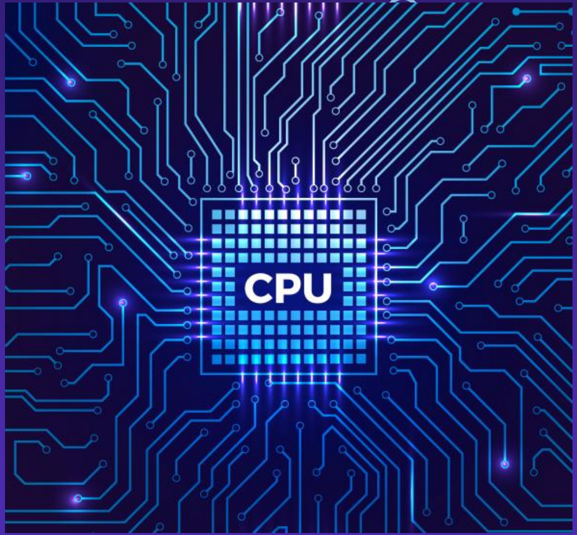
Single cycled mips processor



Agenda

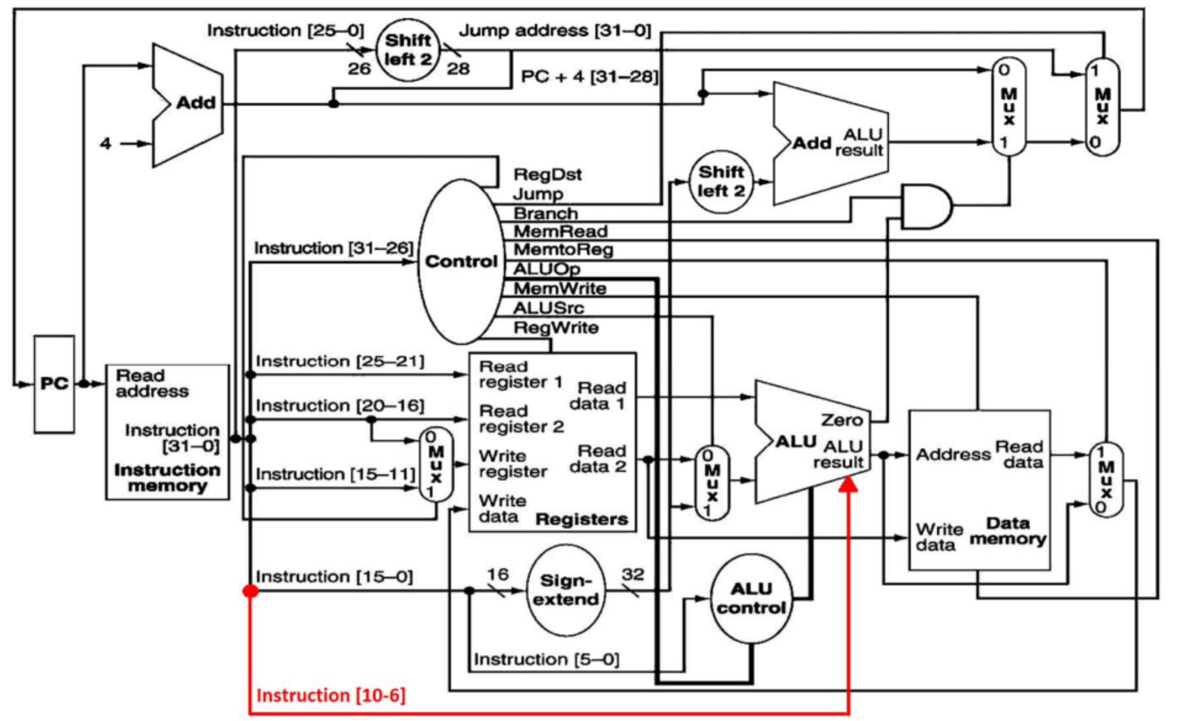
What is MIPS?

MIPS is a RISC (reduced instruction set computing) instruction set architecture developed by several Stanford researchers in the mid 1980s. Originally, the name was an acronym for Microprocessor without Interlocked Pipeline Stages, but interlocks between pipeline stages were eventually reintroduced, probably for performance reasons as other processors became more advanced. The decision for making a processor without interlocking pipeline stages was based on performance and simplicity of design. With interlocks, operations such as integer division, which is very time-consuming, would cause other pipeline phases to wait until the execute unit was done with the division. This defeats the purpose of pipelining because it causes sections of the processor to idle. Reducing all phases to one clock cycle removes idling (however it might force the clock to be slower).

MIPS follows the classic 5-stage RISC pipeline: Instruction Fetch (IF), Instruction Decode/Operand Fetch (ID), Execute (EX), Memory Access (MEM), and Write Back (WB). MIPS is a load-store architecture, which means that to do arithmetic on data, values must explicitly be read from memory with a special load instruction and written to memory with a store instruction; arithmetic instructions only operate on registers.

