CSE 331 Computer Organization

Final Project – Single cycle MIPS with Structural Verilog

REPORT

151044062 - MUAZ KURT

R TYPES			[31:26] = 00 0000		[5:0]
Add	add	R	R[rd] = R[rs] + R[rt]	(1)	0/20 hex: 10 0000
Add Unsigned	addu	R	R[rd] = R[rs] + R[rt]		0/21 hex: 10 0001
And	and	R	R[rd] = R[rs] & R[rt]		0/24 hex: 10 0100
Nor	nor	R	R[rd] = ~ (R[rs] | R[rt])		0/27 hex: 10 0111
Or	or	R	R[rd] = R[rs] | R[rt]		0/25 hex 10 0101
Set Less Than	slt	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0		0/2a hex 10 1010
Set Less Than Unsig.	sltu	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0	(6)	0/2b hex 10 1011
Shift Left Logical	sll	R	R[rd] = R[rt] << shamt		0/00 hex 00 0000
Shift Right Logical	srl	R	R[rd] = R[rt] > > shamt		0/02 hex 00 0010
Subtract	sub	R	R[rd] = R[rs] - R[rt]	(1)	0/22 hex 10 0010
Subtract Unsigned	subu	R	R[rd] = R[rs] - R[rt]		0/23 hex 10 0011

I TYPES					[31: 26]
Add Immediate	addi	I	R[rt] = R[rs] + SignExtImm	(1,2)	8 hex: 00 1000
Add Imm. Unsigned	addiu	I	R[rt] = R[rs] + SignExtImm	(2)	9 hex: 00 1001
And Immediate	andi	I	R[rt] = R[rs] & ZeroExtImm	(3)	c hex: 00 1100
Branch On Equal	beq	I	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	(4)	4 hex: 00 0100
Load Word	lw	I	R[rt] = M[R[rs]+SignExtImm]	(2)	23 hex: 10 0011
Or Immediate	ori	I	R[rt] = R[rs] | ZeroExtImm	(3)	d hex: 00 1101
Store Word	sw	I	M[R[rs]+SignExtImm] = R[rt]	(2)	2b hex: 10 1011
Set Less Than Imm.	slti	I	R[rt] = (R[rs] < SignExtImm)? 1 : 0	(2)	a hex: 00 1010

J TYPES					
Jump	j	J	PC=JumpAddr	(5)	2 hex: 00 0010

CONTROL SIGNAL LIST

1. BRANCH ENABLE
2. ALU RESULT OR MEMORY RESULT
3. MEMORY WRITE ENABLE
4. MEMORY READ ENABLE
5. IMMEDIATE SELECT SIGNAL
6. SHIFT SELECT SIGNAL
7. RD/RT SIGNAL
8. REGISTER WRITE ENABLE
9. ZERO EXTEND SIGNAL
10. JUMP INSTR SIGNAL
11. SET LESS THAN SELECT SIGNAL
12. ALU CONTROL SIGNALS (3 BITS)

R Type Instructions Signals List | [31:26] → 00 0000

- Same Signals:
  - Branch enable signal: 0
  - ALU Result/Memory Result S: 0
  - Memory Read Enable S: 0
  - Memory Write Enable S: 0
  - Immediate Select Signal: 0
  - Rd/Rt signal: 0
  - Register Write enable: 1
  - Zero/Sign Extend signal: X
  - Jump Instr Signal: 0
- Separate Signals:
  - Add [5:0] → 10 0000
  - Add Unsigned [5:0] → 10 0001
    - ALU Control signals: ADD = 010
    - Set less than signal: 0
  - And [5:0] → 10 0100
    - ALU Control signals: AND = 000
    - Set less than signal: 0
  - Nor [5:0] → 10 0111
    - ALU Control signals: NOR = 111
    - Set less than signal: 0
  - Or [5:0] → 10 0101
    - ALU Control signals: OR = 001
    - Set less than signal: 0
  - Set Less Than [5:0] → 10 1010
  - Set Less Than Unsig. [5:0] → 10 1011
    - Set less than signal: 1
  - Subtract [5:0] → 10 0010
  - Subtract Unsigned [5:0] → 10 0011
    - ALU Control signals: SUB = 100
    - Set less than signal: 0
  - Shift Left Logical [5:0] → 00 0000
    - Shift Select Signal: 1
    - ALU Control signals: Shift Left = 110
    - Set less than signal: 0
  - Shift Right Logical [5:0] → 00 0010
    - Shift Select Signal: 1
    - ALU Control signals: Shift Right = 101
    - Set less than signal: 0

