[Microprocessor Applications]
Lab 2: Memory Access

Chester Sungchung Park

SoC Design Lab, Konkuk University

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

Outline

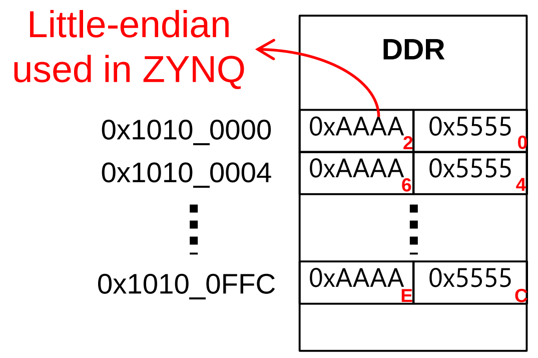
- ❑ Objectives
- ❑ Description
- ❑ Block diagram
- ❑ Address map
- ❑ Section map
- ❑ Source codes
- ❑ Evaluation

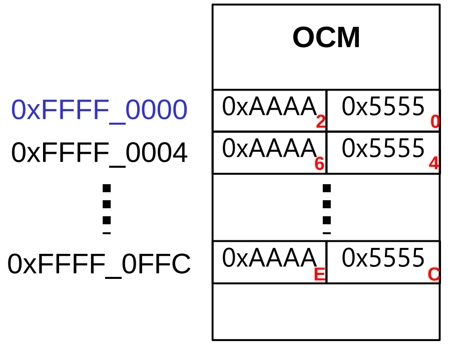
Objectives

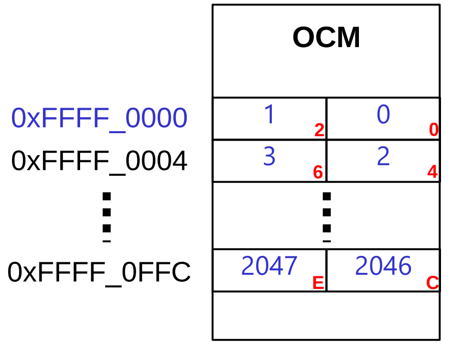
- ❑ Running a C application that writes and reads different regions of memory
- ❑ Running a C application that writes and reads memory addresses with different access sizes
- ❑ Understanding the corresponding assembly codes

Description

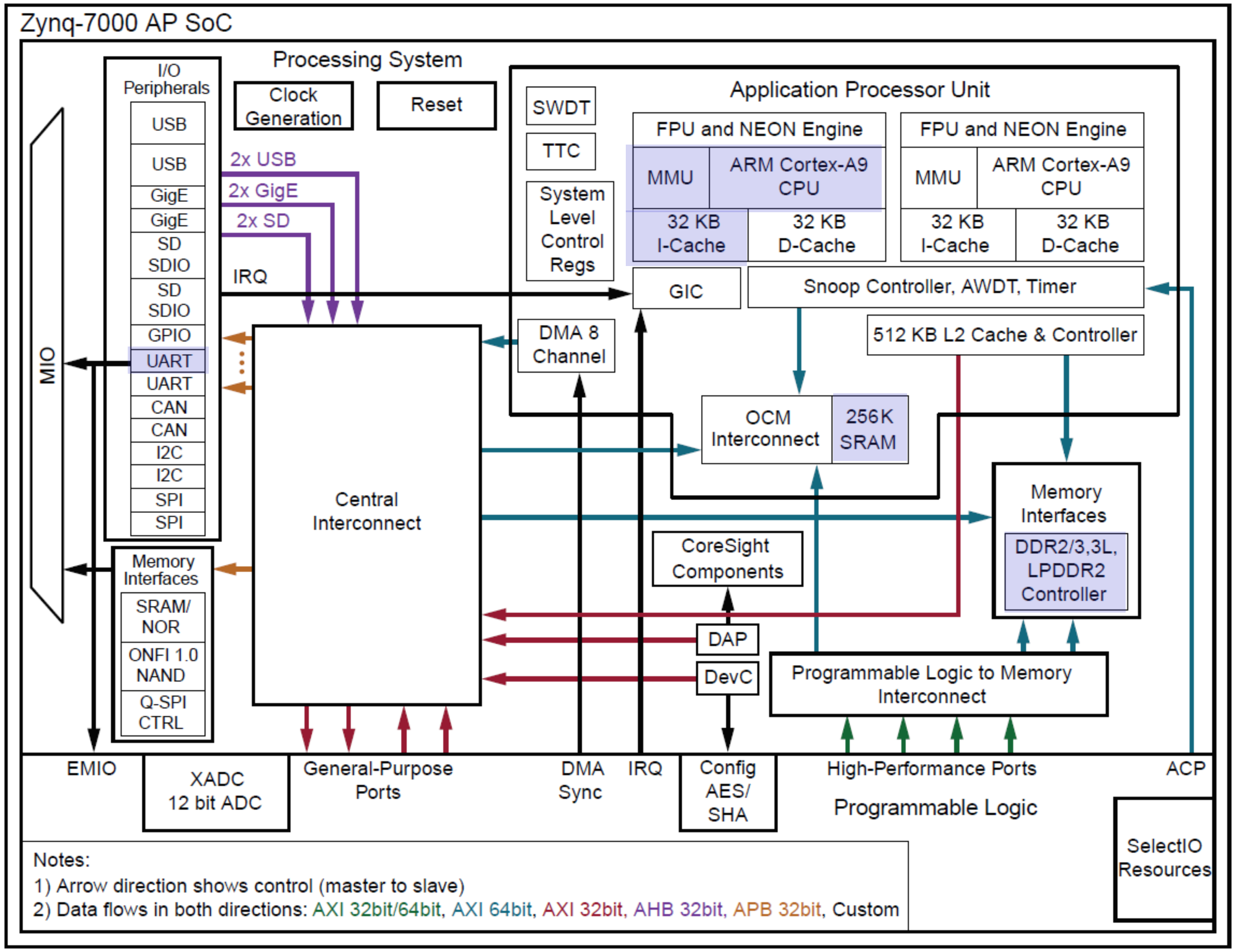
- Example
  - Access size: 32 bits (word)
  - Pattern: 0xAAAA_5555 (1024 words)
  - Region: 0x1010_0000 ~ 0x1010_0FFC (DDR)
- Test0
  - Access size: 32 bits (word)
  - Pattern: 0xAAAA_5555 (1024 words)
  - Region: 0xFFFF_0000 ~ 0xFFFF_0FFC (OCM)
- Test1
  - Access size: 16 bits (halfword)
  - Pattern: 0xAAAA_5555 (1024 words)
  - Region: 0xFFFF_0000 ~ 0xFFFF_0FFC (OCM)
- Test2
  - Access size: 16 bits (halfword)
  - Pattern: halfword offset (0 ~ 2047)
  - Region: 0xFFFF_0000 ~ 0xFFFF_0FFC (OCM)





