CONTROL													
INSTRUCTION	OPCODE (op)		REG_DEST	BEQ	MEM_READ	MEM_TO_REG	ALU_OP_2	ALU_OP_1	ALU_OP_0	MEM_WRITE	ALU_SRC	REG_WRITE	BNE
R_TYPE	0000		1	0	0	0	0	0	0	0	0	1	0
ADDI	0001		0	0	0	0	0	0	1	0	1	1	0
ANDI	0010		0	0	0	0	1	1	0	0	1	1	0
ORI	0011		0	0	0	0	0	1	1	0	1	1	0
NORI	0100		0	0	0	0	1	0	0	0	1	1	0
BEQ	0101		0	1	0	0	0	1	0	0	0	0	0
BNE	0110		0	0	0	0	0	1	0	0	0	0	1
SLTI	0111		0	0	0	0	1	1	1	0	1	1	0
LW	1000		0	0	1	1	0	0	1	0	1	1	0
SW	1001		0	0	0	0	0	0	1	1	1	0	0

LOGICAL EXPRESSIONS FOR CONTROL:	
REG_DEST	(op3+op2+op1+op0)'
BEQ	(op2 op1' op0)
BNE	(op2 op1 op0')
MEM_READ	op3 op2' op1' op0'
MEM_TO_REG	op3 op2' op1' op0'
ALU_OP_2	(op3' op2' op1 op0')+(op3' op2 op1' op0')+(op3' op2 op1 op0)
ALU_OP_1	(op3' op2' op1 op0')+(op3' op2' op1 op0)+(op3' op2 op1' op0)+(op3' op2 op1 op0')+(op3' op2 op1 op0)
ALU_OP_0	(op3' op2' op1' op0)+(op3' op2' op1 op0)+(op3' op2 op1 op0)+(op3 op2' op1' op0')+(op3 op2' op1' op0)
MEM_WRITE	op3 op2' op1' op0
ALU_SRC	((op3+op2+op1+op0)'+(op2 op1' op0)+(op2 op1 op0'))'
REG_WRITE	((op3' op2 op1' op0)+(op3' op2 op1 op0')+(op3 op2' op1' op0))'

ALU CONTROL					
INSTRUCTION	OPCODE	FUNC(f)	ALUOP(a)	ACTION	ALU_CONTROL
AND	0000	000	000	AND	110
ADD	0000	001	000	ADD	000
SUB	0000	010	000	SUB	010
XOR	0000	011	000	XOR	001
NOR	0000	100	000	NOR	101
OR	0000	101	000	OR	111
ADDI	0001	XXX	001	ADD	000
ANDI	0010	XXX	110	AND	110
ORI	0011	XXX	011	OR	111
NORI	0100	XXX	100	NOR	101
BEQ	0101	XXX	010	SUB	010
BNE	0110	XXX	010	SUB	010
SLTI	0111	XXX	111	SLT	100
LW	1000	XXX	001	ADD	000
SW	1001	XXX	001	ADD	000

LOGICAL EXPRESSIONS FOR ALU CONTROL:	
ALU CTRL 2	((a2' a1' a0') ((f2' f1' f0') + (f2 f1' f0') + (f2 f1' f0))) + (a2 a1 a0') + (a2' a1 a0)+ (a2 a1' a0') + (a2 a1 a0)
ALU CTRL 1	((a2' a1' a0') ((f2' f1' f0') + (f2' f1 f0') + (f2 f1' f0))) + (a2 a1 a0') + (a2' a1 a0) + (a2' a1 a0')
ALU CTRL 0	((a2' a1' a0') ((f2' f1 f0) + (f2 f1' f0') + (f2 f1' f0))) + (a2' a1 a0) + (a2 a1' a0')

More detail about **Control** and **ALU Control** could be found in **minimips_design.ods** file.