Immediate Operations Control Signals

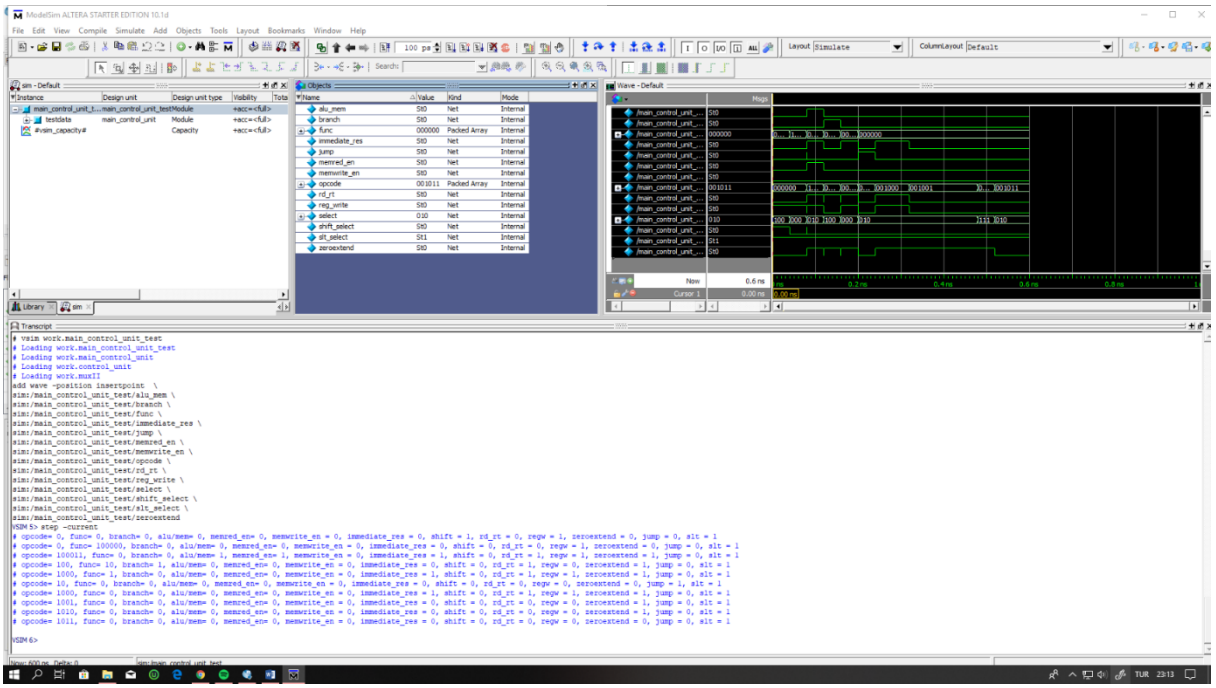
- Same Signals
  - Rd/Rt signal: 1
  - Shift Select Signal: 0
  - Jump Instr Signal: 0
  - Group 1 (Arithmetic Operations)
    - i. Same signals
      1. Branch enable signal: 0
      2. ALU Result/Memory Result S: 0
      3. Memory Read Enable S: 0
      4. Memory Write Enable S: 0
      5. Immediate Select Signal: 1
      6. Register Write enable: 1
    - ii. Separate Signals
      1. Add Immediate: 00 1000, 00 1001
         a. Zero/Sign Extend signal: 1
         b. Set less than signal: 0
         c. ALU Control signals: ADD = 010
      2. And Immediate: 00 1100
         a. Zero/Sign Extend signal: 0
         b. Set less than signal: 0
         c. ALU Control signals: AND = 000
      3. Or Immediate: 00 1101
         a. Zero/Sign Extend signal: 0
         b. Set less than signal: 0
         c. ALU Control signals: OR = 001
      4. Set Less Than Immediate: 00 1010
         a. Zero/Sign Extend signal: 1
         b. Set less than signal: 1
         c. ALU Control signals: SUB =100
  - Group 2 (Memory)
    - i. Same signals
      1. Branch enable signal: 0
      2. Immediate Select Signal: 1
      3. Zero/Sign Extend signal: 1
      4. Shift Select Signal: 0
      5. Set less than signal: 0
      6. ALU Control signals: ADD = 010
    - ii. Separate Signals
      1. Load Word: 10 0011
         a. ALU Result/Memory Result S: 1
         b. Memory Read Enable S: 1
         c. Memory Write Enable S: 0
         d. Register Write enable: 1
      2. Store Word: 10 1011

