**1. Multiplication of two numbers:**

**Venus Simulator Code:**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256 # make room for a0, ra

addi a0, x0, 5 # n

addi a1, x0, 100 # n2

add ra, x0, x0

add t0, x0, x0

slt t1, a1, a0 # slt instruction: slt rd, rs1, rs2; rd is set to 1 if rs1<rs2 else 0.

# if t1 is 1 proceed to normal multiplication; if 0 then switch operands.

addi t2, x0, 1 # to compare if result of t1 is

bne t1, t2, start2 # operand switching.

# normal addition

start: add ra, ra, a0

addi t0, t0, 1

bne t0, a1, start

beq t0, a1, end

start2: add t1, x0, x0

add t1, x0, a1

start3: add ra, ra, t1

addi t0, t0, 1

bne t0, a0, start3

end:

sw ra, 0(sp)

add sp, s0, x0

loop: jal zero, loop

**Result:**

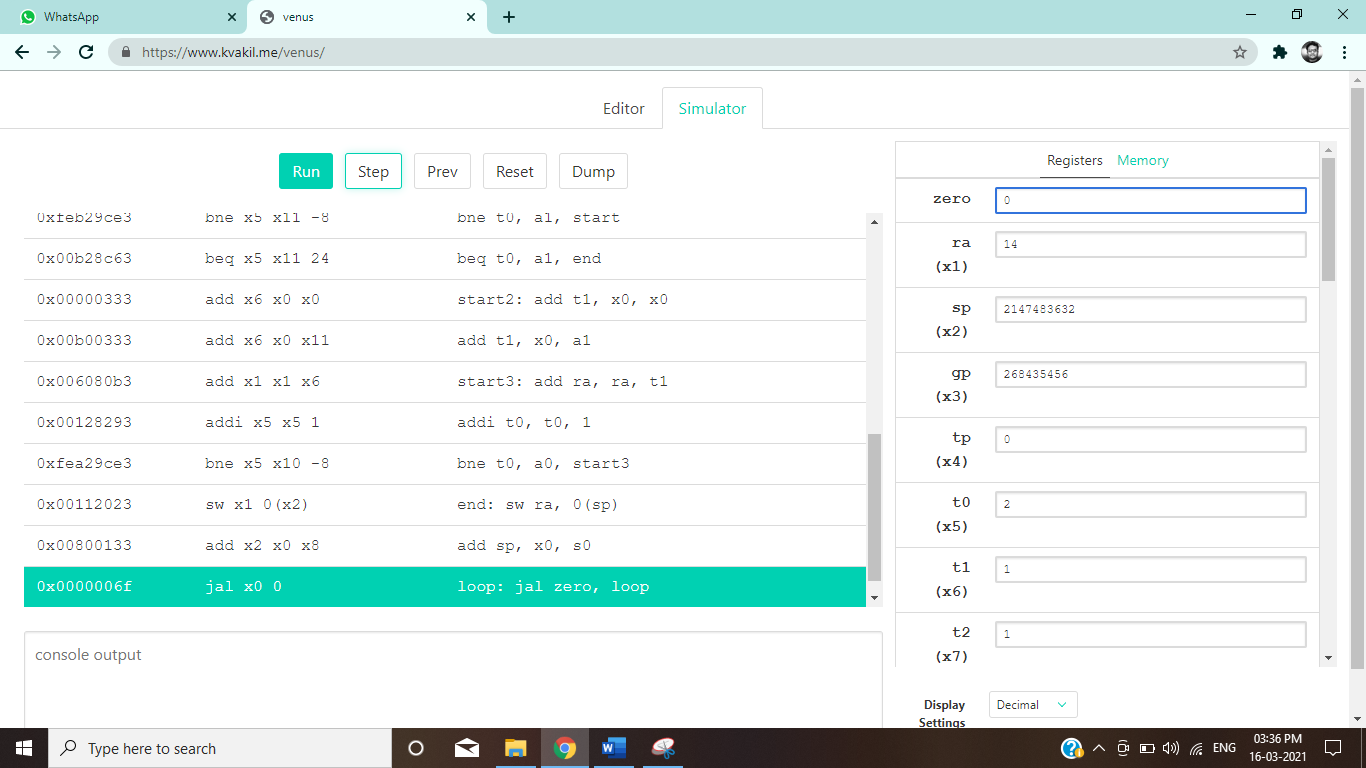
* **Input:**





* **Output**





**Modelsim Code:**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256 # make room for a0, ra

lw a0, 4(sp)

lw a1, 8(sp)

lw ra, 0(sp)

add t0, x0, x0

slt t1, a1, a0 # slt instruction: slt rd, rs1, rs2; rd is set to 1 if rs1<rs2 else 0.

# if t1 is 1 proceed to normal multiplication; if 0 then switch operands.

addi t2, x0, 1 # to compare if result of t1 is

bne t1, t2, start2 # operand switching.

# normal addition

start: add ra, ra, a0

addi t0, t0, 1

bne t0, a1, start

beq t0, a1, end

start2: add t1, x0, x0

add t1, x0, a1

start3: add ra, ra, t1

addi t0, t0, 1

bne t0, a0, start3

end:

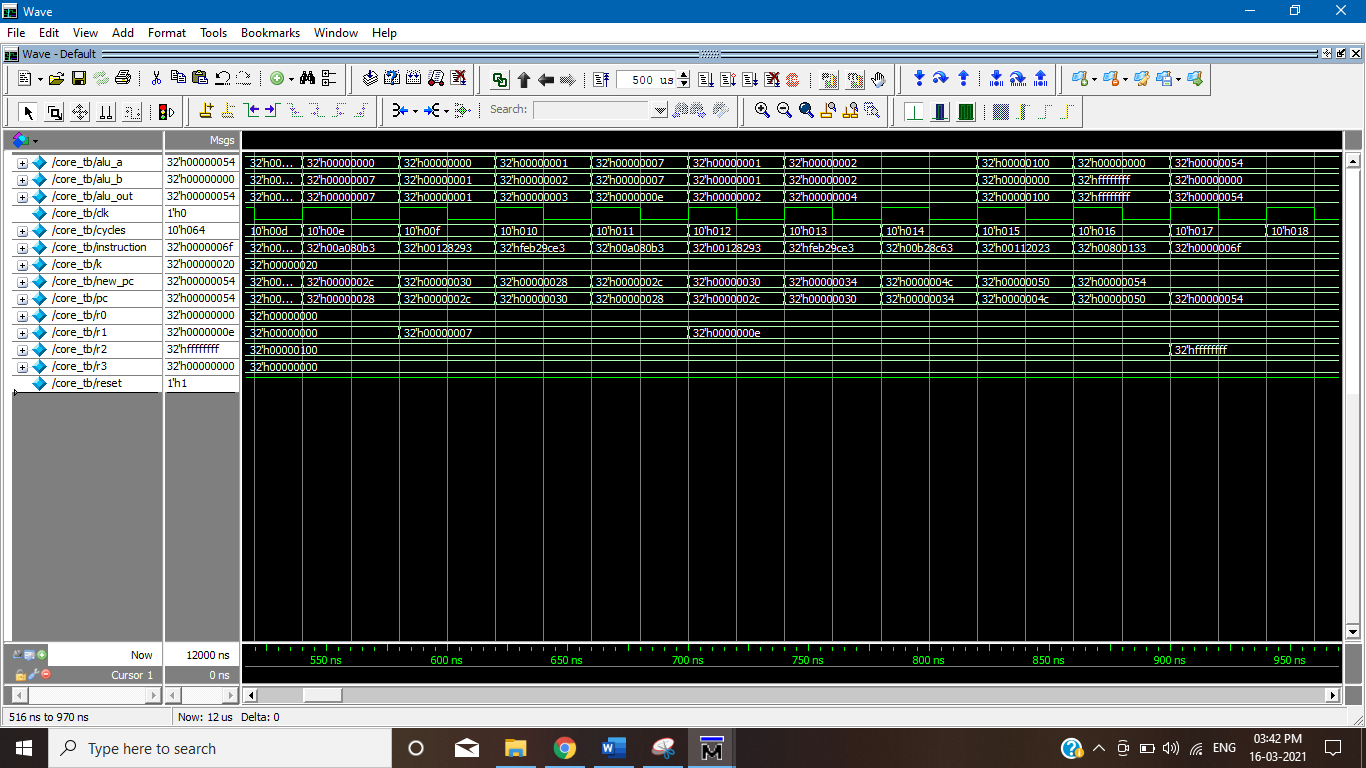
sw ra, 0(sp)

add sp, x0, s0

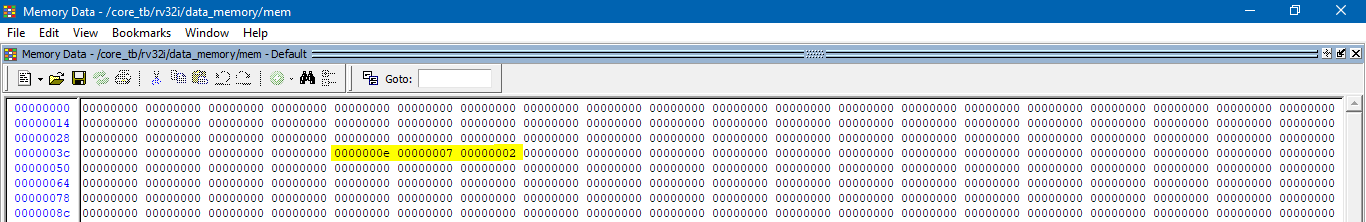
loop: jal zero, loop

**Result:**

* **Wave Output:**



* **Data Memory Output:**



**2. Factorial:**

**Venus Simulator Code**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256 # make room for a0, ra

addi a2, x0, 5 # load the first operand n

add ra, x0, x0

add t0, x0, x0

addi t1, x0, 1

# if n==0 ans = 1

addi ra, x0, 1

beq t0, a2, end

# if n==1 ans = 1

addi ra, x0, 1

beq t1, a2, end

# else

#Final answer must be in ra,

start1:

addi t1, t1, 1 # so it starts from 2.

# assigning multiplication values.

add a0, x0, x0

add a0, x0, ra # setting the a0 inner loop variable to operand 1

add a1, x0, x0

add a1, x0, t1 # setting the a1 inner loop variable to operand 2

add t0, x0, x0 # setting inner counter back to zero

add ra, x0, x0 # initialize ra to 0

#Inner Loop and the loop variables are t0(for counter) a1 and a2 (the operands),

# this inner loop does the function a0, a1.

start2: add ra, ra, a0

addi t0, t0, 1

bne t0, a1, start2

#Inner Loop ends

bne t1, a2, start1

end:

sw ra, 0(sp)

