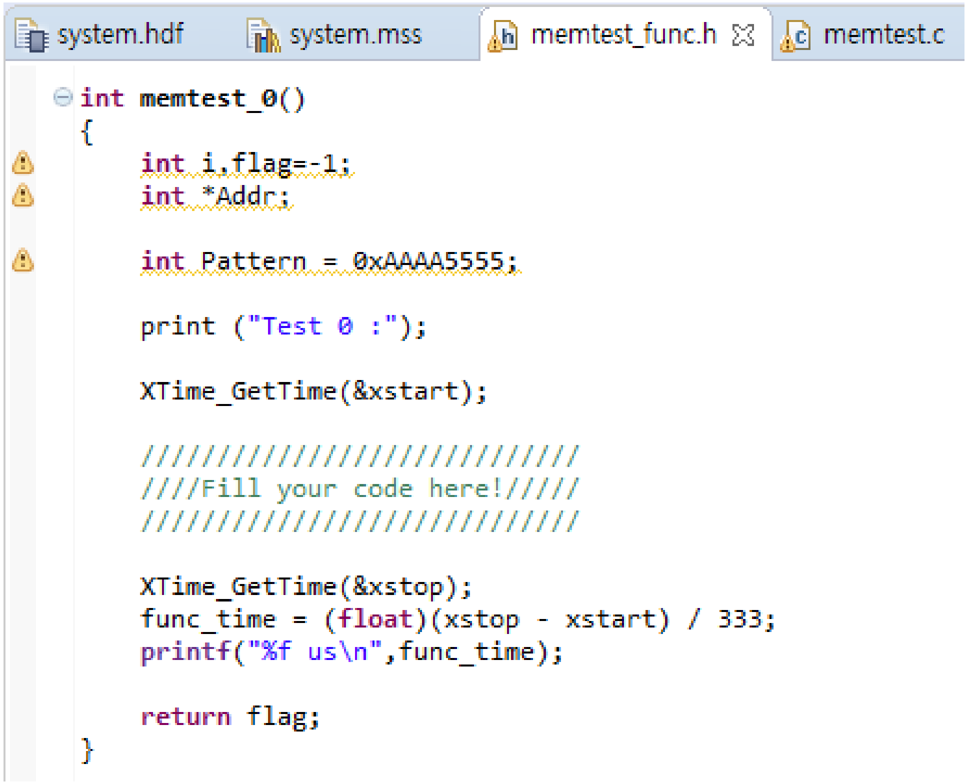






Programming C Applications

- ❑ Modify the C source code (‘**memtest_func.h**’)
  - Add lines to ‘**Fill your code here**’ in the functions, memtest_0( ), memtest_1( ) and memtest_2( )



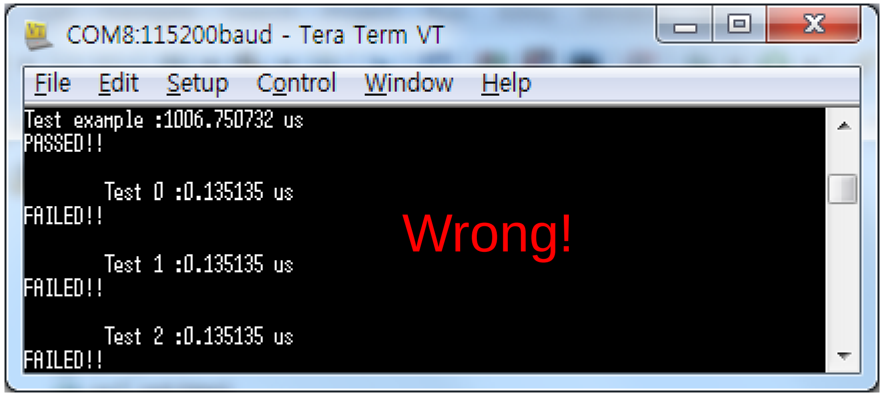
```
int memtest_0()
{
    int i,flag=-1;
    int *Addr;

    int Pattern = 0xAAAA5555;

    print ("Test 0 :");

    XTime_GetTime(&xstart);

    ///////////////////////////////
    ////Fill your code here!//////
    ///////////////////////////////

    XTime_GetTime(&xstop);
    func_time = (float)(xstop - xstart) / 333;
    printf("%f us\n",func_time);

    return flag;
}
```

Running C Applications

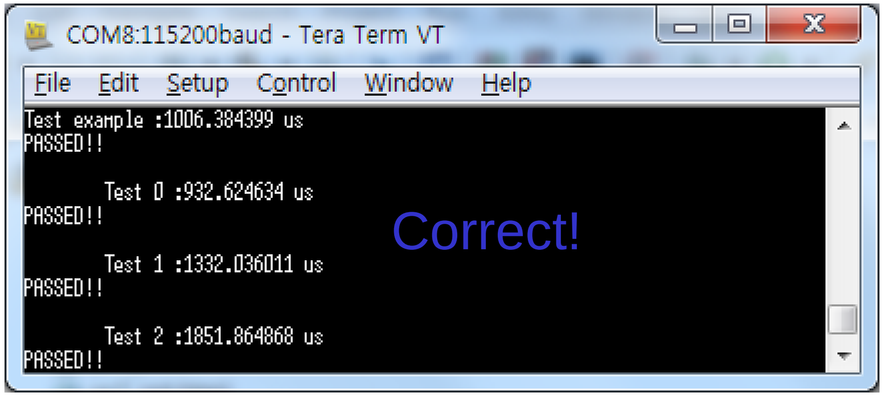
- Repeat the previous steps
  - Follow pp. 30~33 of the following lab workbook: **Lab_MP2022_1_work.pdf**

Running C Applications

- ❑ Run the application
  - Check the output on ***'Tera Term'***
    - ✓ Whether all memory tests are done correctly
    - ✓ How long each memory test takes


COM8:115200baud - Tera Term VT
File Edit Setup Control Window Help
Wrong!

```
Test example :1006.750732 us
PASSED!!

        Test 0 :0.135135 us
FAILED!!

        Test 1 :0.135135 us
FAILED!!

        Test 2 :0.135135 us
FAILED!!
```

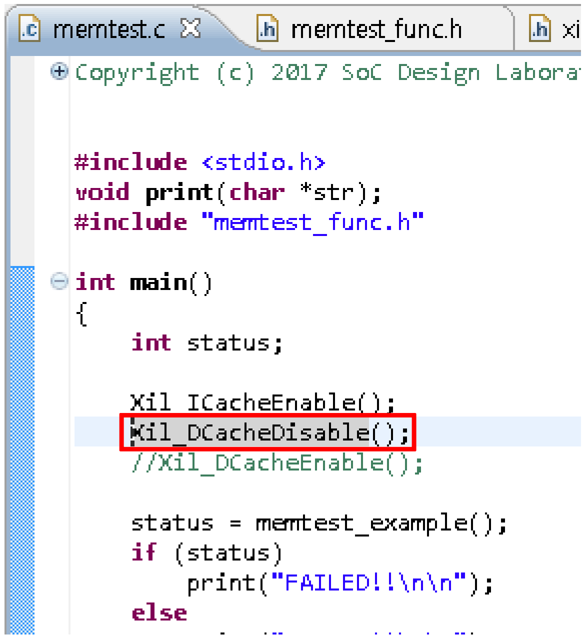

COM8:115200baud - Tera Term VT
File Edit Setup Control Window Help
Correct!

```
Test example :1006.384399 us
PASSED!!

        Test 0 :932.624634 us
PASSED!!

        Test 1 :1332.036011 us
PASSED!!

        Test 2 :1851.864868 us
PASSED!!
```