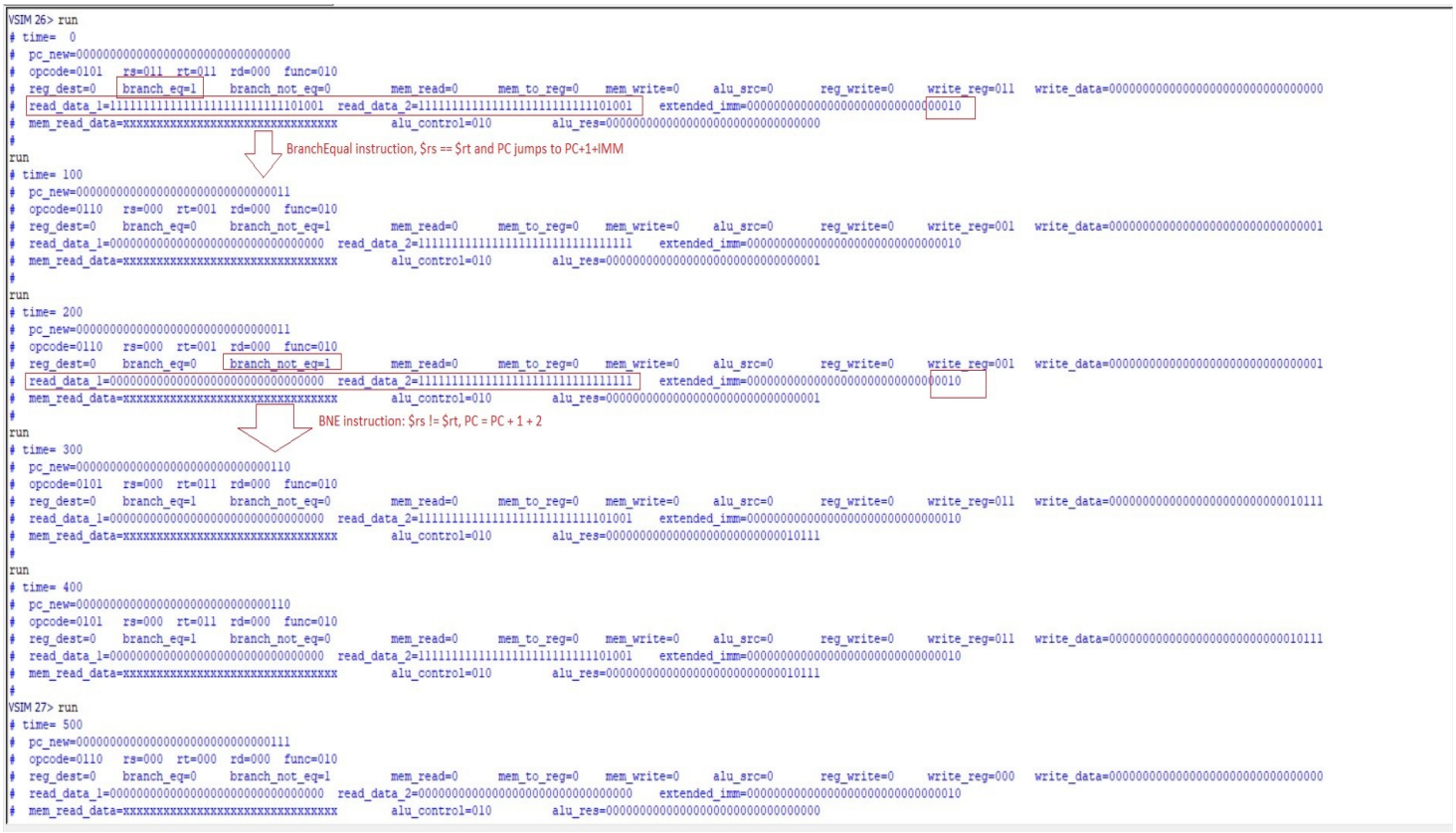
General Structure

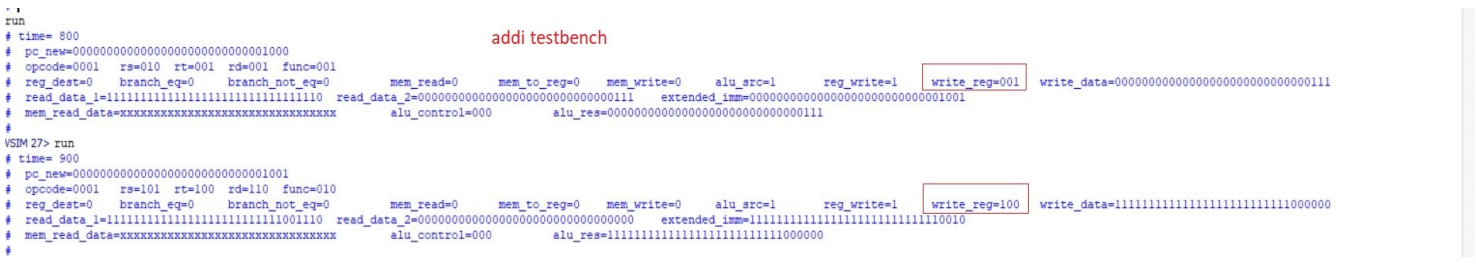
MiniMips processor is designed according to the single path datapath that is given in the book. All desing is identical except hw requirements.

Test Cases

Pc is updated at posedge and pc clock is updated at every 100 ps, there is another clock for datapath component and it is updated every 50 ps. While testing project it would be sufficient to adjuct multisim clock step, if step by step test is desired.

In miniMips_testbench.v file all functionalities could be tested. There are at least 2 example of each instruction type in instruction memory. Some of testbench results are shown in below.





```
#
run
# time= 1200
# pc_new=00000000000000000000000000001010
# opcode=0010  rs=011  rt=110  rd=010  func=101
# reg_dest=0    branch_eq=0     branch_not_eq=0         mem_read=0      mem_to_reg=0    mem_write=0     alu_src=1       reg_write=1     write_reg=110   write_data=00000000000000000000000000000001
# read_data_1=11111111111111111111111111101001  read_data_2=00000000000000000000000000000001    extended_imm=00000000000000000000000000010101
# mem_read_data=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx        alu_control=110         alu_res=00000000000000000000000000000001
#
VSIM 27> run
# time= 1300
# pc_new=00000000000000000000000000001011
# opcode=0010  rs=011  rt=111  rd=010  func=100
# reg_dest=0    branch_eq=0     branch_not_eq=0         mem_read=0      mem_to_reg=0    mem_write=0     alu_src=1       reg_write=1     