add sp, s0, x0

loop: jal zero, loop

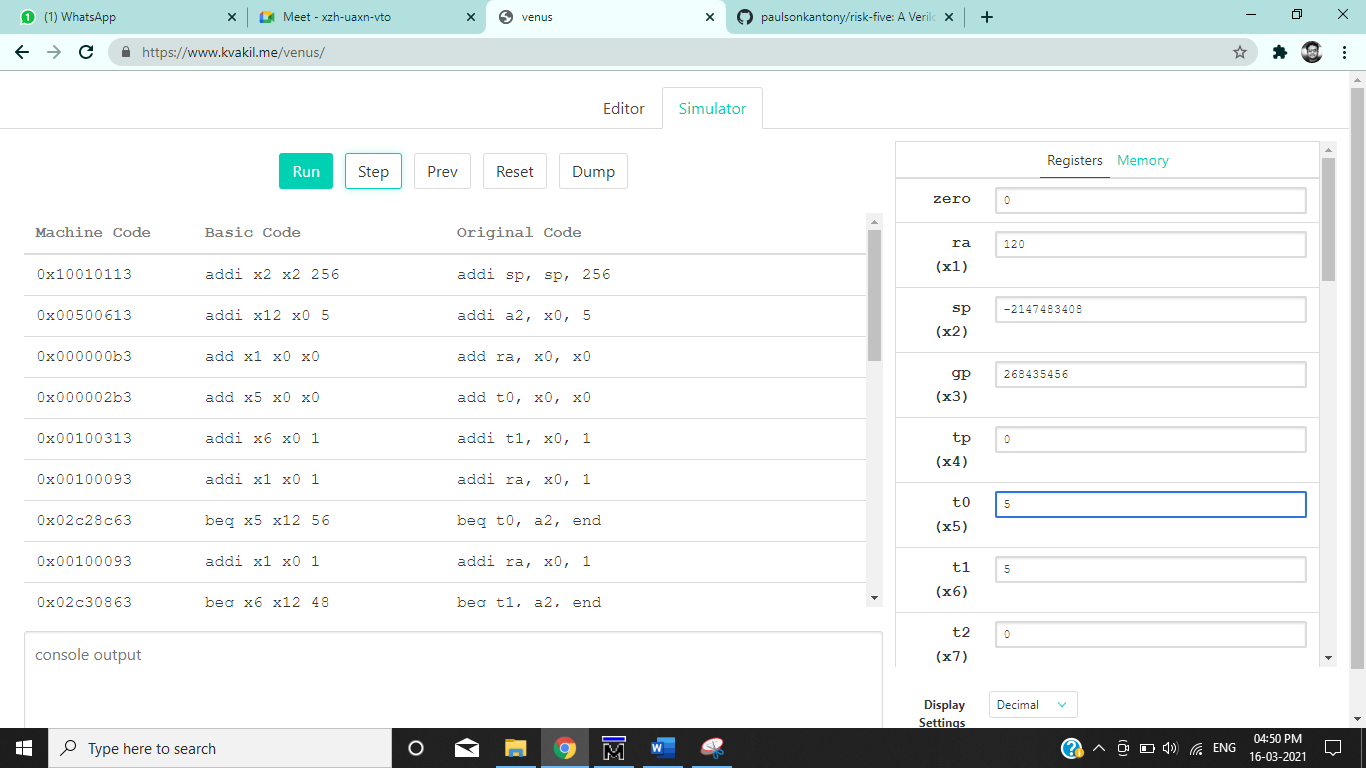
**Result:**

* **Input:**



* **Output:**





**Modelsim Code**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256 # make room for a0, ra

lw a2, 4(sp) # load the first operand n

lw ra, 0(sp)

add t0, x0, x0

addi t1, x0, 1

# if n==0 ans = 1

addi ra, x0, 1

beq t0, a2, end

# if n==1 ans = 1

addi ra, x0, 1

beq t1, a2, end

# else

#Final answer must be in ra,

start1:

addi t1, t1, 1 # so it starts from 2.

# assigning multiplication values.

add a0, x0, x0

add a0, x0, ra # setting the a0 inner loop variable to operand 1

add a1, x0, x0

add a1, x0, t1 # setting the a1 inner loop variable to operand 2

add t0, x0, x0 # setting inner counter back to zero

add ra, x0, x0 # initialize ra to 0

#Inner Loop and the loop variables are t0(for counter) a1 and a2 (the operands),

# this inner loop does the function a0, a1.

start2: add ra, ra, a0

addi t0, t0, 1

bne t0, a1, start2

#Inner Loop ends

bne t1, a2, start1

end:

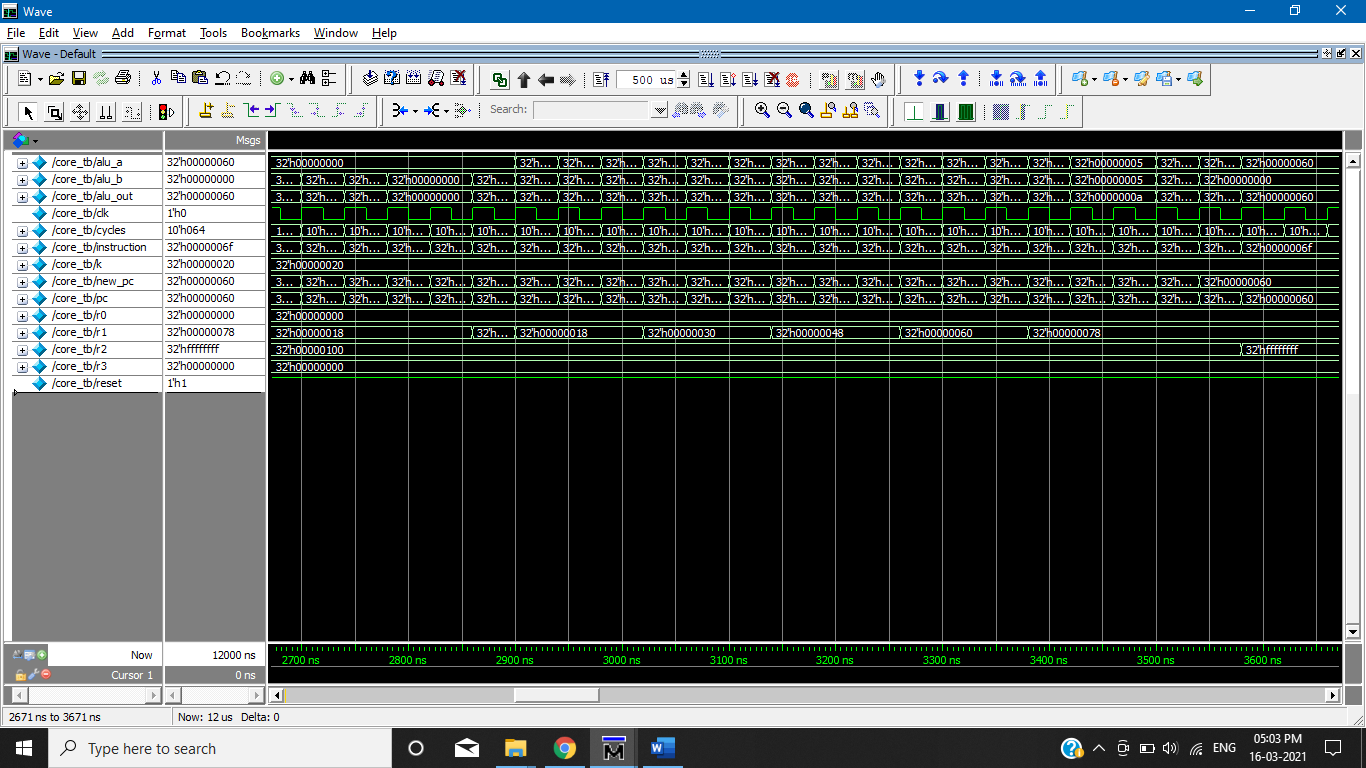
sw ra, 0(sp)