Running C Applications

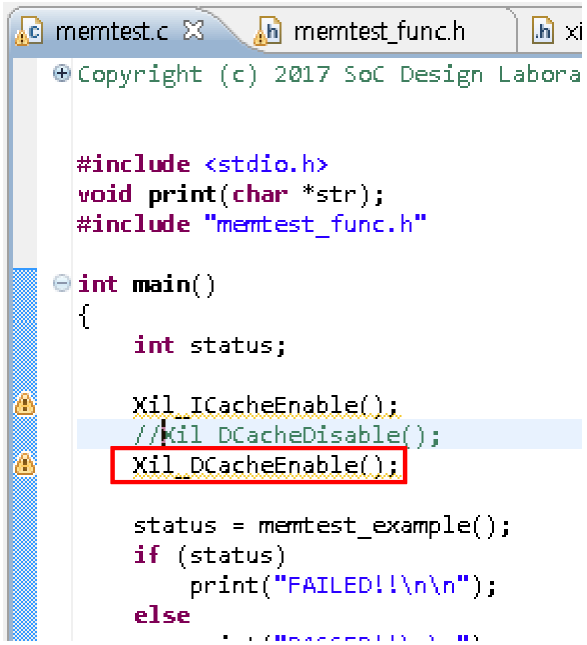
❑ Modify the application

D-Cache ***Disabled***



```
memtest.c    memtest_func.h    xi
Copyright (c) 2017 SoC Design Labora

#include <stdio.h>
void print(char *str);
#include "memtest_func.h"

int main()
{
    int status;

    Xil_ICacheEnable();
    Xil_DCacheDisable();
    //Xil_DCacheEnable();

    status = memtest_example();
    if (status)
        print("FAILED!!\n\n");
    else
```

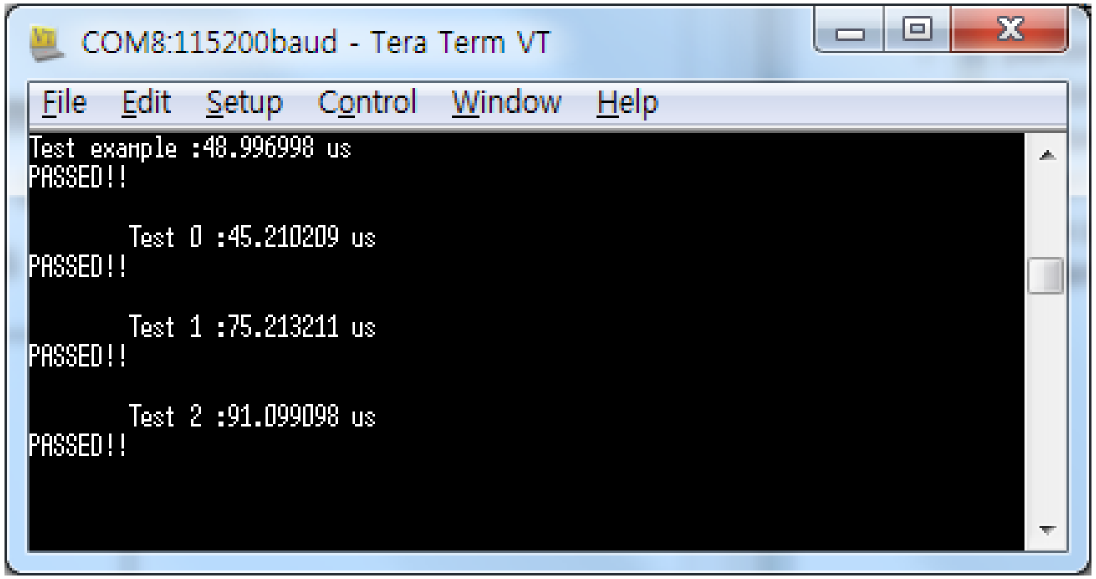
D-Cache ***Enabled***



```
memtest.c    memtest_func.h    xi
Copyright (c) 2017 SoC Design Labora

#include <stdio.h>
void print(char *str);
#include "memtest_func.h"

int main()
{
    int status;

    Xil_ICacheEnable();
    //Xil_DCacheDisable();
    Xil_DCacheEnable();

    status = memtest_example();
    if (status)
        print("FAILED!!\n\n");
    else
```

Running C Applications

- Run the application
  - Check the output on ***'Tera Term'***
    - ✓ Whether all the memory tests are done correctly
    - ✓ How long each memory test takes

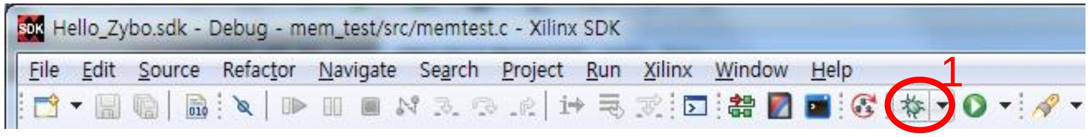

COM8:115200baud - Tera Term VT
File Edit Setup Control Window Help

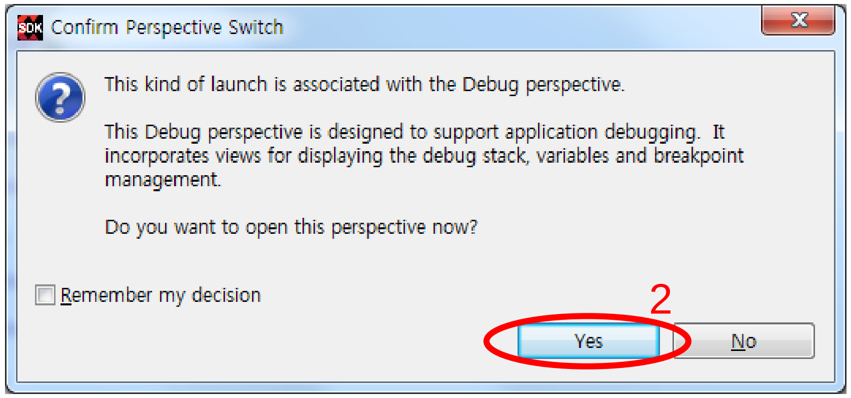
```
Test example :48.996998 us
PASSED!!

        Test 0 :45.210209 us
PASSED!!

        Test 1 :75.213211 us
PASSED!!

        Test 2 :91.099098 us
PASSED!!
```

Debugging C Applications

❑ Debug the application

- Click the ***'Debug System Debugger'*** icon and then click ***'Yes'***


Hello_Zybo.sdk - Debug - mem_test/src/memtest.c - Xilinx SDK
File Edit Source Refactor Navigate Search Project Run Xilinx Window Help
1



This kind of launch is associated with the Debug perspective.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