write_reg=111   write_data=00000000000000000000000000000000
# read_data_1=11111111111111111111111111101001  read_data_2=00000000000000000000000000010110    extended_imm=00000000000000000000000000010100
# mem_read_data=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx        alu_control=110         alu_res=00000000000000000000000000000000
#
```

ANDI

```
run
# time= 1600
# pc_new=00000000000000000000000000001100
# opcode=0011  rs=011  rt=100  rd=000  func=110
# reg_dest=0    branch_eq=0     branch_not_eq=0         mem_read=0      mem_to_reg=0    mem_write=0     alu_src=1       reg_write=1     write_reg=100   write_data=11111111111111111111111111101111
# read_data_1=11111111111111111111111111101001  read_data_2=11111111111111111111111111101111    extended_imm=00000000000000000000000000001110
# mem_read_data=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx        alu_control=111         alu_res=11111111111111111111111111101111
#
run
# time= 1700
# pc_new=00000000000000000000000000001101
# opcode=0011  rs=100  rt=011  rd=001  func=110
# reg_dest=0    branch_eq=0     branch_not_eq=0         mem_read=0      mem_to_reg=0    mem_write=0     alu_src=1       reg_write=1     write_reg=011   write_data=11111111111111111111111111101111
# read_data_1=11111111111111111111111111101111  read_data_2=11111111111111111111111111101001    extended_imm=00000000000000000000000000001110
# mem_read_data=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx        alu_control=111         alu_res=11111111111111111111111111101111
#
```