a.	ALU Result/Memory Result S:	X
b.	Memory Read Enable S:	0
c.	Memory Write Enable S:	1
d.	Register Write enable:	0

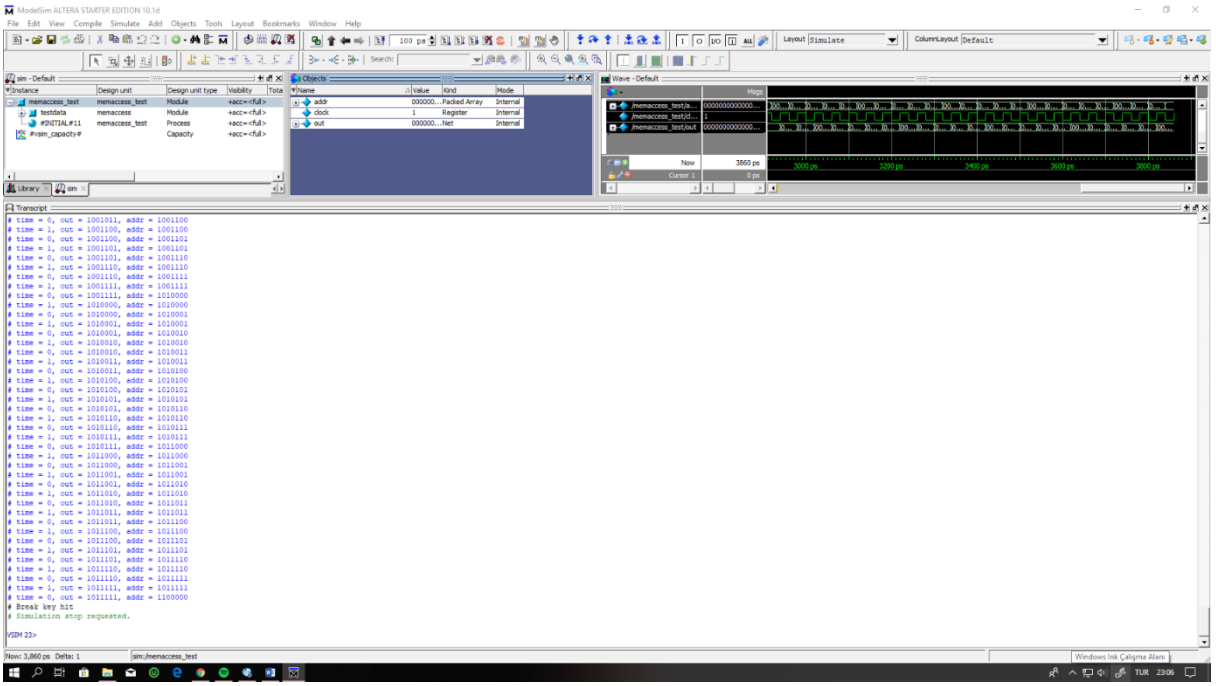
- Branch On Equal: 00 0100
    i. Branch enable signal: 1
    ii. ALU Result/Memory Result S: X
    iii. Memory Read Enable S: 0
    iv. Memory Write Enable S: 0
    v. Immediate Select Signal: 0
    vi. Register Write enable: 0
    vii. Zero/Sign Extend signal: X
    viii. Set less than signal: 0
    ix. ALU Control signals: SUB = 100