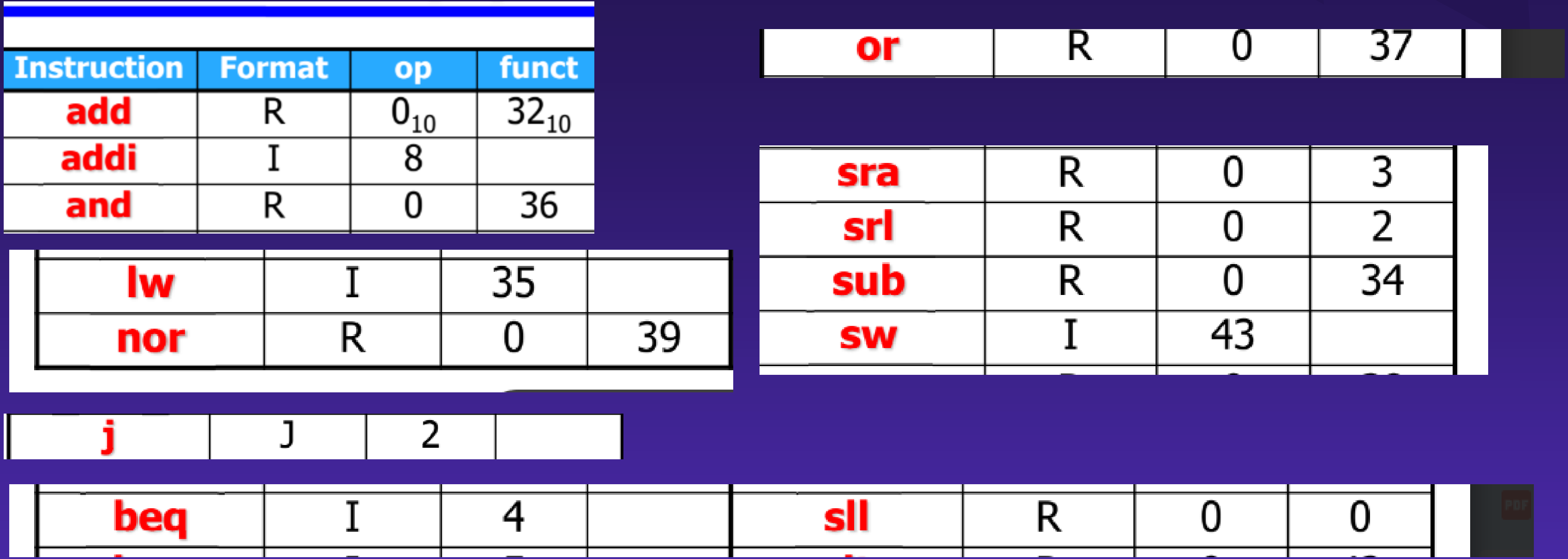
In my opinion, MIPS is a really great architecture if you’re interested in learning about ISAs, computer architecture, and computer organization because it is straightforward and simple.

Architecture



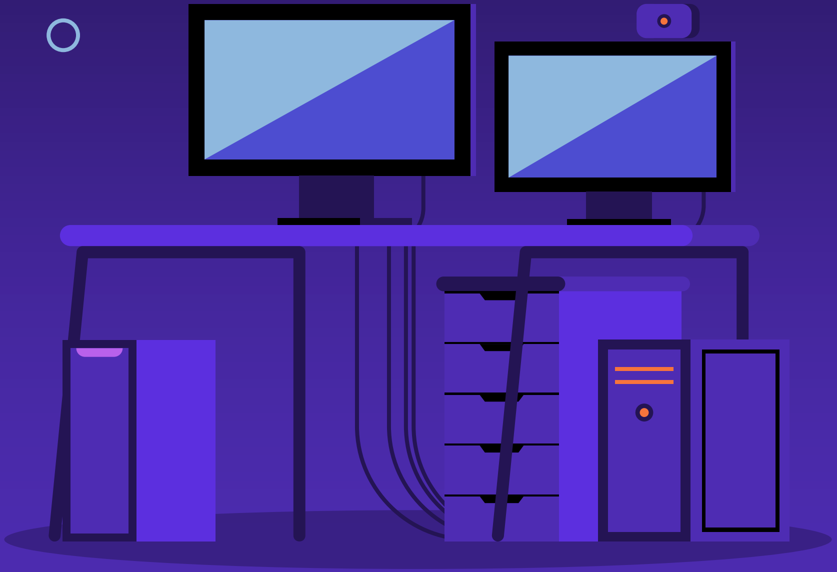
Instruction [25–0]
Shift left 2
Jump address [31–0]
26
28
PC + 4 [31–28]
Add
4
Add
ALU result
0 Mux 1
1 Mux 0
Shift left 2
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
0 Mux 1
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
0 Mux 1
Zero
ALU
ALU result
Address
Read data
Write data
Data memory
1 Mux 0
Instruction [15–0]
16
Sign-extend
32
ALU control
Instruction [5–0]
Instruction [10-6]

This MIPS support the following instructions :



