

CS 342000 | CS343000

Instructor: Professor Izidor Gertner

Spring 2022

Azwad Shameem, 4/25/2022

Take Home Test 2

Please hand write and sign statements affirming that you will not cheat here and in your submission:

"I will neither give nor receive unauthorized assistance on this LAB. I will use only one computing device to perform this LAB".

I will neither give nor receive unauthorized assistance on this exam. I will only use one computing device to perform this test

Azwad Shameem

Start Time and Date: 8:00 AM 04/24/22

End Time and Date: 8:00 PM 04/24/22

Table of Contents

Objective.....	3
MIPS.....	4
INTEL X32 ISA Windows 32-bit.....	25
X86_64 ISA Linux 64-bit.....	35
Conclusions.....	43

Objective

The objective of this Take Home Test was to test, understand and demonstrate the recursive calls and stack frames in the three different architecture MARS Simulator, Visual Studio and Linux GDB and GCC. The Take Home Test objective includes testing and explaining the MIPS instructions on MARS Simulator, explaining the registers and memory on Visual Studio and explaining the debugger on Linux GDB and GCC. Furthermore, in this Take Home Test, we will be using the recursive version of the Euclidean algorithm for $\text{GCD}(a, b)$ where integers $a > 0$ and $b > 0$.

Solutions:

MIPS:

The screenshot shows the MARS 4.5 assembly editor interface. The top menu bar includes File, Edit, Run, Settings, Tools, and Help. The toolbar contains various assembly language instructions and tools. The main window has tabs for 'Edit' and 'Execute'. The code area displays the Shameem_GCD.asm file with assembly code for calculating the GCD of two numbers, a and b. The registers window on the right lists all 32 general-purpose registers (\$zero to \$t31) and their corresponding values, which are mostly zero. The stack pointer (\$sp) is at address 0x7ffffe00, and the program counter (\$pc) is at 0x00000000.

```

1 .data
2     a: .word 5
3     b: .word 10
4     answer: .word 0
5 .text
6 main:
7     lw $t0, a
8     lw $t1, b
9     jal gcd
10    sw $t4, answer
11    li $t4, 10
12    syscall
13
14 gcd:
15    addi $sp, $sp, -12
16    sw $t3, 8($sp)
17    sw $t2, 4($sp)
18    sw $t0, 0($sp)
19    add $t3, $t0, $zero
20    add $t3, $t1, $zero
21    div $t0, $t1
22    mfhi $t2
23    sw $t2, 4($sp)
24    bne $t2, $zero, L1
25    add $t4, $zero, $t1
26    addi $sp, $sp 12
27    jr $ra
28
29 L1:
30    add $t0, $t1, $zero
31    lw $t2, 4($sp)
32    add $t1, $t2, $zero
33    jal gcd
34    lw $t0, 0($sp)
35    lw $t2, 4($sp)
36    lw $t3, 8($sp)
37    addi $sp, $sp, 12
38    jr $ra
39

```

Figure 1: Shameem_GCD.asm code in MARS

This screenshot continues from Figure 1, showing the assembly code and register values for the Shameem_GCD.asm program. The code is identical to Figure 1, but the registers window shows different initial values. The stack pointer (\$sp) is now at 0x7ffffe00, and the program counter (\$pc) is at 0x00000000. The values for registers \$t0 through \$t31 are all set to zero.

```

1 .data
2     a: .word 5
3     b: .word 10
4     answer: .word 0
5 .text
6 main:
7     lw $t0, a
8     lw $t1, b
9     jal gcd
10    sw $t4, answer
11    li $t4, 10
12    syscall
13
14 gcd:
15    addi $sp, $sp, -12
16    sw $t3, 8($sp)
17    sw $t2, 4($sp)
18    sw $t0, 0($sp)
19    add $t3, $t0, $zero
20    add $t3, $t1, $zero
21    div $t0, $t1
22    mfhi $t2
23    sw $t2, 4($sp)
24    bne $t2, $zero, L1
25    add $t4, $zero, $t1
26    addi $sp, $sp 12
27    jr $ra
28
29 L1:
30    add $t0, $t1, $zero
31    lw $t2, 4($sp)
32    add $t1, $t2, $zero
33    jal gcd
34    lw $t0, 0($sp)
35    lw $t2, 4($sp)
36    lw $t3, 8($sp)
37    addi $sp, $sp, 12
38    jr $ra
39

```

Figure 2: Shameem_GCD.asm code in MARS continued

The screenshot shows the MARS 4.5 assembly debugger interface. The assembly code for `Shameem_GCD.asm` is displayed in the Text Segment window. The code starts at line 7 and includes instructions for initializing registers, performing calculations, and returning the result. The Registers window shows the stack pointer (\$sp) at address 0x7FFFEFFC. The Data Segment window shows variables `a`, `b`, and `answer` stored at address 0x10010000.

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$a0	16	0x00000000
\$a1	17	0x00000000
\$a2	18	0x00000000
\$a3	19	0x00000000
\$t9	20	0x00000000
\$x0	21	0x00000000
\$x1	22	0x00000000
\$gp	23	0x00000000
\$sp	24	0x00000000
\$fp	25	0x00000000
\$ra	26	0x00000000
pc	27	0x00000000
hi	28	0x10000000
lo	29	0x7fffffc
	30	0x00000000
	31	0x00000000

Figure 3: Shameem_GCD.asm code assembled in MARS (Lines 1-6)

Text Segment: This window shows that the code starts from line 7 because the

previous lines don't use registers.

Registers: This window shows that the stack pointer is 0x7FFFEFFC.

Data Segment: This window shows the variables stored in the addresses.

Variable `a` is stored at address 0x10010000, with a value of 0x00000005.

Variable `b` is stored at address 0x10010000, with a value of 0x0000000a.

Variable `answer` is stored at address 0x10010000, with a value of 0x00000000.

C:\Users\Azwad\Desktop\Shameem_Take_Home_Test_2\Shameem_GCD.asm - MARS 4.5

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000005
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7fffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400008
hi		0x00000000
lo		0x00000000

Figure 4: Shameem_GCD.asm code assembled in MARS (Line 7)

C:\Users\Azwad\Desktop\Shameem_Take_Home_Test_2\Shameem_GCD.asm - MARS 4.5

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$t0	8	0x00000005
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$t8	23	0x00000000
\$t9	24	0x00000000
\$k0	25	0x00000000
\$gp	26	0x10000000
\$sp	27	0x00000000
\$fp	28	0x00000000
\$ra	29	0xffffffff
pc	30	0x00000000
hi	31	0x00000000
lo		0x00000000

Figure 5: Shameem_GCD.asm code assembled in MARS (Line 8)

For Figure 4 and Figure 5.

Text Segment: This window shows that MARS has ran lines 7 and 8.

Registers: This window shows that the stack pointer is 0x7FFFEFFC. This window also shows that the variables have been loaded into the registers at \$t0 and \$t1. In the registers window in figure 4, \$t0 is highlighted with the value of 0x00000005 because variable a's value has recently been put into the registers. In the registers window in figure 5, \$t1 is highlighted with the value of 0x0000000a because variable b's value has recently been put into the registers window.

Data Segment: This window shows the variables stored in the addresses.

Variable a is stored at address 0x10010000, with a value of 0x00000005.

Variable b is stored at address 0x10010000, with a value of 0x0000000a.

Variable answer is stored at address 0x10010000, with a value of 0x00000000.

There has been no change in the Data Segment window figure 5 and figure 6, remains the same as figure 6.

The screenshot shows the MARS assembly debugger interface. The top menu bar includes File, Edit, Run, Settings, Tools, and Help. The main window has three main sections:

- Text Segment:** Shows assembly code with addresses, opcodes, and comments. Line 9 is highlighted in yellow: `addi \$sp, \$sp, -12` (opcode 0x3b3dff4). Line 15 is also highlighted: `addi \$sp, \$sp, -12` (opcode 0x3b3dff4).
- Registers:** A table showing register names, numbers, and values. The \$ra register (number 31) is highlighted in green and has a value of 0x00400014.
- Data Segment:** A table showing memory addresses and their corresponding values.

Figure 6: Shameem_GCD.asm code assembled in MARS (Line 9)

Text Segment: This window shows that MARS has ran line 9 and jumped to line 15 where the gcd function starts.

Registers: This window shows that the \$ra register has been updated to 0x00400014.

The other values in the registers window remain the same from figure 4 and figure 5.

Data Segment: This window shows the variables stored in the addresses.

Variable a is stored at address 0x10010000, with a value of 0x00000005.

Variable b is stored at address 0x10010000, with a value of 0x0000000a.

Variable answer is stored at address 0x10010000, with a value of 0x00000000.

There has been no change in the Data Segment window figure 5 and figure 6, remains the same as figure 6.

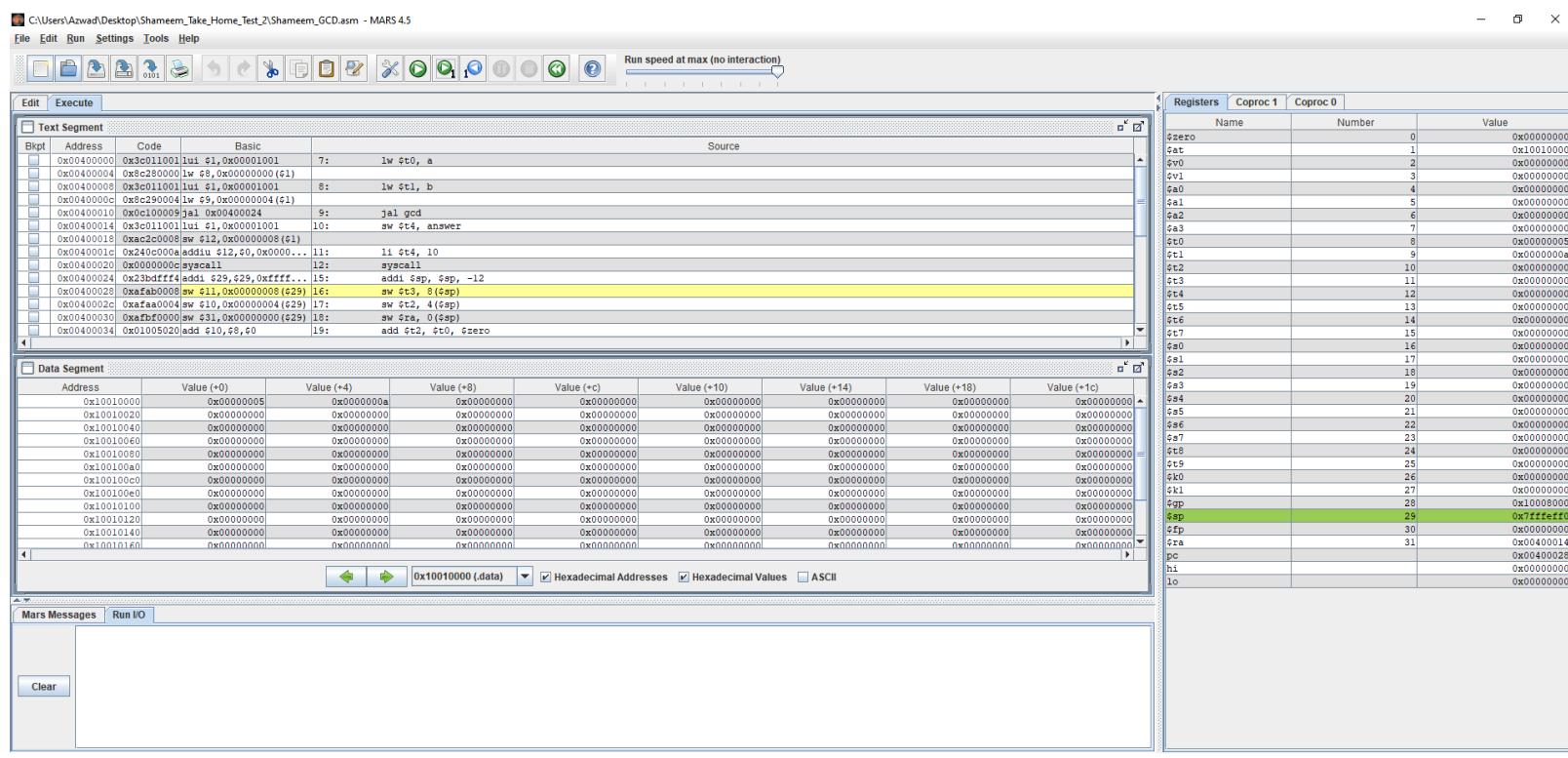


Figure 7: Shameem_GCD.asm code assembled in MARS (Line 15)

Text Segment: This window shows that MARS has ran line 15.

