GEBZE TECHNICAL UNIVERSITY

CSE 331/503 Computer Organization

Homework # 4

Ahmet YAZICI

1801042639

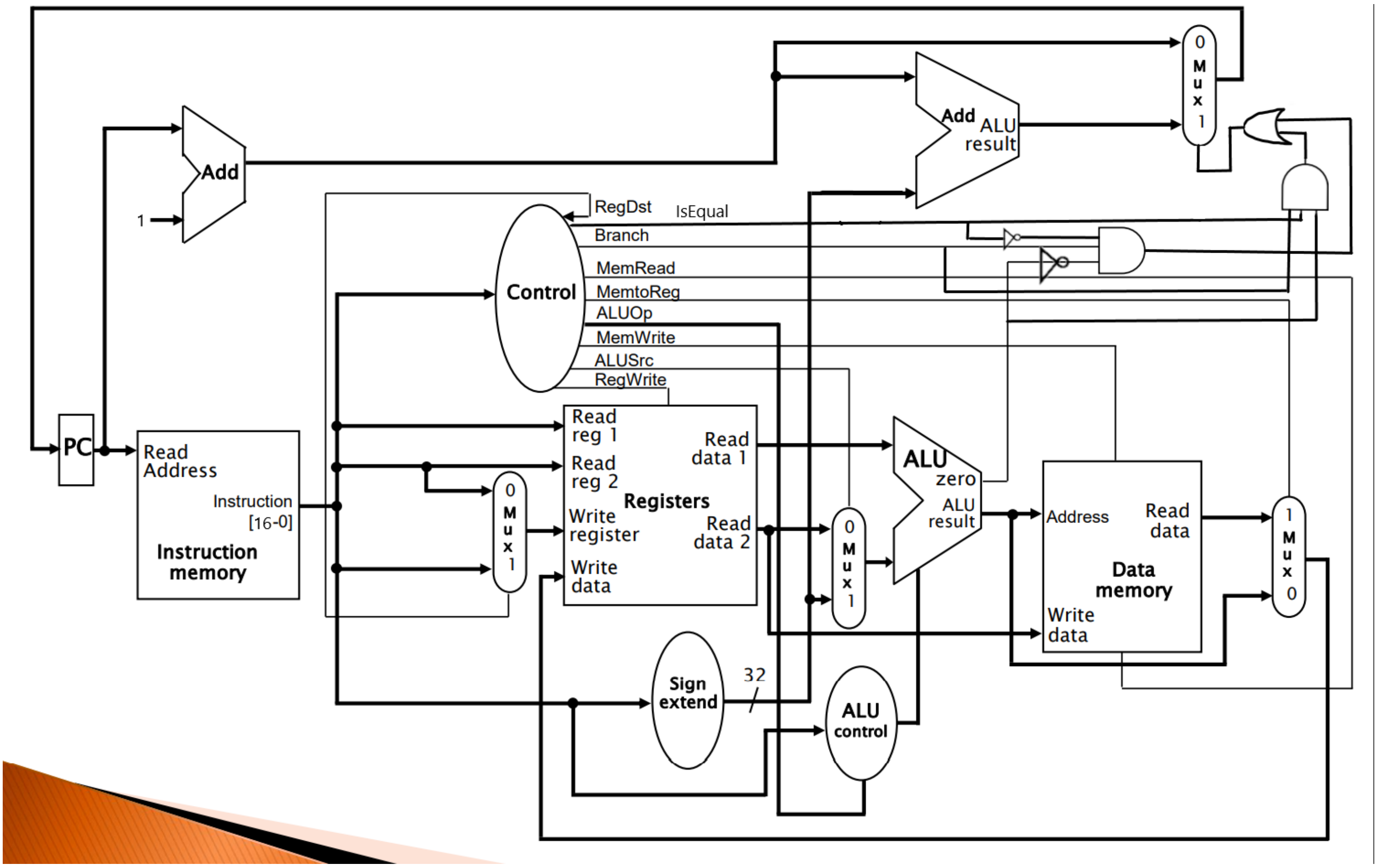
The Truth Table for the Main Control

	000	0001	0010	0011	0100	0101	0110	0111	1000	1001
	R-type	addi	andi	ori	nori	beq	bne	slti	lw	sw
RegDst	1	0	0	0	0	x	x	x	x	x
ALUSrc	0	1	1	1	1	0	0	1	1	1
MemtoReg	0	0	0	0	0	x	x	0	1	x
RegWrite	1	1	1	1	1	0	0	1	1	0
MemRead	0	0	0	0	0	0	0	0	1	0
MemWrite	0	0	0	0	0	0	0	0	0	1
Branch	0	0	0	0	0	1	1	0	0	0
IsEqual	0	0	0	0	0	1	0	0	0	0
ALUop(Symbolic)	r	add	and	or	nor	sub	sub	slt	add	add
ALUop2	1	0	0	1	1	0	0	1	0	0
ALUop1	1	1	1	0	0	0	0	1	0	0
ALUop2	1	0	1	0	1	1	1	0	0	0

The Truth Table for the ALU Control

Instruction opcode	P2 P1 P0 ALUop	F2 F1 F0 Function	Desired ALU action	C2 C1 C0 ALU control
lw	000	xxx	add	000
sw	000	xxx	add	000
beq	001	xxx	sub	010
bne	001	xxx	sub	010
addi	010	xxx	add	000
andi	011	xxx	and	110
ori	100	xxx	or	111
nori	101	xxx	nor	101
slti	110	xxx	slt	100
r-type	111	000	and	110
r-type	111	001	add	000
r-type	111	010	sub	010
r-type	111	011	xor	001
r-type	111	100	nor	101
r-type	111	101	or	111

The Single Cycle Datapath

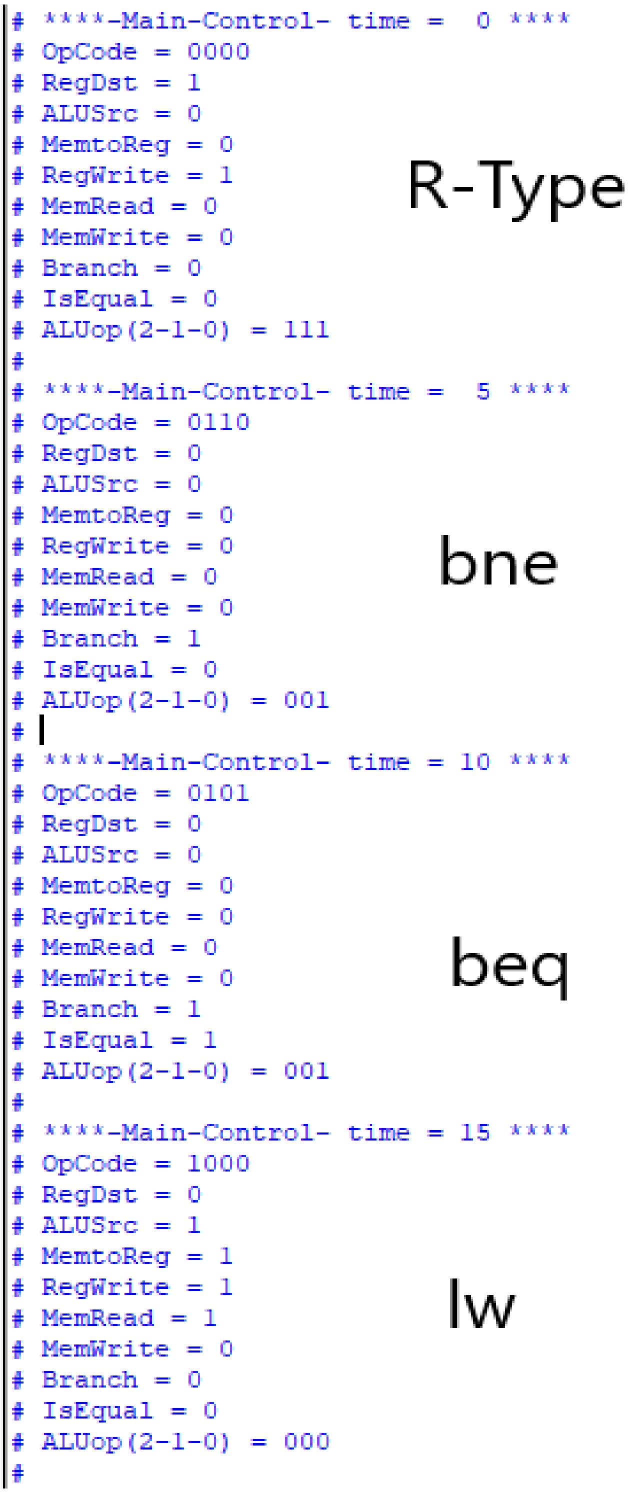


I implemented the MiniMIPS processor according to this datapath which is taken from the book. For perform bne and beq instructions, I add additional logical gates and one extra main control signa(IsEqual). The new signal will be 1 when the instruction is beq. With the help of this extra signal, the processor can distinguish between beq and bne. For program counter, I use 32-bit program counter and for next instruction,I incremented it by one. For branch calculation, I did not shift the immediate value.All the other parts exactly same with book’s datapath.