add sp, s0, x0

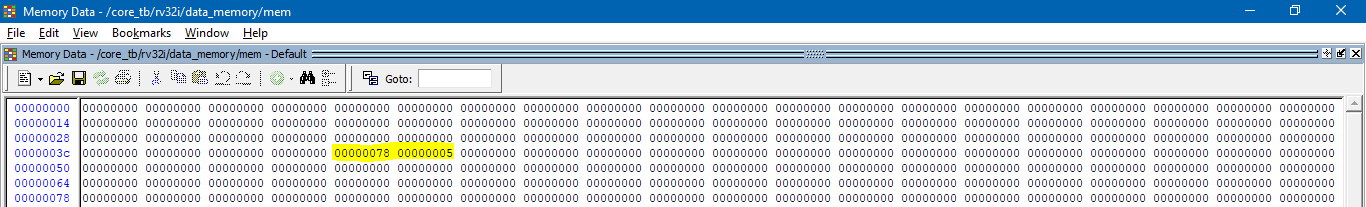
loop: jal zero, loop

**Result:**

* **Wave Output:**



* **Data Memory Output:**



**3. Exponentiation:**

**Venus Simulator Code:**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256 # make room for a0, ra

addi a0, x0, 1 # number

addi a1, x0, 5 # exponent

add ra, x0, x0

add t0, x0, x0

addi t1, x0, 1

#if n==0 ans=0

beq ra, a0, end

#if n==1 ans=1

addi ra, x0, 1

beq t1, a0, end

# if exp==0 ans = 1

addi ra, x0, 1

beq x0, a1, end

# if exp==1 ans = a0

add ra, x0, a0

beq t1, a1, end

start1:

addi t1, t1, 1 # so it starts from 2.

# assigning multiplication values.

add a2, x0, x0

add a2, x0, ra # setting the a2 inner loop variable to operand 1

add a3, x0, x0

add a3, x0, a0 # setting the a3 inner loop variable to operand 2

add t0, x0, x0 # setting inner counter back to zero

add ra, x0, x0 # initialize ra to 0

#Inner Loop and the loop variables are t0(for counter) a1 and a2 (the operands),

# this inner loop does the function a2 \* a3 and stores the result in ra.

start2: add ra,ra, a2

addi t0, t0, 1

bne t0, a3, start2

#Inner Loop ends

bne t1, a1, start1

end:

sw ra, 0(sp)

add sp, s0, x0

loop: jal zero, loop

**Result:**

* **Input:**





* **Output**



**Modelsim Code:**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256

lw a0, 4(sp) # number

lw a1, 8(sp) # exponent

lw ra, 0(sp)

add t0, x0, x0

addi t1, x0,

#if n==0 ans=0

beq ra, a0, end

#if n==1 ans=1

addi ra, x0, 1

beq t1, a0, end

# if exp==0 ans = 1

addi ra, x0, 1

beq x0, a1, end

# if exp==1 ans = a0

add ra, x0, a0

beq t1, a1, end

start1:

addi t1, t1, 1 # so it starts from 2.

# assigning multiplication values.

add a2, x0, x0

add a2, x0, ra # setting the a2 inner loop variable to operand 1

add a3, x0, x0

add a3, x0, a0 # setting the a3 inner loop variable to operand 2

add t0, x0, x0 # setting inner counter back to zero

add ra, x0, x0 # initialize ra to 0

#Inner Loop and the loop variables are t0(for counter) a1 and a2 (the operands),

# this inner loop does the function a2 \* a3 and stores the result in ra.

start2: add ra,ra, a2

addi t0, t0, 1

bne t0, a3, start2

#Inner Loop ends

bne t1, a1, start1

end:

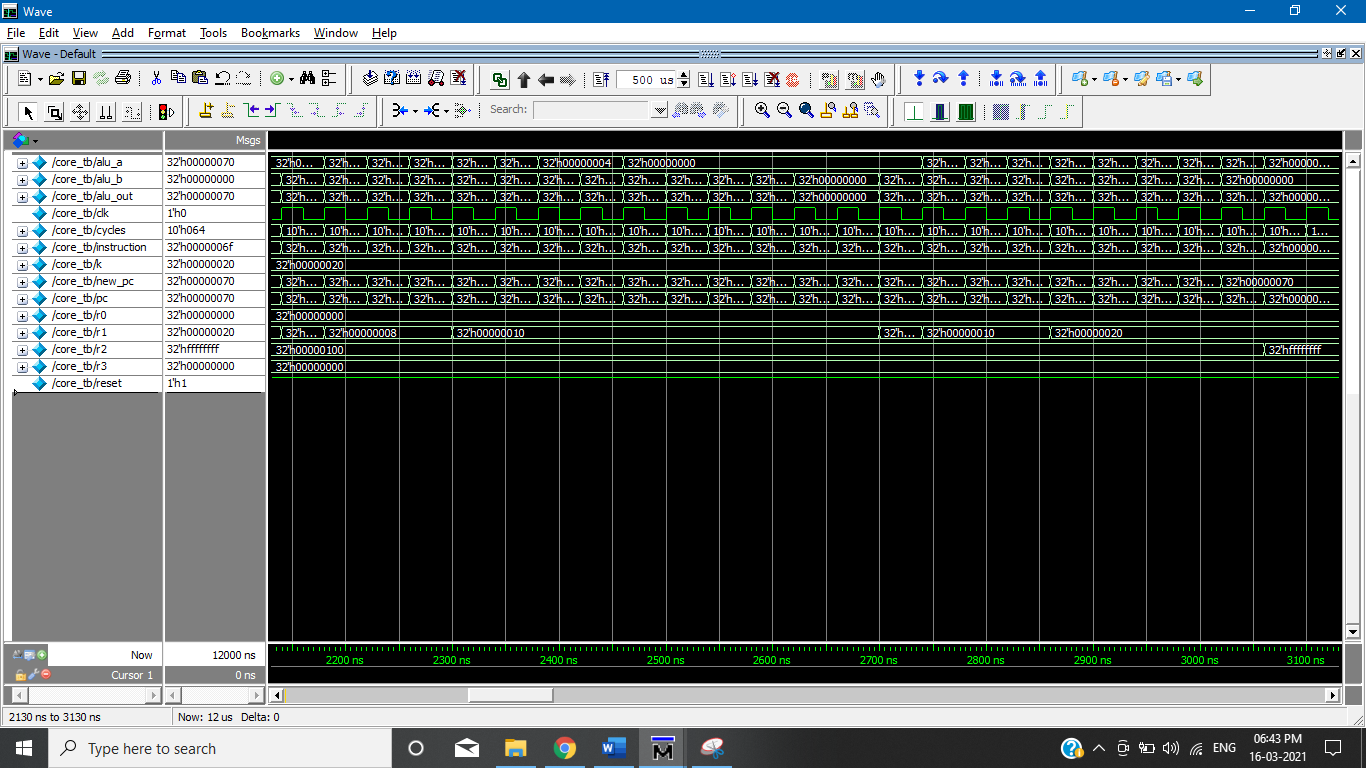
sw ra, 0(sp)

