CSE 331/503

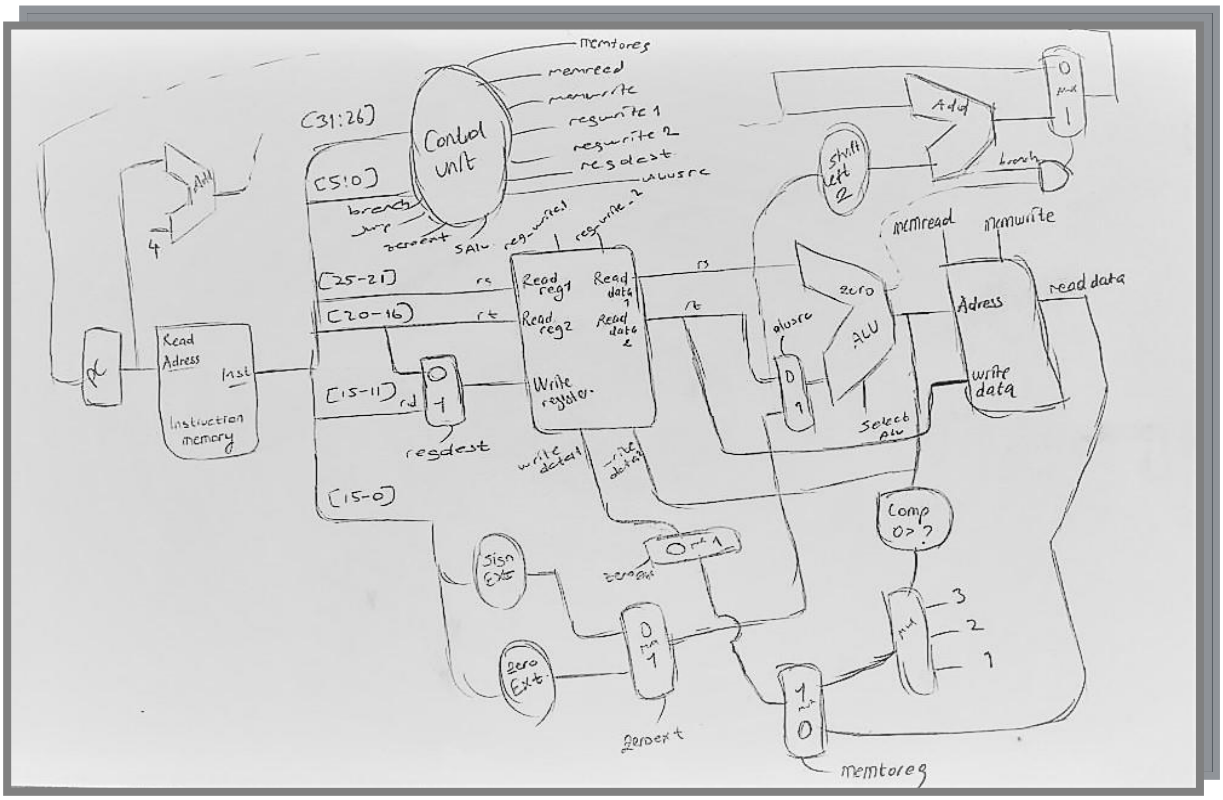
Computer Organization

Homework 4 Report

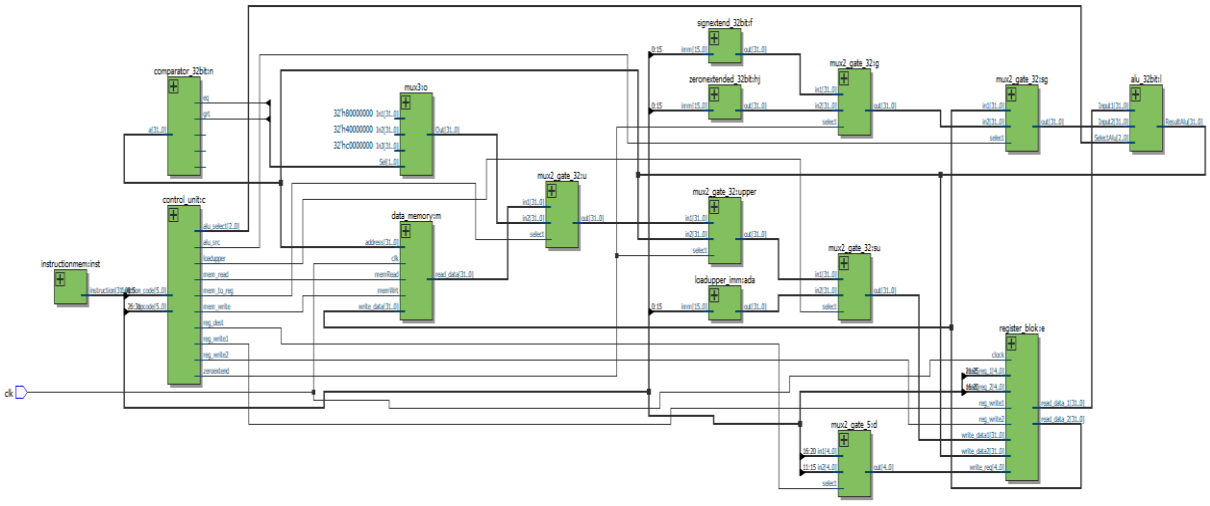
MUHAMMET FİKRET ATAR

1801042693

Single Cycle Datapath: I used following schema.



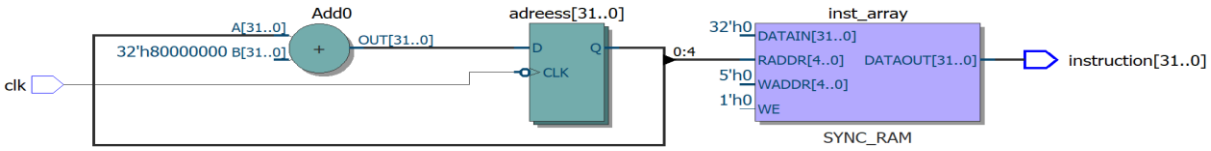
RTL View:



Modules

Instruction Memory:

I used instruction to keep instructions. Also it changes read adreess .



Control Unit:

I used following logical operations:

This table for type check;

	F.op5	F.op4	F.op3	F.op2	F.op1	F.op0	ALU0P1	ALUOP0