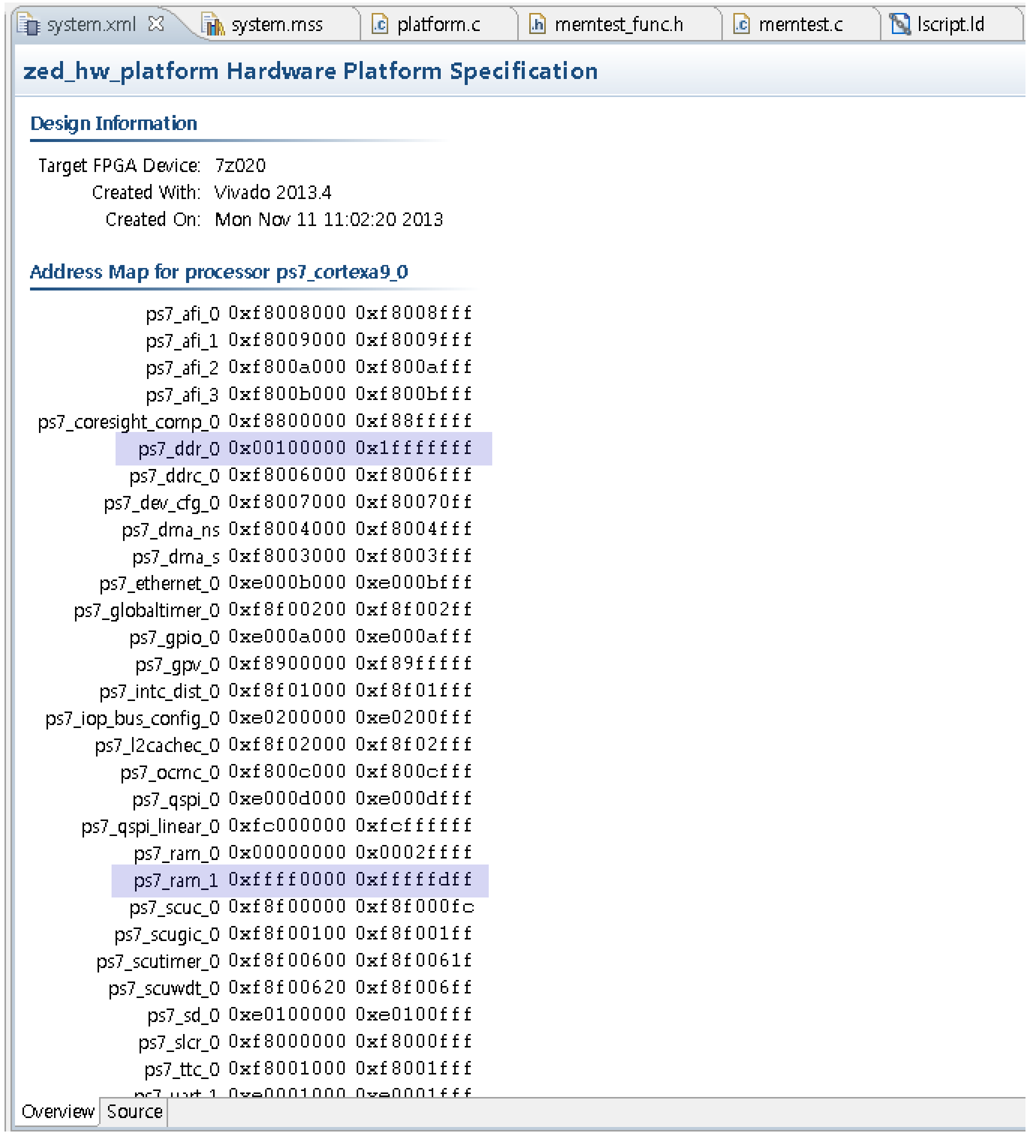


Block Diagram



Notes:
1) Arrow direction shows control (master to slave)
2) Data flows in both directions: AXI 32bit/64bit, AXI 64bit, AXI 32bit, AHB 32bit, APB 32bit, Custom

Address Map



zed_hw_platform Hardware Platform Specification

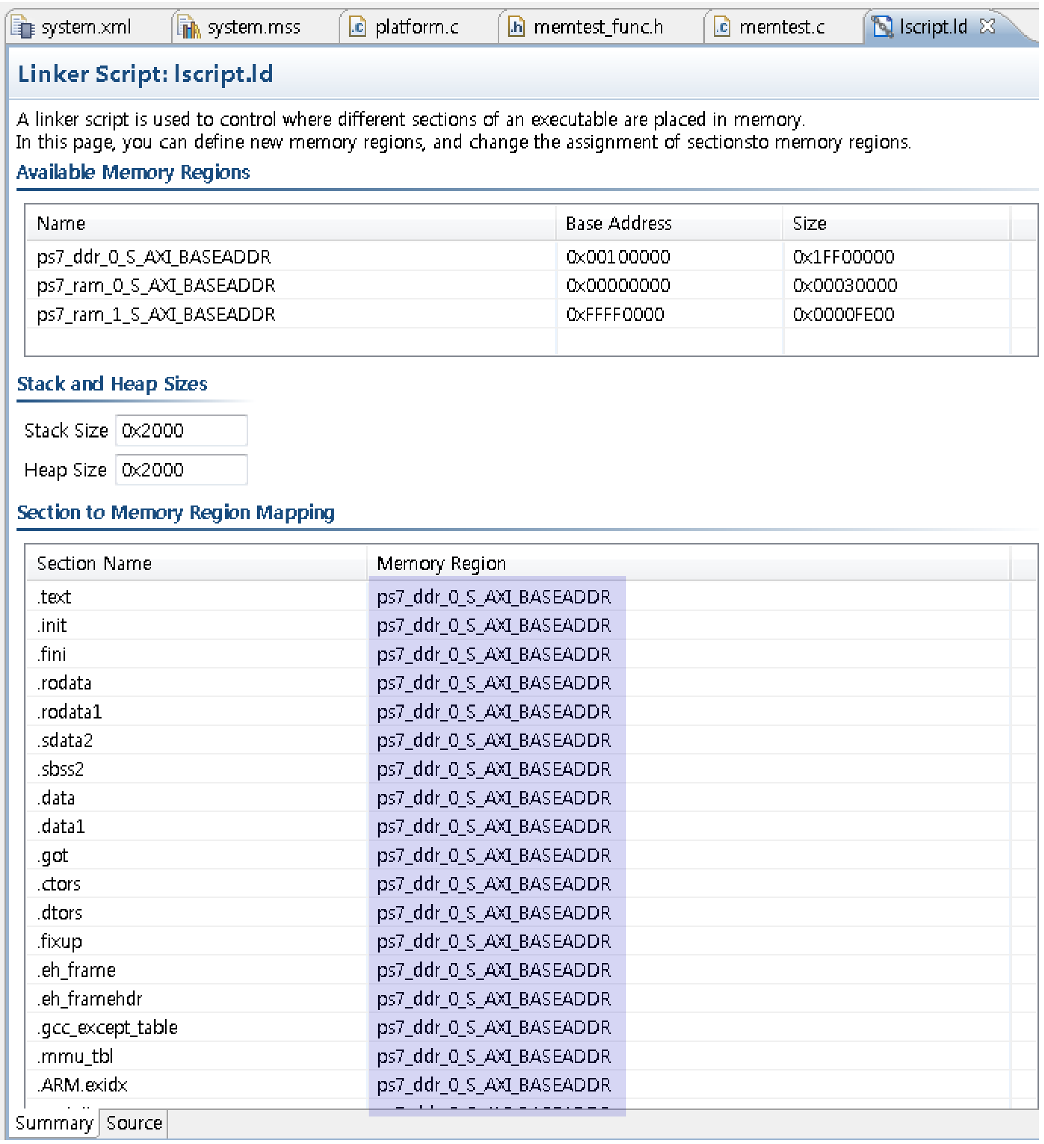
Design Information

Target FPGA Device:	7z020
Created With:	Vivado 2013.4
Created On:	Mon Nov 11 11:02:20 2013