Registers: This window shows that the stack pointer, \$sp register has been decreased by the value of 12. The other values in the registers window remain the same from figure 6.

Data Segment: This window shows the variables stored in the addresses.

Variable a is stored at address 0x10010000, with a value of 0x00000005.

Variable b is stored at address 0x10010004, with a value of 0x0000000a.

Variable answer is stored at address 0x10010008, with a value of 0x00000000.

There has been no change in the Data Segment window figure 5 and figure 6, remains the same as figure 6.

The screenshot shows the MARS 4.5 assembly editor interface. The assembly window displays the following code:

```

Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c011001 lui $t0, 0x00000001
0x00400004 0x3c200000 lw $t1, 0x00000000($t1)
0x00400008 0x3c011001 lui $t1, 0x00000001
0x0040000c 0x3c200000 lw $t2, 0x00000000($t1)
0x00400010 0x00000000jal 0x00000004($t1)
0x00400014 0x3c011001 lui $t1, 0x00000001
0x00400018 0x3c200000 sw $t1, 0x00000008($t1)
0x00400020 0x00000000addiu $t2, $t0, 0x00000000
0x00400024 0x230dffff addi $t2, $t2, 0xffff...16:
0x00400028 0x0afab0008 sw $t1, 0x00000008($t1)16:
0x0040002c 0x0afaa0004 sw $t1, 0x00000004($t1)17:
0x00400030 0x0afbf0000 sw $t1, 0x00000000($t1)18:
0x00400034 0x01005020 add $t1, $t2, $t0, $zero19:

```

The assembly window has tabs for Text Segment and Data Segment. The Registers window on the right lists the processor's registers with their current values.

Figure 8: Shameem_GCD.asm code assembled in MARS (Line 16)

The screenshot shows the MARS 4.5 assembly editor interface. The assembly window displays the following code:

```

Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c011001 lui $t0, 0x00000001
0x00400004 0x3c200000 lw $t1, 0x00000000($t1)
0x00400008 0x3c011001 lui $t1, 0x00000001
0x0040000c 0x3c200000 lw $t2, 0x00000000($t1)
0x00400010 0x00000000jal 0x00000004($t1)
0x00400014 0x3c011001 lui $t1, 0x00000001
0x00400018 0x3c200000 sw $t1, 0x00000008($t1)
0x00400020 0x00000000addiu $t2, $t0, 0x00000000
0x00400024 0x230dffff addi $t2, $t2, 0xffff...15:
0x00400028 0x0afab0008 sw $t1, 0x00000008($t1)16:
0x0040002c 0x0afaa0004 sw $t1, 0x00000004($t1)17:
0x00400030 0x0afbf0000 sw $t1, 0x00000000($t1)18:
0x00400034 0x01005020 add $t1, $t2, $t0, $zero19:

```

The assembly window has tabs for Text Segment and Data Segment. The Registers window on the right lists the processor's registers with their current values.

Figure 9: Shameem_GCD.asm code assembled in MARS (Line 17)

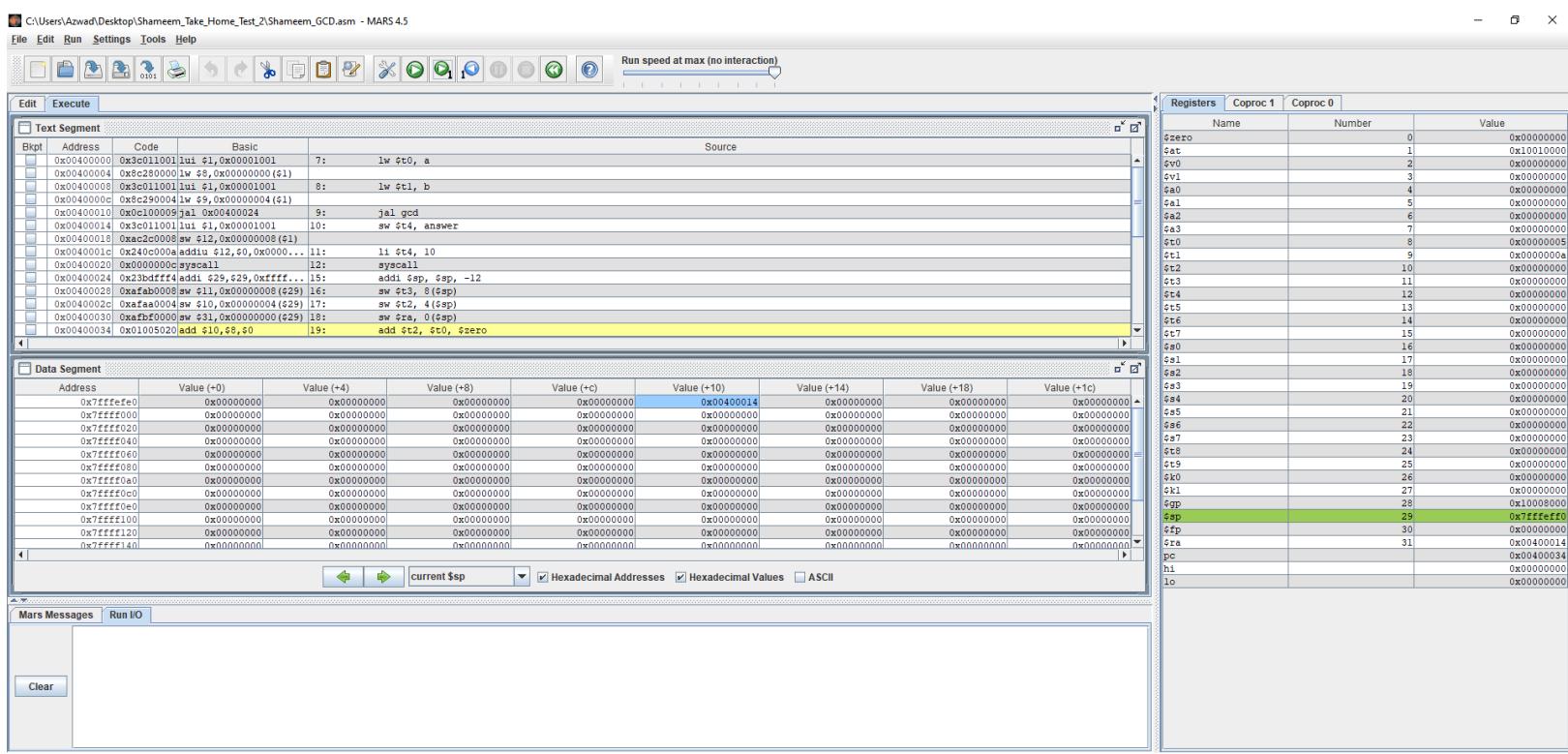


Figure 10: Shameem_GCD.asm code assembled in MARS (Line 18)

For Figure 8, Figure 9 and Figure 10.

Text Segment: This window shows that MARS has ran lines 16-18.

Registers: This window shows no change from Figure 7.

Data Segment: This window shows the variables stored in the addresses.

The value of \$t3 and \$t2, which is 0, is stored at the addresses 0x7FFF010 and 0x7FFFEFFE respectively. The value of \$ra, which is 0x00400014 is stored at addresses 0x7FFFEFF0. There has been no change in the previous values that have been added into the Data Segment window from figure 5 and figure 6.

C:\Users\Azwad\Desktop\Shameem_Take_Home_Test_2\Shameem_GCD.asm - MARS 4.5

The screenshot shows the MARS 4.5 assembly editor interface. The main window displays the assembly code for Shameem_GCD.asm. The code includes instructions like `lw \$t1, b`, `jal gcd`, and various arithmetic operations involving registers \$t1, \$t2, \$t3, and \$t4. The Registers window on the right shows the initial values of all general-purpose registers (\$zero to \$t1) set to 0. The Data Segment window shows memory starting at address 0x3c011001, containing the value 1. The Registers window also lists Coprocessor 1 and Coprocessor 0 registers, all initialized to 0.

Figure 11: Shameem_GCD.asm code assembled in MARS (Line 19)

C:\Users\Azwad\Desktop\Shameem_Take_Home_Test_2\Shameem_GCD.asm - MARS 4.5

This screenshot shows the same assembly code as Figure 11, but with a different assembly point highlighted. The instruction at address 0x04000013, `add \$t1,\$t0,\$zero` (Line 20), is highlighted in yellow. The Registers window shows the state of the registers after the previous instructions have been executed. The Data Segment window shows the memory starting at address 0x3c011001 now contains the value 2. The Registers window also lists Coprocessor 1 and Coprocessor 0 registers, all initialized to 0.

Figure 12: Shameem_GCD.asm code assembled in MARS (Line 20)

For Figure 11 and Figure 12.

Text Segment: This window shows that MARS has ran lines 16-18.

Registers: This window shows that the registers have made a copy of the values of variables a and variable b in order to do computation. The value of the \$t0 register, which is 0x00000005, plus the value of \$zero register, which is 0x00000000, is added into the highlighted \$t3 register in figure 11. The \$t3 register is now updated to the value of 0x00000005, which is the value of the variable a. The value of the \$t3 register, which is 0x0000000a, plus the value of the \$zero register, which is 0x00000000, is added into the highlighted \$t4 register in figure 12. The value of the \$t4 with the value of 0x0000000a, which is the value of the variable b.

Data Segment: This window shows no change since Figure 10.

The screenshot shows the MARS 4.5 assembly debugger interface. The top status bar indicates the file path: C:\Users\Azwad\Desktop\Shameem_Take_Home_Test_2\Shameem_GCD.asm - MARS 4.5. The menu bar includes File, Edit, Run, Settings, Tools, and Help. The toolbar contains various assembly and debugging icons.

Text Segment: This window displays the assembly code. Line 21, which contains the instruction `mfhi \$10`, is highlighted in yellow. The assembly code includes instructions like `lw \$9,0x00000004(\$1)`, `jal gcd`, `sw \$t4, answer`, and `mfhi \$10`.

Registers: This window shows the register values. The \$hi register is highlighted in green and has a value of 0x00000005. Other registers like \$t0 through \$t9, \$gp, \$sp, \$fp, and \$ra also have values displayed.

Data Segment: This window shows a memory dump of the data segment. It lists addresses from 0x7ffffe00 to 0x7fffff140 and their corresponding memory values, mostly showing 0x00000000.

Figure 13: Shameem_GCD.asm code assembled in MARS (Line 21)

Text Segment: This window shows that MARS has ran line 21.

Registers: This window shows that the registers have been updated, the hi register shows the value of 0x00000005 and the lo register is highlighted and shows the value of 0x00000000.

Data Segment: This window shows no change since Figure 10.