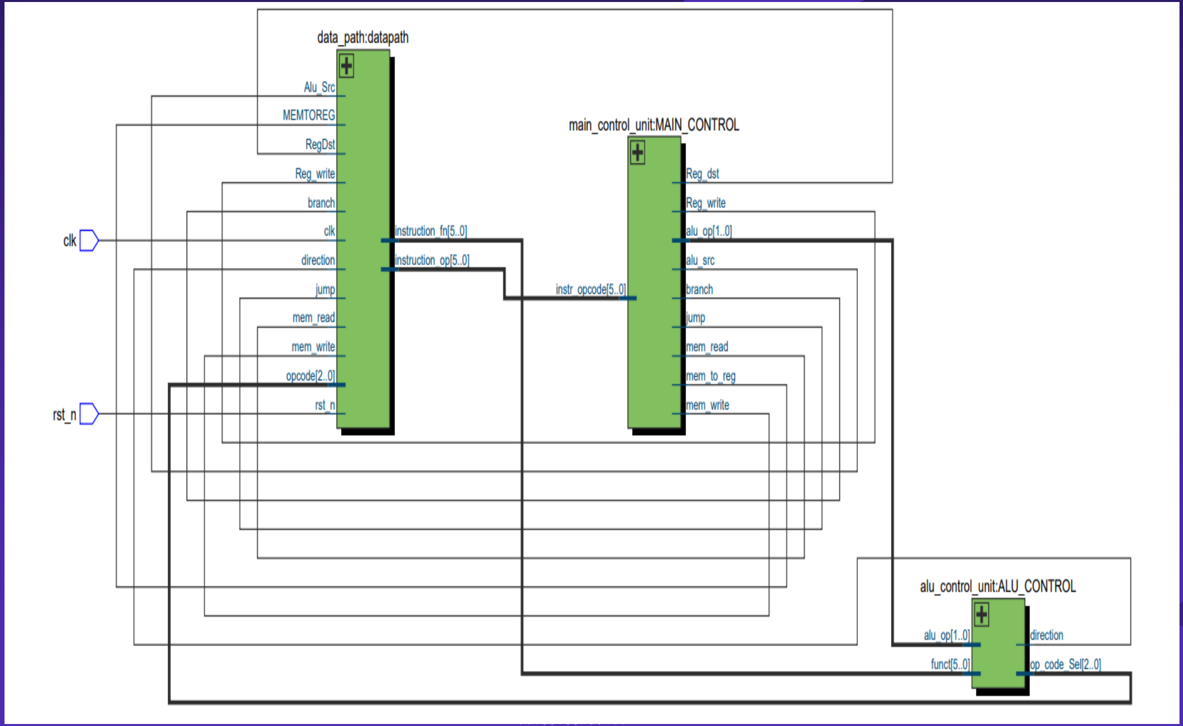
Instruction	Format	op	funct
add	R	0_{10}	32_{10}
addi	I	8	
and	R	0	36
lw	I	35	
nor	R	0	39
j	J	2	
beq	I	4	
or	R	0	37
sra	R	0	3
srl	R	0	2
sub	R	0	34
sw	I	43	
sll	R	0	0