Address Map for processor ps7_cortexa9_0

ps7_afi_0	0xf8008000	0xf8008fff
ps7_afi_1	0xf8009000	0xf8009fff
ps7_afi_2	0xf800a000	0xf800afff
ps7_afi_3	0xf800b000	0xf800bfff
ps7_coresight_comp_0	0xf8800000	0xf88fffff
ps7_ddr_0	0x00100000	0x1fffffff
ps7_ddrc_0	0xf8006000	0xf8006fff
ps7_dev_cfg_0	0xf8007000	0xf80070ff
ps7_dma_ns	0xf8004000	0xf8004fff
ps7_dma_s	0xf8003000	0xf8003fff
ps7_ethernet_0	0xe000b000	0xe000bfff
ps7_globaltimer_0	0xf8f00200	0xf8f002ff
ps7_gpio_0	0xe000a000	0xe000afff
ps7_gpv_0	0xf8900000	0xf89fffff
ps7_intc_dist_0	0xf8f01000	0xf8f01fff
ps7_iop_bus_config_0	0xe0200000	0xe0200fff
ps7_l2cachec_0	0xf8f02000	0xf8f02fff
ps7_ocmc_0	0xf800c000	0xf800cfff
ps7_qspi_0	0xe000d000	0xe000dfff
ps7_qspi_linear_0	0xfc000000	0xfcffffff
ps7_ram_0	0x00000000	0x0002ffff
ps7_ram_1	0xffff0000	0xfffffdff
ps7_scuc_0	0xf8f00000	0xf8f000fc
ps7_scugic_0	0xf8f00100	0xf8f001ff
ps7_scutimer_0	0xf8f00600	0xf8f0061f
ps7_scuwdt_0	0xf8f00620	0xf8f006ff
ps7_sd_0	0xe0100000	0xe0100fff
ps7_slcr_0	0xf8000000	0xf8000fff
ps7_ttc_0	0xf8001000	0xf8001fff
ps7_uart_1	0xe0001000	0xe0001fff

Section Map


system.xml | system.mss | platform.c | memtest_func.h | memtest.c | lscript.ld

Linker Script: lscript.ld

A linker script is used to control where different sections of an executable are placed in memory.
In this page, you can define new memory regions, and change the assignment of sectionsto memory regions.

Available Memory Regions

Name	Base Address	Size
ps7_ddr_0_S_AXI_BASEADDR	0x00100000	0x1FF00000
ps7_ram_0_S_AXI_BASEADDR	0x00000000	0x00030000
ps7_ram_1_S_AXI_BASEADDR	0xFFFF0000	0x0000FE00

Stack and Heap Sizes

Stack Size 0x2000

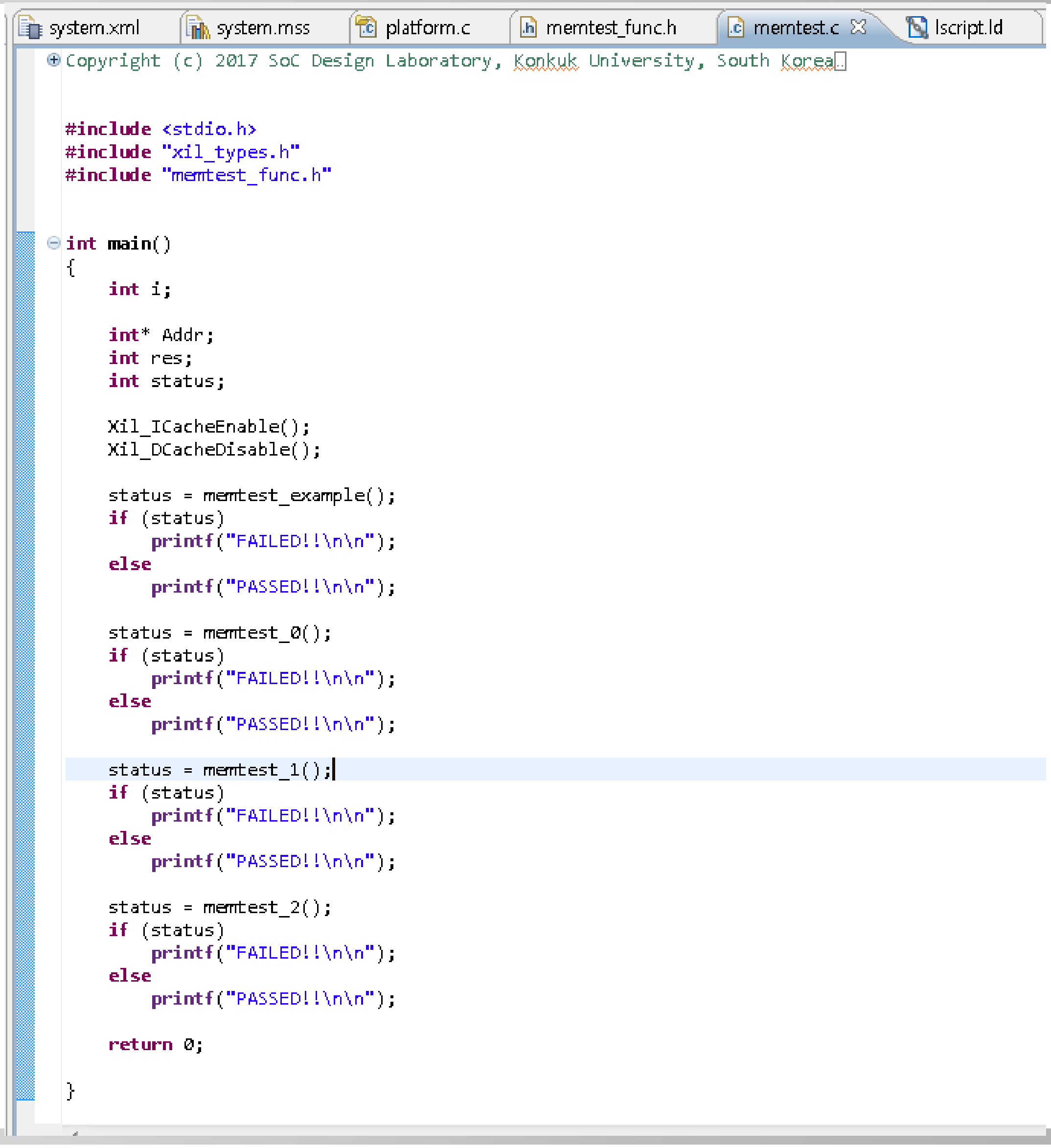
Heap Size 0x2000

Section to Memory Region Mapping

Section Name	Memory Region
.text	ps7_ddr_0_S_AXI_BASEADDR
.init	ps7_ddr_0_S_AXI_BASEADDR
.fini	ps7_ddr_0_S_AXI_BASEADDR
.rodata	ps7_ddr_0_S_AXI_BASEADDR
.rodata1	ps7_ddr_0_S_AXI_BASEADDR
.sdata2	ps7_ddr_0_S_AXI_BASEADDR
.sbss2	ps7_ddr_0_S_AXI_BASEADDR
.data	ps7_ddr_0_S_AXI_BASEADDR
.data1	ps7_ddr_0_S_AXI_BASEADDR
.got	ps7_ddr_0_S_AXI_BASEADDR
.ctors	ps7_ddr_0_S_AXI_BASEADDR
.dtors	ps7_ddr_0_S_AXI_BASEADDR
.fixup	ps7_ddr_0_S_AXI_BASEADDR
.eh_frame	ps7_ddr_0_S_AXI_BASEADDR
.eh_framehdr	ps7_ddr_0_S_AXI_BASEADDR
.gcc_except_table	ps7_ddr_0_S_AXI_BASEADDR
.mmu_tbl	ps7_ddr_0_S_AXI_BASEADDR
.ARM.exidx	ps7_ddr_0_S_AXI_BASEADDR