Added and subtracted parts can be seen in the datapath photo.

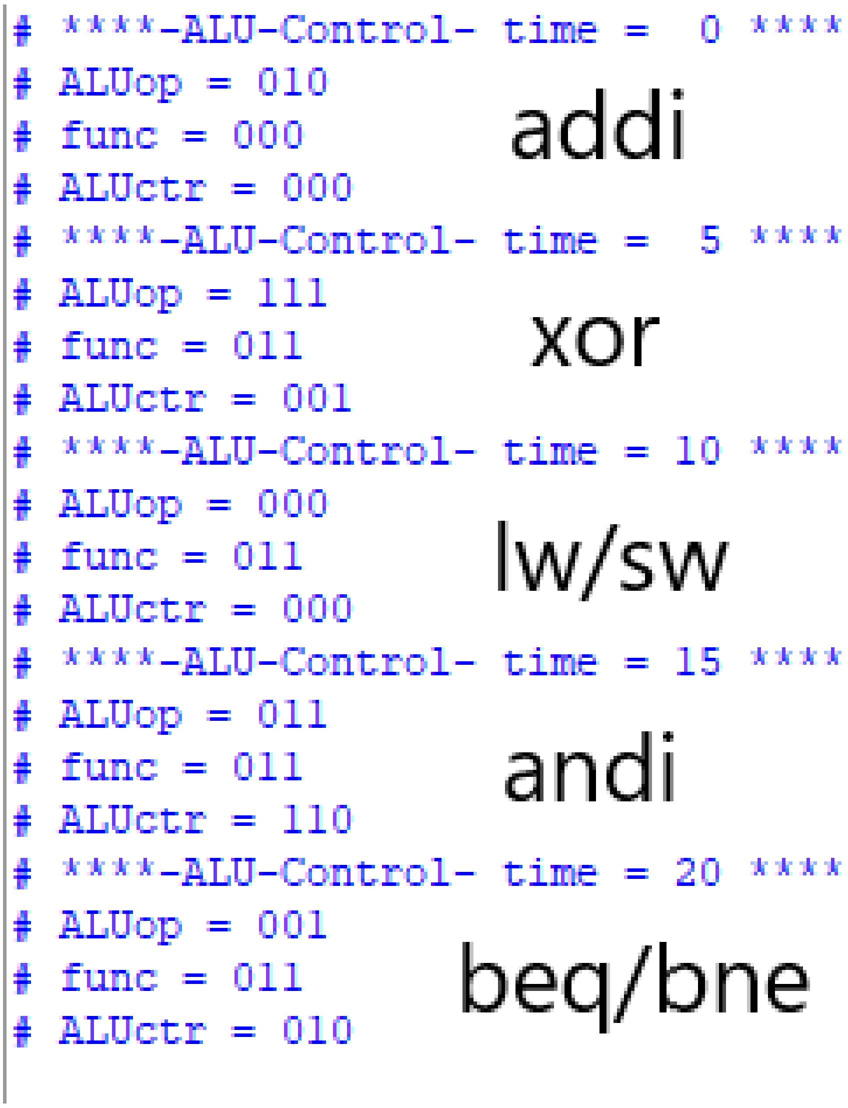
Testbench Results

-Main Control



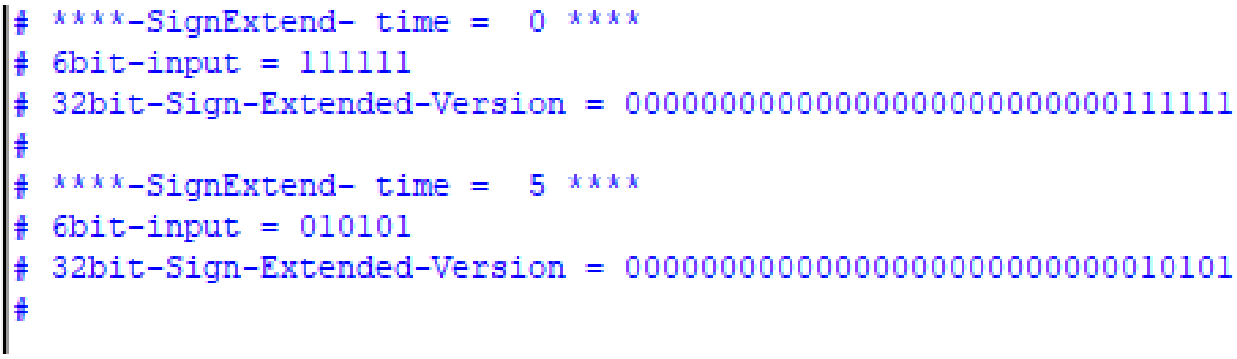
```
# ****-Main-Control- time =  0 ****
# OpCode = 0000
# RegDst = 1
# ALUSrc = 0
# MemtoReg = 0
# RegWrite = 1
# MemRead = 0
# MemWrite = 0
# Branch = 0
# IsEqual = 0
# ALUop(2-1-0) = 111
#
# ****-Main-Control- time =  5 ****
# OpCode = 0110
# RegDst = 0
# ALUSrc = 0
# MemtoReg = 0
# RegWrite = 0
# MemRead = 0
# MemWrite = 0
# Branch = 1
# IsEqual = 0
# ALUop(2-1-0) = 001
#
# ****-Main-Control- time = 10 ****
# OpCode = 0101
# RegDst = 0
# ALUSrc = 0
# MemtoReg = 0
# RegWrite = 0
# MemRead = 0
# MemWrite = 0
# Branch = 1
# IsEqual = 1
# ALUop(2-1-0) = 001
#
# ****-Main-Control- time = 15 ****
# OpCode = 1000
# RegDst = 0
# ALUSrc = 1
# MemtoReg = 1
# RegWrite = 1
# MemRead = 1
# MemWrite = 0
# Branch = 0
# IsEqual = 0
# ALUop(2-1-0) = 000
#
```

