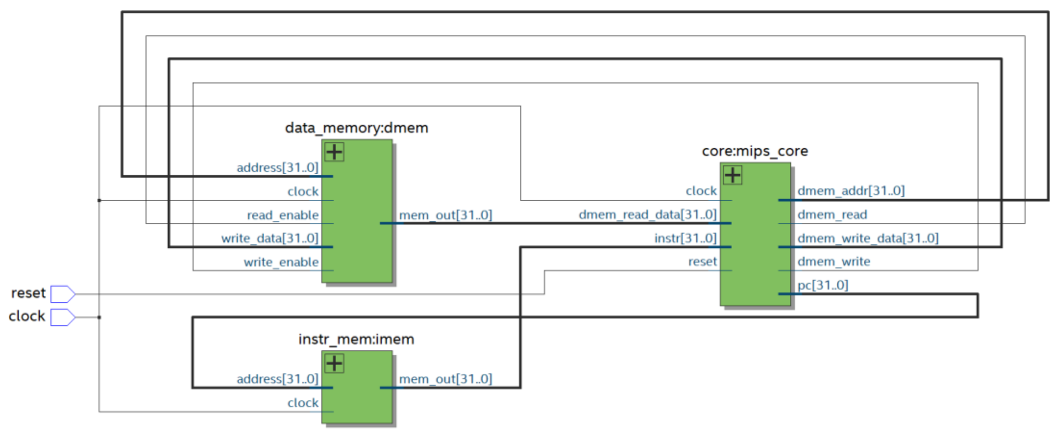
MIPS32

32-bit Single Cycle MIPS Processor

Emirkan Burak Yılmaz

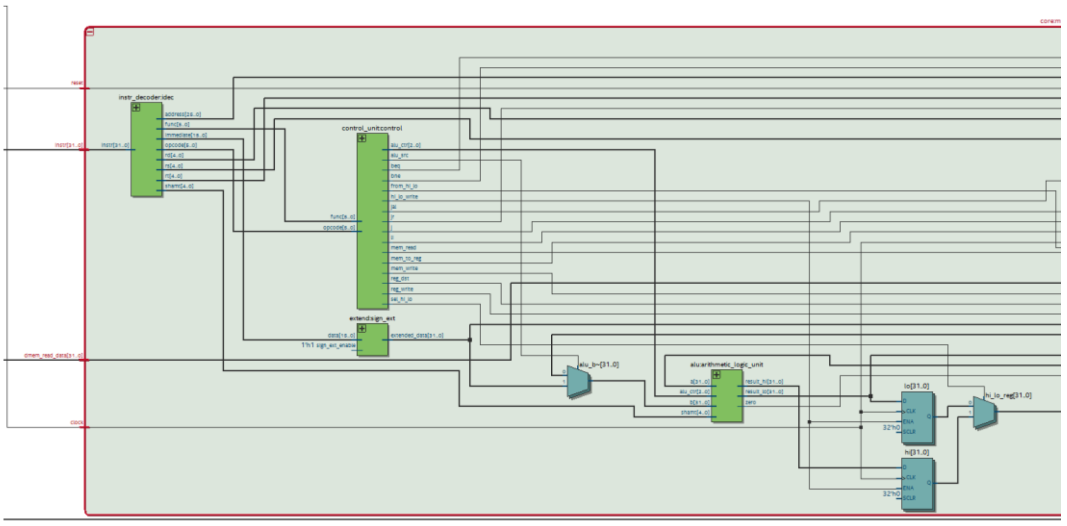
1. **mips32:** Consist of instruction memory, data memory and core.