Source Codes

- helloworld.c

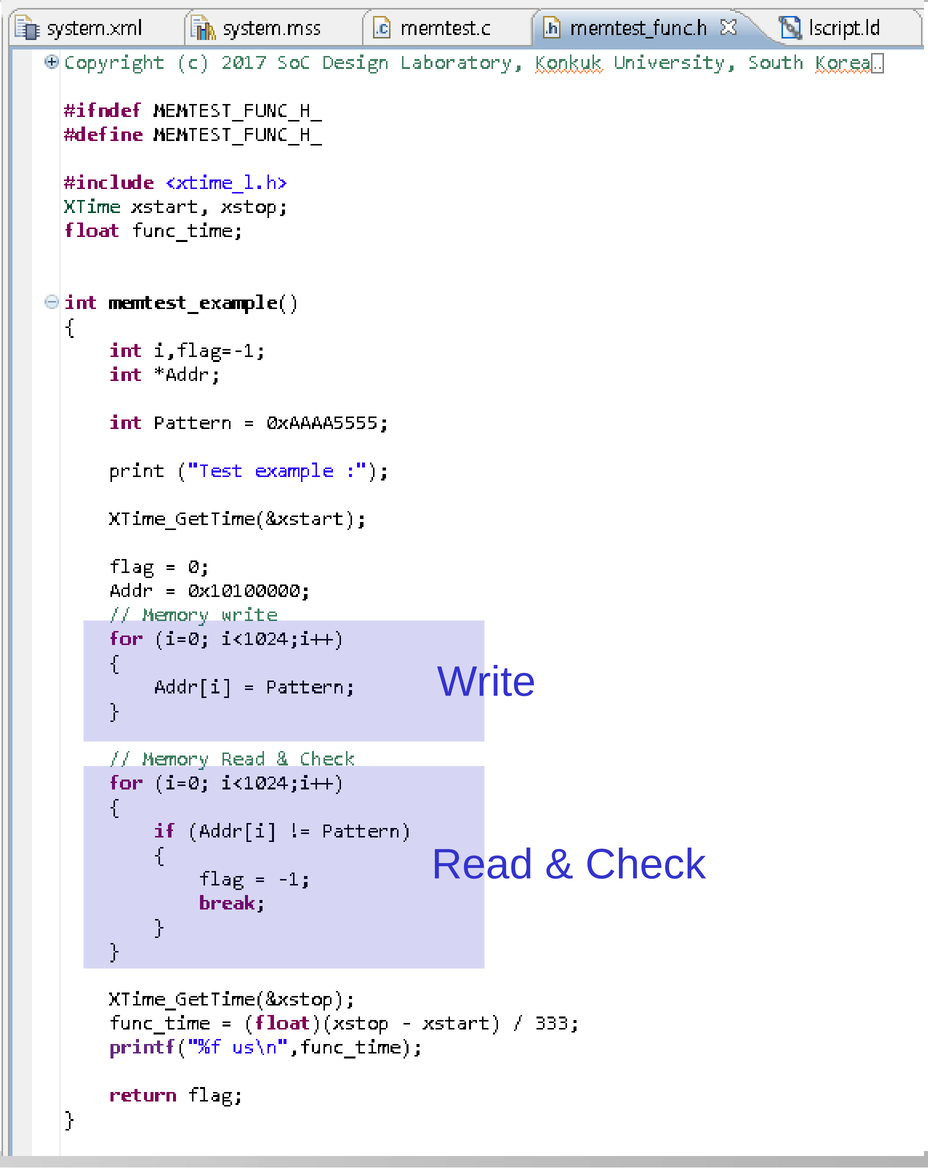


system.xml | system.mss | platform.c | memtest_func.h | memtest.c | lscript.ld

```c
Copyright (c) 2017 SoC Design Laboratory, Konkuk University, South Korea


#include <stdio.h>
#include "xil_types.h"
#include "memtest_func.h"


int main()
{
    int i;

    int* Addr;
    int res;
    int status;

    Xil_ICacheEnable();
    Xil_DCacheDisable();

    status = memtest_example();
    if (status)
        printf("FAILED!!\n\n");
    else
        printf("PASSED!!\n\n");

    status = memtest_0();
    if (status)
        printf("FAILED!!\n\n");
    else
        printf("PASSED!!\n\n");

    status = memtest_1();
    if (status)
        printf("FAILED!!\n\n");
    else
        printf("PASSED!!\n\n");

    status = memtest_2();
    if (status)
        printf("FAILED!!\n\n");
    else
        printf("PASSED!!\n\n");

    return 0;

}
```

Source Codes

- memtest_func.h



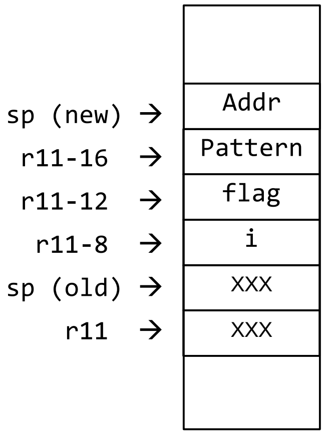
```c
Copyright (c) 2017 SoC Design Laboratory, Konkuk University, South Korea

#ifndef MEMTEST_FUNC_H_
#define MEMTEST_FUNC_H_

#include <xtime_l.h>
XTime xstart, xstop;
float func_time;


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);

    return flag;
}
```