It also supports multiplication through the alu



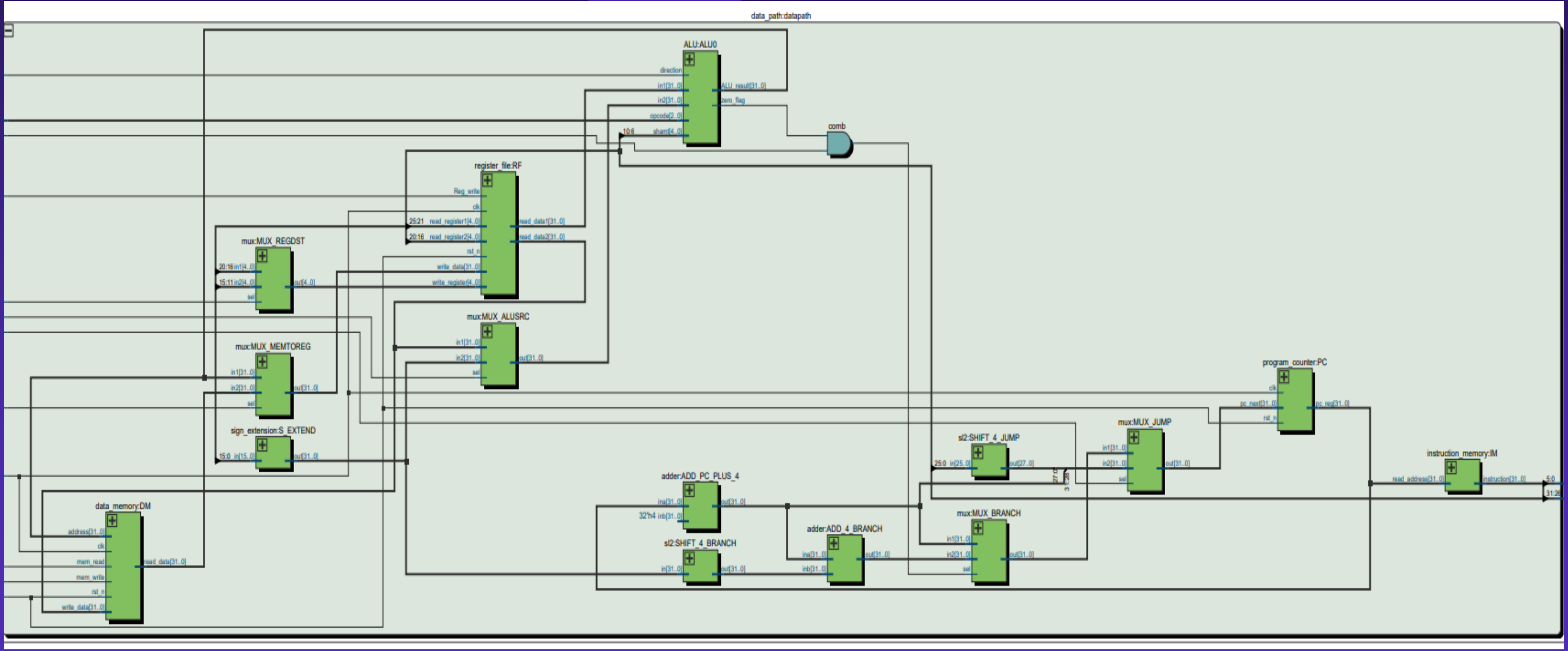
synthesis

Top module

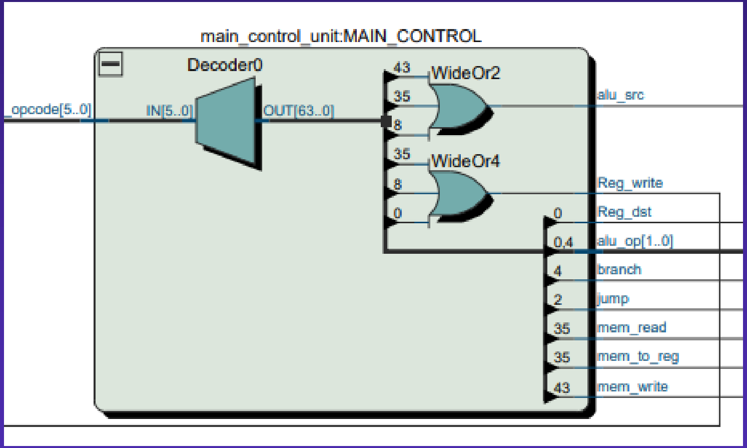


data_path:datapath
Alu_Src
MEMTOREG
RegDst
Reg_write
branch
clk
direction
jump
mem_read
mem_write
opcode[2..0]
rst_n
instruction_fn[5..0]
instruction_op[5..0]
main_control_unit:MAIN_CONTROL
instr_opcode[5..0]
Reg_dst
Reg_write
alu_op[1..0]
alu_src
branch
jump
mem_read
mem_to_reg
mem_write
clk
rst_n
alu_control_unit:ALU_CONTROL
alu_op[1..0]
funct[5..0]
direction
op_code_Sel[2..0]