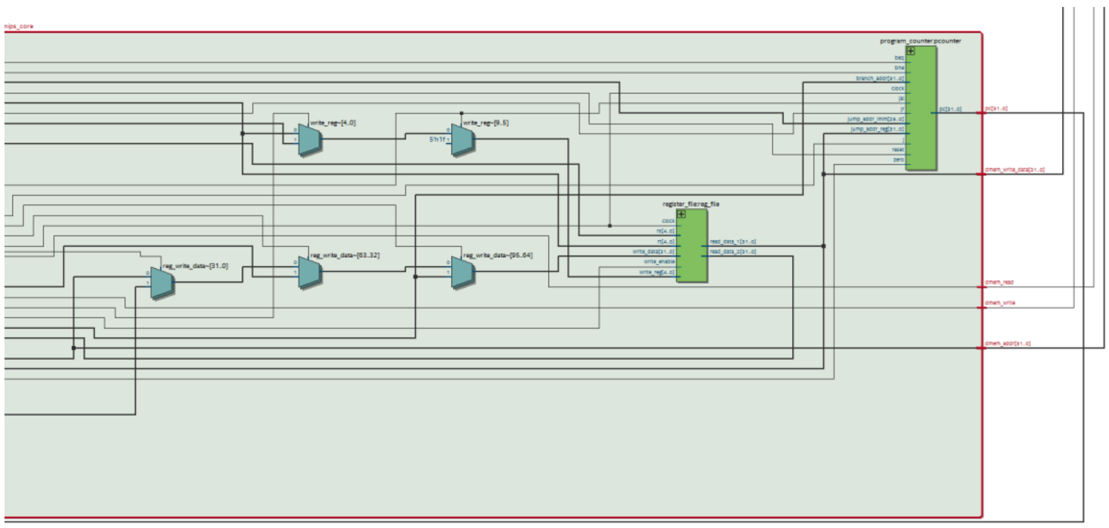


Supported instructions are given below.

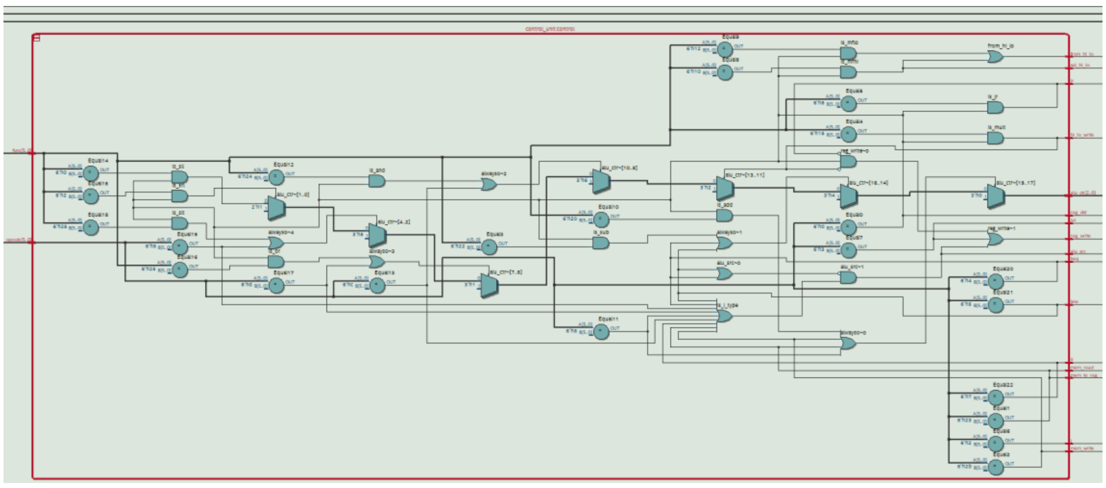
Instruction	**Instruction Type**
add	R
sub	R
mult	R
or	R
and	R
slt	R
jr	R
mfhi	R
mflo	R
addi	I
ori	I
andi	I
slti	I
lw	I
li	I
sw	I
beq	I
bne	I
j	J
jal	J

2. **core:** Contains register file, ALU and multiplexers for control signals.