-ALU Control



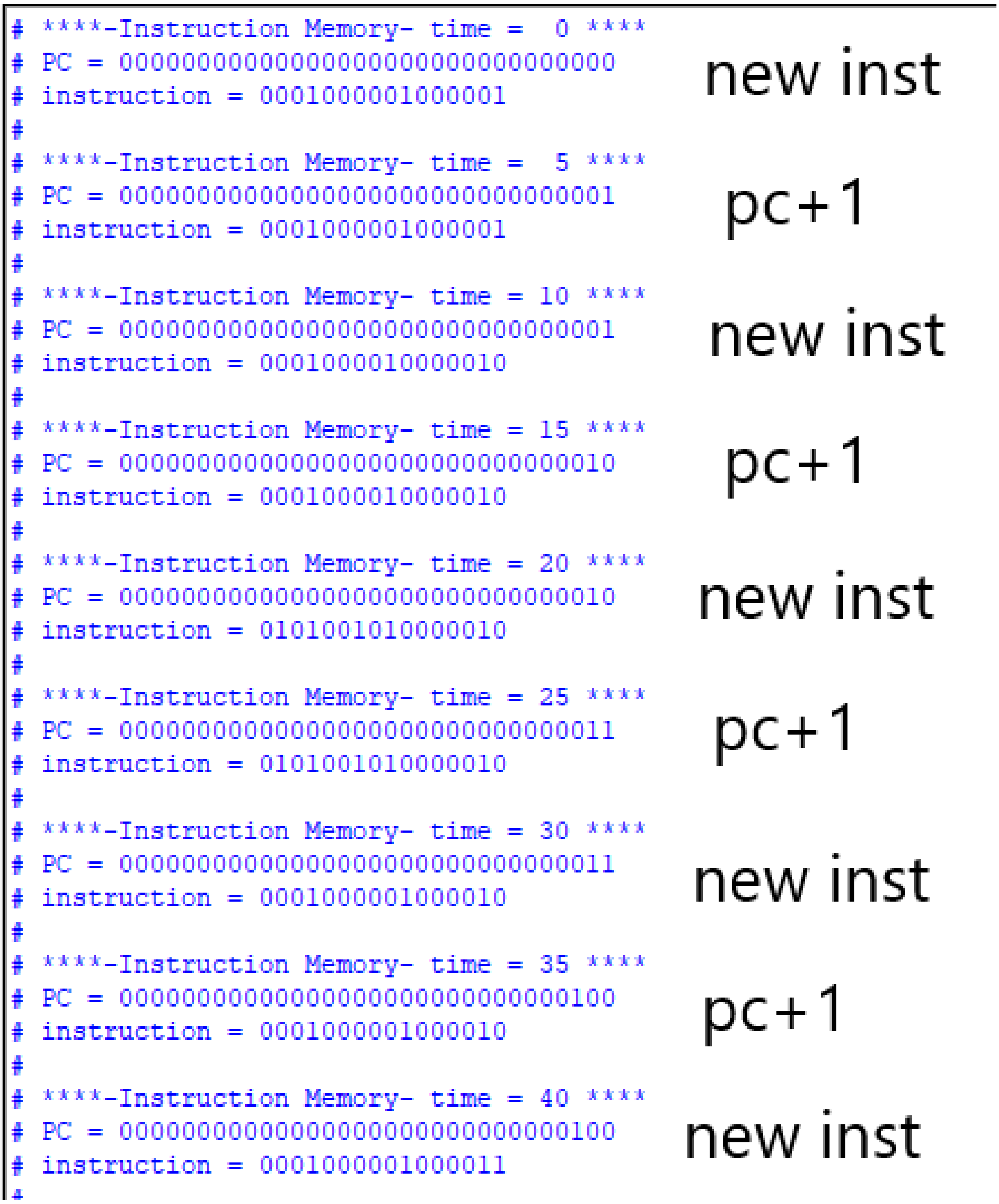
```
# ****-ALU-Control- time =  0 ****
# ALUop = 010
# func = 000
# ALUctr = 000
# ****-ALU-Control- time =  5 ****
# ALUop = 111
# func = 011
# ALUctr = 001
# ****-ALU-Control- time = 10 ****
# ALUop = 000
# func = 011
# ALUctr = 000
# ****-ALU-Control- time = 15 ****
# ALUop = 011
# func = 011
# ALUctr = 110
# ****-ALU-Control- time = 20 ****
# ALUop = 001
# func = 011
# ALUctr = 010
```

-Sign Extend



```
# ****-SignExtend- time =  0 ****
# 6bit-input = 111111
# 32bit-Sign-Extended-Version = 00000000000000000000000000111111
#
# ****-SignExtend- time =  5 ****
# 6bit-input = 010101
# 32bit-Sign-Extended-Version = 00000000000000000000000000010101
#
```

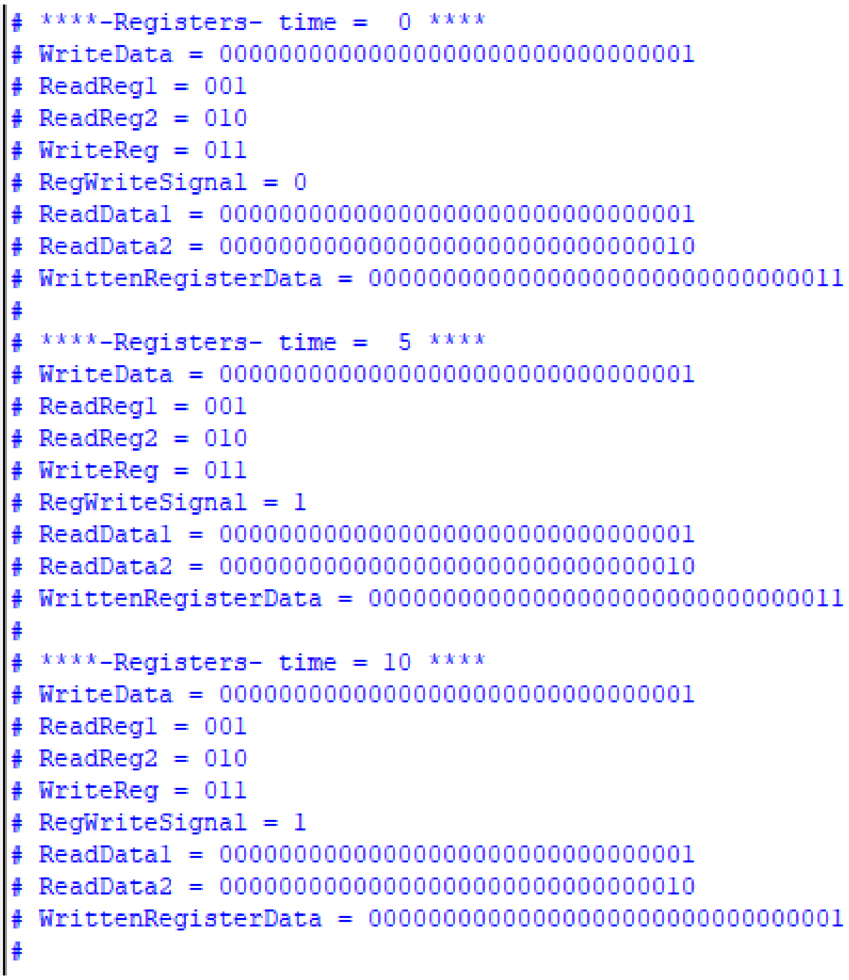
-Instruction Memory



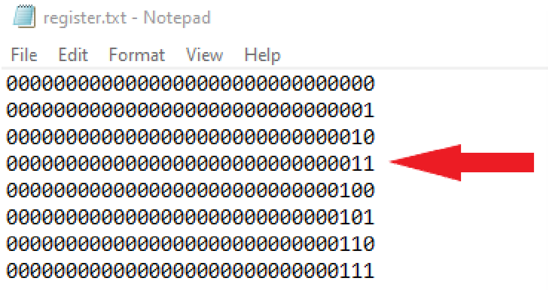
```
# ****-Instruction Memory- time =  0 ****
# PC = 00000000000000000000000000000000
# instruction = 0001000001000001
#
# ****-Instruction Memory- time =  5 ****
# PC = 00000000000000000000000000000001
# instruction = 0001000001000001
#
# ****-Instruction Memory- time = 10 ****
# PC = 00000000000000000000000000000001
# instruction = 0001000010000010
#
# ****-Instruction Memory- time = 15 ****
# PC = 00000000000000000000000000000010
# instruction = 0001000010000010
#
# ****-Instruction Memory- time = 20 ****
# PC = 00000000000000000000000000000010
# instruction = 0101001010000010
#
# ****-Instruction Memory- time = 25 ****
# PC = 00000000000000000000000000000011
# instruction = 0101001010000010
#
# ****-Instruction Memory- time = 30 ****
# PC = 00000000000000000000000000000011
# instruction = 0001000001000010
#
# ****-Instruction Memory- time = 35 ****
# PC = 00000000000000000000000000000100
# instruction = 0001000001000010
#
# ****-Instruction Memory- time = 40 ****
# PC = 00000000000000000000000000000100
# instruction = 0001000001000011
#
```

(The instructions in testbench not the final instructions)