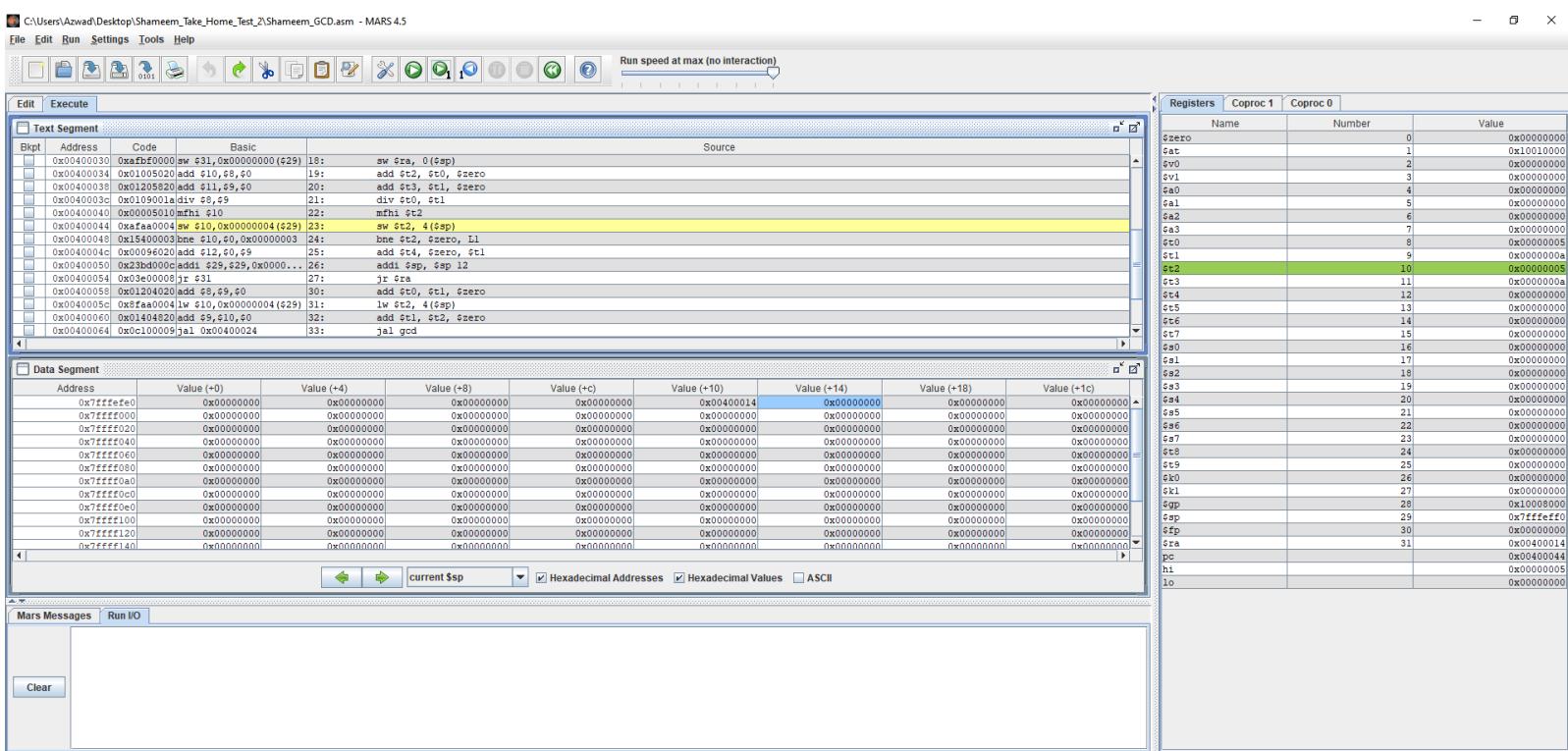


Figure 14: Shameem_GCD.asm code assembled in MARS (Line 22)

Text Segment: This window shows that MARS has ran line 22.

Registers: This window shows that the registers have been updated. The register `$t2` is highlighted and has been inputted with the remainder of the division of `$t0` and `$t1` which is `0x00000005`.

Data Segment: This window shows no change since Figure 10.

The screenshot shows the MARS 4.5 assembly debugger interface. The assembly code for `Shameem_GCD.asm` is displayed in the Text Segment window. Line 23, which contains the instruction `sw $t2, 4($sp)`, is highlighted. The Registers window shows the state of various registers, including \$zero through \$t1, \$t2, and \$t3. The Data Segment window shows a memory dump starting at address 0x7ffffe00, where the value at address 0x7fffffe is 0x00000005, corresponding to the value of \$t2.

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000005
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000004
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$a3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$t7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7ffffeff0
\$fp	30	0x00000000
\$ra	31	0x00000014
pc		0x00000048
hi		0x00000005
lo		0x00000000

Figure 15: Shameem_GCD.asm code assembled in MARS (Line 23)

Text Segment: This window shows that MARS has ran line 23.

Registers: This window shows that the registers have not changed since figure 14.

Data Segment: This window shows that the values at addresses have been updated. At the highlighted address 0xFFFFE, the value of \$t2 is stored, which is 0x00000005, which is also the value of the remainder of the division of variable a and variable b.

The screenshot shows the MARS 4.5 assembly debugger interface. The assembly code for `Shameem_GCD.asm` is displayed in the Text Segment window. Line 24, which contains the BNE instruction, is highlighted. The Registers window shows the state of all registers, with `$t2` highlighted in green. The Data Segment window shows a memory dump from address `0x00000000` to `0x0000ffff`. The status bar at the bottom shows `current $sp`.

Figure 16: Shameem_GCD.asm code assembled in MARS (Line 24)

Text Segment: This window shows that MARS has ran line 24, which has made it jump to line 30 because of the BNE instruction.

Registers: This window shows that the registers have not changed since figure 14.

Data Segment: This window shows that there has been no change since figure 15.

C:\Users\Azwad\Desktop\Shameem_Take_Home_Test_2\Shameem_GCD.asm - MARS 4.5

The screenshot shows the assembly editor interface with several windows:

- Text Segment:** Shows assembly instructions from Line 18 to Line 30. The highlighted instruction is Line 30: `lw $t0, 0($sp)`.
- Data Segment:** Shows memory starting at address 0x7ffffe00.
- Registers:** Shows the state of registers r0 through r31, \$zero, \$at, \$v0, \$v1, \$a0, \$a1, \$a2, \$a3, \$t0, \$t1, \$t2, \$t3, \$t4, \$t5, \$t6, \$t7, \$t8, \$t9, \$t10, \$t11, \$t12, \$t13, \$t14, \$t15, \$t16, \$t17, \$t18, \$t19, \$t20, \$t21, \$t22, \$t23, \$t24, \$t25, \$t26, \$t27, \$t28, \$t29, \$t30, \$t31, \$gp, \$sp, \$fp, \$ra, \$hi, and \$lo.
- Mars Messages:** Shows the message "Run I/O".

Figure 17: Shameem_GCD.asm code assembled in MARS (Line 30)

C:\Users\Azwad\Desktop\Shameem_Take_Home_Test_2\Shameem_GCD.asm - MARS 4.5

The screenshot shows the assembly editor interface with several windows:

- Text Segment:** Shows assembly instructions from Line 21 to Line 36. The highlighted instruction is Line 31: `lw $t0, 0($sp)`.
- Data Segment:** Shows memory starting at address 0x7ffffe00.
- Registers:** Shows the state of registers r0 through r31, \$zero, \$at, \$v0, \$v1, \$a0, \$a1, \$a2, \$a3, \$t0, \$t1, \$t2, \$t3, \$t4, \$t5, \$t6, \$t7, \$t8, \$t9, \$t10, \$t11, \$t12, \$t13, \$t14, \$t15, \$t16, \$t17, \$t18, \$t19, \$t20, \$t21, \$t22, \$t23, \$t24, \$t25, \$t26, \$t27, \$t28, \$t29, \$t30, \$t31, \$gp, \$sp, \$fp, \$ra, \$hi, and \$lo.
- Mars Messages:** Shows the message "Run I/O".

Figure 18: Shameem_GCD.asm code assembled in MARS (Line 31)

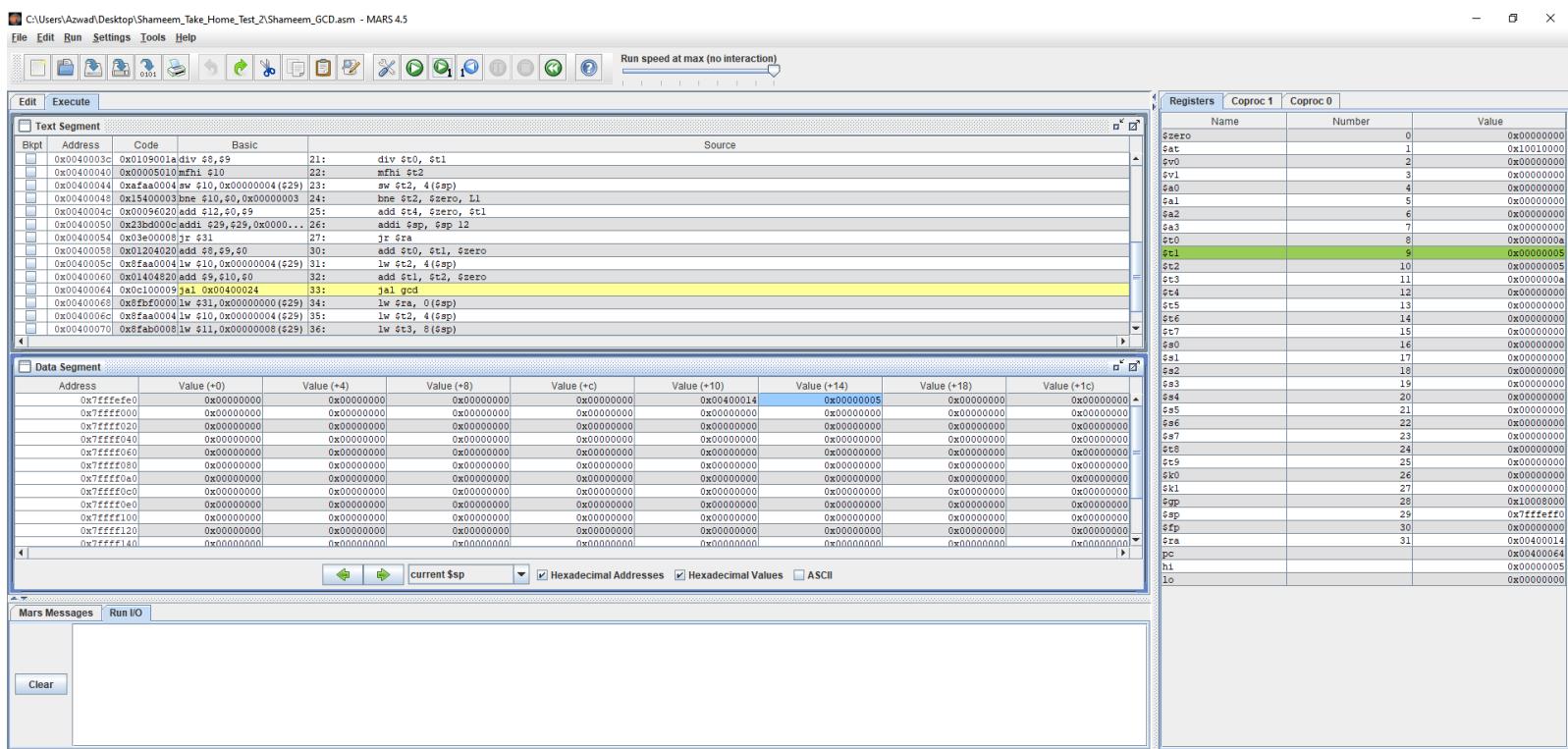


Figure 19: Shameem_GCD.asm code assembled in MARS (Line 32)

For Figure 17, Figure 18 and Figure 19.

Text Segment: This window shows that MARS has ran lines 30-32.