ORI

```
#
run
# time= 2000
# pc_new=00000000000000000000000000001110
# opcode=0100  rs=100  rt=001  rd=000  func=101
# reg_dest=0    branch_eq=0     branch_not_eq=0         mem_read=0      mem_to_reg=0    mem_write=0     alu_src=1       reg_write=1     write_reg=001   write_data=00000000000000000000000000010000
# read_data_1=11111111111111111111111111101111  read_data_2=00000000000000000000000000010000    extended_imm=00000000000000000000000000000101
# mem_read_data=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx        alu_control=101         alu_res=00000000000000000000000000010000
#
VSIM 27> run
# time= 2100
# pc_new=00000000000000000000000000001111
# opcode=0100  rs=111  rt=011  rd=000  func=111
# reg_dest=0    branch_eq=0     branch_not_eq=0         mem_read=0      mem_to_reg=0    mem_write=0     alu_src=1       reg_write=1     write_reg=011   write_data=11111111111111111111111111111000
# read_data_1=00000000000000000000000000000000  read_data_2=11111111111111111111111111101111    extended_imm=00000000000000000000000000000111
# mem_read_data=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx        alu_control=101         alu_res=11111111111111111111111111111000
#
```

NORI

```
#
VSIM 27> run
# time= 2400
# pc_new=00000000000000000000000000010000
# opcode=0111  rs=011  rt=111  rd=000  func=001
# reg_dest=0    branch_eq=0     branch_not_eq=0         mem_read=0      mem_to_reg=0    mem_write=0     alu_src=1       reg_write=1     write_reg=111   write_data=00000000000000000000000000000001
# read_data_1=11111111111111111111111111111000  read_data_2=00000000000000000000000000000001    extended_imm=00000000000000000000000000000001
# mem_read_data=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx        alu_control=100         alu_res=00000000000000000000000000000001
#
# time= 2500
# pc_new=00000000000000000000000000010001
# opcode=0111  rs=000  rt=011  rd=111  func=111
# reg_dest=0    branch_eq=0     branch_not_eq=0         mem_read=0      mem_to_reg=0    mem_write=0     alu_src=1       reg_write=1     write_reg=011   write_data=00000000000000000000000000000000
# read_data_1=00000000000000000000000000000000  read_data_2=11111111111111111111111111111000    extended_imm=11111111111111111111111111111111
# mem_read_data=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx        alu_control=100         alu_res=00000000000000000000000000000000
#
```

SLTI

```
VSIM 27> run
# time= 2800
# pc_new=00000000000000000000000000010010
# opcode=1000  rs=111  rt=001  rd=000  func=011
# reg_dest=0    branch_eq=0     branch_not_eq=0         mem_read=1      mem_to_reg=1    mem_write=0     alu_src=1       reg_write=1     write_reg=001   write_data=11111111111111111111111111111111
# read_data_1=00000000000000000000000000000001  read_data_2=11111111111111111111111111111111    extended_imm=00000000000000000000000000000011
# mem_read_data=11111111111111111111111111111111        alu_control=000         alu_res=00000000000000000000000000000100
#
# time= 2900
# pc_new=00000000000000000000000000010011
# opcode=1000  rs=000  rt=011  rd=000  func=111
# reg_dest=0    branch_eq=0     branch_not_eq=0         mem_read=1      mem_to_reg=1    mem_write=0     alu_src=1       reg_write=1     write_reg=011   write_data=11111111111111111111111111111111
# read_data_1=00000000000000000000000000000000  read_data_2=00000000000000000000000000000000    extended_imm=00000000000000000000000000000111
# mem_read_data=11111111111111111111111111111111        alu_control=000         alu_res=00000000000000000000000000000111
#
```