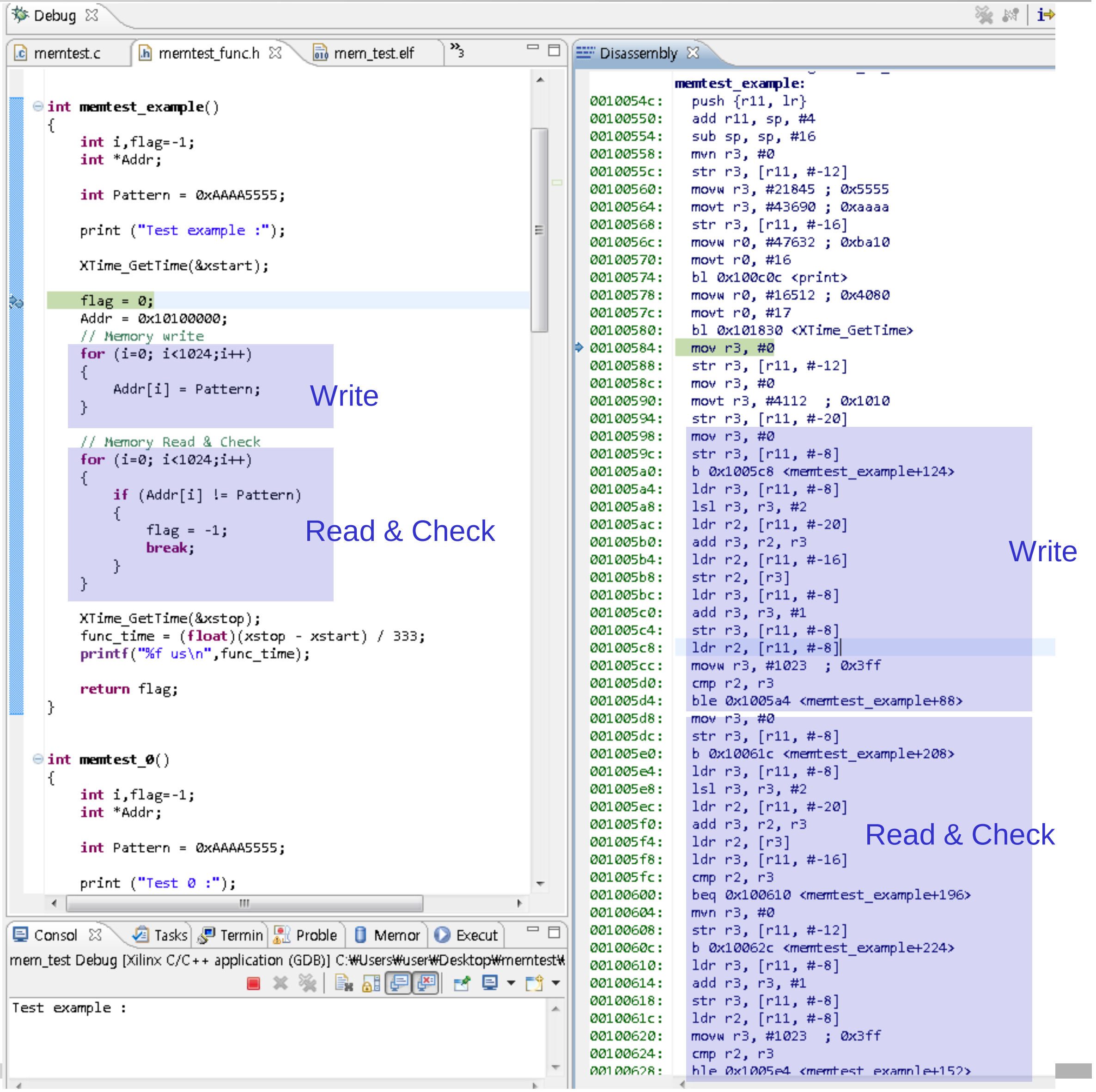
Evaluation

❑ Compiler optimization levels (ARM gcc)

- -O0: No optimization is performed.
- -O1: Enables the most common forms of optimization that do not require decisions regarding size or speed.
- -O2: Enables further optimizations, such as instruction scheduling.
- -O3: Enables more aggressive optimizations, such as aggressive function inlining, and it typically increases speed at the expense of image size. Moreover, this option enables -ftree-vectorize, causing the compiler to attempt to automatically generate NEON code.
- -Os: Selects optimizations that attempt to minimize the size of the image, even at the expense of speed.

Assembly Codes (-O0)





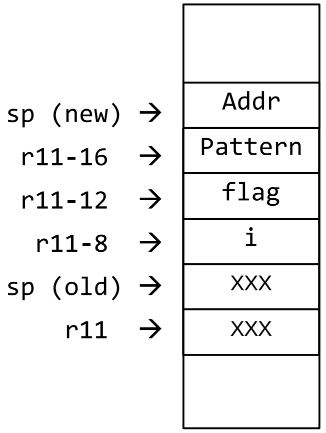
```
int memtest_example()
{
    int i,flag=-1;
    int *Addr;

    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);

    return flag;
}

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

    int Pattern = 0xAAAA5555;

    print ("Test 0 :");
```