Registers: This window shows that the registers have been updated. The value of \$t0 register is now 0x0000000a and the value of the \$t1 register is now 0x00000005.

Data Segment: This window shows that there has been no change since figure 15.

Figure 22 shows the assembly code for Shameem_GCD.asm assembled in MARS 4.5. The code implements the Euclidean algorithm for finding the GCD of two numbers. It uses registers \$t0 through \$t9 and \$sp. The assembly code is shown in the Text Segment and Data Segment panes, with the Registers pane on the right showing initial values for all registers.

Figure 22: Shameem_GCD.asm code assembled in MARS (Line 16)

Figure 23 shows the assembly code for Shameem_GCD.asm assembled in MARS 4.5. The code implements the Euclidean algorithm for finding the GCD of two numbers. It uses registers \$t0 through \$t9 and \$sp. The assembly code is shown in the Text Segment and Data Segment panes, with the Registers pane on the right showing initial values for all registers.

Figure 23: Shameem_GCD.asm code assembled in MARS (Line 17)

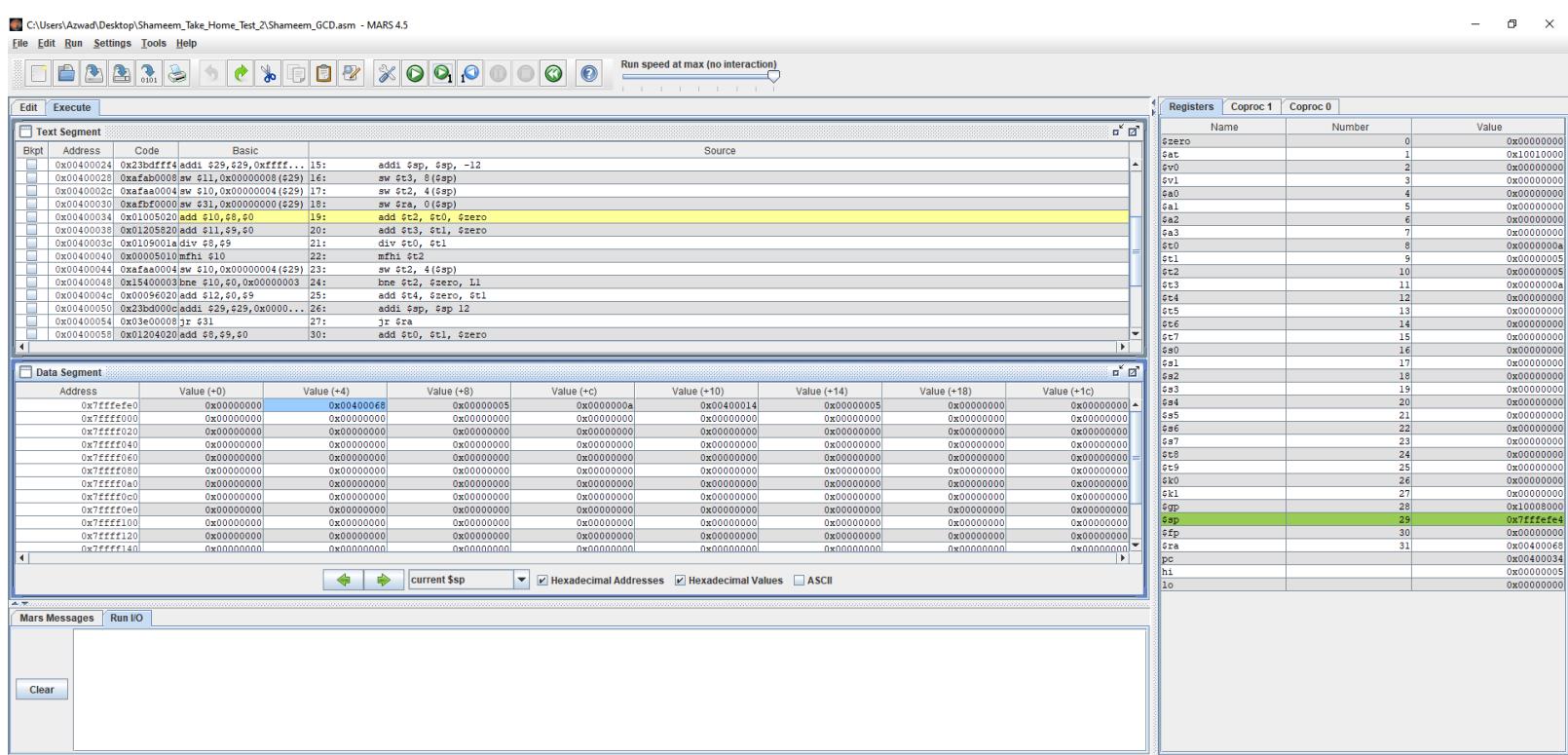


Figure 24: Shameem_GCD.asm code assembled in MARS (Line 18)

For Figure 20, Figure 21, Figure 22, Figure 23, Figure 24.

Text Segment: This window shows that MARS has ran line 30 and jumped to 15 and then ran lines 15-18.

Registers: This window shows that the registers have been updated. The \$ra register is now 0x00400068 and the stack pointer register, \$sp is now 0x7FFFEFE4.

Data Segment: This window shows that at addresses 0x7FFFEC, 0x7FFE8, 0x7FFFEFE4, the values have been updated with 0x00400068, 0x0000000A and 0x00000005 respectively.

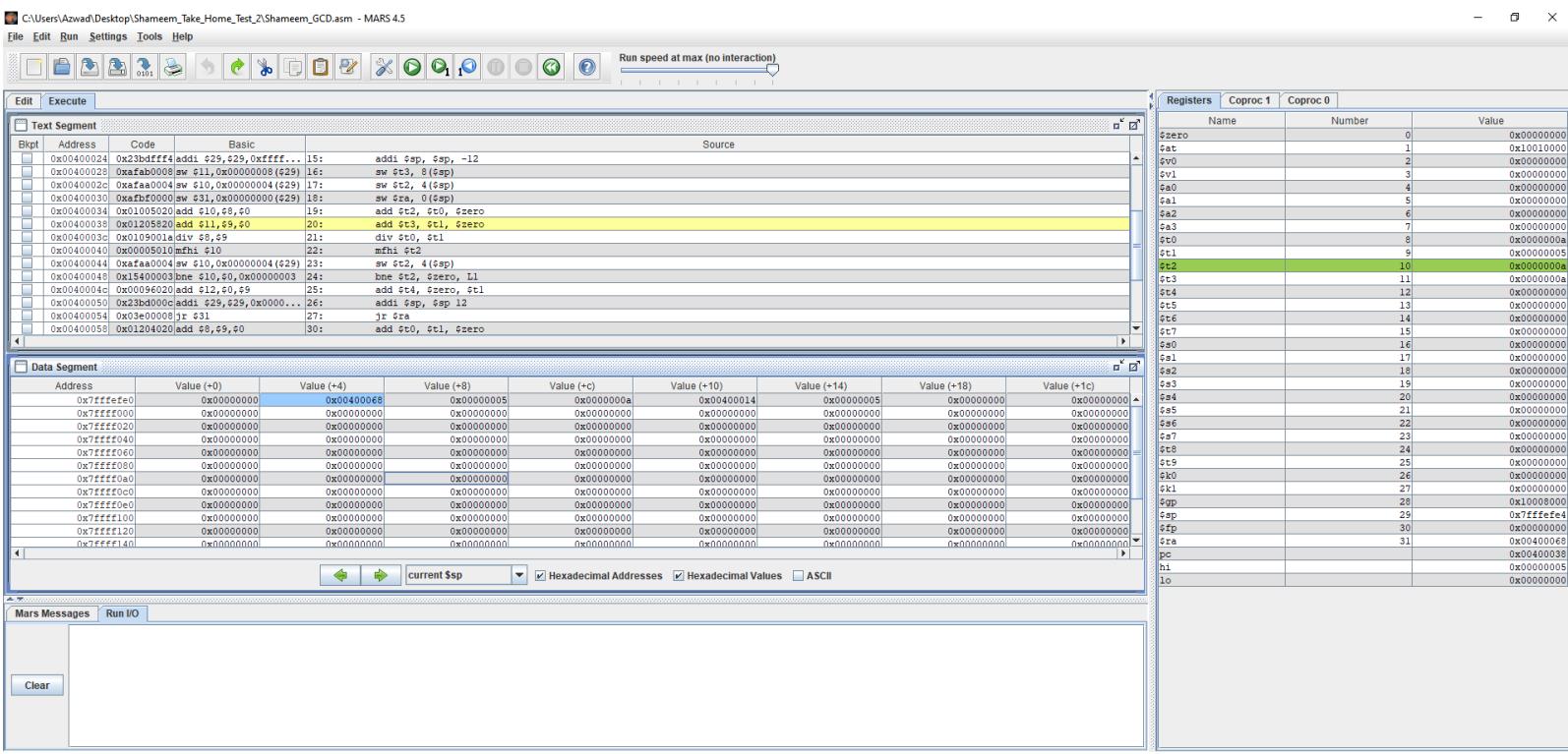


Figure 25: Shameem_GCD.asm code assembled in MARS (Line 19)

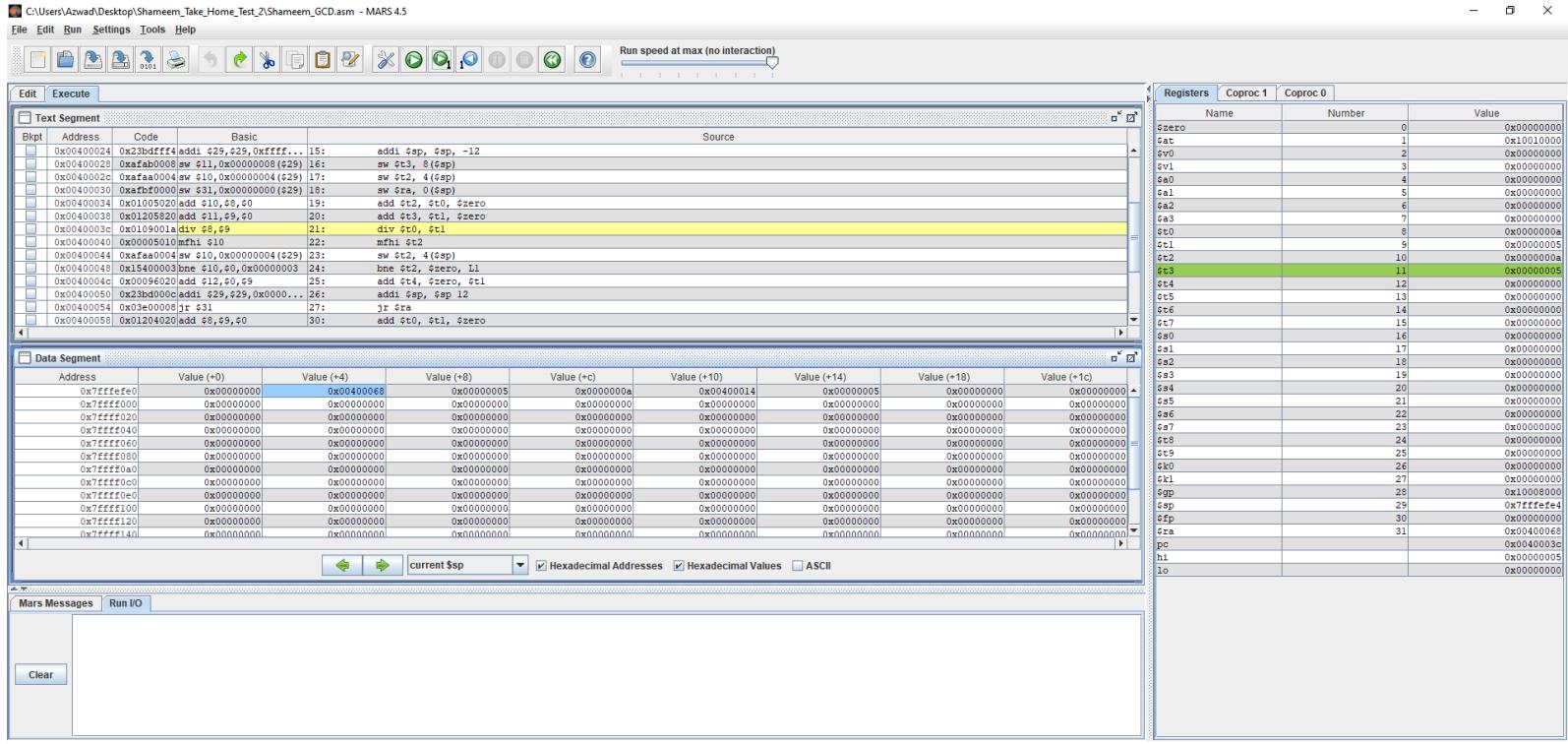


Figure 26: Shameem_GCD.asm code assembled in MARS (Line 20)

Figure 27 shows the assembly code for Shameem_GCD.asm assembled in MARS 4.5. The code implements the Euclidean algorithm for finding the Greatest Common Divisor (GCD) of two numbers. The assembly code uses the following registers:

- \$t0, \$t1, \$t2, \$t3 for temporary storage.
- \$r0, \$r1 for input values.
- \$zero for zero.
- \$sp for stack pointer.
- \$gp for global pointer.
- \$fp for frame pointer.
- \$ra for return address.
- \$pc for program counter.
- \$hi for high part of quotient.
- \$lo for low part of quotient.