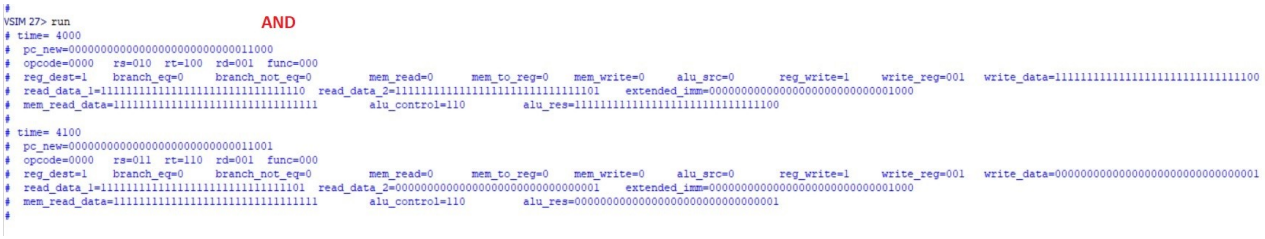
LW

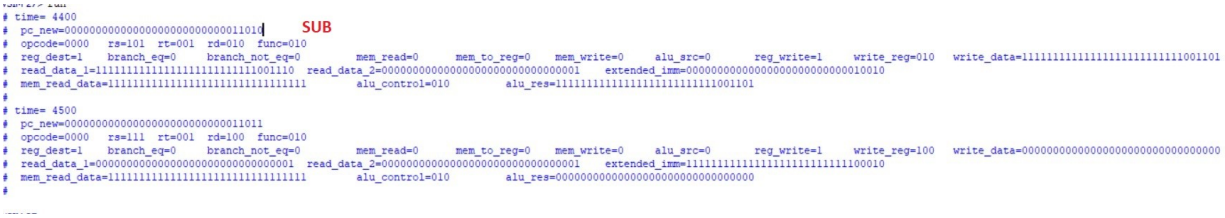
```
# time= 3200
# pc_new=00000000000000000000000000010100
# opcode=1001  rs=100  rt=101  rd=000  func=010
# reg_dest=0    branch_eq=0     branch_not_eq=0         mem_read=0      mem_to_reg=0    mem_write=1     alu_src=1       reg_write=0     write_reg=101   write_data=11111111111111111111111111110001
# read_data_1=11111111111111111111111111101111  read_data_2=11111111111111111111111111110001    extended_imm=00000000000000000000000000000010
# mem_read_data=11111111111111111111111111111111        alu_control=000         alu_res=11111111111111111111111111110001
#
# time= 3300
# pc_new=00000000000000000000000000010101
# opcode=1001  rs=010  rt=011  rd=000  func=110
# reg_dest=0    branch_eq=0     branch_not_eq=0         mem_read=0      mem_to_reg=0    mem_write=1     alu_src=1       reg_write=0     write_reg=011   write_data=00000000000000000000000000000100
# read_data_1=11111111111111111111111111111110  read_data_2=11111111111111111111111111111111    extended_imm=00000000000000000000000000000110
# mem_read_data=11111111111111111111111111111111        alu_control=000         alu_res=00000000000000000000000000000100
#

VSIM 27>
```