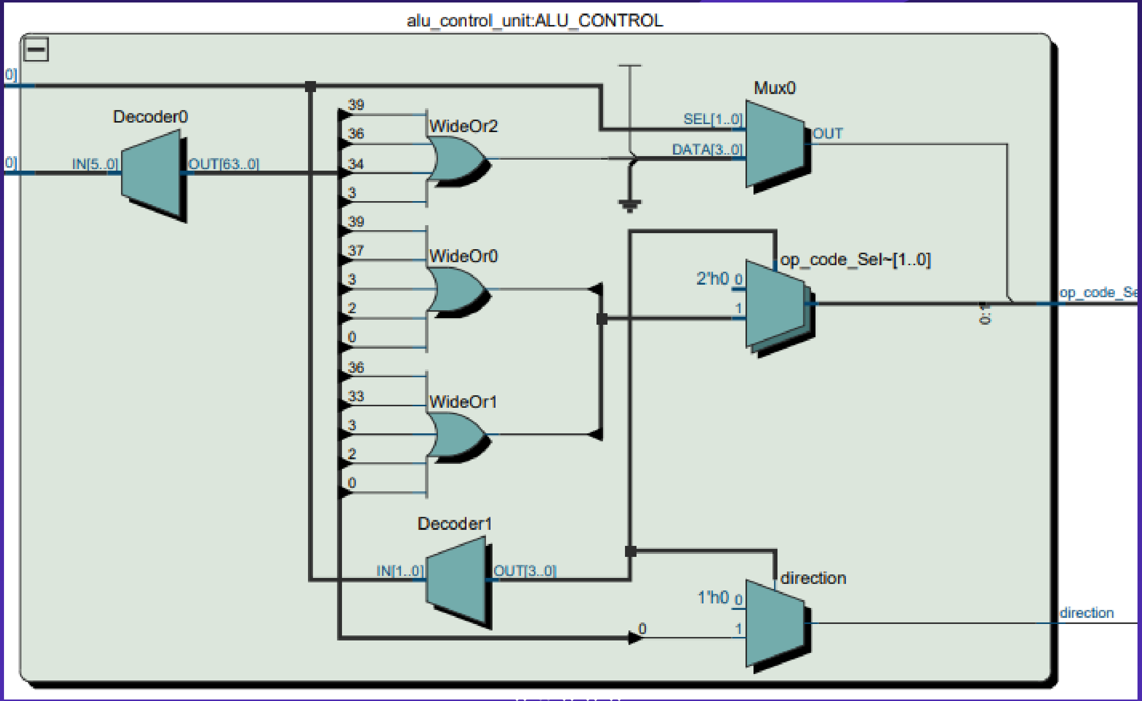
Data path

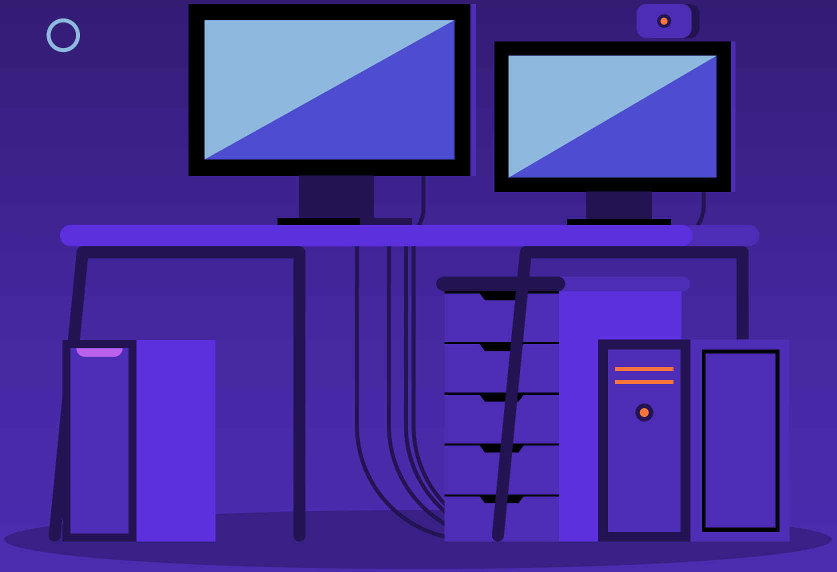

data_path:datapath
ALU:ALU0
direction
in1[31..0]
in2[31..0]
opcode[2..0]
shamt[4..0]
ALU_result[31..0]
zero_flag
comb
10:6
register_file:RF
Reg_write
clk
read_register1[4..0]
read_register2[4..0]
rst_n
write_data[31..0]
write_register[4..0]
read_data1[31..0]
read_data2[31..0]
25:21
20:16
mux:MUX_REGDST
20:16 in1[4..0]
15:11 in2[4..0]
sel
out[4..0]
mux:MUX_ALUSRC
mux:MUX_MEMTOREG
sign_extension:S_EXTEND
15:0 in[15..0]
out[31..0]
data_memory:DM
address[31..0]
clk
mem_read
mem_write
rst_n
write_data[31..0]
read_data[31..0]
adder:ADD_PC_PLUS_4
ina[31..0]
32'h4 inb[31..0]
sl2:SHIFT_4_BRANCH
adder:ADD_4_BRANCH
sl2:SHIFT_4_JUMP
25:0 in[25..0]
out[27..0]
mux:MUX_BRANCH
mux:MUX_JUMP
program_counter:PC
pc_next[31..0]
pc_reg[31..0]
instruction_memory:IM
read_address[31..0]
instruction[31..0]
5:0
31:26