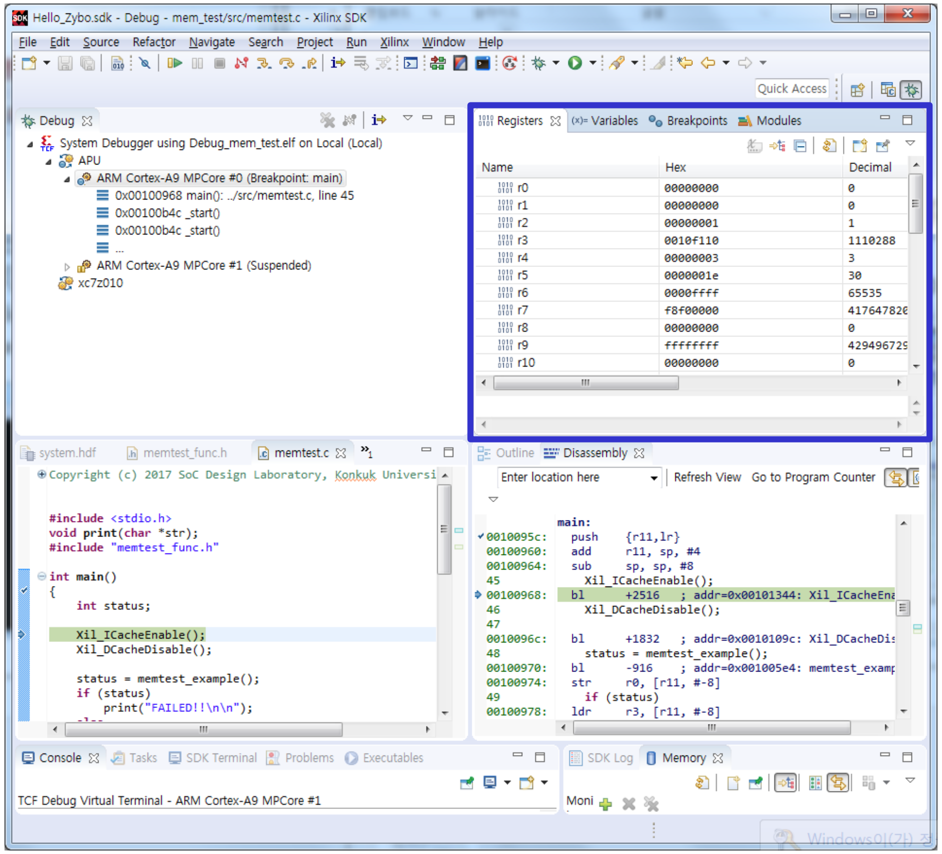
Do you want to open this perspective now?

☐ Remember my decision

Debugging C Applications

❑ Debug the application (cont'd)

- Choose the ***'Registers'*** or ***'Variable'*** tab



```
System Debugger using Debug_mem_test.elf on Local (Local)
  APU
    ARM Cortex-A9 MPCore #0 (Breakpoint: main)
      0x00100968 main(): ../src/memtest.c, line 45
      0x00100b4c _start()
      0x00100b4c _start()
      ...
    ARM Cortex-A9 MPCore #1 (Suspended)
  xc7z010
```

Name	Hex	Decimal
r0	00000000	0
r1	00000000	0
r2	00000001	1
r3	0010f110	1110288
r4	00000003	3
r5	0000001e	30
r6	0000ffff	65535
r7	f8f00000	417647820
r8	00000000	0
r9	ffffffff	429496729
r10	00000000	0

```c
Copyright (c) 2017 SoC Design Laboratory, Konkuk Universi


#include <stdio.h>
void print(char *str);
#include "memtest_func.h"

int main()
{
    int status;

    Xil_ICacheEnable();
    Xil_DCacheDisable();

    status = memtest_example();
    if (status)
        print("FAILED!!\n\n");
```

```
          main:
0010095c:   push    {r11,lr}
00100960:   add     r11, sp, #4
00100964:   sub     sp, sp, #8
45            Xil_ICacheEnable();
00100968:   bl      +2516   ; addr=0x00101344: Xil_ICacheEna
46            Xil_DCacheDisable();
47
0010096c:   bl      +1832   ; addr=0x0010109c: Xil_DCacheDis
48            status = memtest_example();
00100970:   bl      -916    ; addr=0x001005e4: memtest_examp
00100974:   str     r0, [r11, #-8]
49            if (status)
00100978:   ldr     r3, [r11, #-8]
```

```
TCF Debug Virtual Terminal - ARM Cortex-A9 MPCore #1
```

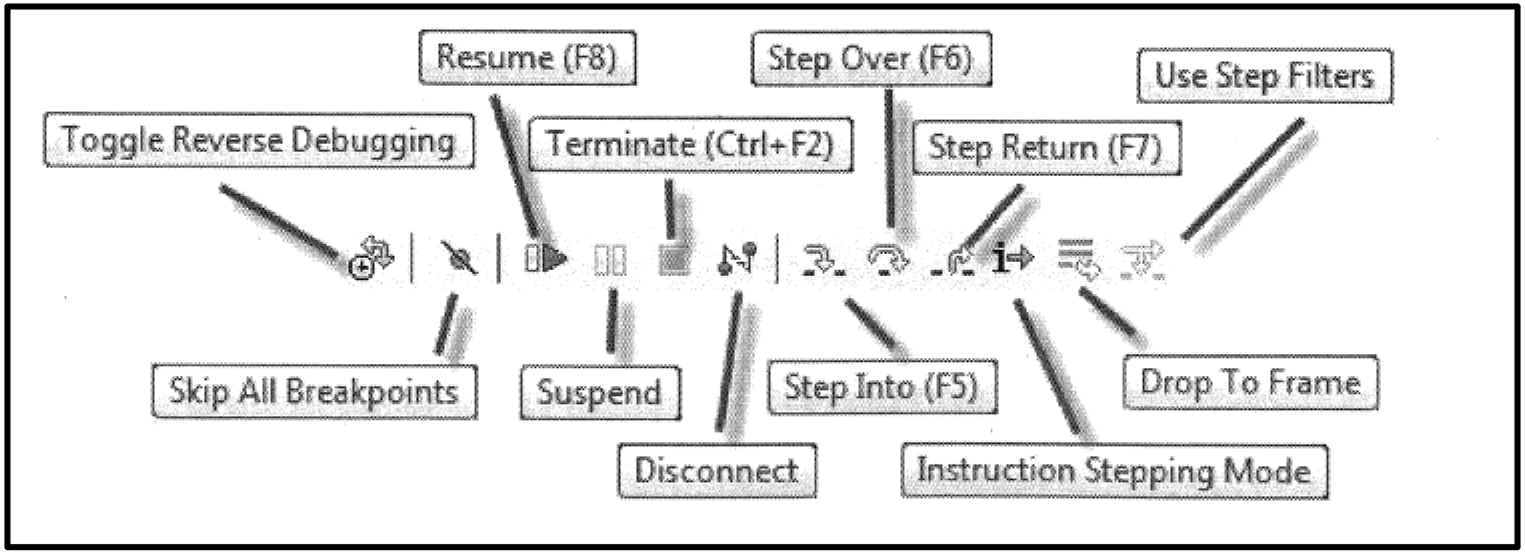
KU KONKUK UNIVERSITY

System-on-a-Chip Design LAB

Debugging C Applications

- Debug the application (cont'd)

