**INTRODUCTION:**

This activity is used to run a sample assembly code in both MARS and MIPSASM and compare the machine code of both assemblers, logging few registers’ values and memory content of selected addresses.

**MY GROUP:**

Name: Student\_Team 6

Harish Marepalli – 016707314

Tirumala Saiteja Goruganthu - 016707210

**SOURCE CODE:**

# mipstest.smd

# Test the following MIPS instructions.

# add, sub, and, or, slt, addi, lw, sw, beq, j

# Assembly Description Address Machine

main: addi $2, $0, 5 # initialize $2 = 5 0 20020005

addi $3, $0, 12 # initialize $3 = 12 4 2003000c

addi $7, $3, -9 # initialize $7 = 3 8 2067fff7

or $4, $7, $2 # $4 <= 3 or 5 = 7 c 00e22025

and $5, $3, $4 # $5 <= 12 and 7 = 4 10 00642824

add $5, $5, $4 # $5 = 4 + 7 = 11 14 00a42820

beq $5, $7, end # shouldn't be taken 18 10a7000a

slt $4, $3, $4 # $4 = 12 < 7 = 0 1c 0064202a

beq $4, $0, around # should be taken 20 10800001

addi $5, $0, 0 # shouldn't execute 24 20050000

around: slt $4, $7, $2 # $4 = 3 < 5 = 1 28 00e2202a

add $7, $4, $5 # $7 = 1 + 11 = 12 2c 00853820

sub $7, $7, $2 # $7 = 12 - 5 = 7 30 00e23822

sw $7, 68($3) # [80] = 7 34 ac670044

lw $2, 80($0) # $2 = [80] = 7 38 8c020050

j end # should be taken 3c 08000011

addi $2, $0, 1 # shouldn't execute 40 20020001

end: sw $2, 84($0) # write adr 84 = 7 44 ac020054

j main # go back to beginning 48 08000c00

**TEST LOG:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adr | Machine Code  for MARS | Machine Code  for MIPSASM | PC | Registers | | | | | Memory Content | |
| $v0 | $v1 | $a0 | $a1 | $a3 | [80] | [84] |
| 3000 | 0x20020005 | 0x20020005 | 0x00003004 | 0x00000005 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 3004 | 0x2003000c | 0x2003000C | 0x00003008 | 0x00000005 | 0x0000000c | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 3008 | 0x2067fff7 | 0x2067FFF7 | 0x0000300c | 0x00000005 | 0x0000000c | 0x00000000 | 0x00000000 | 0x00000003 | 0x00000000 | 0x00000000 |
| 300c | 0x00e22025 | 0x00E22025 | 0x00003010 | 0x00000005 | 0x0000000c | 0x00000007 | 0x00000000 | 0x00000003 | 0x00000000 | 0x00000000 |
| 3010 | 0x00642824 | 0x00642824 | 0x00003014 | 0x00000005 | 0x0000000c | 0x00000007 | 0x00000004 | 0x00000003 | 0x00000000 | 0x00000000 |
| 3014 | 0x00a42820 | 0x00A42820 | 0x00003018 | 0x00000005 | 0x0000000c | 0x00000007 | 0x0000000b | 0x00000003 | 0x00000000 | 0x00000000 |
| 3018 | 0x10a7000a | 0x10E5000A | 0x0000301c | 0x00000005 | 0x0000000c | 0x00000007 | 0x0000000b | 0x00000003 | 0x00000000 | 0x00000000 |
| 301c | 0x0064202a | 0x0064202A | 0x00003020 | 0x00000005 | 0x0000000c | 0x00000000 | 0x0000000b | 0x00000003 | 0x00000000 | 0x00000000 |
| 3020 | 0x10800001 | 0x10040001 | 0x00003028 | 0x00000005 | 0x0000000c | 0x00000000 | 0x0000000b | 0x00000003 | 0x00000000 | 0x00000000 |
| 3024 | 0x20050000 | 0x20050000 |  |  |  |  |  |  |  |  |
| 3028 | 0x00e2202a | 0x00E2202A | 0x0000302c | 0x00000005 | 0x0000000c | 0x00000001 | 0x0000000b | 0x00000003 | 0x00000000 | 0x00000000 |
| 302c | 0x00853820 | 0x00853820 | 0x00003030 | 0x00000005 | 0x0000000c | 0x00000001 | 0x0000000b | 0x0000000c | 0x00000000 | 0x00000000 |
| 3030 | 0x00e23822 | 0x00E23822 | 0x00003034 | 0x00000005 | 0x0000000c | 0x00000001 | 0x0000000b | 0x00000007 | 0x00000000 | 0x00000000 |
| 3034 | 0xac670044 | 0xAC670044 | 0x00003038 | 0x00000005 | 0x0000000c | 0x00000001 | 0x0000000b | 0x00000007 | 0x00000007 | 0x00000000 |
| 3038 | 0x8c020050 | 0x8C020050 | 0x0000303c | 0x00000007 | 0x0000000c | 0x00000001 | 0x0000000b | 0x00000007 | 0x00000007 | 0x00000000 |
| 303c | 0x08000c11 | 0x08000011 | 0x00003044 | 0x00000007 | 0x0000000c | 0x00000001 | 0x0000000b | 0x00000007 | 0x00000007 | 0x00000000 |
| 3040 | 0x20020001 | 0x20020001 |  |  |  |  |  |  |  |  |
| 3044 | 0xac020054 | 0xAC020054 | 0x00003048 | 0x00000007 | 0x0000000c | 0x00000001 | 0x0000000b | 0x00000007 | 0x00000007 | 0x00000007 |
| 3048 | 0x08000c00 | 0x08000000 | 0x00003000 | 0x00000007 | 0x0000000c | 0x00000001 | 0x0000000b | 0x00000007 | 0x00000007 | 0x00000007 |