Main control unit



Alu control unit



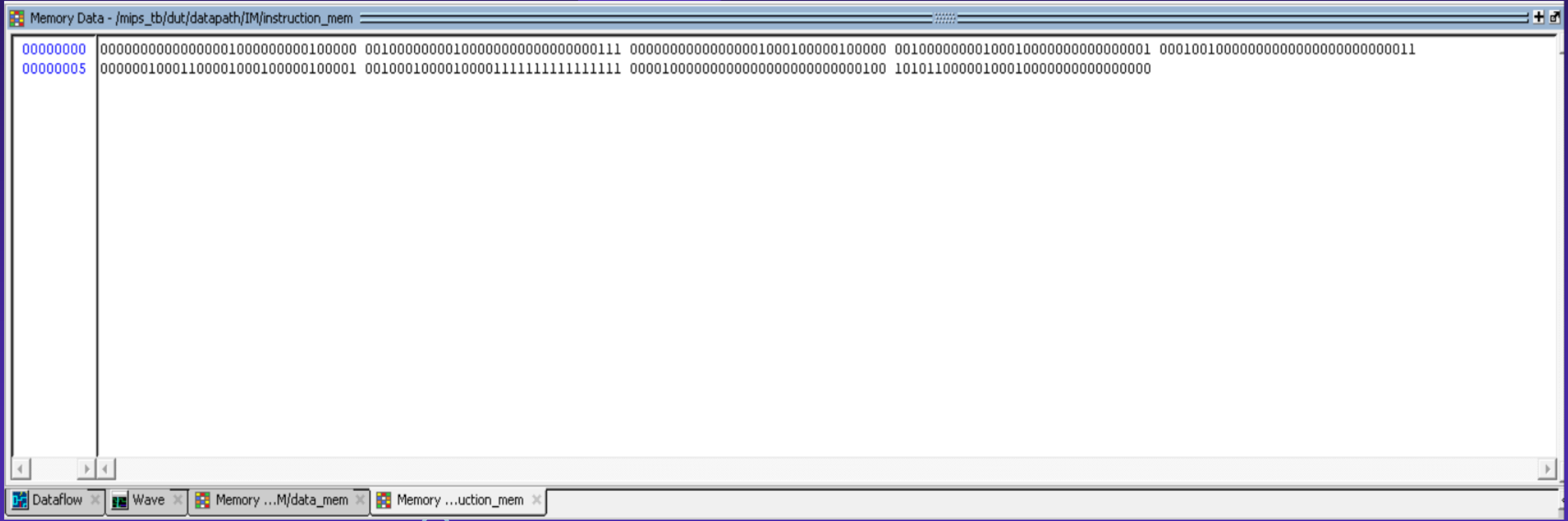


Functional verification

```
add $s0, $0, $0
addi $s0, $0, 7
add $s1, $0, $0
addi $s1, $0, 1
beq $s1, $0, 3
mul $s1, $s1, $s0
addi $s0, $s0, -1
J  4
sw $s1, 0($0)
```

This program 1 that I used to test the MIPS and it calculates the factorial of 7

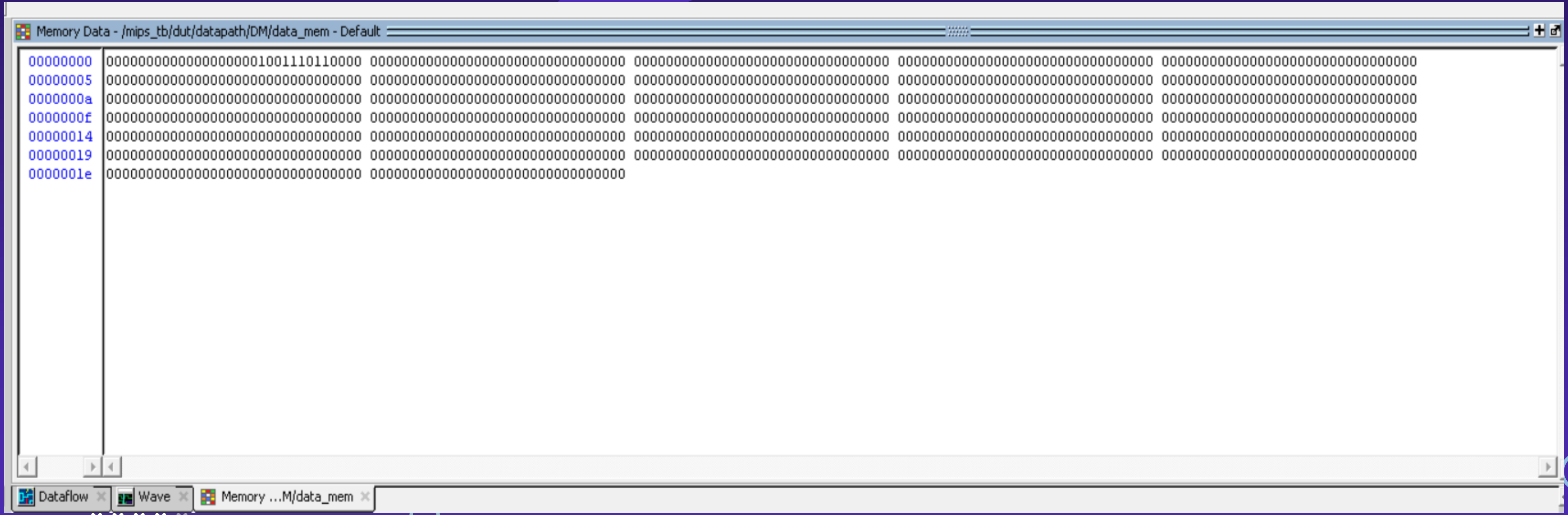
Instruction memory



Memory Data - /mips_tb/dut/datapath/IM/instruction_mem

```
00000000 00000000000000001000000000100000 00100000000100000000000000000111 00000000000000001000100000100000 00100000000100010000000000000001 00010010000000000000000000000011
00000005 00000010001100001000100000100001 00100010000100001111111111111111 00001000000000000000000000000100 10101100000100010000000000000000
```