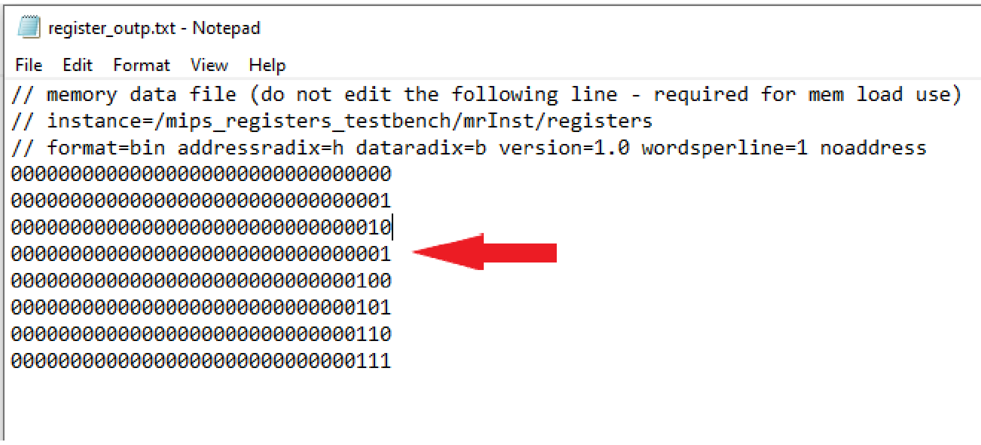
-Registers



```
# ****-Registers- time =  0 ****
# WriteData = 00000000000000000000000000000001
# ReadReg1 = 001
# ReadReg2 = 010
# WriteReg = 011
# RegWriteSignal = 0
# ReadData1 = 00000000000000000000000000000001
# ReadData2 = 00000000000000000000000000000010
# WrittenRegisterData = 00000000000000000000000000000011
#
# ****-Registers- time =  5 ****
# WriteData = 00000000000000000000000000000001
# ReadReg1 = 001
# ReadReg2 = 010
# WriteReg = 011
# RegWriteSignal = 1
# ReadData1 = 00000000000000000000000000000001
# ReadData2 = 00000000000000000000000000000010
# WrittenRegisterData = 00000000000000000000000000000011
#
# ****-Registers- time = 10 ****
# WriteData = 00000000000000000000000000000001
# ReadReg1 = 001
# ReadReg2 = 010
# WriteReg = 011
# RegWriteSignal = 1
# ReadData1 = 00000000000000000000000000000001
# ReadData2 = 00000000000000000000000000000010
# WrittenRegisterData = 00000000000000000000000000000001
#
```

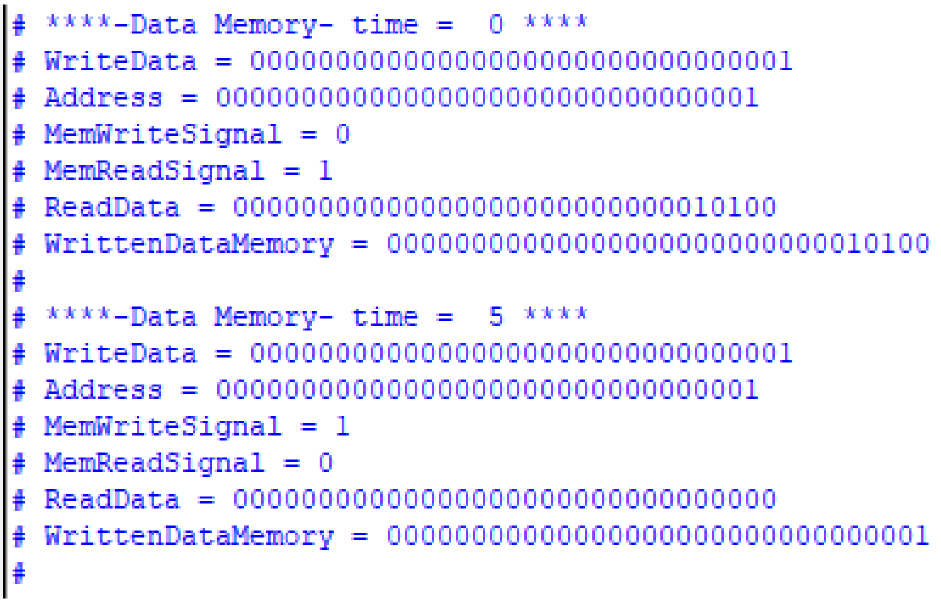


```
0000000000000000000000000000000
0000000000000000000000000000001
0000000000000000000000000000010
0000000000000000000000000000011
0000000000000000000000000000100
0000000000000000000000000000101
0000000000000000000000000000110
0000000000000000000000000000111
```

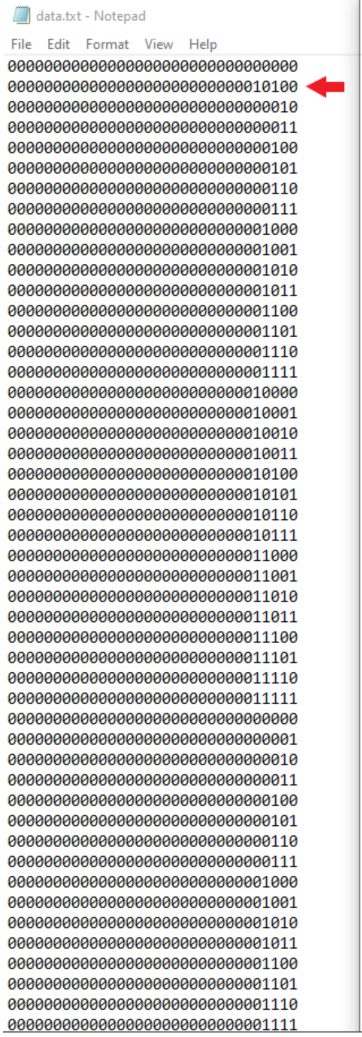


```
// memory data file (do not edit the following line - required for mem load use)
// instance=/mips_registers_testbench/mrInst/registers
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000001
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

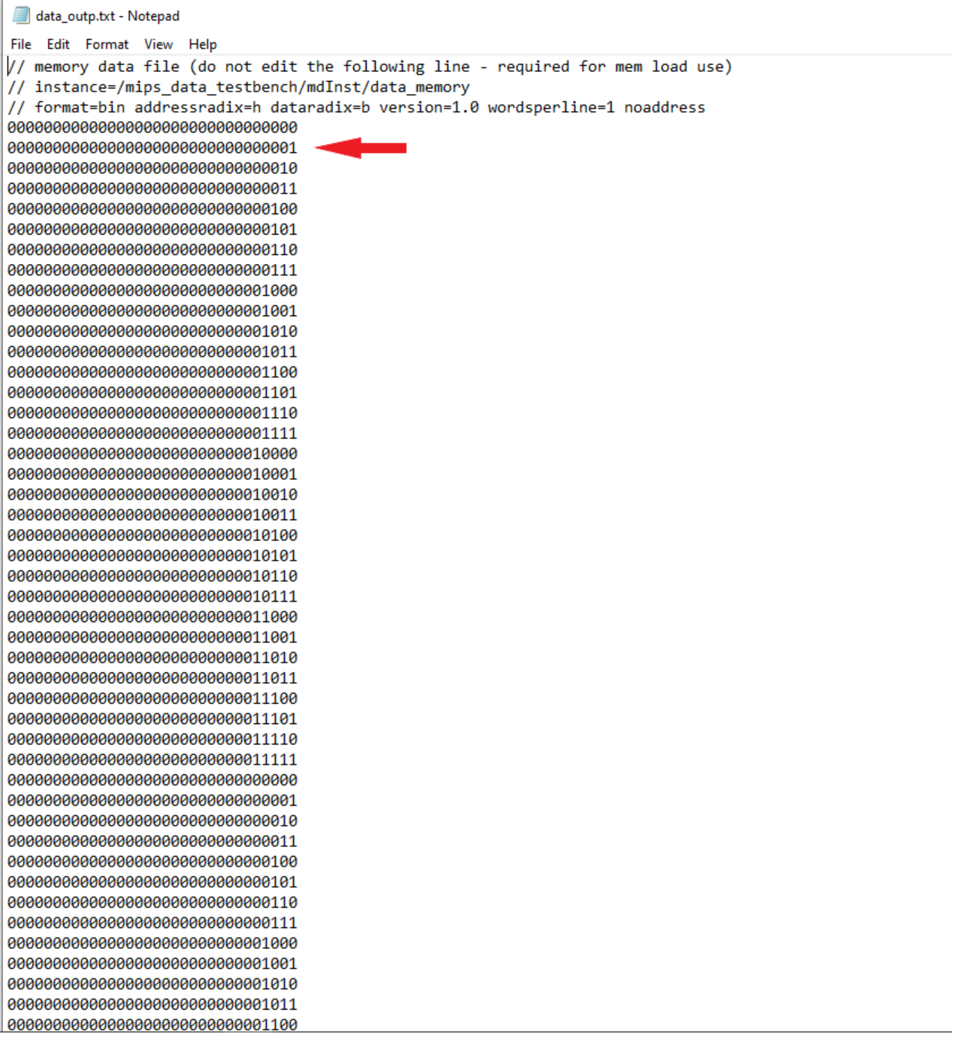
-Data Memory