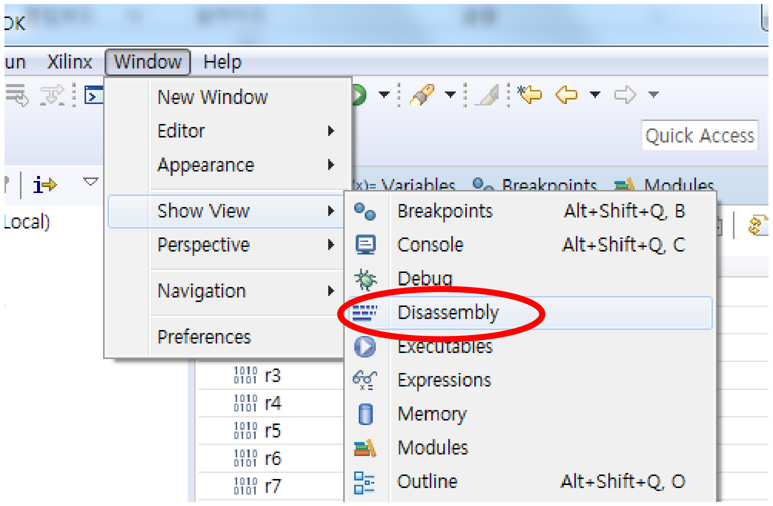
Debug Tool Bar

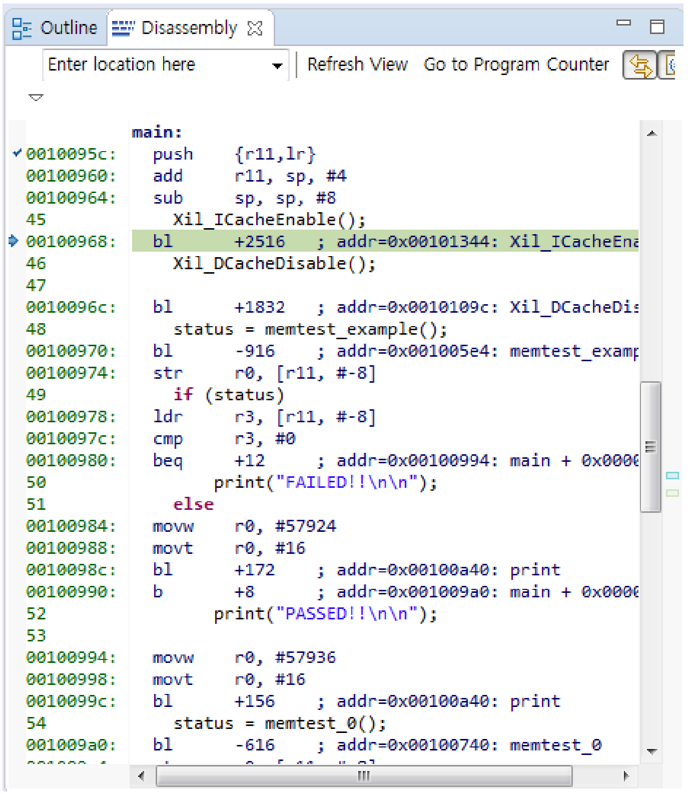


Debugging C Applications

❑ Review the disassembly

- Open the ***'Window'* > *'Show View'*** menu and then click ***'Disassembly'***



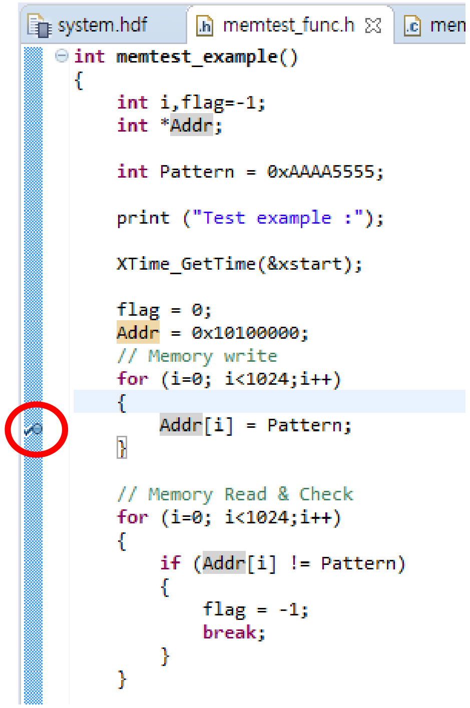


```
                main:
0010095c:         push    {r11,lr}
00100960:         add     r11, sp, #4
00100964:         sub     sp, sp, #8
45                  Xil_ICacheEnable();
00100968:         bl      +2516    ; addr=0x00101344: Xil_ICacheEna
46                  Xil_DCacheDisable();
47
0010096c:         bl      +1832    ; addr=0x0010109c: Xil_DCacheDis
48                  status = memtest_example();
00100970:         bl      -916     ; addr=0x001005e4: memtest_examp
00100974:         str     r0, [r11, #-8]
49                  if (status)
00100978:         ldr     r3, [r11, #-8]
0010097c:         cmp     r3, #0
00100980:         beq     +12      ; addr=0x00100994: main + 0x0000
50                      print("FAILED!!\n\n");
51                  else
00100984:         movw    r0, #57924
00100988:         movt    r0, #16
0010098c:         bl      +172     ; addr=0x00100a40: print
00100990:         b       +8       ; addr=0x001009a0: main + 0x0000
52                      print("PASSED!!\n\n");
53
00100994:         movw    r0, #57936
00100998:         movt    r0, #16
0010099c:         bl      +156     ; addr=0x00100a40: print
54                  status = memtest_0();
001009a0:         bl      -616     ; addr=0x00100740: memtest_0
```

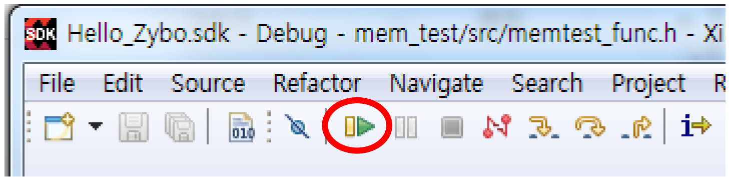
Debugging C Applications

❑ Review the disassembly (cont'd)

- Double-click on the left side of a code line to add a breakpoint
- Click the ***'Resume'*** icon to continue debugging

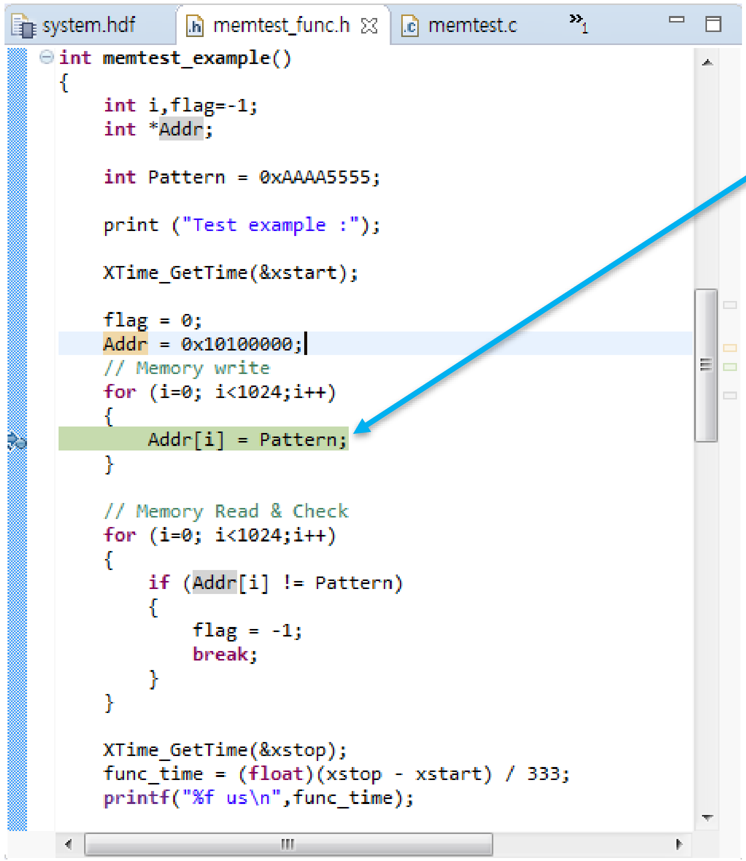


```
int memtest_example()
{
    int i,flag=-1;
    int *Addr;

    int Pattern = 0xAAAA5555;

    print ("Test example :");

    XTime_GetTime(&xstart);

    flag = 0;
    Addr = 0x10100000;
    // Memory write
    for (i=0; i<1024;i++)
    {
        Addr[i] = Pattern;
    }

    // Memory Read & Check
    for (i=0; i<1024;i++)
    {
        if (Addr[i] != Pattern)
        {
            flag = -1;
            break;
        }
    }
```