add sp, s0, x0

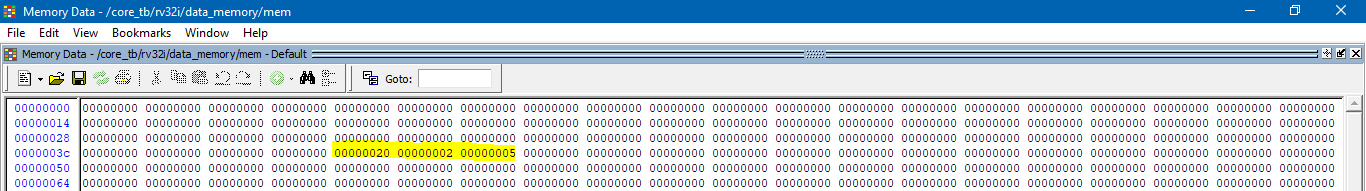
loop: jal zero, loop

**Result:**

* **Wave Output:**



* **Data Memory Output:**



**4. Comparator**

**Venus Simulator Code:**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256

addi a0, x0, 5

addi a1, x0, 5

add ra, x0, x0

addi t0, x0, 2

xor ra, a0, a1

addi ra, ra, 2

beq ra, t0, end

slt ra, a0, a1

end:

sw ra, 0(sp)

add sp, s0, x0

loop: jal zero, loop

# output 2 if equal, output 1 if operand 1 is less, output 0 if operand 1 is greater

**Result:**

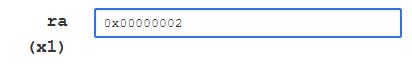
**Case-1:** Input-A: 5 Input-B: 5 Output: 2

* **Input**





* **Output**



**Case-2:** Input-A: 5 Input-B: 2 Output: 0

* **Input**





* **Output**



**Case-3:** Input-A: 2 Input-B: 5 Output: 1

* **Input**





* **Output**



**Modelsim Code:**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256

lw a0, 4(sp) # operand 1

lw a1, 8(sp) # operand 2

lw ra, 0(sp)

addi t0, x0, 2

xor ra, a0, a1

addi ra, ra, 2

beq ra, t0, end

slt ra, a0, a1

end:

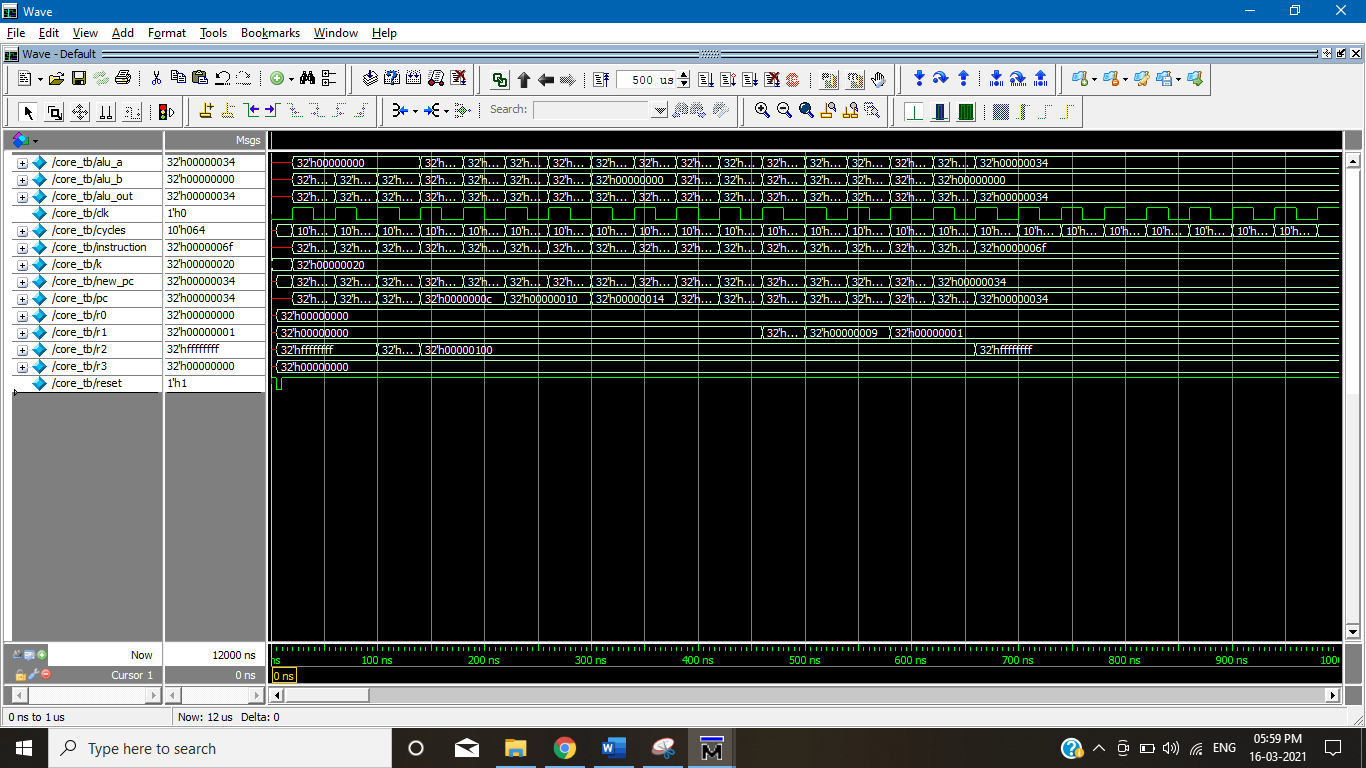
sw ra, 0(sp)