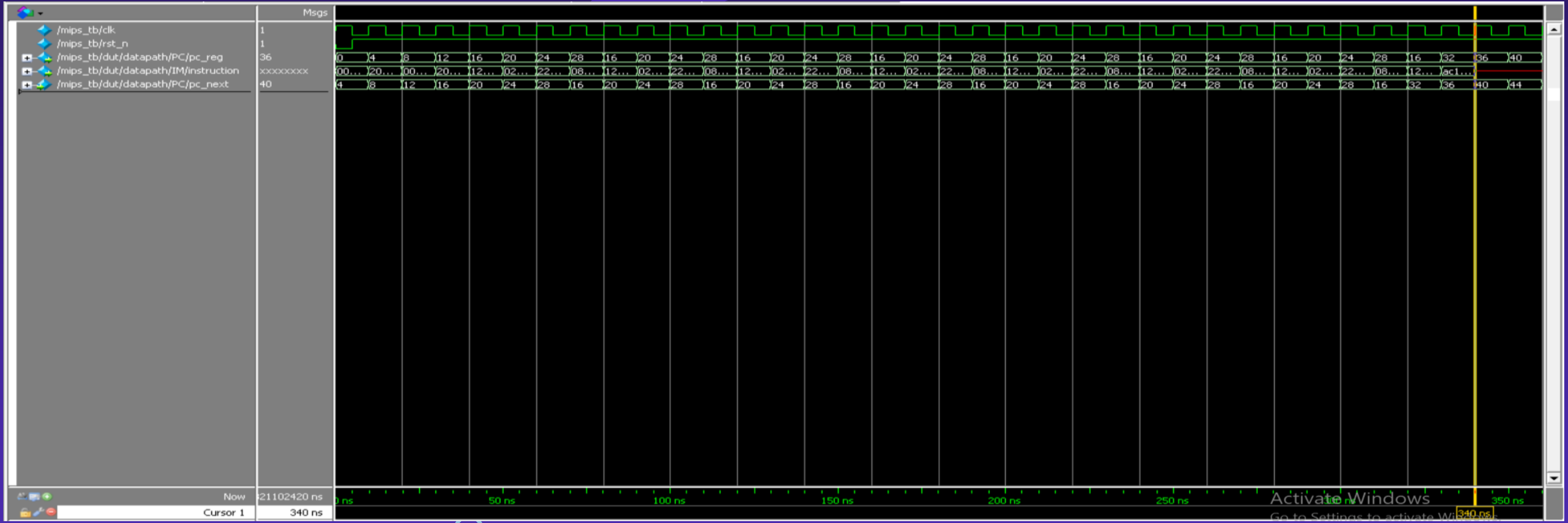
data memory



Memory Data - /mips_tb/dut/datapath/DM/data_mem - Default

```
00000000 00000000000000000001001110110000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000005 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000000a 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000000f 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000014 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000019 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000001e 00000000000000000000000000000000 00000000000000000000000000000000
```

Snippet that shows how many cycles the program takes



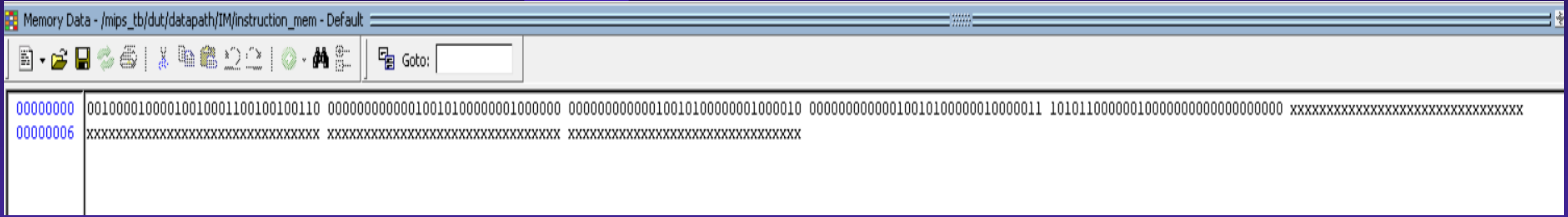
```
                                         Msgs
/mips_tb/clk                             1
/mips_tb/rst_n                           1
/mips_tb/dut/datapath/PC/pc_reg          36        0 4 8 12 16 20 24 28 16 20 24 28 16 20 24 28 16 20 24 28 16 20 24 28 16 20 24 28 16 20 24 28 16 32 36 40
/mips_tb/dut/datapath/IM/instruction     xxxxxxxx  00... 20... 00... 20... 12... 02... 22... 08... 12... 02... 22... 08... 12... 02... 22... 08... 12... 02... 22... 08... 12... 02... 22... 08... 12... 02... 22... 08... 12... 02... 22... 08... 12... ac1...
/mips_tb/dut/datapath/PC/pc_next         40        4 8 12 16 20 24 28 16 20 24 28 16 20 24 28 16 20 24 28 16 20 24 28 16 20 24 28 16 20 24 28 16 32 36 40 44

Now        21102420 ns    0 ns   50 ns   100 ns   150 ns   200 ns   250 ns   300 ns   350 ns
Cursor 1   340 ns                                                               340 ns
```

```
ADDI $t1 $t0 0x1926
SLL $t0 $t1 1
SRL $t0 $t1 1
SRA $t0 $t1 2
SW $t0 0($zero)
```