```
# ****-Data Memory- time =  0 ****
# WriteData = 00000000000000000000000000000001
# Address = 00000000000000000000000000000001
# MemWriteSignal = 0
# MemReadSignal = 1
# ReadData = 00000000000000000000000000010100
# WrittenDataMemory = 00000000000000000000000000010100
#
# ****-Data Memory- time =  5 ****
# WriteData = 00000000000000000000000000000001
# Address = 00000000000000000000000000000001
# MemWriteSignal = 1
# MemReadSignal = 0
# ReadData = 00000000000000000000000000000000
# WrittenDataMemory = 00000000000000000000000000000001
#
```

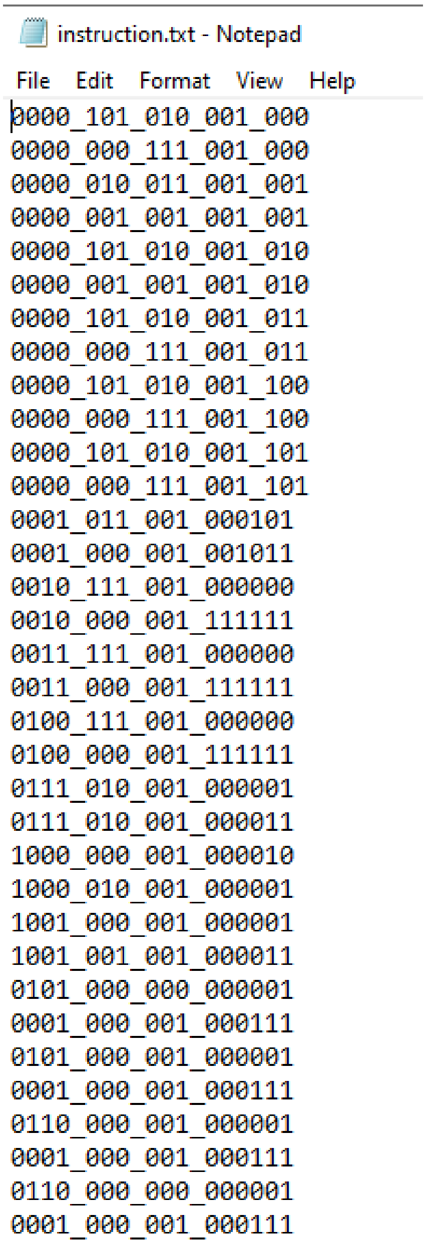


```
00000000000000000000000000000000
00000000000000000000000000010100
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
00000000000000000000000000001101
00000000000000000000000000001110
00000000000000000000000000001111
00000000000000000000000000010000
00000000000000000000000000010001
00000000000000000000000000010010
00000000000000000000000000010011
00000000000000000000000000010100
00000000000000000000000000010101
00000000000000000000000000010110
00000000000000000000000000010111
00000000000000000000000000011000
00000000000000000000000000011001
00000000000000000000000000011010
00000000000000000000000000011011
00000000000000000000000000011100
00000000000000000000000000011101
00000000000000000000000000011110
00000000000000000000000000011111
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
00000000000000000000000000001101
00000000000000000000000000001110
00000000000000000000000000001111
```



```
// memory data file (do not edit the following line - required for mem load use)
// instance=/mips_data_testbench/mdInst/data_memory
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
00000000000000000000000000001101
00000000000000000000000000001110
00000000000000000000000000001111
00000000000000000000000000010000
00000000000000000000000000010001
00000000000000000000000000010010
00000000000000000000000000010011
00000000000000000000000000010100
00000000000000000000000000010101
00000000000000000000000000010110
00000000000000000000000000010111
00000000000000000000000000011000
00000000000000000000000000011001
00000000000000000000000000011010
00000000000000000000000000011011
00000000000000000000000000011100
00000000000000000000000000011101
00000000000000000000000000011110
00000000000000000000000000011111
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
```

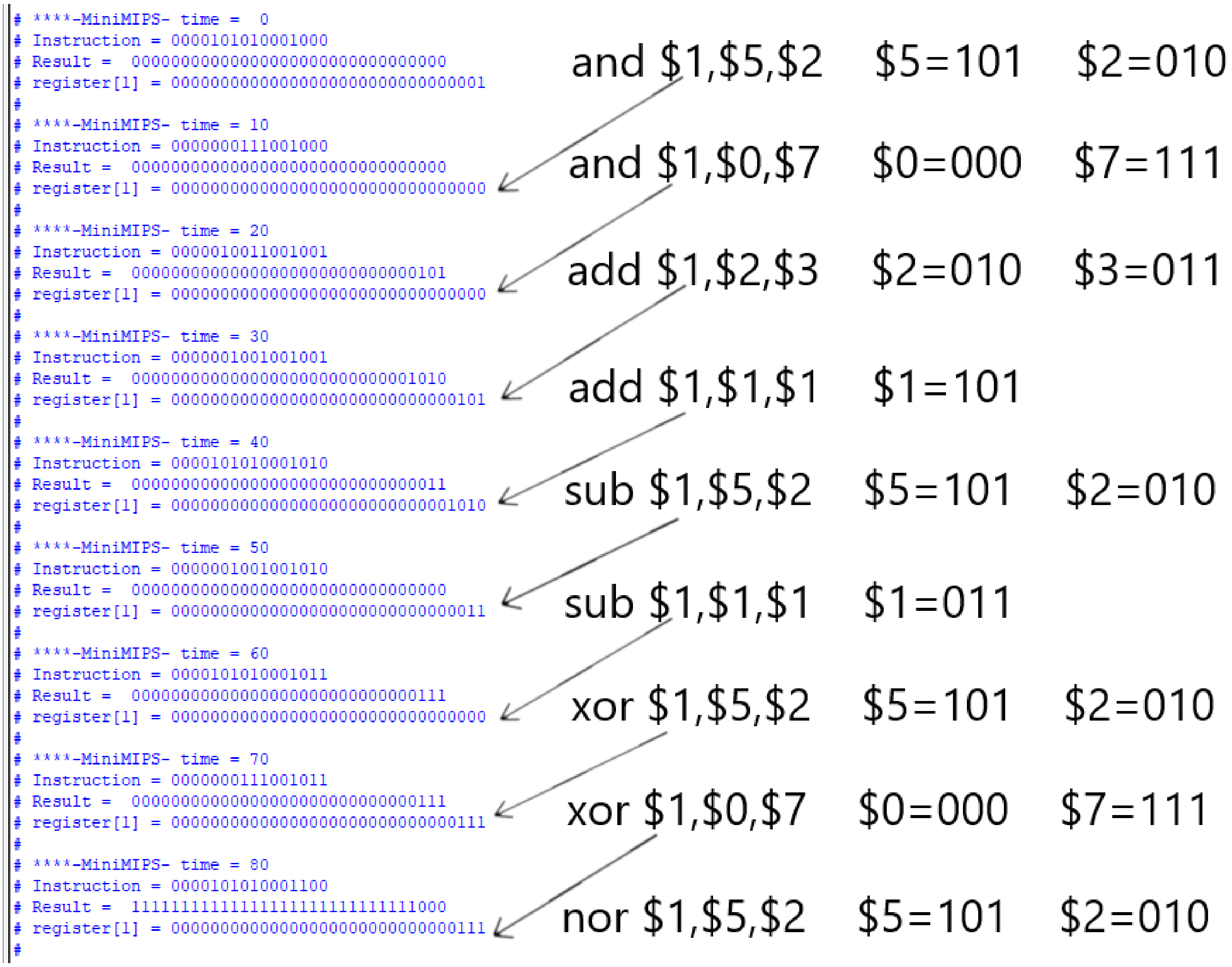
-MiniMIPS



```
0000_101_010_001_000
0000_000_111_001_000
0000_010_011_001_001
0000_001_001_001_001
0000_101_010_001_010
0000_001_001_001_010
0000_101_010_001_011
0000_000_111_001_011
0000_101_010_001_100
0000_000_111_001_100
0000_101_010_001_101
0000_000_111_001_101
0001_011_001_000101
0001_000_001_001011
0010_111_001_000000
0010_000_001_111111
0011_111_001_000000
0011_000_001_111111
0100_111_001_000000
0100_000_001_111111
0111_010_001_000001
0111_010_001_000011
1000_000_001_000010
1000_010_001_000001
1001_000_001_000001
1001_001_001_000011
0101_000_000_000001
0001_000_001_000111
0101_000_001_000001
0001_000_001_000111
0110_000_001_000001
0001_000_001_000111
0110_000_000_000001
0001_000_001_000111
```

I used these instructions to test all the instructions twice. For testing the beq and bne, I added extra addi instructions. To make the test easier to understand, I always save the results in the register 1($1) and print out the register 1.

The arrows show the content of register 1($1) is changing. The result of operation stores in register 1.