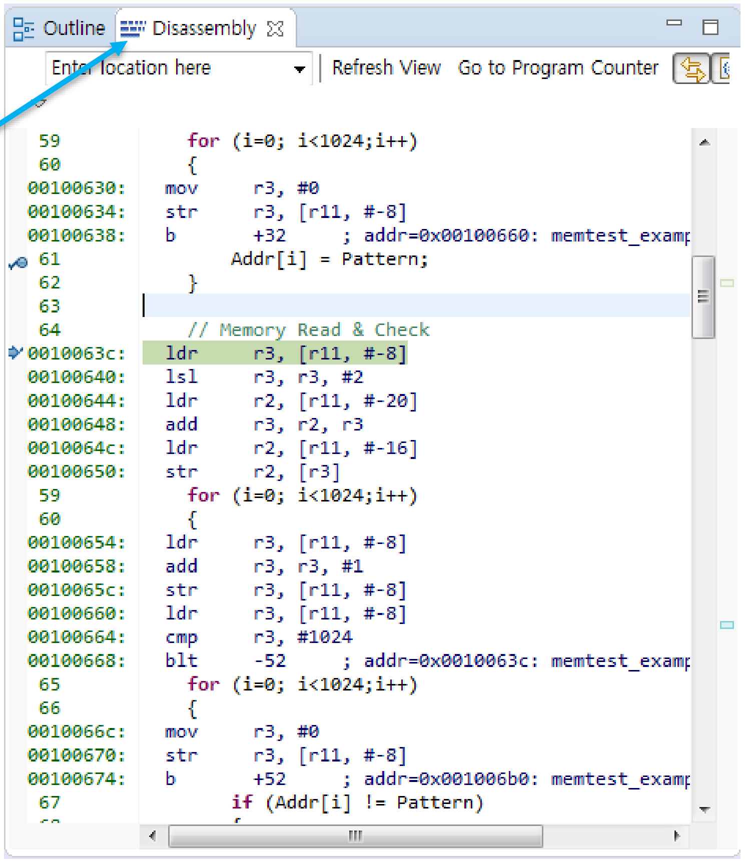


Debugging C Applications

❑ Review the disassembly (cont'd)



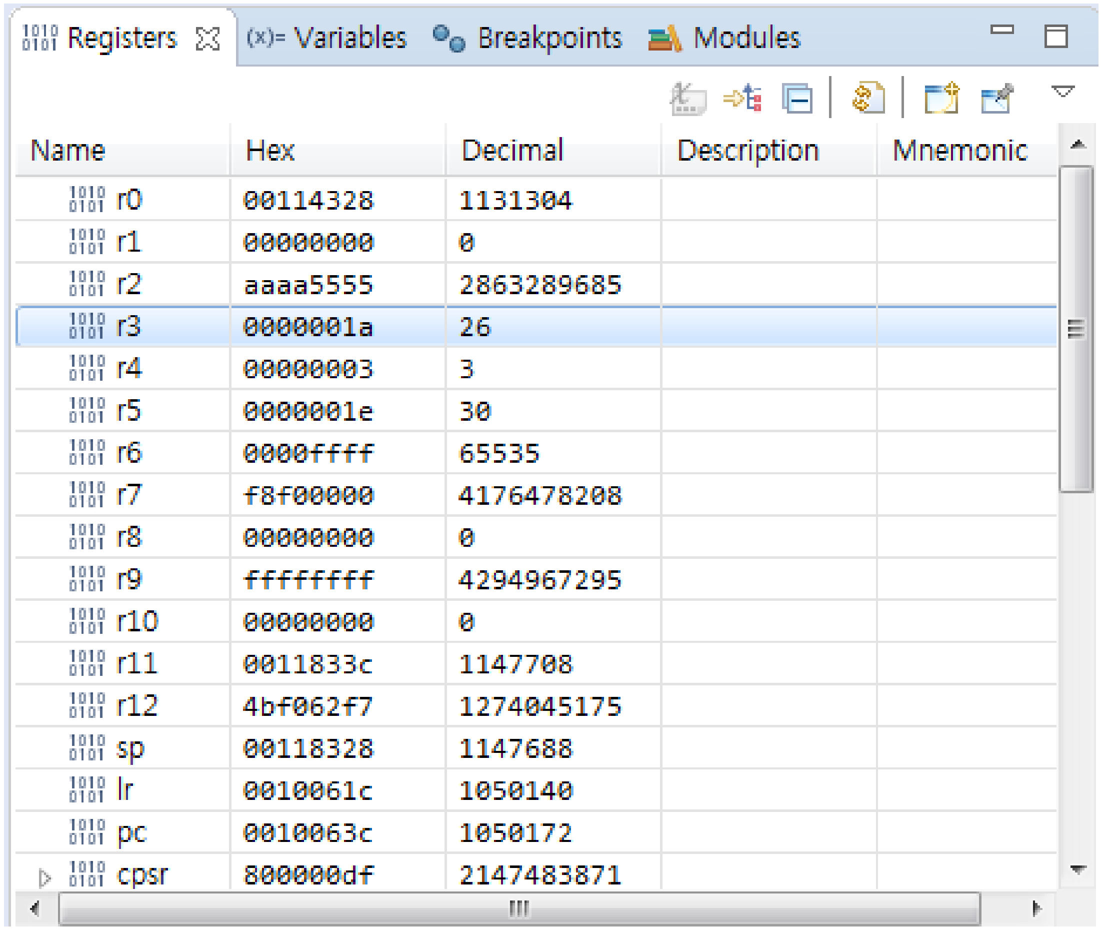
```
int memtest_example()
{
    int i,flag=-1;
    int *Addr;

    int Pattern = 0xAAAA5555;

    print ("Test example :");

    XTime_GetTime(&xstart);

    flag = 0;
    Addr = 0x10100000;
    // Memory write
    for (i=0; i<1024;i++)
    {
        Addr[i] = Pattern;
    }

    // Memory Read & Check
    for (i=0; i<1024;i++)
    {
        if (Addr[i] != Pattern)
        {
            flag = -1;
            break;
        }
    }

    XTime_GetTime(&xstop);
    func_time = (float)(xstop - xstart) / 333;
    printf("%f us\n",func_time);
```



```
59              for (i=0; i<1024;i++)
60              {
00100630:   mov     r3, #0
00100634:   str     r3, [r11, #-8]
00100638:   b       +32      ; addr=0x00100660: memtest_examp
61                  Addr[i] = Pattern;
62              }
63
64              // Memory Read & Check
0010063c:   ldr     r3, [r11, #-8]
00100640:   lsl     r3, r3, #2
00100644:   ldr     r2, [r11, #-20]
00100648:   add     r3, r2, r3
0010064c:   ldr     r2, [r11, #-16]
00100650:   str     r2, [r3]
59              for (i=0; i<1024;i++)
60              {
00100654:   ldr     r3, [r11, #-8]
00100658:   add     r3, r3, #1
0010065c:   str     r3, [r11, #-8]
00100660:   ldr     r3, [r11, #-8]
00100664:   cmp     r3, #1024
00100668:   blt     -52      ; addr=0x0010063c: memtest_examp
65              for (i=0; i<1024;i++)
66              {
0010066c:   mov     r3, #0
00100670:   str     r3, [r11, #-8]
00100674:   b       +52      ; addr=0x001006b0: memtest_examp
67                  if (Addr[i] != Pattern)
```