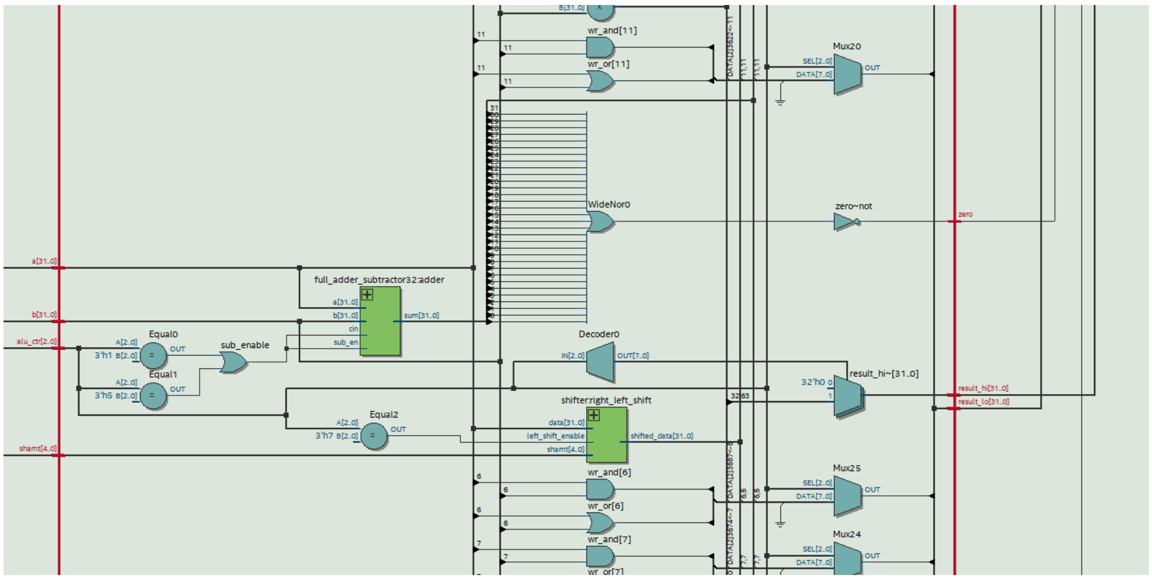
3. **control_unit:** Produces control signals according to opcode and function field of the instruction.



The following table explains the control signals with their meaning.

Signal	**Meaning**
reg_dst	Selects rt (0) or rd (1) register for write register address
beq	Selects extended branch address for PC (ALU zero should be 1)
bne	Selects extended branch address for PC (ALU zero should be 0)
j	Selects extended immediate address value for PC
jr	Selects register rs content for PC
jal	Selects extended immediate address for PC and $ra for destination register
li	Selects extended immediate value for destination register
hi_lo_write	Write the multiplication result into hi and lo registers
from_hi_lo	Selects the hi or lo register content for destination register
sel_hi_lo	Selects register hi (1) or lo (0) content for destination register
reg_write	Enables write for register file
alu_ctr	Selects ALU operation
alu_src	Select extended immediate value (1) or rt register content (0)
mem_read	Enables read for data memory
mem_write	Enables write for data memory
mem_to_reg	Select the data comes from memory for destination register

4. **alu:** Provides arithmetic and logical operations and desired operation is selected by the control signal alu_ctr.



alu_ctr	Operation
000	ADD
001	SUB
010	MULT
011	AND
100	OR
101	SLT
110	SRL
111	SLL

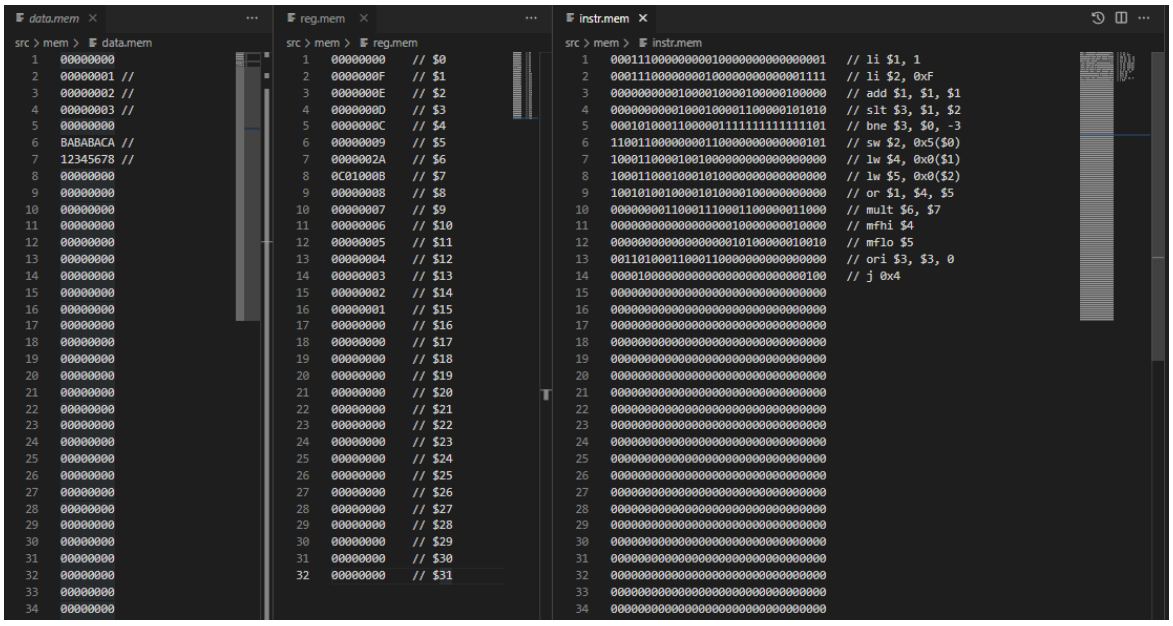
instruction	ALU Operation
add, addi	ADD
sub	SUB
mult	MULT
and	AND
or	OR
sll	SLL
srl	SRL
mflo	X
mfhi	X
li	X
lw	ADD
sw	ADD
beq, bne	SUB
slt, slti	SUB
j, jr, jal	X