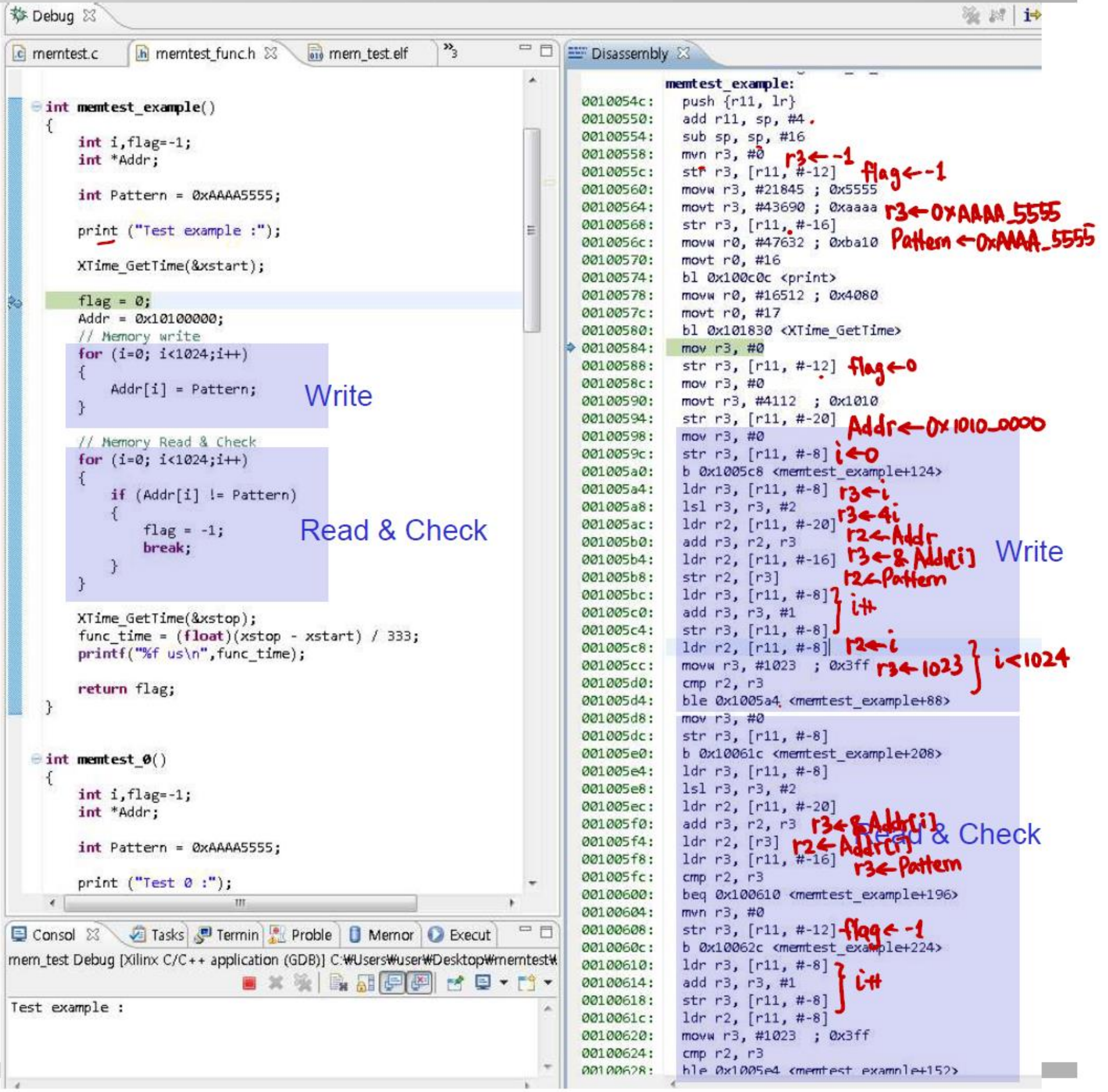
```
          memtest_example:
0010054c:   push {r11, lr}
00100550:   add r11, sp, #4
00100554:   sub sp, sp, #16
00100558:   mvn r3, #0
0010055c:   str r3, [r11, #-12]
00100560:   movw r3, #21845 ; 0x5555
00100564:   movt r3, #43690 ; 0xaaaa
00100568:   str r3, [r11, #-16]
0010056c:   movw r0, #47632 ; 0xba10
00100570:   movt r0, #16
00100574:   bl 0x100c0c <print>
00100578:   movw r0, #16512 ; 0x4080
0010057c:   movt r0, #17
00100580:   bl 0x101830 <XTime_GetTime>
00100584:   mov r3, #0
00100588:   str r3, [r11, #-12]
0010058c:   mov r3, #0
00100590:   movt r3, #4112  ; 0x1010
00100594:   str r3, [r11, #-20]
00100598:   mov r3, #0
0010059c:   str r3, [r11, #-8]
001005a0:   b 0x1005c8 <memtest_example+124>
001005a4:   ldr r3, [r11, #-8]
001005a8:   lsl r3, r3, #2
001005ac:   ldr r2, [r11, #-20]
001005b0:   add r3, r2, r3
001005b4:   ldr r2, [r11, #-16]
001005b8:   str r2, [r3]
001005bc:   ldr r3, [r11, #-8]
001005c0:   add r3, r3, #1
001005c4:   str r3, [r11, #-8]
001005c8:   ldr r2, [r11, #-8]
001005cc:   movw r3, #1023  ; 0x3ff
001005d0:   cmp r2, r3
001005d4:   ble 0x1005a4 <memtest_example+88>
001005d8:   mov r3, #0
001005dc:   str r3, [r11, #-8]
001005e0:   b 0x10061c <memtest_example+208>
001005e4:   ldr r3, [r11, #-8]
001005e8:   lsl r3, r3, #2
001005ec:   ldr r2, [r11, #-20]
001005f0:   add r3, r2, r3
001005f4:   ldr r2, [r3]
001005f8:   ldr r3, [r11, #-16]
001005fc:   cmp r2, r3
00100600:   beq 0x100610 <memtest_example+196>
00100604:   mvn r3, #0
00100608:   str r3, [r11, #-12]
0010060c:   b 0x10062c <memtest_example+224>
00100610:   ldr r3, [r11, #-8]
00100614:   add r3, r3, #1
00100618:   str r3, [r11, #-8]
0010061c:   ldr r2, [r11, #-8]
00100620:   movw r3, #1023  ; 0x3ff
00100624:   cmp r2, r3
00100628:   ble 0x1005e4 <memtest_example+152>
```

```
mem_test Debug [Xilinx C/C++ application (GDB)] C:₩Users₩user₩Desktop₩memtest₩
Test example :
```

Assembly Codes (-O0)

❑



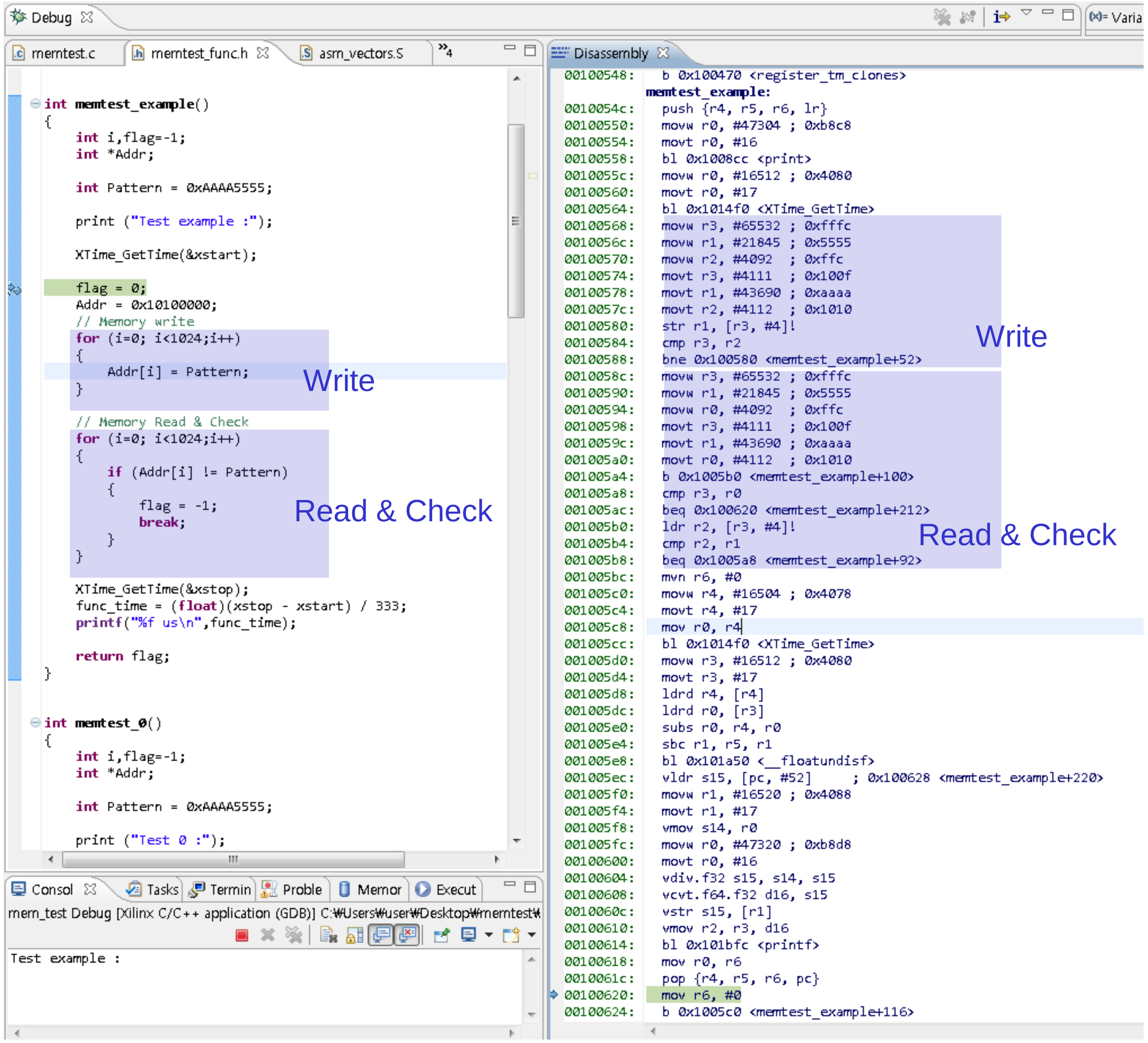


```c
int memtest_example()
{
    int i,flag=-1;
    int *Addr;

    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);

    return flag;
}


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

    int Pattern = 0xAAAA5555;

    print ("Test 0 :");
```

```
mem_test Debug [Xilinx C/C++ application (GDB)] C:₩Users₩user₩Desktop₩memtest₩
Test example :
```

```
          memtest_example:
0010054c:   push {r11, lr}
00100550:   add r11, sp, #4
00100554:   sub sp, sp, #16
00100558:   mvn r3, #0                       r3←-1
0010055c:   str r3, [r11, #-12]              flag←-1
00100560:   movw r3, #21845 ; 0x5555
00100564:   movt r3, #43690 ; 0xaaaa         r3←0xAAAA_5555
00100568:   str r3, [r11, #-16]              Pattern←0xAAAA_5555
0010056c:   movw r0, #47632 ; 0xba10
00100570:   movt r0, #16
00100574:   bl 0x100c0c <print>
00100578:   movw r0, #16512 ; 0x4080
0010057c:   movt r0, #17
00100580:   bl 0x101830 <XTime_GetTime>
00100584:   mov r3, #0
00100588:   str r3, [r11, #-12]              flag←0
0010058c:   mov r3, #0
00100590:   movt r3, #4112  ; 0x1010
00100594:   str r3, [r11, #-20]              Addr←0x1010_0000
00100598:   mov r3, #0
0010059c:   str r3, [r11, #-8]               i←0
001005a0:   b 0x1005c8 <memtest_example+124>
001005a4:   ldr r3, [r11, #-8]               r3←i
001005a8:   lsl r3, r3, #2                   r3←4i
001005ac:   ldr r2, [r11, #-20]              r2←Addr
001005b0:   add r3, r2, r3                   r3←&Addr[i]
001005b4:   ldr r2, [r11, #-16]              r2←Pattern
001005b8:   str r2, [r3]
001005bc:   ldr r3, [r11, #-8]  }
001005c0:   add r3, r3, #1      }  i++
001005c4:   str r3, [r11, #-8]  }
001005c8:   ldr r2, [r11, #-8]               r2←i        }
001005cc:   movw r3, #1023  ; 0x3ff          r3←1023     }  i<1024
001005d0:   cmp r2, r3                                   }
001005d4:   ble 0x1005a4 <memtest_example+88>
001005d8:   mov r3, #0
001005dc:   str r3, [r11, #-8]
001005e0:   b 0x10061c <memtest_example+208>
001005e4:   ldr r3, [r11, #-8]
001005e8:   lsl r3, r3, #2
001005ec:   ldr r2, [r11, #-20]
001005f0:   add r3, r2, r3                   r3←&Addr[i]
001005f4:   ldr r2, [r3]                     r2←Addr[i]
001005f8:   ldr r3, [r11, #-16]              r3←Pattern
001005fc:   cmp r2, r3
00100600:   beq 0x100610 <memtest_example+196>
00100604:   mvn r3, #0
00100608:   str r3, [r11, #-12]              flag←-1
0010060c:   b 0x10062c <memtest_example+224>
00100610:   ldr r3, [r11, #-8]  }
00100614:   add r3, r3, #1      }  i++
00100618:   str r3, [r11, #-8]  }
0010061c:   ldr r2, [r11, #-8]
00100620:   movw r3, #1023  ; 0x3ff
00100624:   cmp r2, r3
00100628:   ble 0x1005e4 <memtest_example+152>
```