Debugging C Applications

❑ Check the content of a register

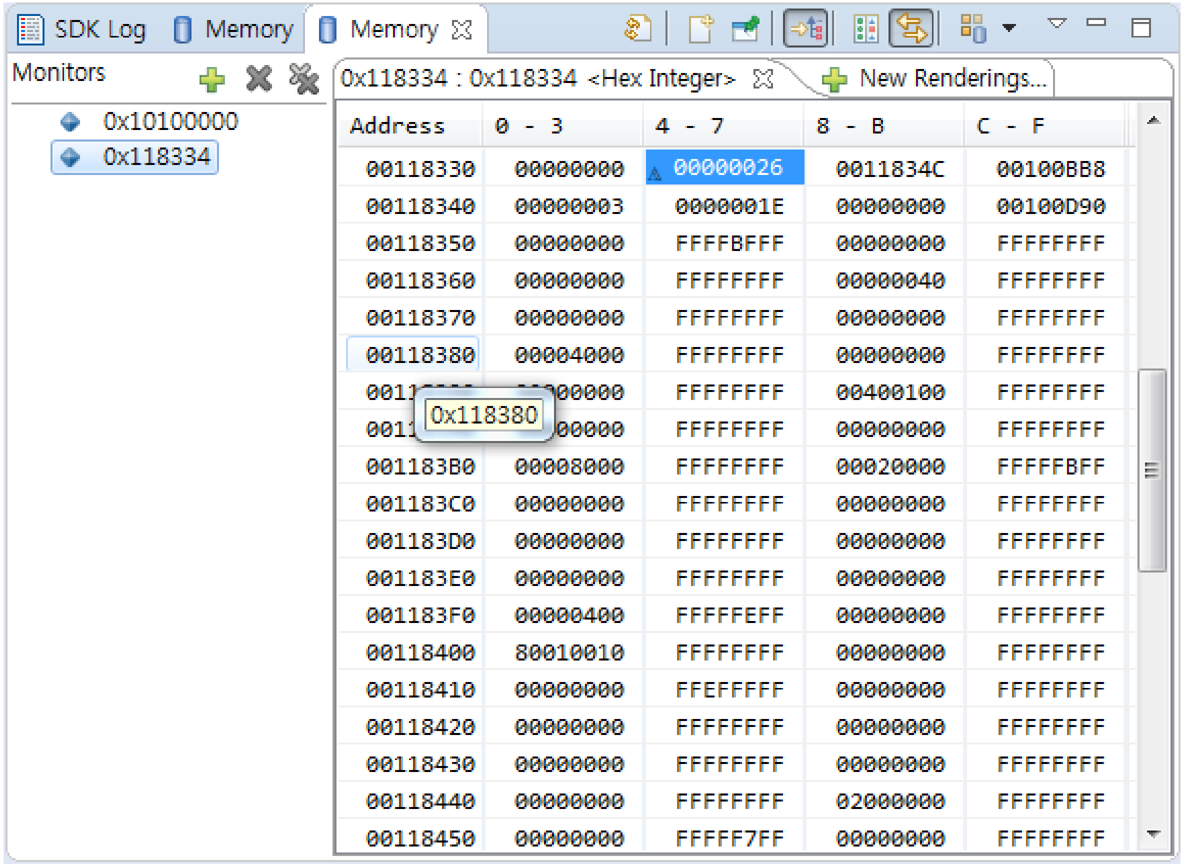


Registers | (x)= Variables | Breakpoints | Modules

Name	Hex	Decimal	Description	Mnemonic
r0	00114328	1131304		
r1	00000000	0		
r2	aaaa5555	2863289685		
r3	0000001a	26		
r4	00000003	3		
r5	0000001e	30		
r6	0000ffff	65535		
r7	f8f00000	4176478208		
r8	00000000	0		
r9	ffffffff	4294967295		
r10	00000000	0		
r11	0011833c	1147708		
r12	4bf062f7	1274045175		
sp	00118328	1147688		
lr	0010061c	1050140		
pc	0010063c	1050172		
cpsr	800000df	2147483871		

Debugging C Applications

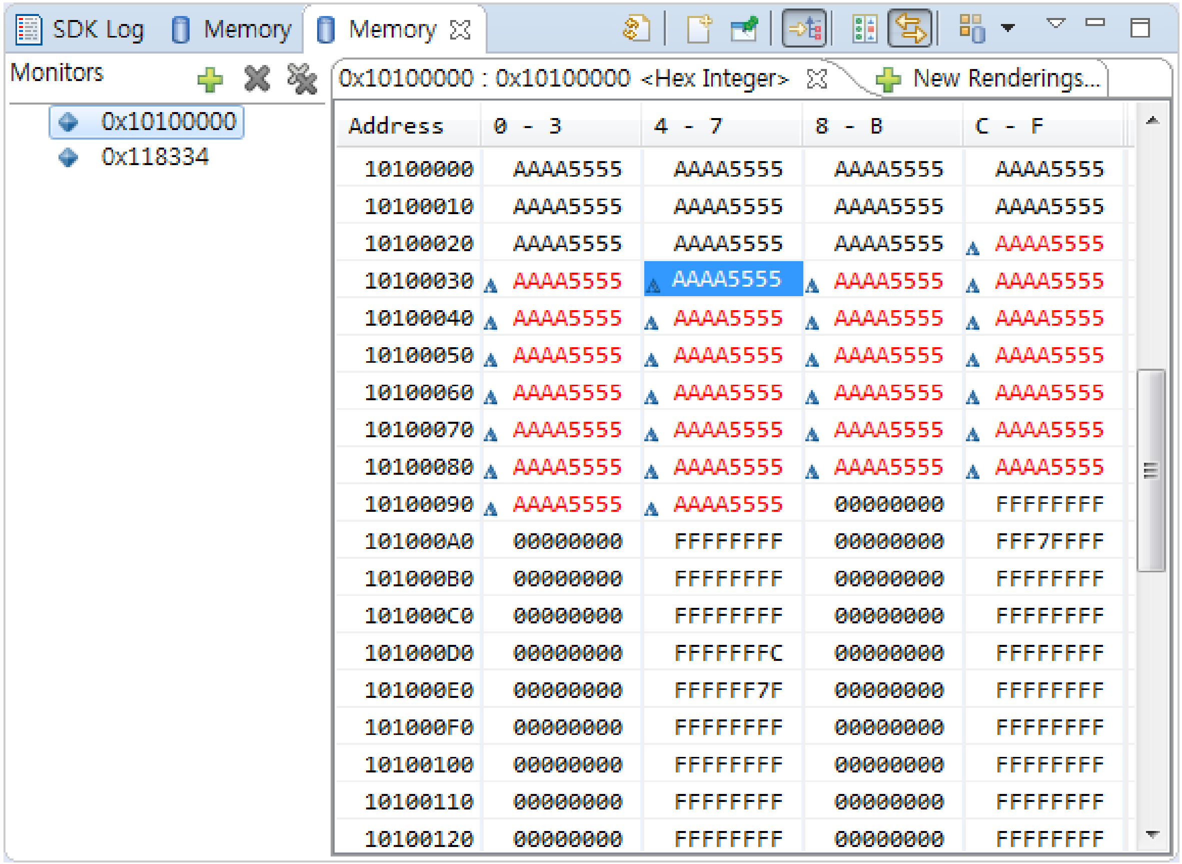
- ❑ Check the content of a memory location
  - Location for the loop variable (i)



```
Address   0 - 3     4 - 7     8 - B     C - F
00118330  00000000  00000026  0011834C  00100BB8
00118340  00000003  0000001E  00000000  00100D90
00118350  00000000  FFFFBFFF  00000000  FFFFFFFF
00118360  00000000  FFFFFFFF  00000040  FFFFFFFF
00118370  00000000  FFFFFFFF  00000000  FFFFFFFF
00118380  00004000  FFFFFFFF  00000000  FFFFFFFF
0011[illegible]  [illegible]00000  FFFFFFFF  00400100  FFFFFFFF
001[illegible]  [illegible]00000  FFFFFFFF  00000000  FFFFFFFF
001183B0  00008000  FFFFFFFF  00020000  FFFFFBFF
001183C0  00000000  FFFFFFFF  00000000  FFFFFFFF
001183D0  00000000  FFFFFFFF  00000000  FFFFFFFF
001183E0  00000000  FFFFFFFF  00000000  FFFFFFFF
001183F0  00000400  FFFFFEFF  00000000  FFFFFFFF
00118400  80010010  FFFFFFFF  00000000  FFFFFFFF
00118410  00000000  FFEFFFFF  00000000  FFFFFFFF
00118420  00000000  FFFFFFFF  00000000  FFFFFFFF
00118430  00000000  FFFFFFFF  00000000  FFFFFFFF
00118440  00000000  FFFFFFFF  02000000  FFFFFFFF
00118450  00000000  FFFFF7FF  00000000  FFFFFFFF
```