5. Testbench and Simulation

The value of the parameters displayed in hex radix. The below table explains the parameters that is used on the simulation.

Parameter	Explanation
_reg_write_data	The data written to destination register. It could be ALU result, extended immediate or data that read from memory
_alu_a	First input of ALU (register rs)
_alu_b	Second input of ALU (register rt or extended immediate)
_instr	Instruction
_pc	Program Counter

The content of memory initialization files (data.mem, instr.mem, reg.mem) are shown here. To simulate the testbench, make sure the correct path for initialization file is provided for the instr_memory, data_memory and register_file modules.

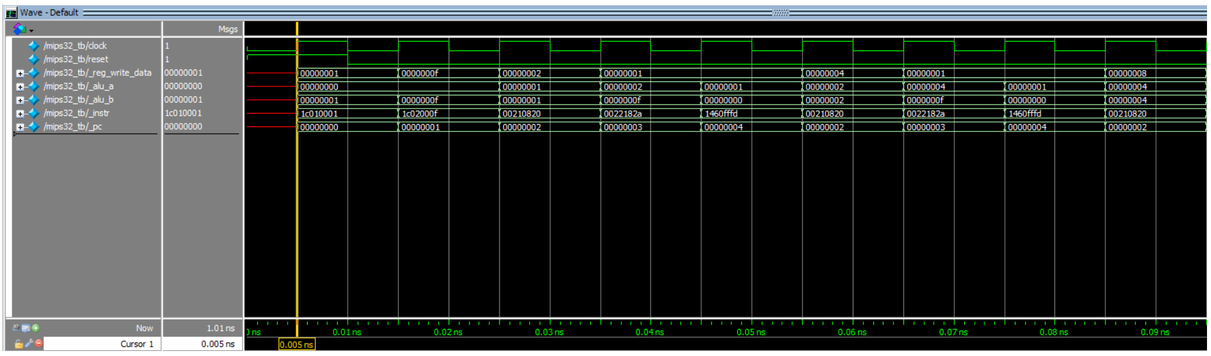


```
00000000
00000001 //
00000002 //
00000003 //
00000000
BABABACA //
12345678 //
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
```

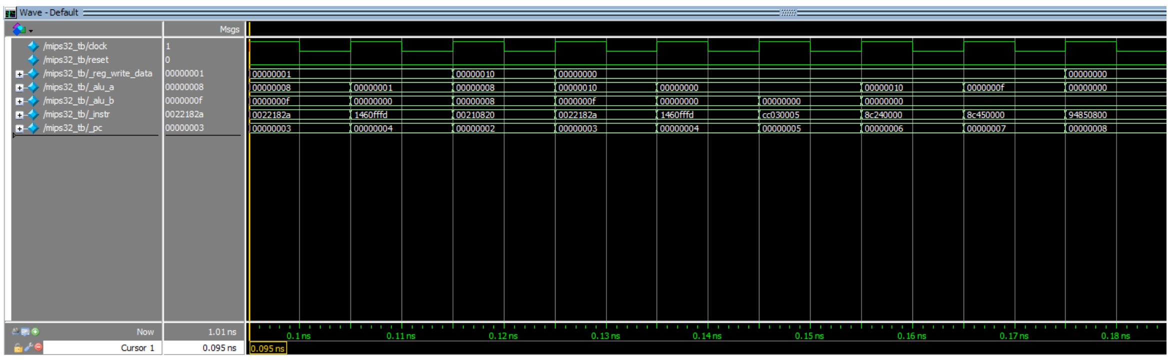
```
00000000    // $0
0000000F    // $1
0000000E    // $2
0000000D    // $3
0000000C    // $4
00000009    // $5
0000002A    // $6
0C01000B    // $7
00000008    // $8
00000007    // $9
00000006    // $10
00000005    // $11
00000004    // $12
00000003    // $13
00000002    // $14
00000001    // $15
00000000    // $16
00000000    // $17
00000000    // $18
00000000    // $19
00000000    // $20
00000000    // $21
00000000    // $22
00000000    // $23
00000000    // $24
00000000    // $25
00000000    // $26
00000000    // $27
00000000    // $28
00000000    // $29
00000000    // $30
00000000    // $31
```