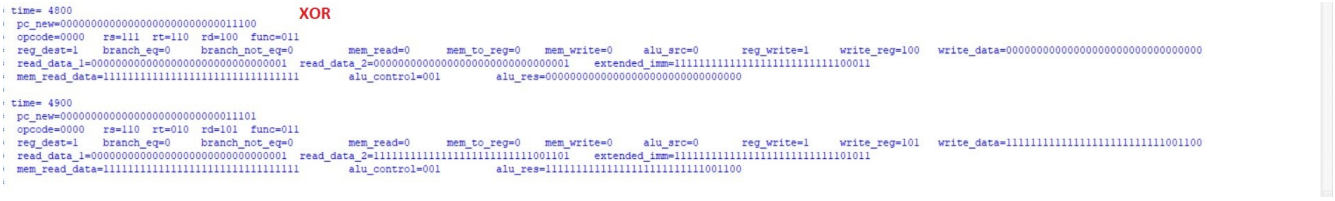
SW

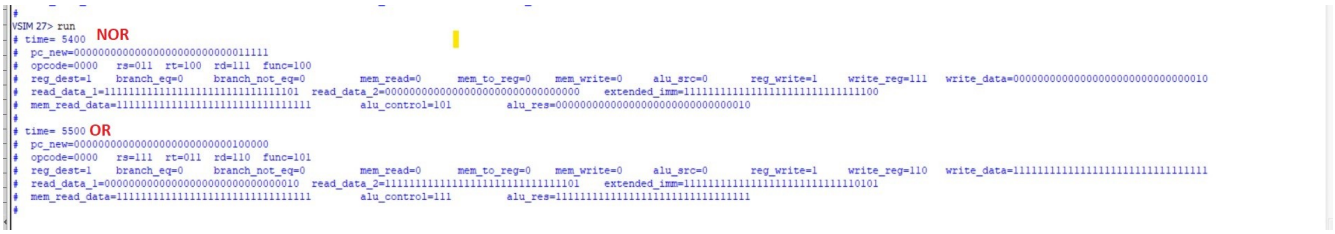
```
VSIM 27> run
# time= 3600
# pc_new=00000000000000000000000000010110
# opcode=0000  rs=001  rt=010  rd=100  func=001
# reg_dest=1    branch_eq=0     branch_not_eq=0         mem_read=0      mem_to_reg=0    mem_write=0     alu_src=0       reg_write=1     write_reg=100   write_data=11111111111111111111111111111101
# read_data_1=11111111111111111111111111111111  read_data_2=11111111111111111111111111111110    extended_imm=11111111111111111111111111100001
# mem_read_data=11111111111111111111111111111111        alu_control=000         alu_res=11111111111111111111111111111101
#
# time= 3700
# pc_new=00000000000000000000000000010111
# opcode=0000  rs=001  rt=010  rd=011  func=001
# reg_dest=1    branch_eq=0     branch_not_eq=0         mem_read=0      mem_to_reg=0    mem_write=0     alu_src=0       reg_write=1     write_reg=011   write_data=11111111111111111111111111111101
# read_data_1=11111111111111111111111111111111  read_data_2=11111111111111111111111111111110    extended_imm=00000000000000000000000000011001
# mem_read_data=11111111111111111111111111111111        alu_control=000         alu_res=11111111111111111111111111111101
#

VSIM 27>
```

ADD









Instruction memory instructions

0101_011_011_000_010 // beq $3 $3 2
0000_000_000_000_000 // nop
0000_000_000_000_000 // nop
0110_000_001_000_010 // bne $0 $1 2
0000_000_000_000_000 // nop
0000_000_000_000_000 // nop
0101_000_011_000_010 // beq $0 $3 2
0110_000_000_000_010 // bne $0 $0 2
0001_010_001_001_001 // addi $2 $1 9
0001_101_100_110_010 // addi $5 $4 (-14)
0010_011_110_010_101 // andi $3 $6 1
0010_011_111_010_100 // andi $3 $7 20
0011_011_100_000_110 // ori $3 $4 6
0011_100_011_001_110 // ori $4 $3 14
0100_100_001_000_101 // nori $4 $1 5

```
0100_111_011_000_111    // nori $7 $3 7
0111_011_111_000_001    // slti $3 $7 1
0111_000_011_111_111    // slti $0 $3 -1
1000_111_001_000_011    // lw   $7 $1 3
1000_000_011_000_111    // lw   $0 $3 7
1001_100_101_000_010    // sw   $4 $5 2
1001_010_011_000_110    // sw   $2 $3 6
0000_001_010_100_001              // add  $2 $4 $1
0000_001_010_011_001               // add $1 $2 $3
0000_010_100_001_000              // and $2 $4 $1
0000_011_110_001_000              // and $3 $6 $1
0000_101_001_010_010              // sub $5 $1 $2
0000_111_001_100_010              // sub $6 $1 $4
0000_111_110_100_011              // xor $7 $6 $4
0000_110_010_101_011              // xor $6 $2 $5
0000_100_001_010_100              // nor $4 $1 $2
0000_011_100_111_100              // nor $3 $4 $7
0000_111_011_110_101              // or  $7 $3 $6
0000_110_111_001_101              // or  $1 $6 $7
```