R TYPE	0	0	0	0	0	0	1	0
LW	1	0	0	0	1	1	0	0
SW	1	0	1	0	1	1	0	1
ORİ	0	0	1	1	0	1	1	1

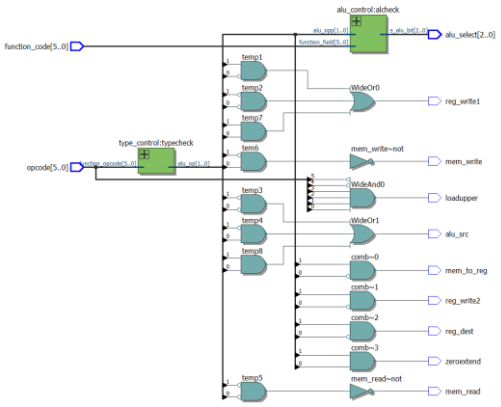
This table for select alu ;

	ALU0P1	ALU0P0	F.field5	F.field4	F.field3	F.field2	F.field1	F.field0	S.alu2	S.alu1	S.alu0
ADD	1	0	1	0	0	0	0	0	0	0	1
SUB	1	0	1	0	0	0	1	0	1	0	0
AND	1	0	1	0	0	1	0	0	0	0	0
OR	1	0	1	0	0	1	0	1	0	0	1
XOR	1	0	1	0	0	1	1	0	0	1	1
LW	0	0	*	*	*	*	*	*	0	1	0
SW	0	1	*	*	*	*	*	*	0	1	0
ORİ	1	1	*	*	*	*	*	*	0	0	1
LUİ	//	//	*	*	*	*	*	*	-	-	-