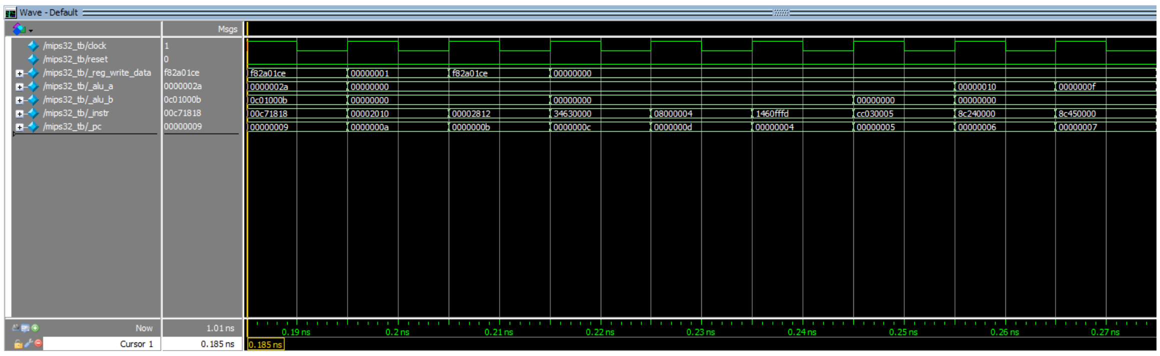
```
00011100000000010000000000000001   // li $1, 1
00011100000000100000000000001111   // li $2, 0xF
00000000001000010000100000100000   // add $1, $1, $1
00000000001000100001100000101010   // slt $3, $1, $2
00010100011000001111111111111101   // bne $3, $0, -3
11001100000000100000000000000101   // sw $2, 0x5($0)
10001100001001000000000000000000   // lw $4, 0x0($1)
10001100010001010000000000000000   // lw $5, 0x0($2)
10010100100001010000100000000000   // or $1, $4, $5
00000000110001110001100000011000   // mult $6, $7
00000000000000000010000000010000   // mfhi $4
00000000000000000010100000010010   // mflo $5
00110100011000110000000000000000   // ori $3, $3, 0
00001000000000000000000000000100   // j 0x4
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
```

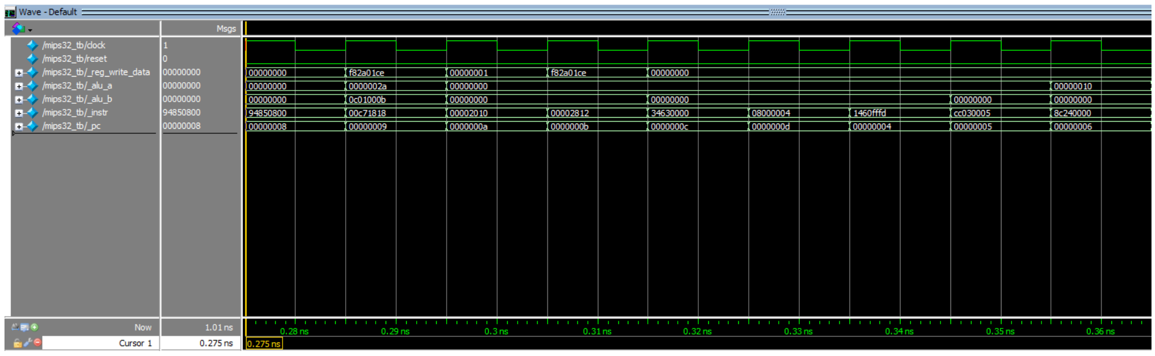
Test case consist of loops and first one is happened between PC of 2 and 4. After that loop is finished, the jump instruction goes address 4 and whole process started again. ModelSim simulation results for the testbench file mips32_tb are provided below.