add sp, s0, x0

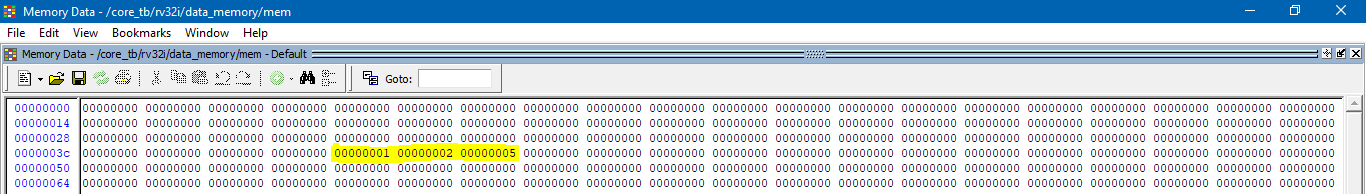
loop: jal zero, loop

**Result:**

* **Wave Output:**



* **Data Memory Output:**



**5. Code for Nth number of a Fibonacci series**

**Venus Simulator Code:**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256 # make room for a0, ra

addi a0, x0, 7 # n1

add ra, x0, x0

addi t0, x0, 1

addi a1, x0, 0

addi a2, x0, 1

# if n==1 ans=0

beq ra, a0,end

# if n==2 ans=1

beq t0, a0, end

# else

addi t0, t0, 1

sub a0, a0, t0

addi t0, x0, 0

start:

add ra, a1, a2

add a1, x0, a2

add a2, x0, ra

addi t0, t0, 1

bne t0, a0, start

end:

sw ra, 0(sp)

add sp, s0, x0

loop: jal zero, loop

**Result:**

* **Input:**



* **Output:**



**Modelsim Code:**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256

lw a0, 4(sp)

add ra, x0, x0

addi t0, x0, 1

addi a1, x0, 0

addi a2, x0, 1

beq ra, a0,end

beq t0, a0, end

addi t0, t0, 1

sub a0, a0, t0

addi t0, x0, 0

start:

add ra, a1, a2

add a1, x0, a2

add a2, x0, ra

addi t0, t0, 1

bne t0, a0, start

end:

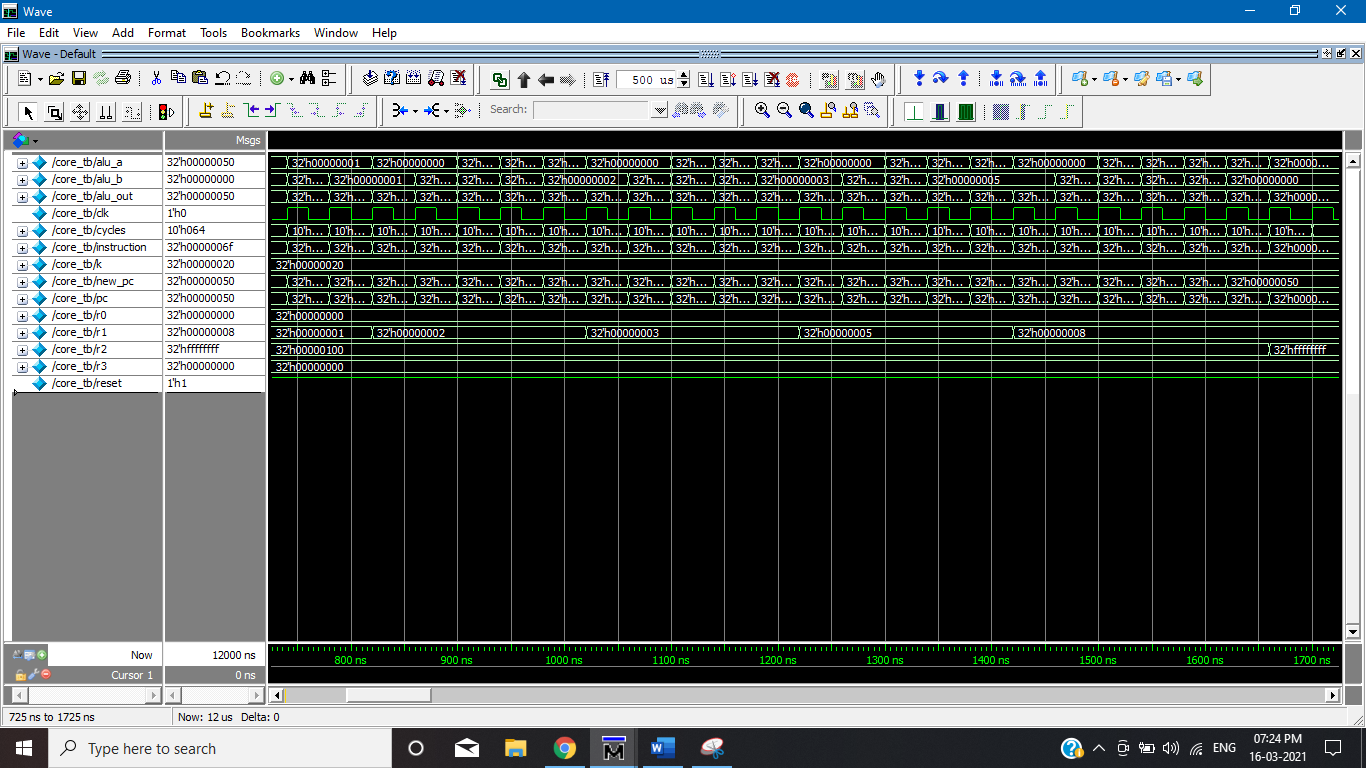
sw ra, 0(sp)

add sp, s0, x0

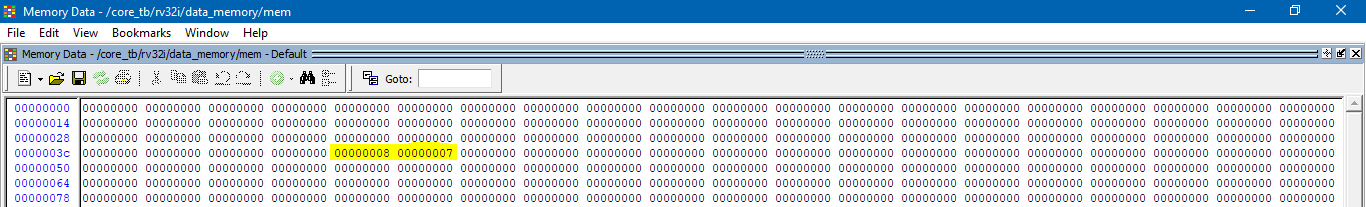
loop: jal zero, loop

**Result:**

* **Wave Output:**



* **Data Memory Output:**



**6. Nth term of an Arithmetic Progression**

**Venus Simulation Code:**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256 # make room for a0, ra

addi a0, x0, 7 # 1st term

addi a1, x0, 7 # common difference

addi a2, x0, 7 # nth term to find

add ra, x0, x0

addi t0, x0, 1

# if n==1 ans=1st term

add ra, x0, a0

beq t0, a2,end

# else 1st term + (n-1)\*d

sub a2, a2, t0

addi t0, x0, 0

start:

add ra, ra, a2

addi t0, t0, 1

bne t0, a1, start

end:

sw ra, 0(sp)