```
# ****-MiniMIPS- time =  0
# Instruction = 0000101010001000
# Result =  00000000000000000000000000000000
# register[1] = 00000000000000000000000000000001
#
# ****-MiniMIPS- time = 10
# Instruction = 0000000111001000
# Result =  00000000000000000000000000000000
# register[1] = 00000000000000000000000000000000
#
# ****-MiniMIPS- time = 20
# Instruction = 0000010011001001
# Result =  00000000000000000000000000000101
# register[1] = 00000000000000000000000000000000
#
# ****-MiniMIPS- time = 30
# Instruction = 0000001001001001
# Result =  00000000000000000000000000001010
# register[1] = 00000000000000000000000000000101
#
# ****-MiniMIPS- time = 40
# Instruction = 0000101010001010
# Result =  00000000000000000000000000000011
# register[1] = 00000000000000000000000000001010
#
# ****-MiniMIPS- time = 50
# Instruction = 0000001001001010
# Result =  00000000000000000000000000000000
# register[1] = 00000000000000000000000000000011
#
# ****-MiniMIPS- time = 60
# Instruction = 0000101010001011
# Result =  00000000000000000000000000000111
# register[1] = 00000000000000000000000000000000
#
# ****-MiniMIPS- time = 70
# Instruction = 0000000111001011
# Result =  00000000000000000000000000000111
# register[1] = 00000000000000000000000000000111
#
# ****-MiniMIPS- time = 80
# Instruction = 0000101010001100
# Result =  11111111111111111111111111111000
# register[1] = 00000000000000000000000000000111
#
```

and $1,$5,$2 $5=101 $2=010

and $1,$0,$7 $0=000 $7=111

add $1,$2,$3 $2=010 $3=011

add $1,$1,$1 $1=101

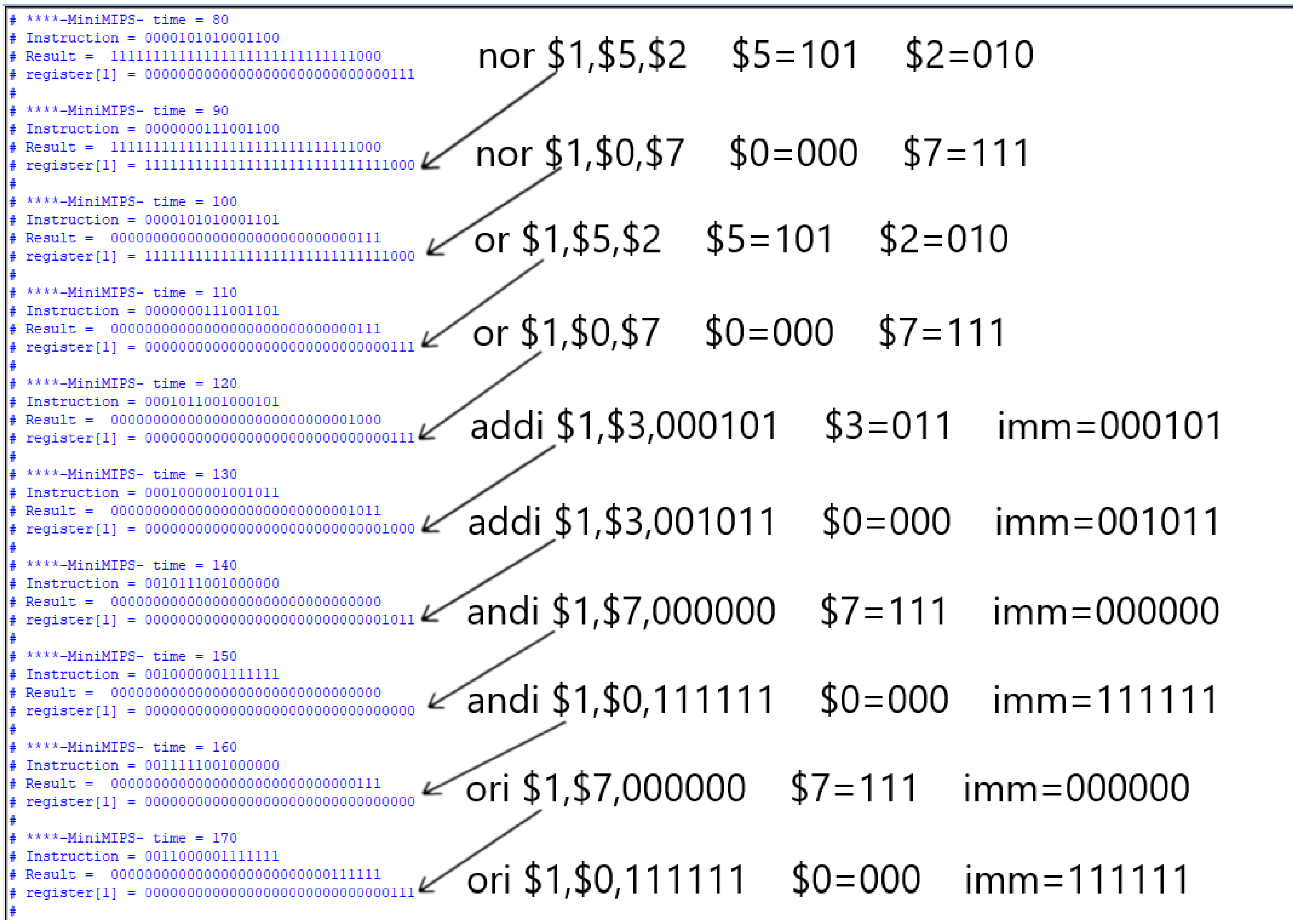
sub $1,$5,$2 $5=101 $2=010

sub $1,$1,$1 $1=011

xor $1,$5,$2 $5=101 $2=010

xor $1,$0,$7 $0=000 $7=111

nor $1,$5,$2 $5=101 $2=010