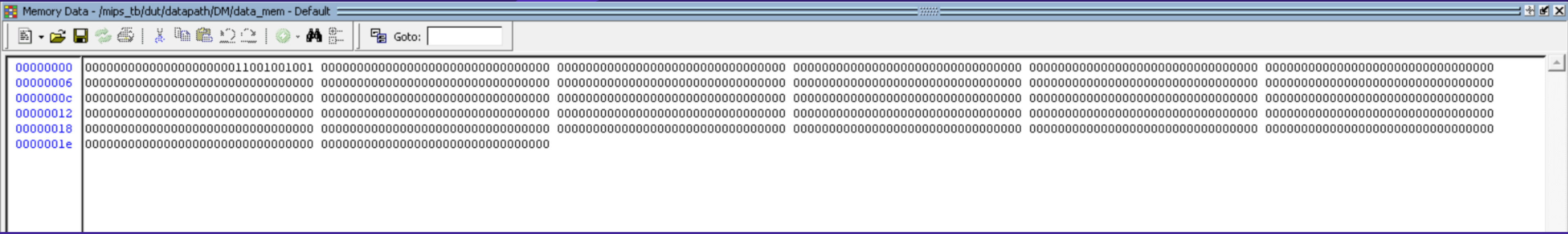
This is program 2 that I used to test the MIPS and it tests the 3 shift instructions and sw

Instruction memory


Memory Data - /mips_tb/dut/datapath/IM/instruction_mem - Default
Goto:

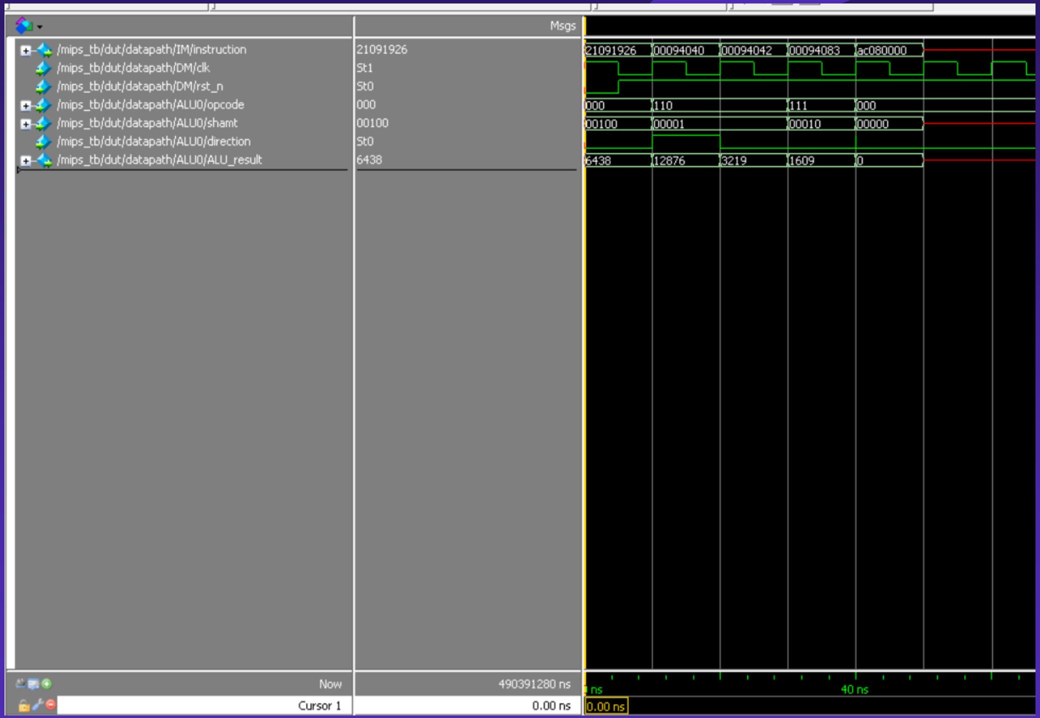
```
00000000 00100001000010010001100100100110 00000000000010010100000001000000 00000000000010010100000001000010 00000000000010010100000001000011 10101100000001000000000000000000 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
00000006 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

data memory


Memory Data - /mips_tb/dut/datapath/DM/data_mem - Default

```
00000000 00000000000000000000011001001001 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000006 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000000c 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000012 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000018 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000001e 00000000000000000000000000000000 00000000000000000000000000000000
```

Snippet that shows how many cycles the program takes and the output of the Alu after every instruction



```
                                              Msgs
/mips_tb/dut/datapath/IM/instruction          21091926   21091926 00094040 00094042 00094083 ac080000
/mips_tb/dut/datapath/DM/clk                  St1
/mips_tb/dut/datapath/DM/rst_n                St0
/mips_tb/dut/datapath/ALU0/opcode             000        000      110               111      000
/mips_tb/dut/datapath/ALU0/shamt              00100      00100    00001             00010    00000
/mips_tb/dut/datapath/ALU0/direction          St0
/mips_tb/dut/datapath/ALU0/ALU_result         6438       6438     12876    3219     1609     0

Now        490391280 ns    ns                       40 ns
Cursor 1   0.00 ns         0.00 ns
```

Hint: direction signal is high in SLL and low in SRL