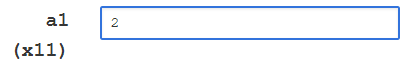
add sp, s0, x0

loop: jal zero, loop

**Results:**

* **Input**







* **Output**



**Modelsim Code:**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256 # make room for a0, ra

lw a0, 4(sp)

lw a1, 8(sp)

lw a2, 12(sp)

add ra, x0, x0

addi t0, x0, 1

# if n==1 ans=1st term

add ra, x0, a0

beq t0, a2,end

# else 1st term + (n-1)\*d

sub a2, a2, t0

addi t0, x0, 0

start:

add ra, ra, a2

addi t0, t0, 1

bne t0, a1, start

end:

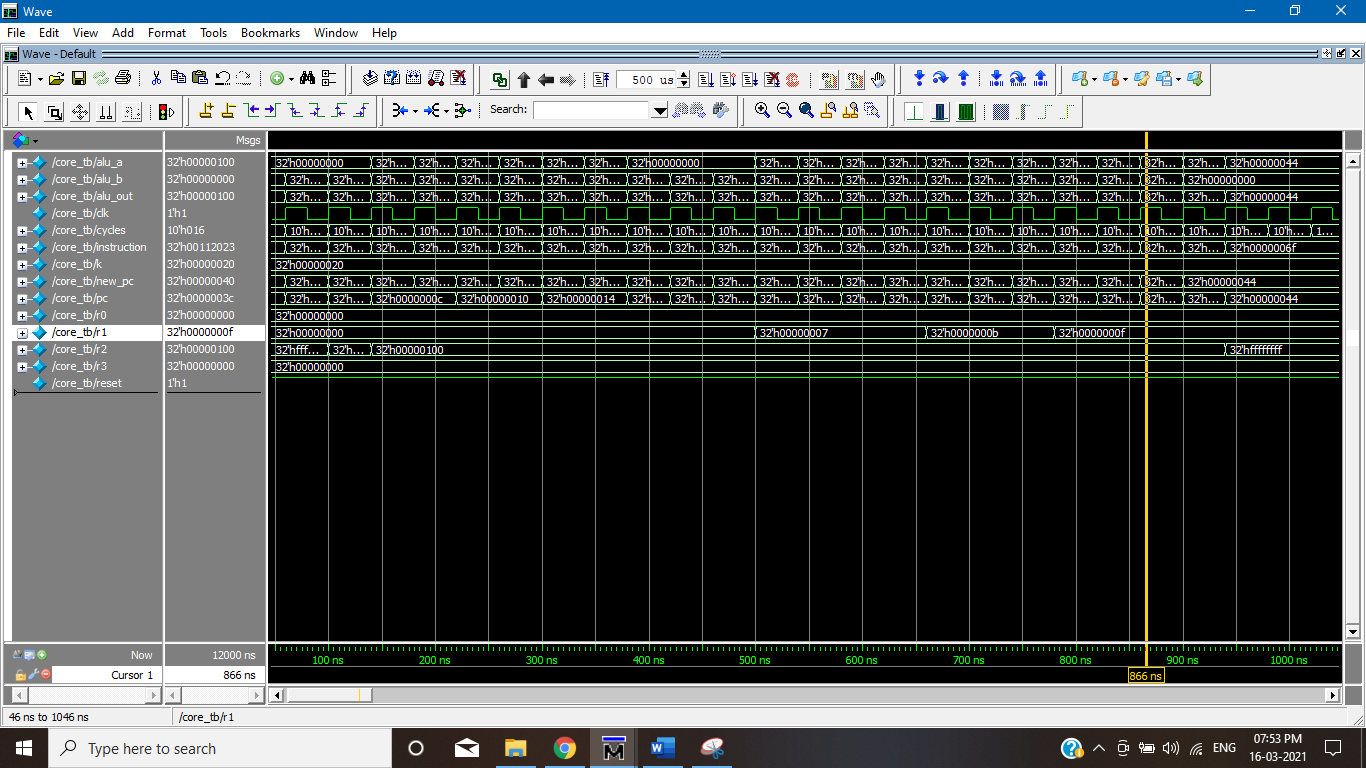
sw ra, 0(sp)

add sp, s0, x0

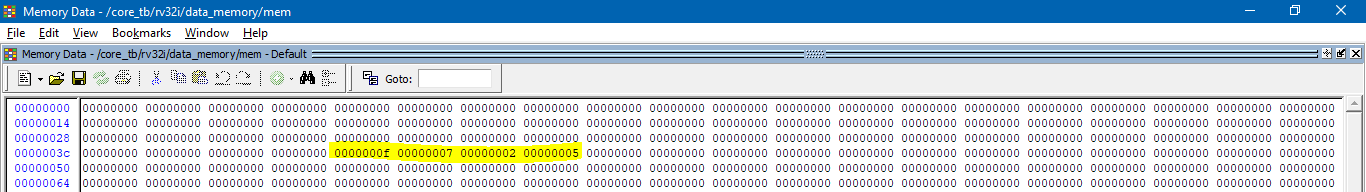
loop: jal zero, loop

**Results:**

* **Wave Output:**



* **Data Memory Output:**



**7. Nth term of a Geometric Progression**

**Venus Simulation Code**

addi sp, sp, 256

addi a0, x0, 5 # 1st term

addi a1, x0, 2 # common ratio

addi a2, x0, 5 # nth term to find

add ra, x0, x0

add t0, x0, x0

addi t1, x0, 1

# if n==1 ans=1st term

add ra, x0, a0

beq t1, a2,end

# else 1st term \* r ^ (n-1).

sub a2, a2, t1

add ra, x0, a1

start1:

addi t1, t1, 1 # so it starts from 2.