```
# ****-MiniMIPS- time = 80
# Instruction = 0000101010001100
# Result =  11111111111111111111111111111000
# register[1] = 00000000000000000000000000000111
#
# ****-MiniMIPS- time = 90
# Instruction = 0000000111001100
# Result =  11111111111111111111111111111000
# register[1] = 11111111111111111111111111111000
#
# ****-MiniMIPS- time = 100
# Instruction = 0000101010001101
# Result =  00000000000000000000000000000111
# register[1] = 11111111111111111111111111111000
#
# ****-MiniMIPS- time = 110
# Instruction = 0000000111001101
# Result =  00000000000000000000000000000111
# register[1] = 00000000000000000000000000000111
#
# ****-MiniMIPS- time = 120
# Instruction = 0001011001000101
# Result =  00000000000000000000000000001000
# register[1] = 00000000000000000000000000000111
#
# ****-MiniMIPS- time = 130
# Instruction = 0001000001001011
# Result =  00000000000000000000000000001011
# register[1] = 00000000000000000000000000001000
#
# ****-MiniMIPS- time = 140
# Instruction = 0010111001000000
# Result =  00000000000000000000000000000000
# register[1] = 00000000000000000000000000001011
#
# ****-MiniMIPS- time = 150
# Instruction = 0010000001111111
# Result =  00000000000000000000000000000000
# register[1] = 00000000000000000000000000000000
#
# ****-MiniMIPS- time = 160
# Instruction = 0011111001000000
# Result =  00000000000000000000000000000111
# register[1] = 00000000000000000000000000000000
#
# ****-MiniMIPS- time = 170
# Instruction = 0011000001111111
# Result =  00000000000000000000000000111111
# register[1] = 00000000000000000000000000000111
#
```

nor $1,$5,$2 $5=101 $2=010

nor $1,$0,$7 $0=000 $7=111

or $1,$5,$2 $5=101 $2=010

or $1,$0,$7 $0=000 $7=111

addi $1,$3,000101 $3=011   imm=000101

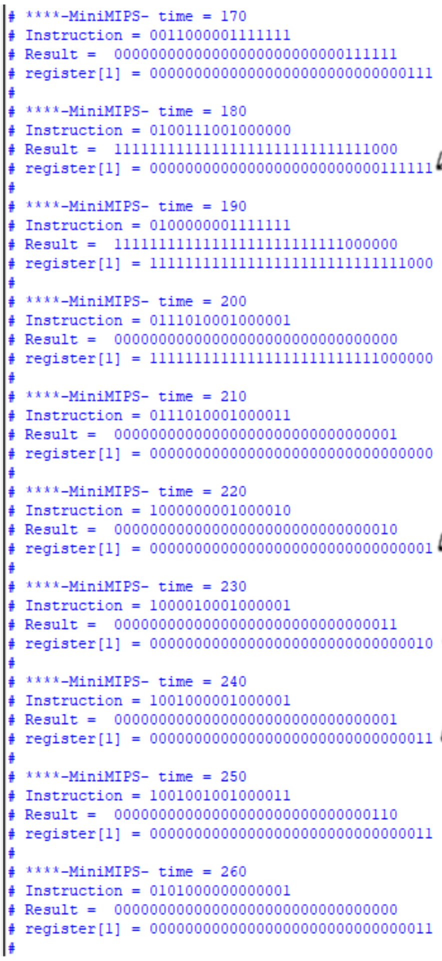
addi $1,$3,001011 $0=000   imm=001011

andi $1,$7,000000 $7=111   imm=000000

andi $1,$0,111111 $0=000   imm=111111

ori $1,$7,000000 $7=111   imm=000000

ori $1,$0,111111 $0=000   imm=111111



```
# ****-MiniMIPS- time = 170
# Instruction = 0011000001111111
# Result =  00000000000000000000000000111111
# register[1] = 00000000000000000000000000000111
#
# ****-MiniMIPS- time = 180
# Instruction = 0100111001000000
# Result =  11111111111111111111111111111000
# register[1] = 00000000000000000000000000111111
#
# ****-MiniMIPS- time = 190
# Instruction = 0100000001111111
# Result =  11111111111111111111111111000000
# register[1] = 11111111111111111111111111111000
#
# ****-MiniMIPS- time = 200
# Instruction = 0111010001000001
# Result =  00000000000000000000000000000000
# register[1] = 11111111111111111111111111000000
#
# ****-MiniMIPS- time = 210
# Instruction = 0111010001000011
# Result =  00000000000000000000000000000001
# register[1] = 00000000000000000000000000000000
#
# ****-MiniMIPS- time = 220
# Instruction = 1000000001000010
# Result =  00000000000000000000000000000010
# register[1] = 00000000000000000000000000000001
#
# ****-MiniMIPS- time = 230
# Instruction = 1000010001000001
# Result =  00000000000000000000000000000011
# register[1] = 00000000000000000000000000000010
#
# ****-MiniMIPS- time = 240
# Instruction = 1001000001000001
# Result =  00000000000000000000000000000001
# register[1] = 00000000000000000000000000000011
#
# ****-MiniMIPS- time = 250
# Instruction = 1001001001000011
# Result =  00000000000000000000000000000110
# register[1] = 00000000000000000000000000000011
#
# ****-MiniMIPS- time = 260
# Instruction = 0101000000000001
# Result =  00000000000000000000000000000000
# register[1] = 00000000000000000000000000000011
#
```

ori $1,$0,111111 $0=000   imm=111111

nori $1,$7,000000 $7=111   imm=000000

nori $1,$0,111111 $0=000   imm=111111

slti $1,$2,000001 $2=010   imm=000001

slti $1,$2,000011 $2=010   imm=000011

lw $1,$0,000010 $0=000   imm=000010   m[2] = 00..010

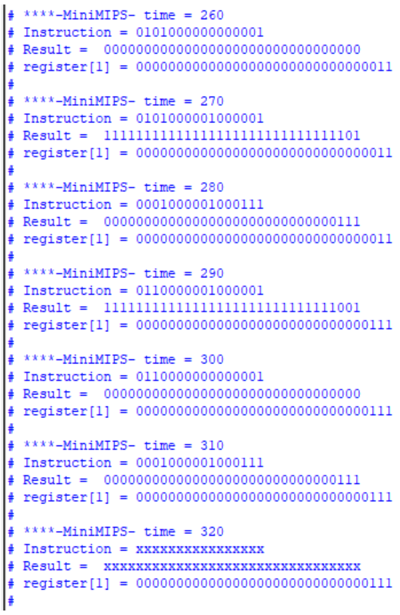
lw $1,$2,000001 $2=010   imm=000001   m[3] = 00..011

sw $1,$0,000001 $0=000   imm=000001 $1=011

sw $1,$1,000011 $1=011   imm=000011 $1=011

beq $0,$0,000001 $0=000   imm=000001

since rs==rt, addi(next inst) instruction should be skipped