Debugging C Applications

- Check the content of a memory location
  - Location for the array (Addr)



SDK Log | Memory | Memory

Monitors

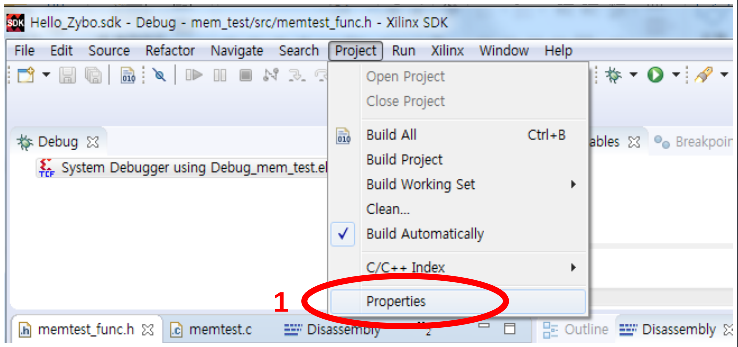
- 0x10100000
- 0x118334

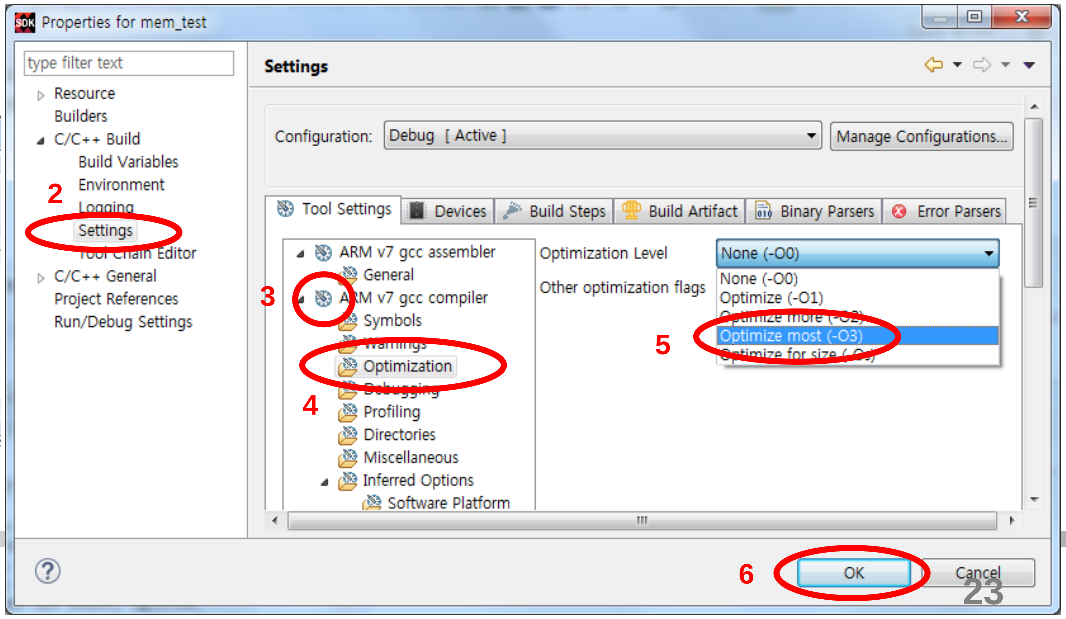
0x10100000 : 0x10100000 <Hex Integer>   New Renderings...

```
Address     0 - 3      4 - 7      8 - B      C - F
10100000    AAAA5555   AAAA5555   AAAA5555   AAAA5555
10100010    AAAA5555   AAAA5555   AAAA5555   AAAA5555
10100020    AAAA5555   AAAA5555   AAAA5555   AAAA5555
10100030    AAAA5555   AAAA5555   AAAA5555   AAAA5555
10100040    AAAA5555   AAAA5555   AAAA5555   AAAA5555
10100050    AAAA5555   AAAA5555   AAAA5555   AAAA5555
10100060    AAAA5555   AAAA5555   AAAA5555   AAAA5555
10100070    AAAA5555   AAAA5555   AAAA5555   AAAA5555
10100080    AAAA5555   AAAA5555   AAAA5555   AAAA5555
10100090    AAAA5555   AAAA5555   00000000   FFFFFFFF
101000A0    00000000   FFFFFFFF   00000000   FFF7FFFF
101000B0    00000000   FFFFFFFF   00000000   FFFFFFFF
101000C0    00000000   FFFFFFFF   00000000   FFFFFFFF
101000D0    00000000   FFFFFFFC   00000000   FFFFFFFF
101000E0    00000000   FFFFFF7F   00000000   FFFFFFFF
101000F0    00000000   FFFFFFFF   00000000   FFFFFFFF
10100100    00000000   FFFFFFFF   00000000   FFFFFFFF
10100110    00000000   FFFFFFFF   00000000   FFFFFFFF
10100120    00000000   FFFFFFFF   00000000   FFFFFFFF
```

Optimizing C Applications

- Set the compiler optimization level
  - Open the ***'Project'*** menu and then click ***'Properties'***
  - Select ***'Settings'*** tap and then click ***'ARM gcc compiler' > 'Optimization'***
  - Select ***'Optimization most (-O3)'*** in the dropdown menu of ***'Optimization Level'***
  - Click ***'OK'***

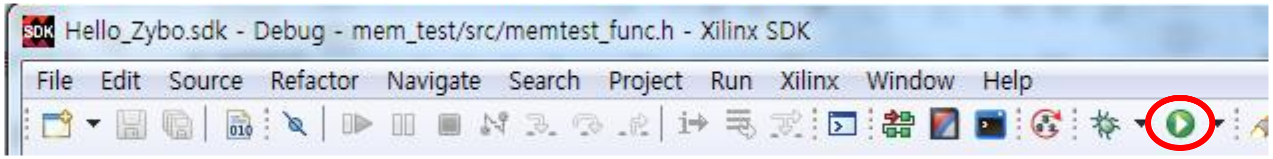

Hello_Zybo.sdk - Debug - mem_test/src/memtest_func.h - Xilinx SDK
File Edit Source Refactor Navigate Search Project Run Xilinx Window Help
Open Project
Close Project
Build All Ctrl+B
Build Project
Build Working Set
Clean...
Build Automatically
C/C++ Index
Properties
1
Debug
System Debugger using Debug_mem_test.elf
memtest_func.h
memtest.c
Disassembly
Outline
Disassembly