Assembly Codes (-O3)

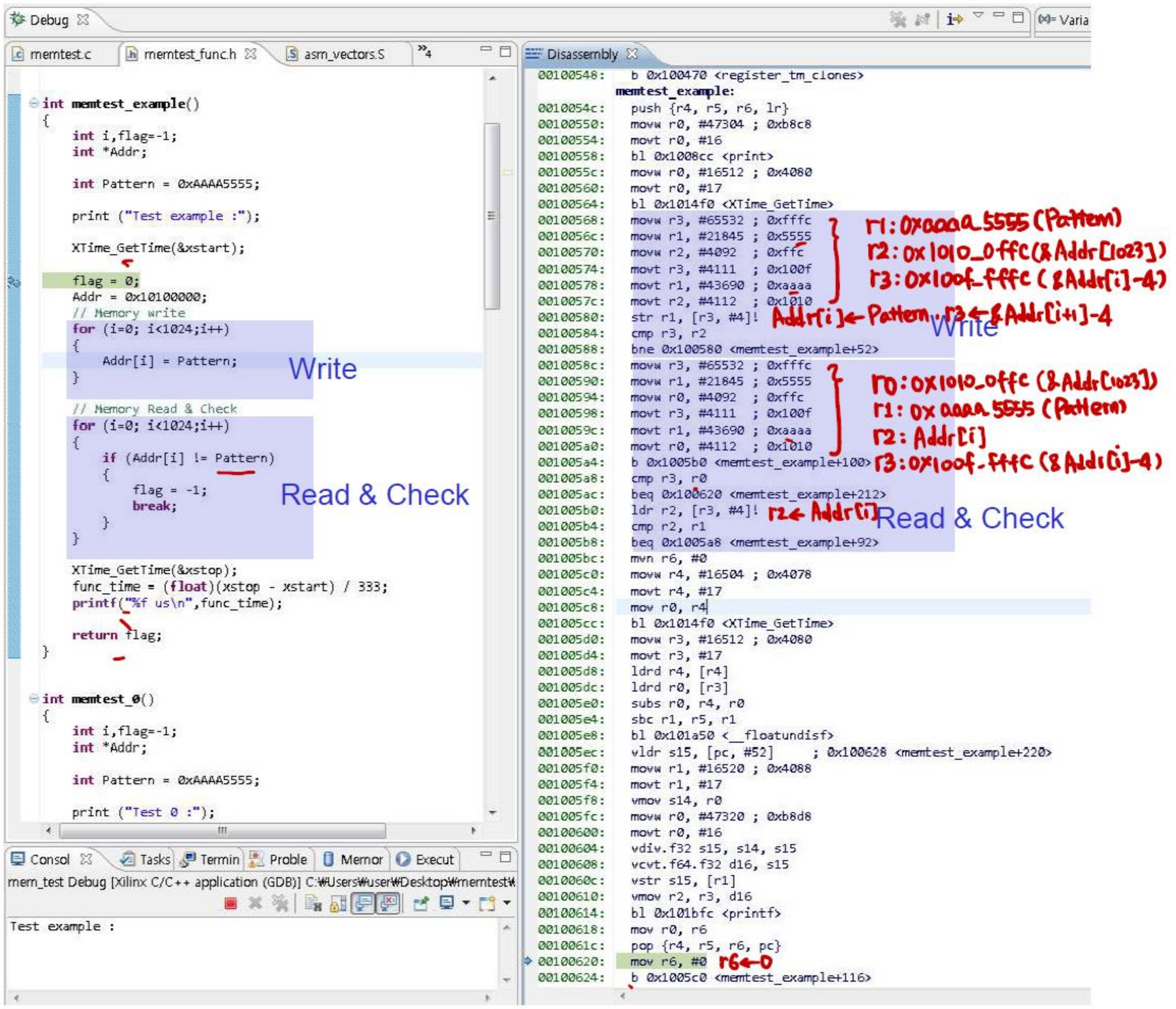


```c
int memtest_example()
{
    int i,flag=-1;
    int *Addr;

    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);

    return flag;
}


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

    int Pattern = 0xAAAA5555;

    print ("Test 0 :");
```

```
00100548:   b 0x100470 <register_tm_clones>
          memtest_example:
0010054c:   push {r4, r5, r6, lr}
00100550:   movw r0, #47304 ; 0xb8c8
00100554:   movt r0, #16
00100558:   bl 0x1008cc <print>
0010055c:   movw r0, #16512 ; 0x4080
00100560:   movt r0, #17
00100564:   bl 0x1014f0 <XTime_GetTime>
00100568:   movw r3, #65532 ; 0xfffc
0010056c:   movw r1, #21845 ; 0x5555
00100570:   movw r2, #4092  ; 0xffc
00100574:   movt r3, #4111  ; 0x100f
00100578:   movt r1, #43690 ; 0xaaaa
0010057c:   movt r2, #4112  ; 0x1010
00100580:   str r1, [r3, #4]!
00100584:   cmp r3, r2
00100588:   bne 0x100580 <memtest_example+52>
0010058c:   movw r3, #65532 ; 0xfffc
00100590:   movw r1, #21845 ; 0x5555
00100594:   movw r0, #4092  ; 0xffc
00100598:   movt r3, #4111  ; 0x100f
0010059c:   movt r1, #43690 ; 0xaaaa
001005a0:   movt r0, #4112  ; 0x1010
001005a4:   b 0x1005b0 <memtest_example+100>
001005a8:   cmp r3, r0
001005ac:   beq 0x100620 <memtest_example+212>
001005b0:   ldr r2, [r3, #4]!
001005b4:   cmp r2, r1
001005b8:   beq 0x1005a8 <memtest_example+92>
001005bc:   mvn r6, #0
001005c0:   movw r4, #16504 ; 0x4078
001005c4:   movt r4, #17
001005c8:   mov r0, r4
001005cc:   bl 0x1014f0 <XTime_GetTime>
001005d0:   movw r3, #16512 ; 0x4080
001005d4:   movt r3, #17
001005d8:   ldrd r4, [r4]
001005dc:   ldrd r0, [r3]
001005e0:   subs r0, r4, r0
001005e4:   sbc r1, r5, r1
001005e8:   bl 0x101a50 <__floatundisf>
001005ec:   vldr s15, [pc, #52]     ; 0x100628 <memtest_example+220>
001005f0:   movw r1, #16520 ; 0x4088
001005f4:   movt r1, #17
001005f8:   vmov s14, r0
001005fc:   movw r0, #47320 ; 0xb8d8
00100600:   movt r0, #16
00100604:   vdiv.f32 s15, s14, s15
00100608:   vcvt.f64.f32 d16, s15
0010060c:   vstr s15, [r1]
00100610:   vmov r2, r3, d16
00100614:   bl 0x101bfc <printf>
00100618:   mov r0, r6
0010061c:   pop {r4, r5, r6, pc}
00100620:   mov r6, #0
00100624:   b 0x1005c0 <memtest_example+116>
```

```
mem_test Debug [Xilinx C/C++ application (GDB)] C:₩Users₩user₩Desktop₩memtest₩
Test example :
```