**SCREEN CAPTURES:**

1. Screenshot of the code in the MARS editor.

![Graphical user interface, text

Description automatically generated with medium confidence]()

1. Screenshot of the execution window after assembling and before executing the code in MARS.

![Graphical user interface, application, table

Description automatically generated]()

1. Screenshot of the execution window after executing the code in MARS.

![Graphical user interface, application, table, Excel

Description automatically generated]()

1. Screenshot of the window containing code in editor, assembled data, and Registers / Memory wizard before execution in MIPSASM.

![Table

Description automatically generated with medium confidence]()

1. Screenshot of the window containing code in editor, assembled data, and Registers / Memory wizard after execution in MIPSASM.

![Text, table

Description automatically generated with medium confidence]()

**DISCUSSION SECTION:**

The explanation of unique instructions in the sample code with the help of the MIPS reference data card.

1. *addi rt, rs, imm => rt = rs + imm;*

Add Immediate value to the source register content.

1. *or rd, rs, rt => rd = rs | rt;*

Perform OR operation between the contents of the register rs, rt and putting the result in register rd.

1. and rd, rs, rt => rd = rs & rt;

Perform AND operation between the contents of the register rs, rt and putting the result in register rd.

1. add rd, rs, rt => rd = rs + rt;

Perform ADD operation between the contents of the register rs, rt and putting the result in register rd.

1. beq rt, rs, Target => if(rt == rs) branch to target;

If the contents of registers rt and rs are equal then the program counter will branch to the target.

1. slt rd, rs, rt => if(rs < rt) rd = 1; else rd = 0;

If the content of register rs is less than rt then rd will become 1 else it becomes 0.

1. sub rd, rs, rt => rd = rs – rt;

Perform SUB operation between the contents of the register rs, rt and putting the result in register rd.

1. sw rt, imm(rs) => Memory[rs + imm] = rt;

Stores the content of rt in memory using the location provided by the combination of register rs and the offset value (imm).

1. lw rt, imm(rs) => rt = Memory[rs + imm];

Loads the content of memory in the location provided by the combination of register rs and the offset value (imm) and puts the result in register rt.

1. j addr => PC = addr;

Jumps to the given target address by assigning this value to the program counter (PC).

Following are the observations related to the machine codes:

1. Explanation with respect to the first line

i.e., addi $2, $0, 5

This is an I-type instruction and the machine code is written as:

6 bits 5 bits 5 bits 16 bits

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Rs | Rt | Immediate |

32 bits

Opcode for addi is 001000

Rs = $0 = 00000

Rt = $2 = 00010

Immediate = 5 = 0000000000000101

So, the 32 bits would be 00100000000000100000000000000101

By dividing it into 4 bits each, we get the machine code.

So, the machine code is 0010 0000 0000 0010 0000 0000 0000 0101 = 20020005 = 0x20020005.

1. Explanation with respect to the fourth line

i.e. or $4, $7, $2

This is an R-type instruction and the machine code is written as:

6 bits 5 bits 5 bits 5 bits 5 bits 6 bits

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Opcode | Rs | Rt | Rd | Shamt | Function |

32 bits

For all R-type instructions, the Opcode is 0 = 000000.

Rs = $7 = 00111

Rt = $2 = 00010

Rd = $4 = 00100

For all non-shift type instructions, the Shamt is 0 = 00000

For or, the function bits are 100101

So, the 32 bits would be 00000000111000100010000000100101

By dividing it into 4 bits each, we get the machine code.

So, the machine code is 0000 0000 1110 0010 0010 0000 0010 0101 = 00e22025 = 0x00e22025.

Following are the observations in the test log:

1. The machine codes for MARS and MIPSASM assemblers for the same assembly program is similar for the most part except for the lines with addresses 0x00003018, 0x00003020, 0x0000303c and 0x00003048. The different machine codes for these addresses are given in the table below.

|  |  |  |
| --- | --- | --- |
| Address | Machine Code in MARS | Machine Code in MIPSASM |
| 0x00003018 | 0x10a7000a | 0x10E5000A |
| 0x00003020 | 0x10800001 | 0x10040001 |
| 0x0000303c | 0x08000c11 | 0x08000011 |
| 0x00003048 | 0x08000c00 | 0x08000000 |

1. The program counter value will never be 0x00003024 and 0x00003040 because of the ‘beq’ instruction, these lines will not get executed during run-time.

**COLLABORATION SECTION:**

1. Installed and setup MARS and MIPSASM by collaborating with each other.
2. Assembled the given MIPS assembly code in the MARS by collaborating with each other.
3. Executed the code and observed all the operations and values.
4. Collaborated with each other to compare the machine code generated by two different assemblers.
5. In addition to the mipstest.asm, modified and run the Fibonacci code in MARS by collaborating and discussing the operations done.
6. By collaborating with each other, debugged each line to know the contents of the relevant registers and recorded the memory values at certain addresses.

**CONCLUSION:**

In conclusion, the machine codes of MARS and MIPSASM for all the addresses cannot be the same. By assembling, simulating, and analyzing the given sample program gained familiarity with the MIPS instruction set.