The code starts by loading the first value into \$t0 and the second value into \$t1. It then enters a loop where it divides \$t0 by \$t1 using the mfhi and mflo instructions. The remainder is stored in \$t2. If \$t2 is zero, the process is complete. Otherwise, the divisor (\$t1) becomes the new dividend (\$t0), and the remainder (\$t2) becomes the new divisor (\$t1). The loop continues until the remainder is zero, at which point the GCD is found in \$t0.

Figure 27: Shameem_GCD.asm code assembled in MARS (Line 21)

Figure 28 shows the assembly code for Shameem_GCD.asm assembled in MARS 4.5, with additional comments and labels added for clarity. The code implements the Euclidean algorithm for finding the Greatest Common Divisor (GCD) of two numbers. The assembly code uses the following registers:

- \$t0, \$t1, \$t2, \$t3 for temporary storage.
- \$r0, \$r1 for input values.
- \$zero for zero.
- \$sp for stack pointer.
- \$gp for global pointer.
- \$fp for frame pointer.
- \$ra for return address.
- \$pc for program counter.
- \$hi for high part of quotient.
- \$lo for low part of quotient.

The code starts by loading the first value into \$t0 and the second value into \$t1. It then enters a loop where it divides \$t0 by \$t1 using the mfhi and mflo instructions. The remainder is stored in \$t2. If \$t2 is zero, the process is complete. Otherwise, the divisor (\$t1) becomes the new dividend (\$t0), and the remainder (\$t2) becomes the new divisor (\$t1). The loop continues until the remainder is zero, at which point the GCD is found in \$t0.

Figure 28: Shameem_GCD.asm code assembled in MARS (Line 22)

The screenshot shows the MARS 4.5 assembly debugger interface with the following windows:

- Text Segment:** Displays assembly code for the `Text Segment`. Lines 18 through 33 are shown, including instructions like `sw $ra, 0($sp)`, `add $t2, $t0, $zero`, and `mfhi $t2`.
- Data Segment:** Displays memory values for the `Data Segment` at addresses from 0x7ffffe00 to 0x7fffff140.
- Registers:** Shows the state of various registers. The `$t2` register is highlighted in green and contains the value 0x00000000. The `$t1` register contains 0x00000005, and the `lo` register contains 0x00000002.
- Registers Coproc 1 Coproc 0:** Shows the state of coprocessor registers.
- Mars Messages:** Shows the message "Run I/O".

Figure 29: Shameem_GCD.asm code assembled in MARS (Line 23)

For Figure 25, Figure 26, Figure 27, Figure 28, Figure 29

Text Segment: This window shows that MARS has ran lines 19-23.

Registers: This window shows that the registers have been updated. The `$t2` register is now 0x00000000 and the `$t1` register is 0x00000005 and the `lo` register is now 0x00000002.

Data Segment: This window shows that at addresses 0x7FFFEF8, the values have been updated with 0x00000000.

The screenshot shows the MARS assembly editor interface with the following details:

- Title Bar:** C:\Users\Azwad\Desktop\Shameem_Take_Home_Test_2\Shameem_GCD.asm - MARS 4.5
- Menu Bar:** File Edit Run Settings Tools Help
- Toolbar:** Includes icons for Open, Save, Run, Stop, Break, and various settings.
- Registers Window:** Shows the state of registers from \$zero to \$lo.
- Text Segment Window:** Displays assembly code with highlighted instructions (e.g., line 24). The code includes operations like add, sub, div, and mthi.
- Data Segment Window:** Shows memory dump with columns for Address, Value (+0), Value (+4), etc.
- Mars Messages and Run I/O Windows:** Both are currently empty.

Figure 30: Shameem_GCD.asm code assembled in MARS (Line 24)

The screenshot shows the MARS assembly editor interface with the following details:

- Title Bar:** C:\Users\Azwad\Desktop\Shameem_Take_Home_Test_2\Shameem_GCD.asm - MARS 4.5
- Menu Bar:** File Edit Run Settings Tools Help
- Toolbar:** Includes icons for Open, Save, Run, Stop, Break, and various settings.
- Registers Window:** Shows the state of registers from \$zero to \$lo.
- Text Segment Window:** Displays assembly code with highlighted instructions (e.g., line 25). The code includes operations like add, sub, div, and mthi.
- Data Segment Window:** Shows memory dump with columns for Address, Value (+0), Value (+4), etc.
- Mars Messages and Run I/O Windows:** Both are currently empty.

Figure 31: Shameem_GCD.asm code assembled in MARS (Line 25)

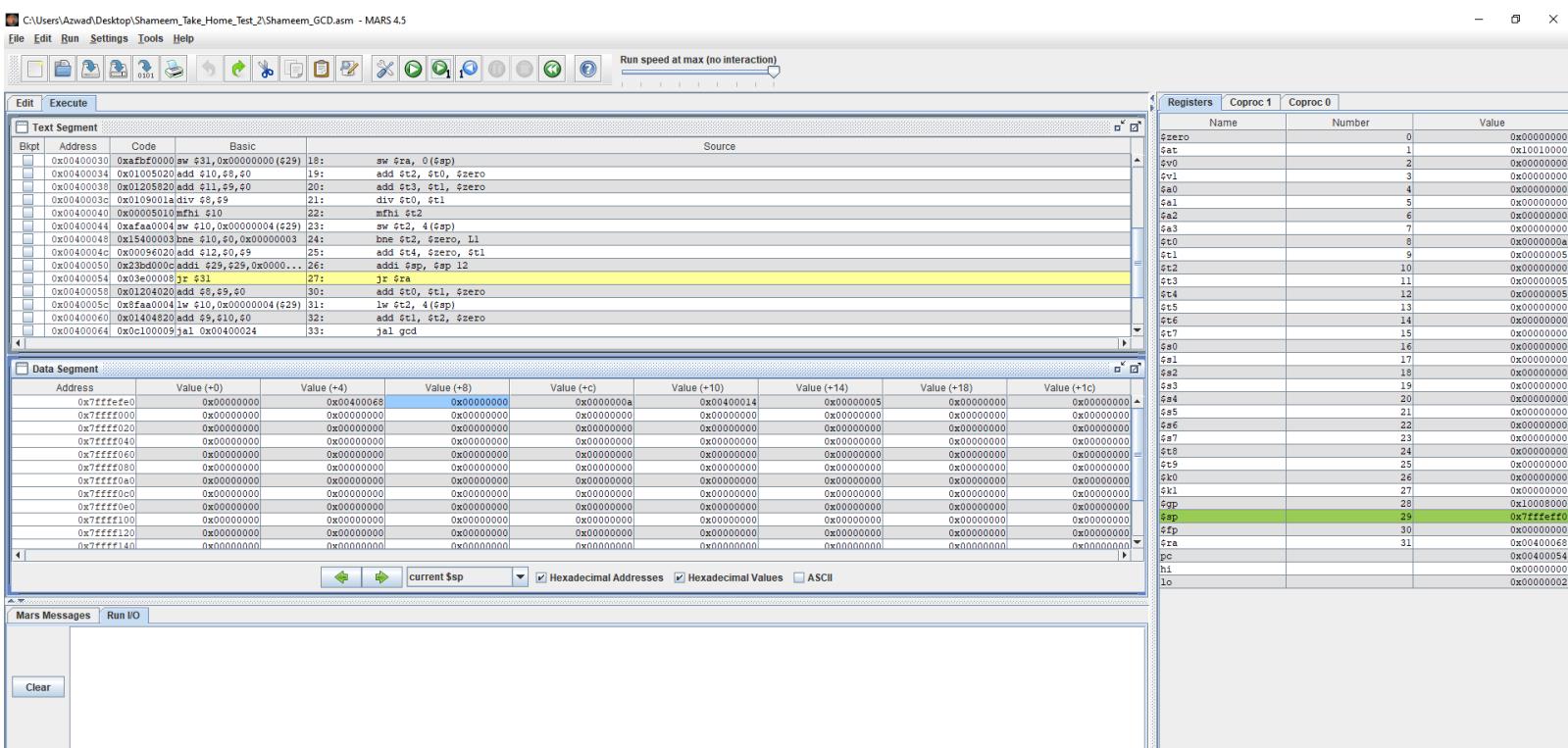


Figure 32: Shameem_GCD.asm code assembled in MARS (Line 26)

For Figure 30, Figure 31, Figure 32

Text Segment: This window shows that MARS has ran lines 24-26.

Registers: This window shows that the \$sp register is now 0x7FFFEF0.

Data Segment: This window shows no change since figure 29.

Figure 33: Shameem_GCD.asm code assembled in MARS (Line 34, jumped from 27)

Figure 34: Shameem_GCD.asm code assembled in MARS (Line 35)

For Figure 33, Figure 34, Figure 35, Figure 36

Text Segment: This window shows that MARS has ran lines 34-37.

Registers: This window shows that the \$sp register is now 0x7FFFEFC. The window also shows that \$t3 is now 0x00000000 and \$ra is now 0x00400014.

Data Segment: This window shows no change since figure 29.