This table for Main Control Unit's Signals;

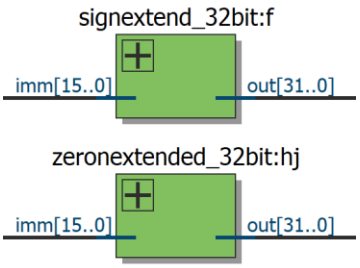
	N1	N0	memtoreg	Mem_read	Mem_write	Reg_write1	Reg_write2	Reg_dest	Alu_src	Zero_ext
Rtype	1	0	1	0	0	1	1	1	0	0
LW	0	0	0	1	0	1	0	0	1	0
SW	0	1	0	0	1	0	0	0	1	0
ORİ	1	1	0	0	0	1	0	0	1	0
LUİ	/	/	0	0	0	1	0	0	0	0

My control unit first performs type control and generates its own signal and the signals of alu's select bit according to the opcode signals.Also , alu control unit is working inside the control unit. You can see the input and output signals in the truth tables above.



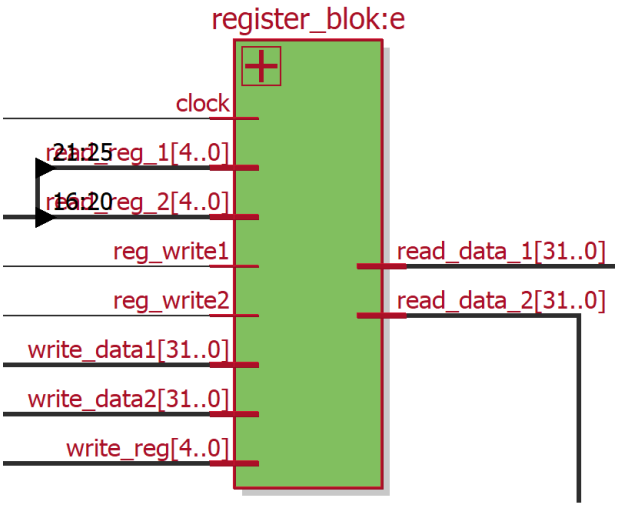
Extenders:

I used two extenders unit, These modules convert the immediate part to 32 bits.



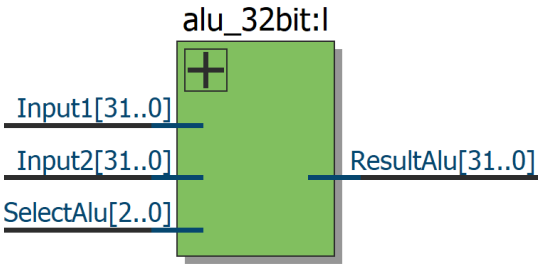
Register Modul:

In order to implement R-type, I had to write two different register addresses in one cycle design my register accordingly.



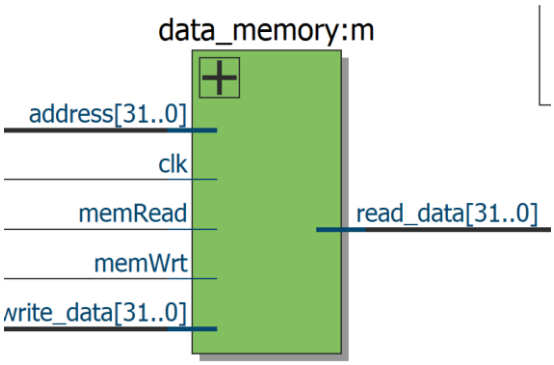
Alu:

Arithmetic operations take place in the alu module.