Assembly Codes (-O3)


Debug
memtest.c
memtest_func.h
asm_vectors.S
Disassembly
Varia
Consol
Tasks
Termin
Proble
Memor
Execut
Write
Read & Check
r1:0xaaaa_5555(Pattern)
r2:0x1010_0ffc(&Addr[1023])
r3:0x100f_fffc(&Addr[i]-4)
Addr[i]←Pattern, r3←&Addr[i+1]-4
r0:0x1010_0ffc(&Addr[1023])
r1: 0x aaaa 5555 (Pattern)
r2: Addr[i]
r3:0x100f-fffc(&Addr[i]-4)
r2←Addr[i]
r6←0

```c
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);

    return flag;
}

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

    int Pattern = 0xAAAA5555;

    print ("Test 0 :");
```

```
mem_test Debug [Xilinx C/C++ application (GDB)] C:₩Users₩user₩Desktop₩memtest₩
Test example :
```

```
00100548:   b 0x100470 <register_tm_clones>
          memtest_example:
0010054c:   push {r4, r5, r6, lr}
00100550:   movw r0, #47304 ; 0xb8c8
00100554:   movt r0, #16
00100558:   bl 0x1008cc <print>
0010055c:   movw r0, #16512 ; 0x4080
00100560:   movt r0, #17
00100564:   bl 0x1014f0 <XTime_GetTime>
00100568:   movw r3, #65532 ; 0xfffc
0010056c:   movw r1, #21845 ; 0x5555
00100570:   movw r2, #4092  ; 0xffc
00100574:   movt r3, #4111  ; 0x100f
00100578:   movt r1, #43690 ; 0xaaaa
0010057c:   movt r2, #4112  ; 0x1010
00100580:   str r1, [r3, #4]!
00100584:   cmp r3, r2
00100588:   bne 0x100580 <memtest_example+52>
0010058c:   movw r3, #65532 ; 0xfffc
00100590:   movw r1, #21845 ; 0x5555
00100594:   movw r0, #4092  ; 0xffc
00100598:   movt r3, #4111  ; 0x100f
0010059c:   movt r1, #43690 ; 0xaaaa
001005a0:   movt r0, #4112  ; 0x1010
001005a4:   b 0x1005b0 <memtest_example+100>
001005a8:   cmp r3, r0
001005ac:   beq 0x100620 <memtest_example+212>
001005b0:   ldr r2, [r3, #4]!
001005b4:   cmp r2, r1
001005b8:   beq 0x1005a8 <memtest_example+92>
001005bc:   mvn r6, #0
001005c0:   movw r4, #16504 ; 0x4078
001005c4:   movt r4, #17
001005c8:   mov r0, r4
001005cc:   bl 0x1014f0 <XTime_GetTime>
001005d0:   movw r3, #16512 ; 0x4080
001005d4:   movt r3, #17
001005d8:   ldrd r4, [r4]
001005dc:   ldrd r0, [r3]
001005e0:   subs r0, r4, r0
001005e4:   sbc r1, r5, r1
001005e8:   bl 0x101a50 <__floatundisf>
001005ec:   vldr s15, [pc, #52]      ; 0x100628 <memtest_example+220>
001005f0:   movw r1, #16520 ; 0x4088
001005f4:   movt r1, #17
001005f8:   vmov s14, r0
001005fc:   movw r0, #47320 ; 0xb8d8
00100600:   movt r0, #16
00100604:   vdiv.f32 s15, s14, s15
00100608:   vcvt.f64.f32 d16, s15
0010060c:   vstr s15, [r1]
00100610:   vmov r2, r3, d16
00100614:   bl 0x101bfc <printf>
00100618:   mov r0, r6
0010061c:   pop {r4, r5, r6, pc}
00100620:   mov r6, #0
00100624:   b 0x1005c0 <memtest_example+116>
```

System-on-a-Chip
Design LAB

References

- Zynq-7000 All Programmable, technical reference manual, Xilinx UG585