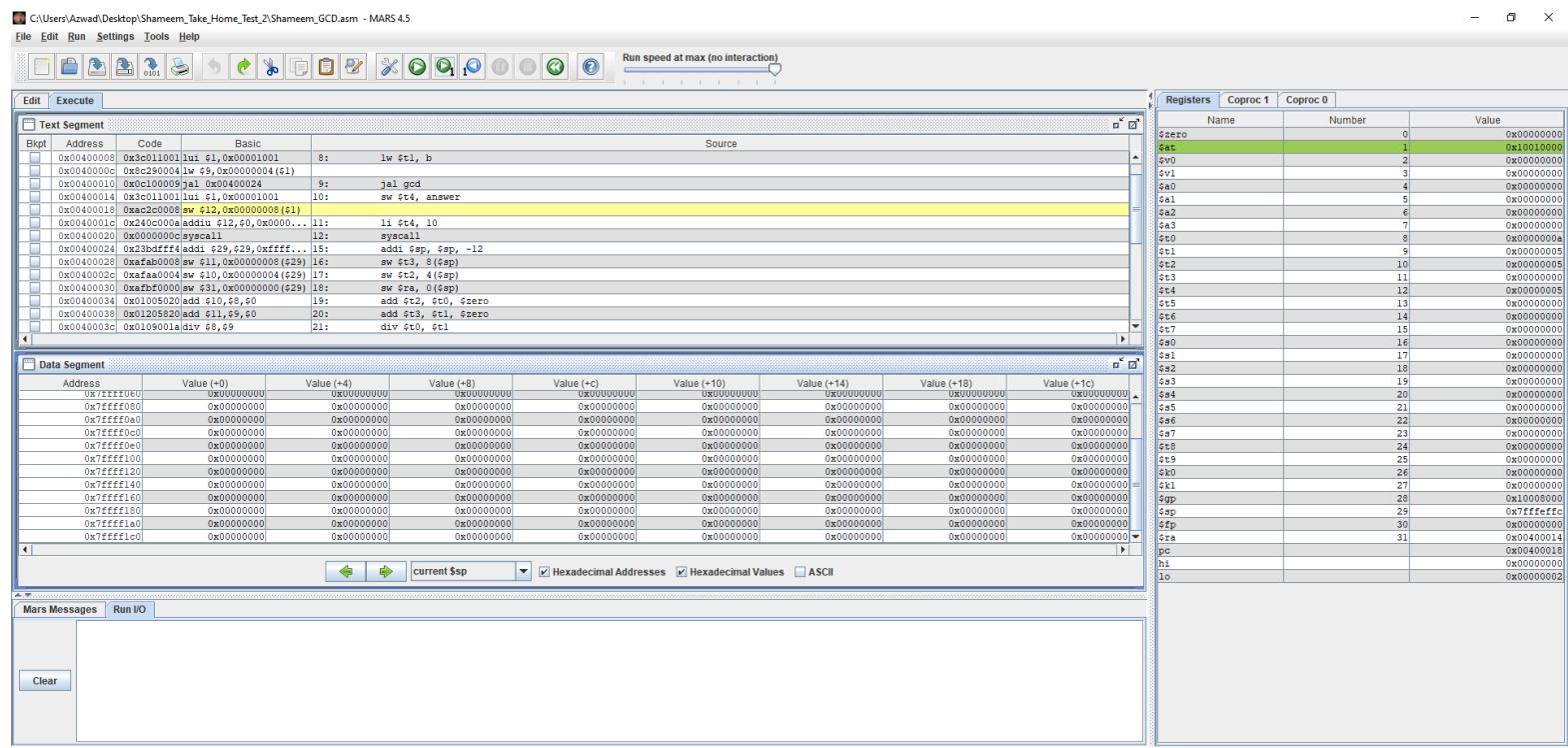


Figure 37: Shameem_GCD.asm code assembled in MARS (jumped to Line 10)

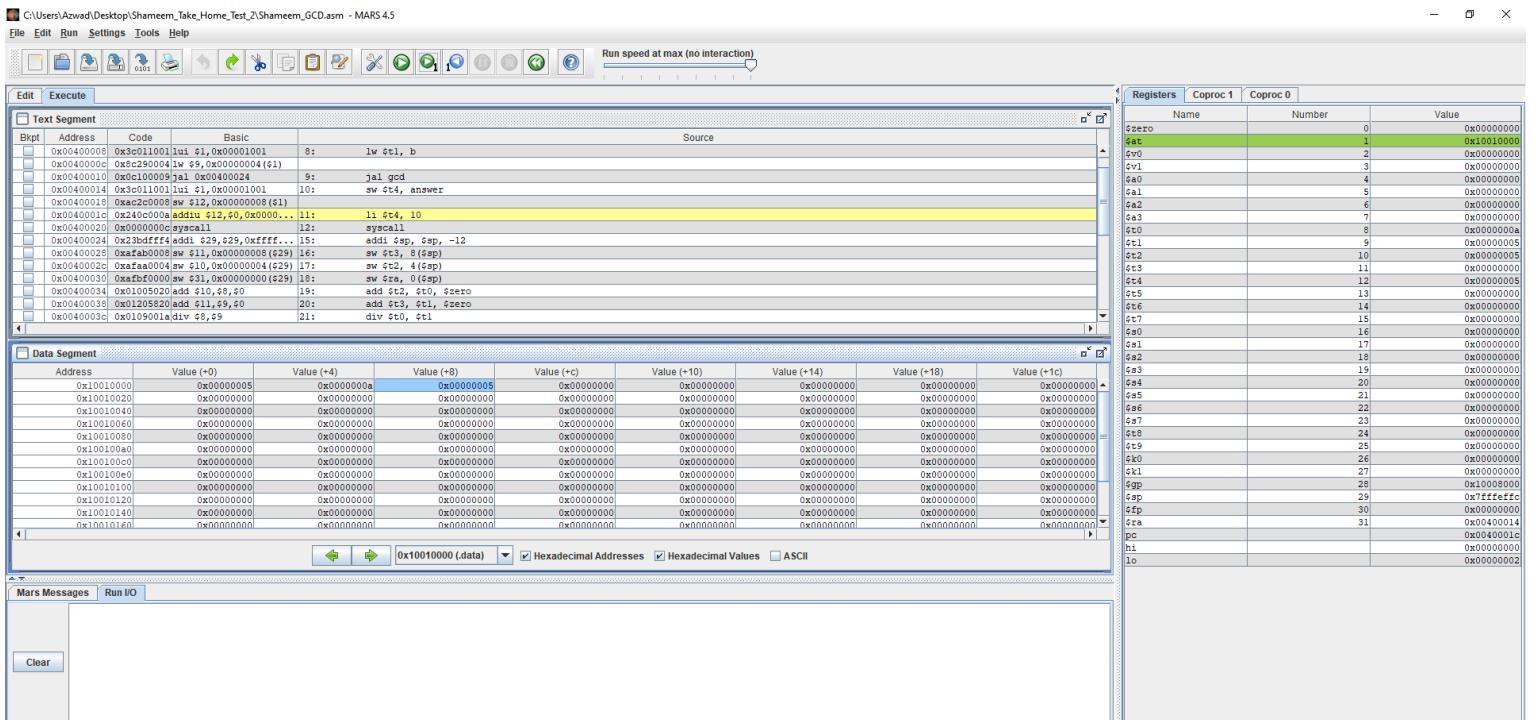


Figure 38: Shameem_GCD.asm code assembled in MARS (Line 10)

The screenshot shows the MARS 4.5 assembly debugger interface. The assembly code for `Shameem_GCD.asm` is displayed in the Text Segment window. Line 11, which contains the instruction `li $t4, 10`, is highlighted. The Registers window shows the state of various registers, with `$t4` having a value of `0x0000000A`. The Data Segment window shows the memory dump starting at address `0x10100000`, where the value `0x00000008` is present.

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000005
\$t2	10	0x00000005
\$t3	11	0x00000000
\$t4	12	0x0000000A
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$a0	16	0x00000000
\$a1	17	0x00000000
\$a2	18	0x00000000
\$a3	19	0x00000000
\$a4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0xfffffeff
\$fp	30	0x00000000
\$ra	31	0x00400014
pc		0x00400020
hi		0x00000000
lo		0x00000002

Figure 39: Shameem_GCD.asm code assembled in MARS (line 11)

For Figure 37, Figure 38, Figure 39

Text Segment: This window shows that MARS has ran lines 10-12 and exited.**Registers:** This window shows that the `$t4` register's value is now `0x0000000A`.**Data Segment:** This window shows the answer is stored in `0x10100008` with the value of `0x00000005`.

Intel X32 ISA Windows 32-bit:

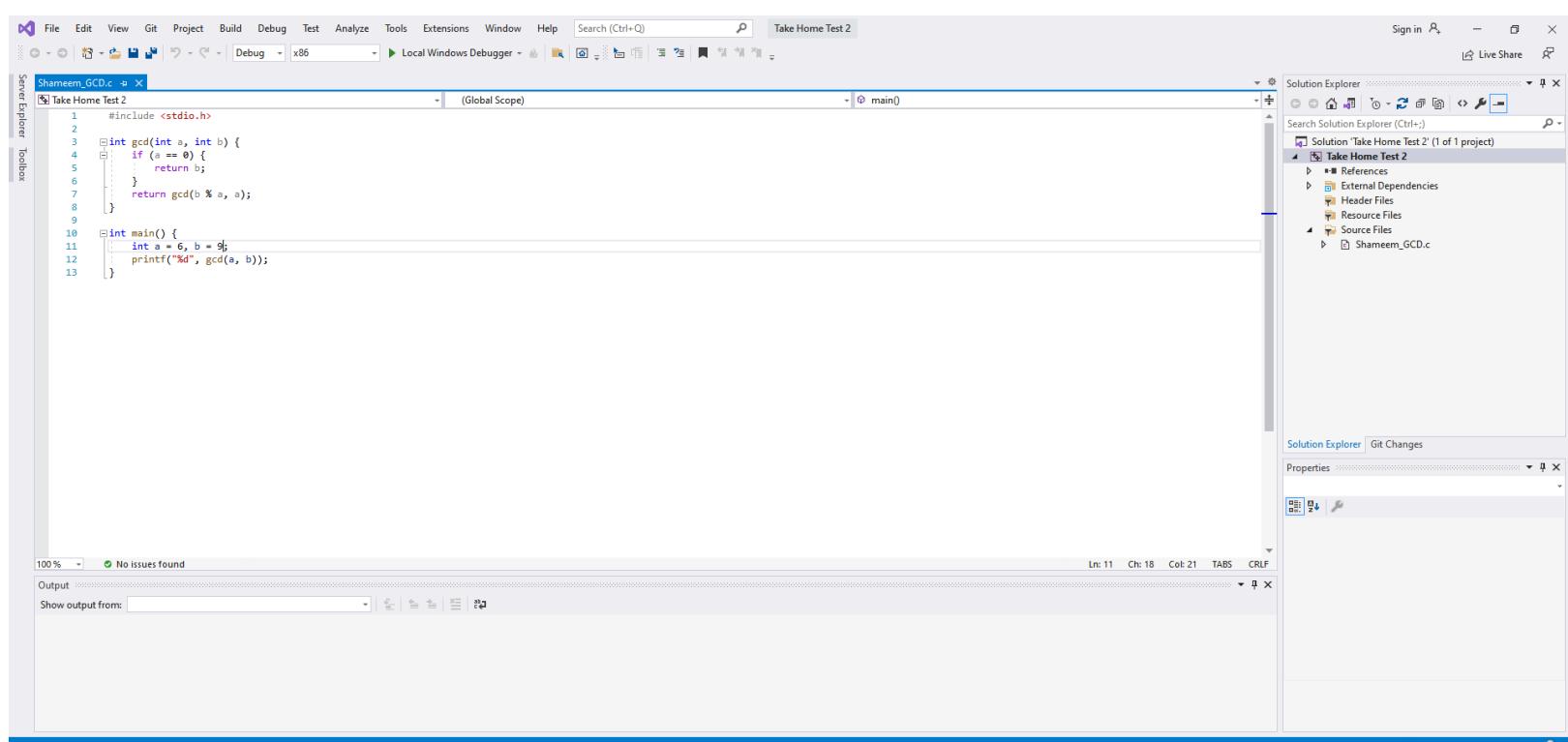


Figure 40: Shameem_GCD.c code in Visual Studio

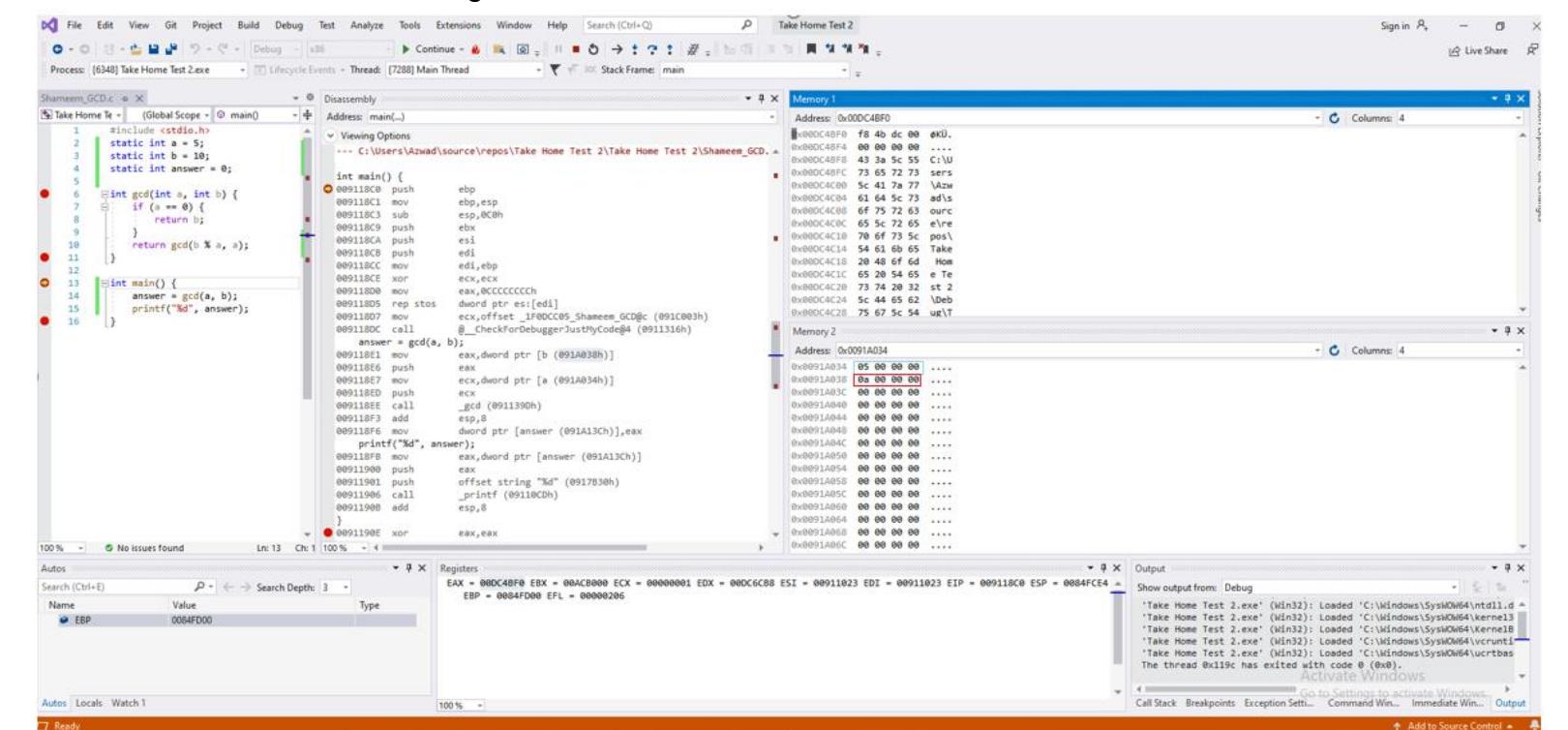


Figure 41: Shameem_GCD.c code in Visual Studio

Disassembly: This window shows that the code starts at line 12.

Registers: This window shows that the value of ESP, the stack pointer, is 0x00084FCE4.

Memory: Memory window 1 shows the values around the stack pointer which is random. While Memory window 2 displays the value of the static variables stored into the addresses.

Variable a is stored in address 0x0091A034 and has the value 05 00 00 00.

Variable b is stored in address 0x0091A038 and has the value 0a 00 00 00.

Variable answer is stored in address 0x0091A03C and has the value 00 00 00 00.

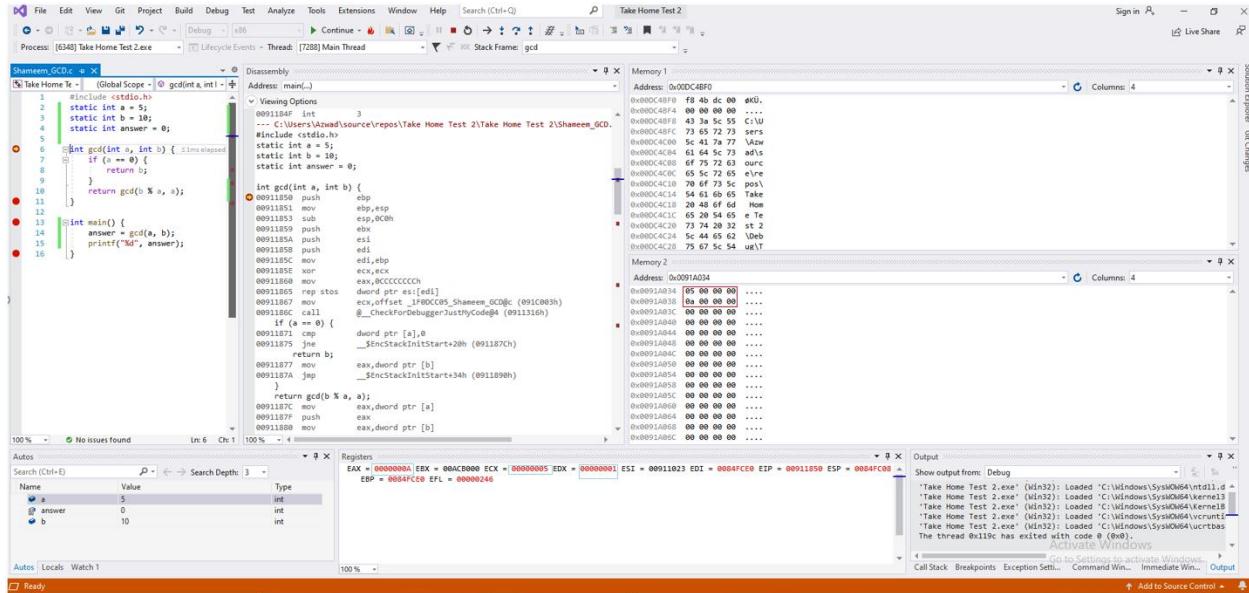


Figure 42: Shameem_GCD.c code in Visual Studio

Disassembly: This window shows that the code runs line 14 which calls the gcd() function at line 6.

Registers: This window shows that the value of EAX is now the value of variable b, which is 0x0000000A, and the value of ECX is now the value of variable a, which is 0x00000005. This means the value of the variables are copied into the register to do computation.

Memory: Memory window 1 shows the values around the stack pointer which is somewhat initialized. While Memory window 2 displays the value of the static variables stored into the addresses.

Variable a is stored in address 0x0091A034 and has the value 05 00 00 00.

Variable b is stored in address 0x0091A038 and has the value 0a 00 00 00.

Variable answer is stored in address 0x0091A03C and has the value 00 00 00 00.

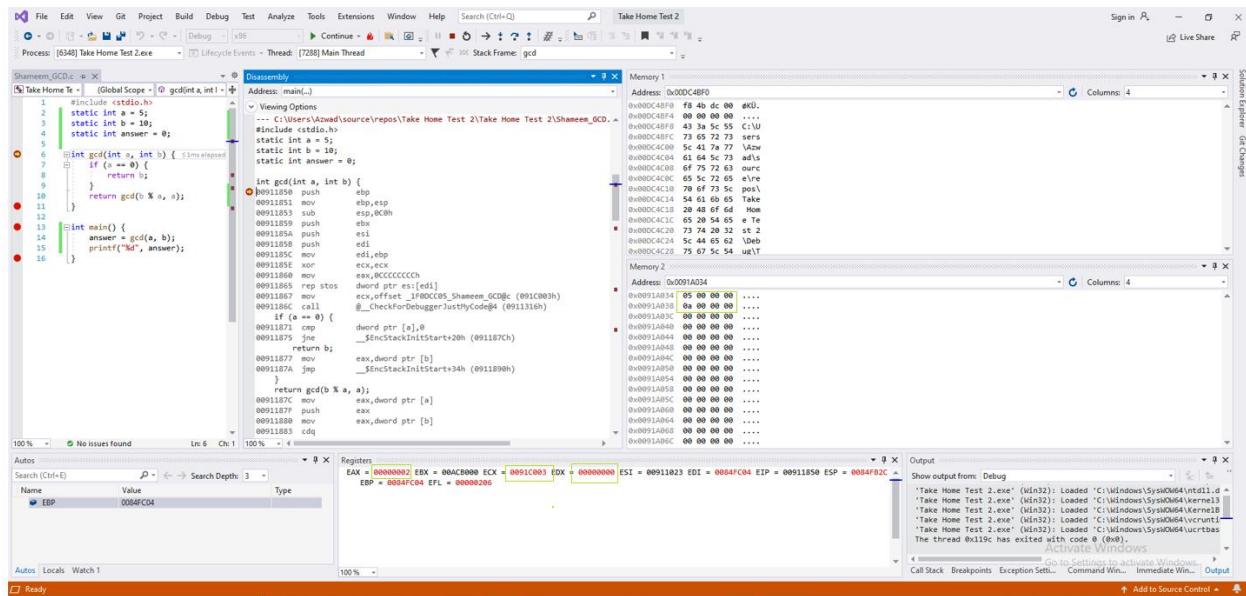


Figure 43: Shameem_GCD.c code in Visual Studio

Disassembly: This window shows that the code runs line 14 which calls the gcd() function at line 6.

Registers: This window shows that the value of EAX is now the value of variable b, which is 0x00000002, and the value of ECX is now the value 0x0091C003.

Memory: window 1 shows the values around the stack pointer which is somewhat initialized. While Memory window 2 displays the value of the static variables stored into the addresses.

Variable a is stored in address 0x0091A034 and has the value 05 00 00 00.

Variable b is stored in address 0x0091A038 and has the value 0a 00 00 00.

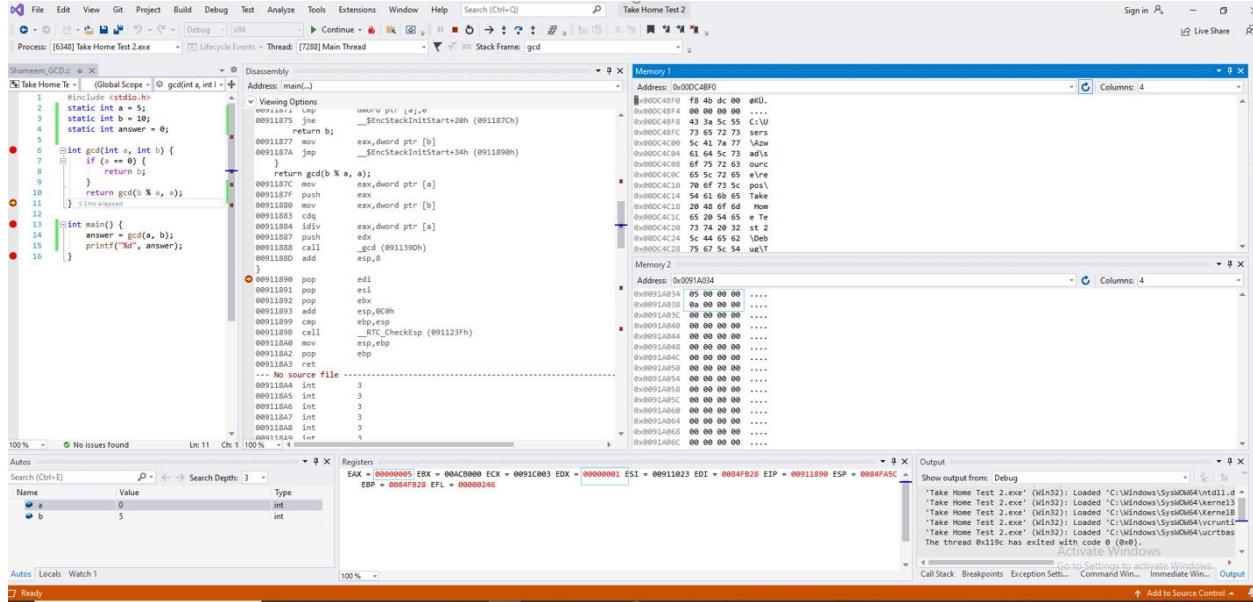


Figure 44: Shameem_GCD.c code in Visual Studio

Disassembly: This window shows that the code runs line 6-11.