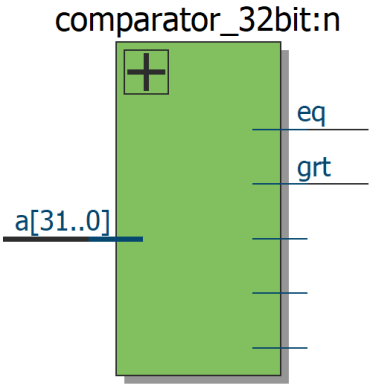
Data memory unit:

I used data memmory for read and write operation to datamem file.



Comparator:

I used comparator for R-Type InstructionS, then comparator selects 1,2,3 muxer's select bit in main modüle.



All instructions work correctly except j, jal, jr, beq, bne. Due to the trouble with my pc counter ,5 instruction does not work correctly. Following are the test of correct running instructions.

Tests;

ADDN : 000000_01000_01001_01010_00000_100000 (addn $r10, $r8, $r9)

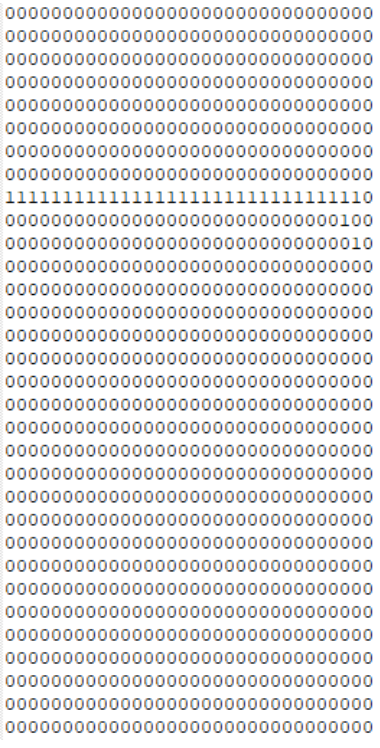
register.txt register.out.txt

ADDN: 000000_01000_01001_01010_00000_100000 (addn $r10, $r8, $r9)

SUBN: 000000_01000_01001_01010_00000_100010 (subn $r10, $r8, $r9)

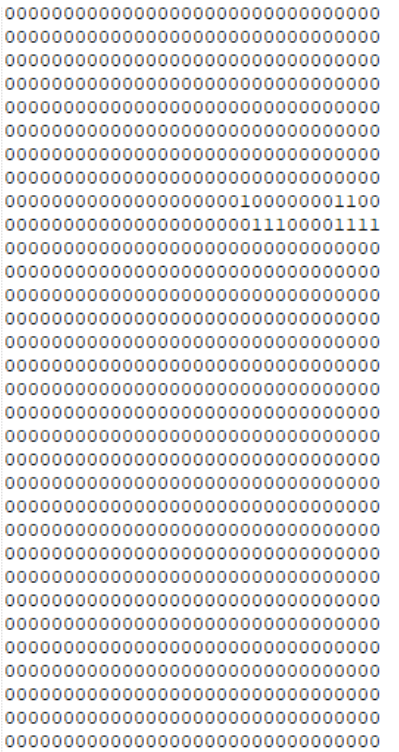
SUBN: 000000_01000_01001_01010_00000_100010 (subn $r10, $r8, $r9)

register.txt

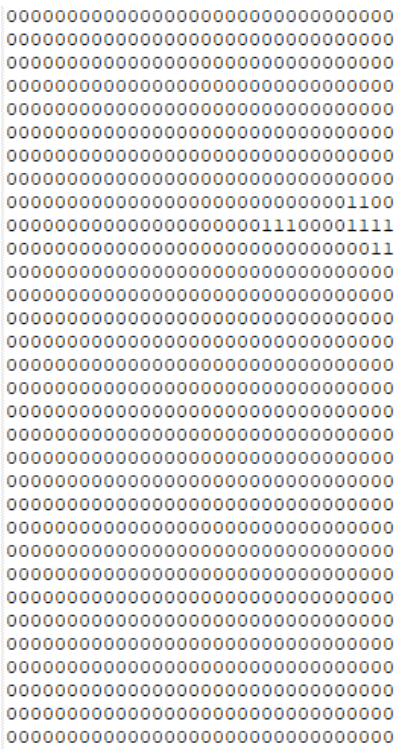


register.out.txt

ANDN: 000000_01000_01001_01010_00000_100100 (andn $r10, $r8, $r9)