J Type Instructions Signal list

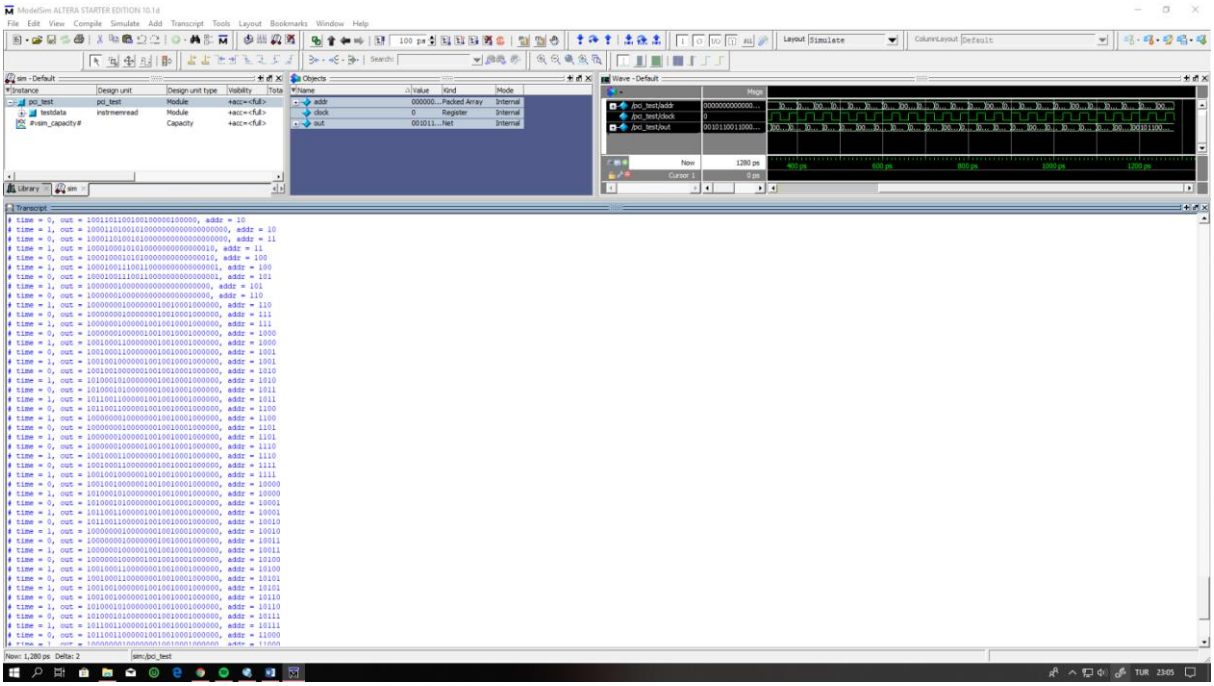
1. Jump Instruction: 00 0010
    a. Branch enable signal: 0
    b. ALU Result/Memory Result S: X
    c. Memory Read Enable S: 0
    d. Memory Write Enable S: 0
    e. Immediate Select Signal: X
    f. Shift Select Signal: X
    g. Rd/Rt signal: X
    h. Register Write enable: 0
    i. Zero/Sign Extend signal: X
    j. Jump Instr Signal: 1
    k. Set less than signal: X
    l. ALU Control signals: XXX