Registers: This window shows that the value of EAX is now the value of variable b, which is 0x00000005, and the value of EDX is now the value 0x00000001.

Memory: window 1 shows the values around the stack pointer which is somewhat initialized. While Memory window 2 displays the value of the static variables stored into the addresses.

Variable a is stored in address 0x0091A034 and has the value 05 00 00 00.

Variable b is stored in address 0x0091A038 and has the value 0a 00 00 00.

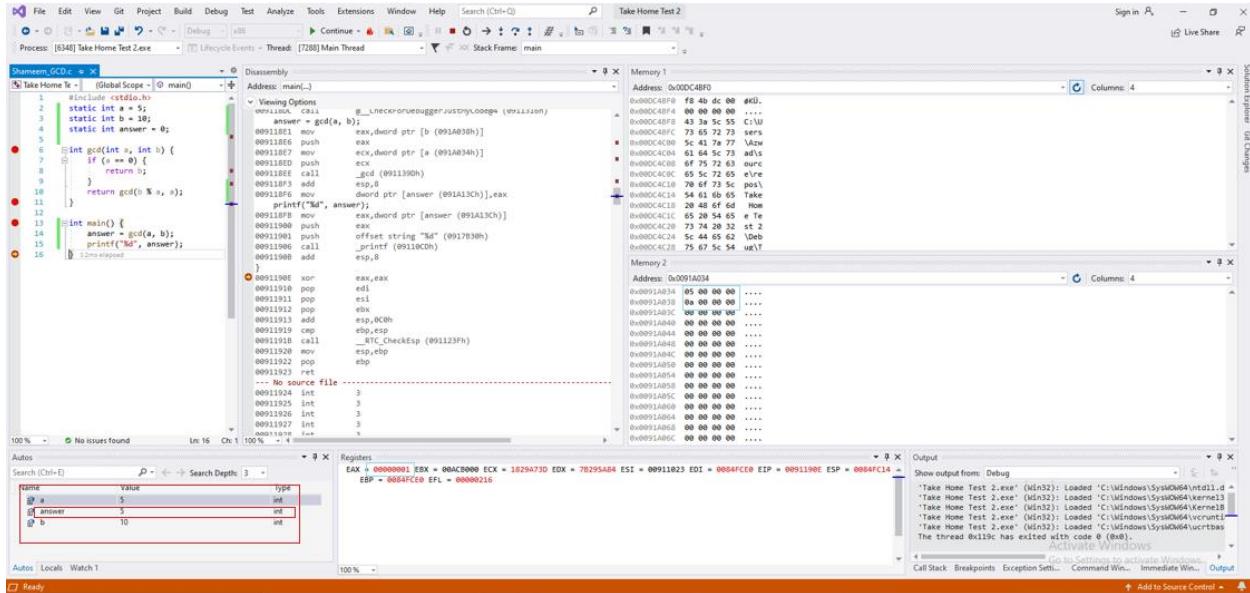


Figure 45: Shameem_GCD.c code in Visual Studio

Disassembly: This window shows that the code runs line 15.

Registers: This window shows that the value of EAX is now the value of variable b, which is 0x00000001, and the value of the other registers have been updated.

Memory: window 1 shows the values around the stack pointer which is somewhat initialized. While Memory window 2 displays the value of the static variables stored into the addresses.

Variable a is stored in address 0x0091A034 and has the value 05 00 00 00.

Variable b is stored in address 0x0091A038 and has the value 0a 00 00 00.

Also notice how the value of answer in the bottom left shows 5, which means the answer is printed out.

X86_64 ISA Linux 64-bit:

```
(gdb) list
1      #include <stdio.h>
2      static int a = 5;
3      static int b = 10;
4      static int answer = 0;
5
6      int gcd(int a, int b) {
7          if (a == 0) {
8              return b;
9          }
10         return gcd(b % a, a);
(gdb)
11     }
12
13     int main() {
14         answer = gcd(a, b);
15         printf("%d", answer);
16     }
(gdb) █
```

Figure 46: Shameem_GCD.c code in GDB

```
(gdb) disassemble
Dump of assembler code for function main:
0x0000555555555163 <+0>:    push   %rbp
0x0000555555555164 <+1>:    mov    %rsp,%rbp
=> 0x0000555555555167 <+4>:    mov    0x2ec7(%rip),%edx      # 0x555555558034 <b>
0x000055555555516d <+10>:   mov    0x2ebd(%rip),%eax      # 0x555555558030 <a>
0x0000555555555173 <+16>:   mov    %edx,%esi
0x0000555555555175 <+18>:   mov    %eax,%edi
0x0000555555555177 <+20>:   call   0x555555555135 <gcd>
0x000055555555517c <+25>:   mov    %eax,0x2eba(%rip)      # 0x55555555803c <answer>
0x0000555555555182 <+31>:   mov    0x2eb4(%rip),%eax      # 0x55555555803c <answer>
0x0000555555555188 <+37>:   mov    %eax,%esi
0x000055555555518a <+39>:   lea    0xe73(%rip),%rdi      # 0x555555556004
0x0000555555555191 <+46>:   mov    $0x0,%eax
0x0000555555555196 <+51>:   call   0x555555555030 <printf@plt>
0x000055555555519b <+56>:   mov    $0x0,%eax
0x00005555555551a0 <+61>:   pop    %rbp
0x00005555555551a1 <+62>:   ret

End of assembler dump.
(gdb) print /x $rsp
$1 = 0x7fffffff5a0
(gdb) print /x $rbp
$2 = 0x7fffffff5a0
(gdb) info registers
rax          0x555555555163      93824992235875
rbx          0x0                0
rcx          0x7fffff7fae718     140737353803544
rdx          0x7fffffff6a8      140737488348840
rsi          0x7fffffff698      140737488348824
rdi          0x1                1
rbp          0x7fffffff5a0      0x7fffffff5a0
rsp          0x7fffffff5a0      0x7fffffff5a0
r8           0x0                0
r9           0x7fffff7fe21b0     140737354015152
r10          0xf                15
r11          0x2                2
r12          0x555555555050     93824992235600
r13          0x0                0
r14          0x0                0
r15          0x0                0
rip          0x555555555167      0x555555555167 <main+4>
eflags        0x246              [ PF ZF IF ]
cs            0x33               51
ss            0x2b               43
ds            0x0                0
es            0x0                0
fs            0x0                0
gs            0x0                0
(gdb) █
```

Figure 47: Shameem_GCD.c code in GDB

Disassembly: This window shows that the code starts to run main().

Registers: This window shows the value that's stored \$rsp which is 0x7fffffff5a0 and the value that's stored in \$rbp which is at 0x0.

```

Dump of assembler code for function main:
0x000055555555163 <+0>:    push   %rbp
0x000055555555164 <+1>:    mov    %rsp,%rbp
0x000055555555167 <+4>:    mov    0xec(%rip),%edx      # 0x555555558034 <b>
0x00005555555516d <+10>:   mov    0xebd(%rip),%eax     # 0x555555558030 <a>
0x000055555555173 <+16>:   mov    %edx,%esi
0x000055555555175 <+18>:   mov    %eax,%edi
0x000055555555177 <+20>:   call   0x55555555135 <gcd>
0x00005555555517c <+25>:   mov    %eax,0xeba(%rip)  # 0x55555555803c <answer>
=> 0x000055555555182 <+31>:   mov    0xeb4(%rip),%eax  # 0x55555555803c <answer>
0x000055555555188 <+37>:   mov    %eax,%esi
0x00005555555518a <+39>:   lea    0xe73(%rip),%rdi  # 0x555555556004
0x000055555555191 <+46>:   mov    $0x0,%eax
0x000055555555196 <+51>:   call   0x55555555030 <printf@plt>
0x00005555555519b <+56>:   mov    $0x0,%eax
0x0000555555551a0 <+61>:   pop    %rbp
0x0000555555551a1 <+62>:   ret

End of assembler dump.
(gdb) print /x $eax
$6 = 0x5
(gdb) print /x $edx
$7 = 0x0
(ndb) ■

```

Figure 48: Shameem_GCD.c code in GD

Registers: The value of a is stored into \$eax. This is done to do computation of division with the value of b.

```
(gdb) disassemble
Dump of assembler code for function main:
0x000055555555163 <+0>:    push   %rbp
0x000055555555164 <+1>:    mov    %rsp,%rbp
0x000055555555167 <+4>:    mov    0x2ec7(%rip),%edx      # 0x555555558034 <b>
0x00005555555516d <+10>:   mov    0x2ebd(%rip),%eax      # 0x555555558030 <a>
0x000055555555173 <+16>:   mov    %edx,%esi
0x000055555555175 <+18>:   mov    %eax,%edi
0x000055555555177 <+20>:   call   0x55555555135 <gcd>
0x00005555555517c <+25>:   mov    %eax,0x2eba(%rip)      # 0x55555555803c <answer>
=> 0x000055555555182 <+31>:   mov    0x2eb4(%rip),%eax      # 0x55555555803c <answer>
0x000055555555188 <+37>:   mov    %eax,%esi
0x00005555555518a <+39>:   lea    0xe73(%rip),%rdi      # 0x555555556004
0x000055555555191 <+46>:   mov    $0x0,%eax
0x000055555555196 <+51>:   call   0x55555555030 <printf@plt>
0x00005555555519b <+56>:   mov    $0x0,%eax
0x0000555555551a0 <+61>:   pop    %rbp
0x0000555555551a1 <+62>:   ret
End of assembler dump.
(gdb) x/8xb 0x555555558034
0x555555558034 <b>: 0x0a 0x00 0x00 0x00 0x00 0x00 0x00 0x00
(gdb) x/8xb 0x555555558030
0x555555558030 <a>: 0x05 0x00 0x00 0x0a 0x00 0x00 0x00 0x00
(gdb) x/8xb 0x55555555803c
0x55555555803c <answer>: 0x05 0x00 0x00 0x00 0x00 0x00 0x00 0x00
(gdb) ■
```

Figure 49: Shameem_GCD.c code in GDB

Memory: This shows the address 0x555555558034 which has the value 0x0a 0x00 0x00 0x00 0x00 0x00 0x00 0x00 which is the variable b. Also, shows the address 0x555555558034 which has the value 0x05 0x000x00 0x00 0x00 0x00 0x00 0x00 0x00 which is the value of variable a. Lastly, shows the address 0x555555558034 which has the value 0x05 0x000x00 0x00 0x00 0x00 0x00 0x00, which is the value of the answer.

Conclusions

In conclusion the Take Home Test helped us to test, understand and demonstrate the recursive calls and stack frames in the three different architecture MARS Simulator, Visual Studio and Linux GDB and GCC. Therefore, by explaining the recursive steps of the $\text{GCD}(a, b)$ function, we can gain a good understanding how the function works in three different architectures, MIPS on MARS Simulator, Intel x86 using Windows MS 32-bit compiler on Visual Studio and Intel X86_64 bit on Linux GDB and GCC. After explaining the recursive $\text{GCD}(a, b)$ function we understand the difference in how the function runs step by step in the different architectures.