# assigning multiplication values.

add a3, x0, x0

add a3, x0, ra # setting the a3 inner loop variable to operand 1

add a4, x0, x0

add a4, x0, a1 # setting the a4 inner loop variable to operand 2

add t0, x0, x0 # setting inner counter back to zero

add ra, x0, x0 # initialize ra to 0

#Inner Loop and the loop variables are t0(for counter) a1 and a2 (the operands),

# this inner loop does the function a3 \* a4 and stores the result in ra.

start2: add ra,ra, a3

addi t0, t0, 1

bne t0, a4, start2

#Inner Loop ends

bne t1, a2, start1

# ra currently has the output of r ^ (n-1), now we multiply this with a0 to get the nth term.

add a1, x0, ra

add ra, x0, x0

add t0, x0, x0

start: add ra, ra, a1

addi t0, t0, 1

bne t0, a0, start

end:

sw ra, 0(sp)

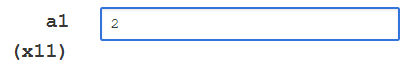
addi sp, sp, -256

loop: jal zero, loop

**Result:**

* **Input**







* **Output**



**Modelsim Code**

add s0, x0, sp

add sp, x0, x0

addi sp, sp, 256

lw a0, 4(sp) # 1st term

lw a1, 8(sp) # common ratio

lw a2, 12(sp) # nth term to find

add ra, x0, x0

add t0, x0, x0

addi t1, x0, 1

# if n==1 ans=1st term

add ra, x0, a0

beq t1, a2,end

# else 1st term \* r ^ (n-1).

sub a2, a2, t1

add ra, x0, a1

start1:

addi t1, t1, 1 # so it starts from 2.

# assigning multiplication values.

add a3, x0, x0

add a3, x0, ra # setting the a3 inner loop variable to operand 1

add a4, x0, x0

add a4, x0, a1 # setting the a4 inner loop variable to operand 2

add t0, x0, x0 # setting inner counter back to zero

add ra, x0, x0 # initialize ra to 0

#Inner Loop and the loop variables are t0(for counter) a1 and a2 (the operands),

# this inner loop does the function a3 \* a4 and stores the result in ra.

start2: add ra,ra, a3

addi t0, t0, 1

bne t0, a4, start2

#Inner Loop ends

bne t1, a2, start1

# ra currently has the output of r ^ (n-1), now we multiply this with a0 to get the nth term.

add a1, x0, ra

add ra, x0, x0

add t0, x0, x0

start: add ra, ra, a1

addi t0, t0, 1

bne t0, a0, start

end:

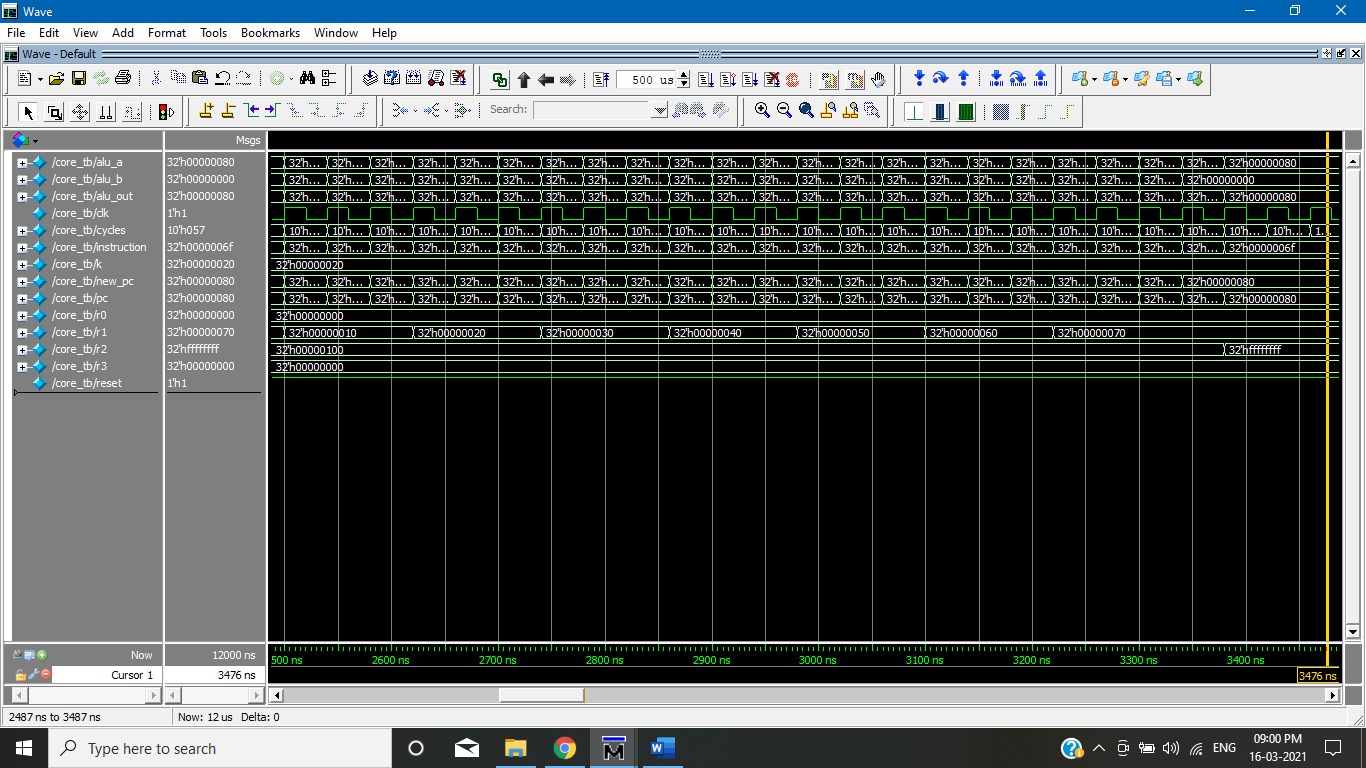
sw ra, 0(sp)

add sp, s0, x0

loop: jal zero, loop

**Results:**

* **Wave Output:**



* **Data Memory Output:**