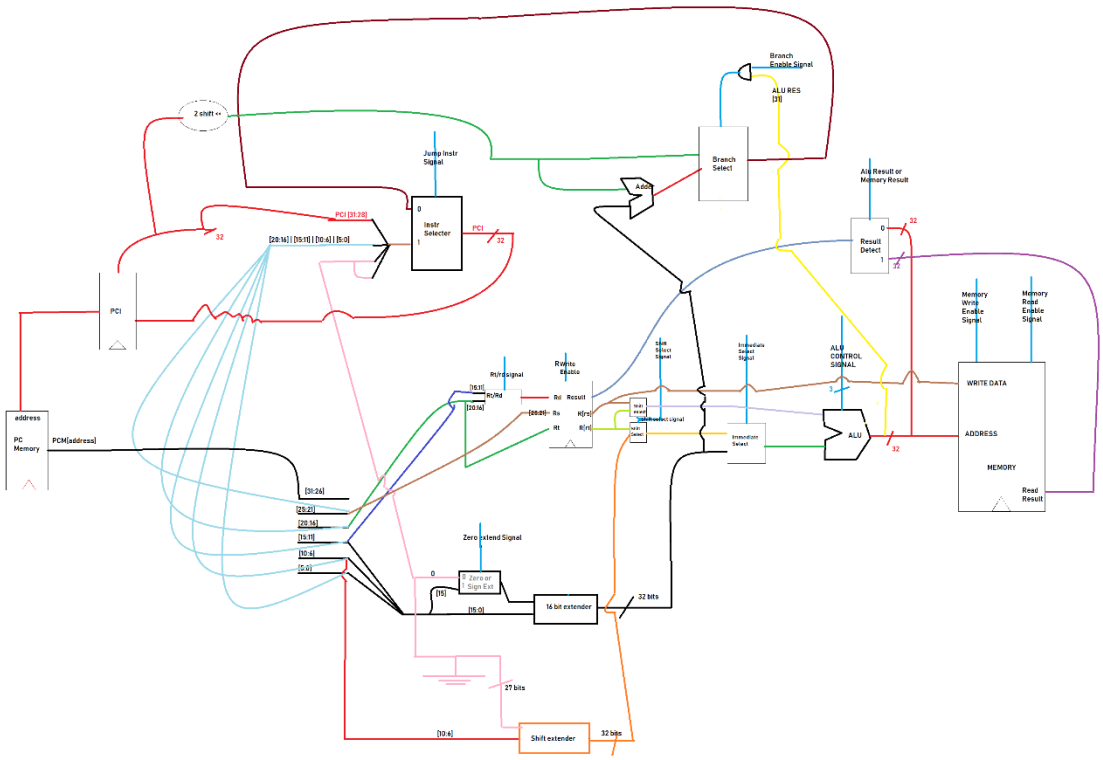
Main control signal results. All are acting as expected.



Can access all the memory one by one. Operation aborted cause of very high time cost and unnecessary modelling. There is 32768 memory blocks.



Instruction memory is accessible. All the memory can be read by pci module.



Design Schema.

Signal generating algebra:

BRANCH ENABLE :

!F5.!F4.!F3.F2.!F1.!F0

ALU R/MEM R :

MEM READ EN :

F5.!F4.!F3.!F2.F1.F0

MEM WRITE EN :

F5.!F4.F3.!F2.F1.F0

IMMEDIATE SELECT :

00 1000

00 1001

00 1100

00 1101

00 1010

10 0011

10 1011

++++++++++++

!F4.(!F5.(F3.!F1 + F3.!F2.!F0) + F5.!F2.F1.F0)

SHIFT SELECT :

00 0000 . 00 00X0

RD/RT :

00 1000

00 1001

00 1100

00 1101

00 1010

00 0100

10 0011

10 1011

++++++++++++

!F4.(!F5.(F2.!F1.!F0 + F3.!F1 + F3.!F2.!F0) + F5.!F2.F1.F0)

REG WRITE :

00 1000

00 1001

00 1100

00 1101

00 1010

10 0011

++++++++++++

!F4.(!F5.(!F2.!F1.!F0 + F3.!F1 + F3.!F2.!F0) + F5.!F3.!F2.F1.F0)

ZERO EXTEND :

10 0011

00 0100

00 1000

00 1001

00 1010

++++++++++++

!F4. (!F5. (F3.!F2.(!F1 + !F0) + !F3.F2.!F1.!F0) + F5.!F3.!F2.F1.F0)

JUMP :

!F5.!F4.!F3.!F2.F1.!F0

ALU CONTROL 2 :

00 0100

00 1010

++++++++++++

!F5.!F4.!F0.(!F3.F2.!F1 + F3.!F2.F1)

ALU CONTROL 1 :

00 100X

10 X011

++++++++++++

!F4.(!F5.F3.!F2.F1 + F5.!F2.F1.F0)

ALU CONTROL 0 :

00 1010

!F5.!F4.F3.!F2.F1.!F0

NOTE:

I CAN’T MAKE INSTRUCTIONS ONE BY ONE OR BRANCHING, BECAUSE I CANT ASSIGN THE NEXT INSTRUCTION ADDRESS TO PCI REGISTER.

I TRIED TO DO IT FOR 3 DAYS, BUT I COULDN’T MADE IT.

THANKS FOR READING 😊