```
# ****-MiniMIPS- time = 260
# Instruction = 0101000000000001
# Result =  00000000000000000000000000000000
# register[1] = 00000000000000000000000000000011
#
# ****-MiniMIPS- time = 270
# Instruction = 0101000001000001
# Result =  11111111111111111111111111111101
# register[1] = 00000000000000000000000000000011
#
# ****-MiniMIPS- time = 280
# Instruction = 0001000001000111
# Result =  00000000000000000000000000000111
# register[1] = 00000000000000000000000000000011
#
# ****-MiniMIPS- time = 290
# Instruction = 0110000001000001
# Result =  11111111111111111111111111111001
# register[1] = 00000000000000000000000000000111
#
# ****-MiniMIPS- time = 300
# Instruction = 0110000000000001
# Result =  00000000000000000000000000000000
# register[1] = 00000000000000000000000000000111
#
# ****-MiniMIPS- time = 310
# Instruction = 0001000001000111
# Result =  00000000000000000000000000000111
# register[1] = 00000000000000000000000000000111
#
# ****-MiniMIPS- time = 320
# Instruction = xxxxxxxxxxxxxxxx
# Result =  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# register[1] = 00000000000000000000000000000111
#
```

beq $0,$0,000001 $0=000   imm=000001

beq $1,$0,000001 $1=011 $0=000   imm=000001

since rs!=rt, addi(next inst) instruction is carried out

bne $1,$0,000001 $1=011 $0=000   imm=000001

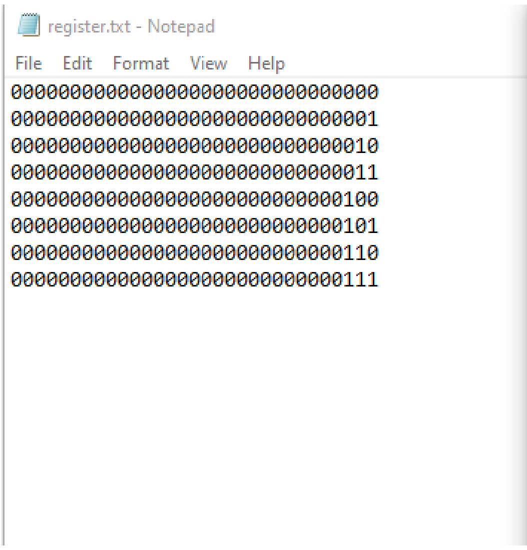
since rs!=rt,addi(next inst) instruction is skipped

bne $0,$0,000001 $0=000   imm=000001

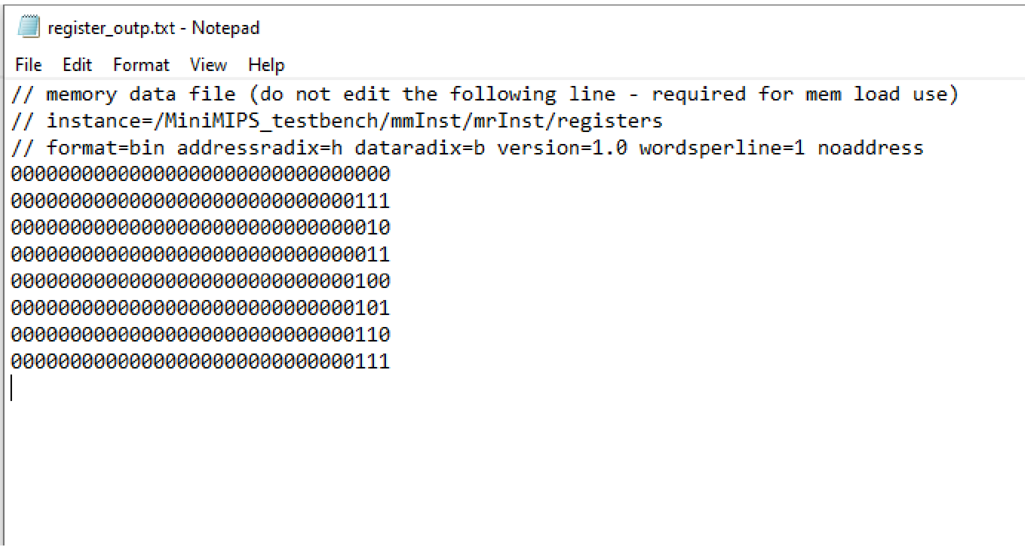
since rs==rt,addi(next inst) instruction is carried out

end of instructions

Registers before and after execution.

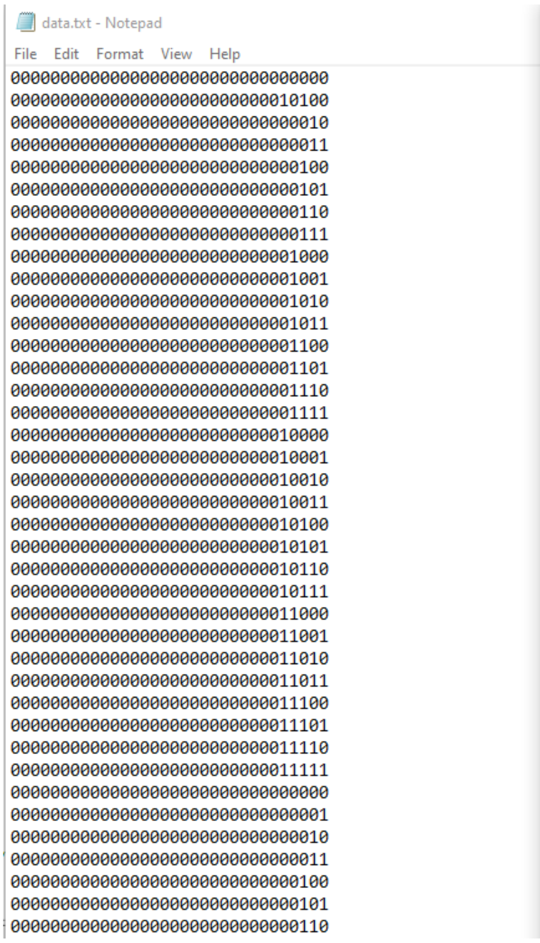


```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

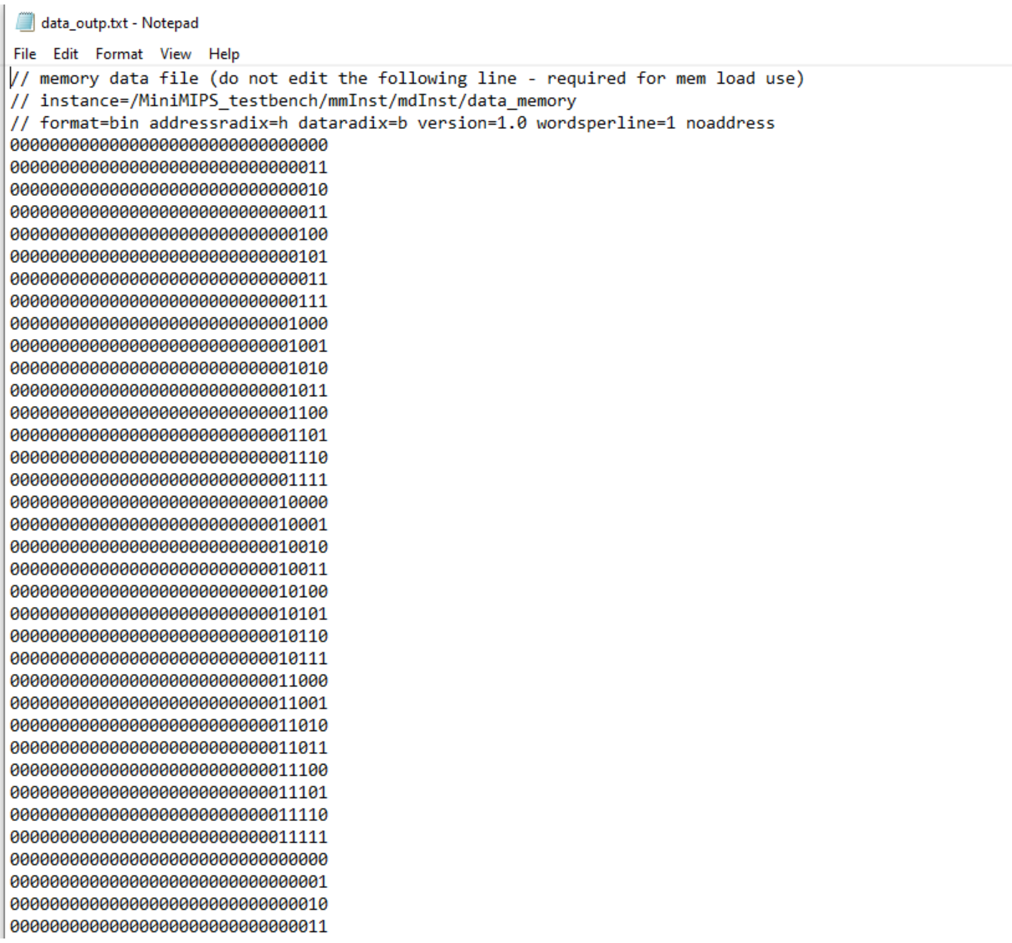


```
// memory data file (do not edit the following line - required for mem load use)
// instance=/MiniMIPS_testbench/mmInst/mrInst/registers
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
00000000000000000000000000000000
00000000000000000000000000000111
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

Data memory before and after execution.



```
00000000000000000000000000000000
00000000000000000000000000010100
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
00000000000000000000000000001101
00000000000000000000000000001110
00000000000000000000000000001111
00000000000000000000000000010000
00000000000000000000000000010001
00000000000000000000000000010010
00000000000000000000000000010011
00000000000000000000000000010100
00000000000000000000000000010101
00000000000000000000000000010110
00000000000000000000000000010111
00000000000000000000000000011000
00000000000000000000000000011001
00000000000000000000000000011010
00000000000000000000000000011011
00000000000000000000000000011100
00000000000000000000000000011101
00000000000000000000000000011110
00000000000000000000000000011111
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
```



```
// memory data file (do not edit the following line - required for mem load use)
// instance=/MiniMIPS_testbench/mmInst/mdInst/data_memory
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
00000000000000000000000000000000
00000000000000000000000000000011
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000011
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
00000000000000000000000000001101
00000000000000000000000000001110
00000000000000000000000000001111
00000000000000000000000000010000
00000000000000000000000000010001
00000000000000000000000000010010
00000000000000000000000000010011
00000000000000000000000000010100
00000000000000000000000000010101
00000000000000000000000000010110
00000000000000000000000000010111
00000000000000000000000000011000
00000000000000000000000000011001
00000000000000000000000000011010
00000000000000000000000000011011
00000000000000000000000000011100
00000000000000000000000000011101
00000000000000000000000000011110
00000000000000000000000000011111
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
```