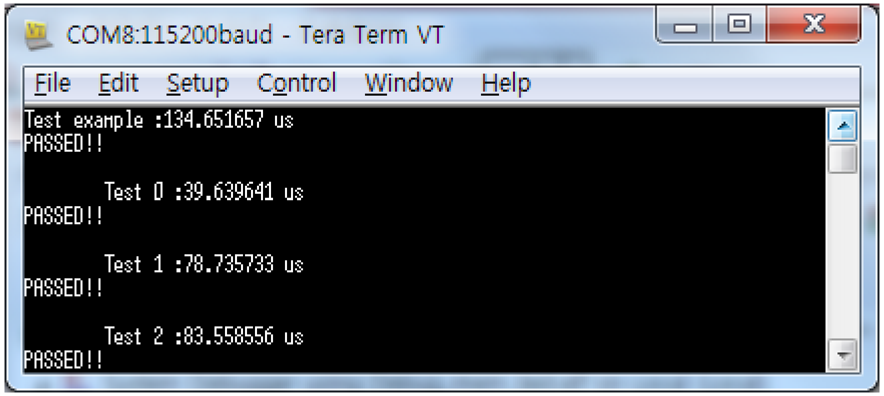


Optimizing C Applications

❑ Run the application

- Click the ***'Run As'*** icon to run the application again
- Check the output on ***'Tera Term'***
  - ✓ Check how much it accelerates the memory tests


Hello_Zybo.sdk - Debug - mem_test/src/memtest_func.h - Xilinx SDK
File Edit Source Refactor Navigate Search Project Run Xilinx Window Help

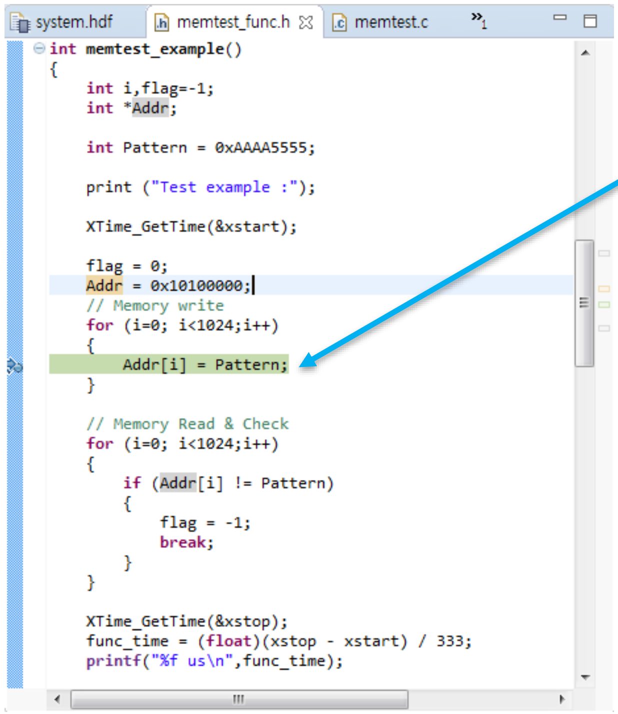

COM8:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
Test example :134.651657 us
PASSED!!

        Test 0 :39.639641 us
PASSED!!

        Test 1 :78.735733 us
PASSED!!

        Test 2 :83.558556 us
PASSED!!
```

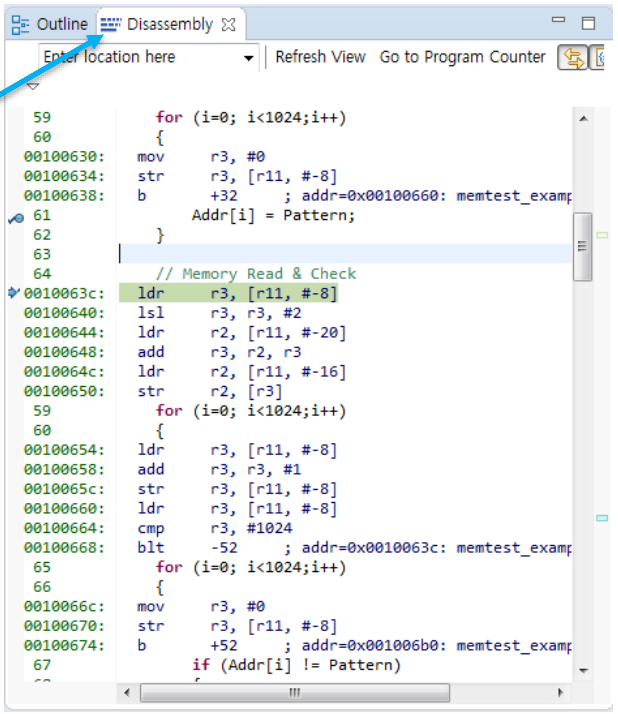
Optimizing C Applications

❑ Debug the application

- Repeat pp.**14~22** of this lab workbookto figure out the impact of the compiler optimization level on the assembly codes



```
int memtest_example()
{
    int i,flag=-1;
    int *Addr;

    int Pattern = 0xAAAA5555;

    print ("Test example :");

    XTime_GetTime(&xstart);

    flag = 0;
    Addr = 0x10100000;
    // Memory write
    for (i=0; i<1024;i++)
    {
        Addr[i] = Pattern;
    }

    // Memory Read & Check
    for (i=0; i<1024;i++)
    {
        if (Addr[i] != Pattern)
        {
            flag = -1;
            break;
        }
    }

    XTime_GetTime(&xstop);
    func_time = (float)(xstop - xstart) / 333;
    printf("%f us\n",func_time);
```



```
59              for (i=0; i<1024;i++)
60              {
00100630:   mov     r3, #0
00100634:   str     r3, [r11, #-8]
00100638:   b       +32         ; addr=0x00100660: memtest_examp
61                  Addr[i] = Pattern;
62              }
63
64              // Memory Read & Check
0010063c:   ldr     r3, [r11, #-8]
00100640:   lsl     r3, r3, #2
00100644:   ldr     r2, [r11, #-20]
00100648:   add     r3, r2, r3
0010064c:   ldr     r2, [r11, #-16]
00100650:   str     r2, [r3]
59              for (i=0; i<1024;i++)
60              {
00100654:   ldr     r3, [r11, #-8]
00100658:   add     r3, r3, #1
0010065c:   str     r3, [r11, #-8]
00100660:   ldr     r3, [r11, #-8]
00100664:   cmp     r3, #1024
00100668:   blt     -52         ; addr=0x0010063c: memtest_examp
65              for (i=0; i<1024;i++)
66              {
0010066c:   mov     r3, #0
00100670:   str     r3, [r11, #-8]
00100674:   b       +52         ; addr=0x001006b0: memtest_examp
67                  if (Addr[i] != Pattern)
```

Demo

- ❑ Compare the (normalized) execution times of all the **four** memory tests as follows
  - Optimization level: O0
    - ✓ D-cache disabled/enabled
  - Optimization level: O3
    - ✓ D-cache disabled/enabled
- ❑ Figure out the reasons for speed up