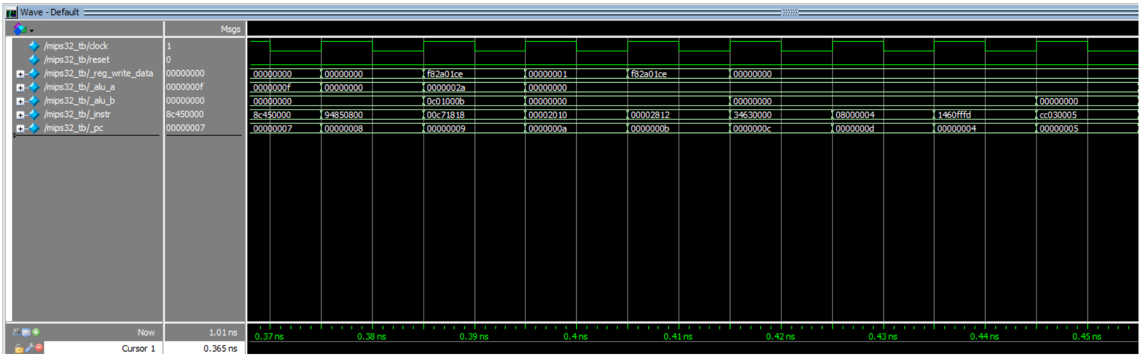


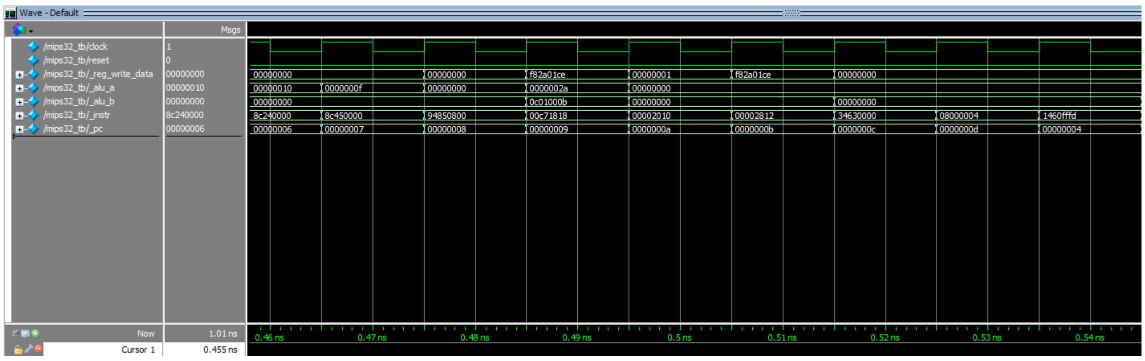
```
/mips32_tb/clock            1
/mips32_tb/reset            1
/mips32_tb/_reg_write_data  00000001  00000001 0000000f 00000002 00000001 00000004 00000001 00000008
/mips32_tb/_alu_a           00000000  00000000 00000001 00000002 00000001 00000002 00000004 00000001 00000004
/mips32_tb/_alu_b           00000001  00000001 0000000f 00000001 0000000f 00000000 00000002 0000000f 00000000 00000004
/mips32_tb/_instr           1c010001  1c010001 1c02000f 00210820 0022182a 1460fffd 00210820 0022182a 1460fffd 00210820
/mips32_tb/_pc              00000000  00000000 00000001 00000002 00000003 00000004 00000002 00000003 00000004 00000002
```


```
Wave - Default
                               Msgs
/mips32_tb/clock               1
/mips32_tb/reset               0
/mips32_tb/_reg_write_data     00000001   00000001 | 00000010 | 00000000 | 00000000
/mips32_tb/_alu_a              00000008   00000008 | 00000001 | 00000008 | 00000010 | 00000000 | 00000000 | 00000010 | 0000000f | 00000000
/mips32_tb/_alu_b              0000000f   0000000f | 00000000 | 00000008 | 0000000f | 00000000 | 00000000 | 00000000
/mips32_tb/_instr              0022182a   0022182a | 1460fffd | 00210820 | 0022182a | 1460fffd | cc030005 | 8c240000 | 8c450000 | 94850800
/mips32_tb/_pc                 00000003   00000003 | 00000004 | 00000002 | 00000003 | 00000004 | 00000005 | 00000006 | 00000007 | 00000008

Now       1.01 ns    0.1 ns  0.11 ns  0.12 ns  0.13 ns  0.14 ns  0.15 ns  0.16 ns  0.17 ns  0.18 ns
Cursor 1  0.095 ns   0.095 ns
```


```
Wave - Default
                               Msgs
/mips32_tb/clock               1
/mips32_tb/reset               0
/mips32_tb/_reg_write_data     f82a01ce   f82a01ce | 00000001 | f82a01ce | 00000000
/mips32_tb/_alu_a              0000002a   0000002a | 00000000 | 00000010 | 0000000f
/mips32_tb/_alu_b              0c01000b   0c01000b | 00000000 | 00000000 | 00000000 | 00000000
/mips32_tb/_instr              00c71818   00c71818 | 00002010 | 00002812 | 34630000 | 08000004 | 1460fffd | cc030005 | 8c240000 | 8c450000
/mips32_tb/_pc                 00000009   00000009 | 0000000a | 0000000b | 0000000c | 0000000d | 00000004 | 00000005 | 00000006 | 00000007

Now       1.01 ns    0.19 ns  0.2 ns  0.21 ns  0.22 ns  0.23 ns  0.24 ns  0.25 ns  0.26 ns  0.27 ns
Cursor 1  0.185 ns   0.185 ns
```


```
Wave - Default
                               Msgs
/mips32_tb/clock               1
/mips32_tb/reset               0
/mips32_tb/_reg_write_data     00000000   00000000 | f82a01ce | 00000001 | f82a01ce | 00000000
/mips32_tb/_alu_a              00000000   00000000 | 0000002a | 00000000 | 00000010
/mips32_tb/_alu_b              00000000   00000000 | 0c01000b | 00000000 | 00000000 | 00000000 | 00000000
/mips32_tb/_instr              94850800   94850800 | 00c71818 | 00002010 | 00002812 | 34630000 | 08000004 | 1460fffd | cc030005 | 8c240000
/mips32_tb/_pc                 00000008   00000008 | 00000009 | 0000000a | 0000000b | 0000000c | 0000000d | 00000004 | 00000005 | 00000006

Now       1.01 ns    0.28 ns  0.29 ns  0.3 ns  0.31 ns  0.32 ns  0.33 ns  0.34 ns  0.35 ns  0.36 ns
Cursor 1  0.275 ns   0.275 ns
```


```
Wave - Default
                               Msgs
/mips32_tb/clock               1
/mips32_tb/reset               0
/mips32_tb/_reg_write_data     00000000   00000000 | 00000000 | f82a01ce | 00000001 | f82a01ce | 00000000
/mips32_tb/_alu_a              0000000f   0000000f | 00000000 | 0000002a | 00000000
/mips32_tb/_alu_b              00000000   00000000 | 0c01000b | 00000000 | 00000000 | 00000000
/mips32_tb/_instr              8c450000   8c450000 | 94850800 | 00c71818 | 00002010 | 00002812 | 34630000 | 08000004 | 1460fffd | cc030005
/mips32_tb/_pc                 00000007   00000007 | 00000008 | 00000009 | 0000000a | 0000000b | 0000000c | 0000000d | 00000004 | 00000005

Now       1.01 ns    0.37 ns  0.38 ns  0.39 ns  0.4 ns  0.41 ns  0.42 ns  0.43 ns  0.44 ns  0.45 ns
Cursor 1  0.365 ns
```


```
Wave - Default
                               Msgs
/mips32_tb/clock               1
/mips32_tb/reset               0
/mips32_tb/_reg_write_data     00000000   00000000 | 00000000 | f82a01ce | 00000001 | f82a01ce | 00000000
/mips32_tb/_alu_a              00000010   00000010 | 0000000f | 00000000 | 0000002a | 00000000
/mips32_tb/_alu_b              00000000   00000000 | 0c01000b | 00000000 | 00000000
/mips32_tb/_instr              8c240000   8c240000 | 8c450000 | 94850800 | 00c71818 | 00002010 | 00002812 | 34630000 | 08000004 | 1460fffd
/mips32_tb/_pc                 00000006   00000006 | 00000007 | 00000008 | 00000009 | 0000000a | 0000000b | 0000000c | 0000000d | 00000004

Now       1.01 ns    0.46 ns  0.47 ns  0.48 ns  0.49 ns  0.5 ns  0.51 ns  0.52 ns  0.53 ns  0.54 ns
Cursor 1  0.455 ns
```