register.txt



register.out.txt

ANDN: 000000_01000_01001_01010_00000_100100 (andn $r10, $r8, $r9)

register.txt register.out.txt

ORN: 000000_01000_01001_01010_00000_100101 (andn $r10, $r8, $r9)

register.txt register.out.txt

ORN: 000000_01000_01001_01010_00000_100101 (andn $r10, $r8, $r9)

register.txt register.out.txt

XORN: 000000_01000_01001_01010_00000_100110 (andn $r10, $r8, $r9)

register.txt register.out.txt

XORN: 000000_01000_01001_01010_00000_100110 (andn $r10, $r8, $r9)

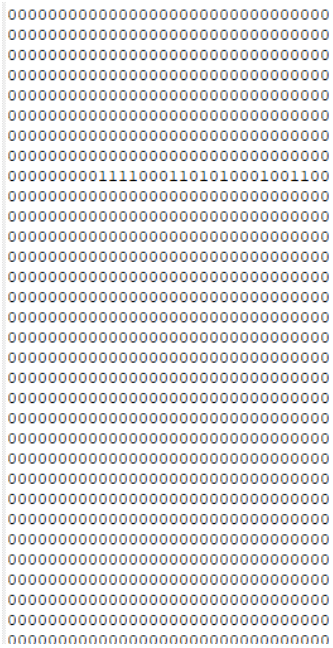
register.txt register.out.txt

LW: 100011_01001_01010_0000000000000100 (lw $10,14($0))

register.txt data.mem.txt register.out.txt

LW: 100011_01001_01010_0000000000000100 (lw $10,13($0))

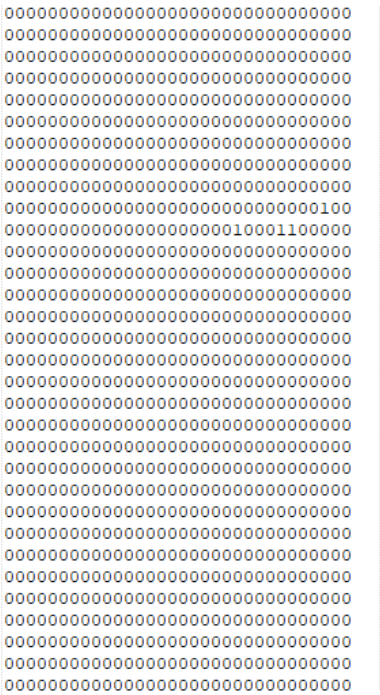
register.txt



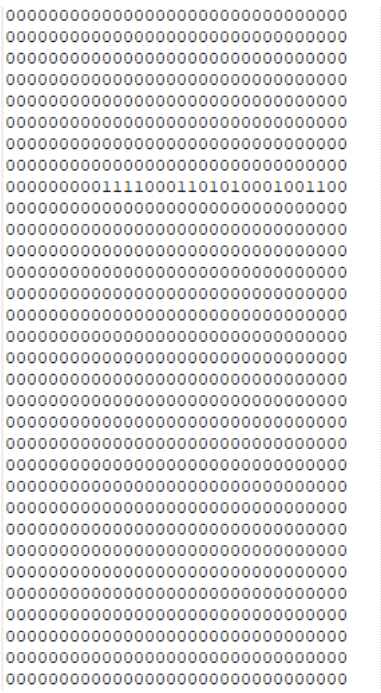
data.mem.txt

register.out.txt

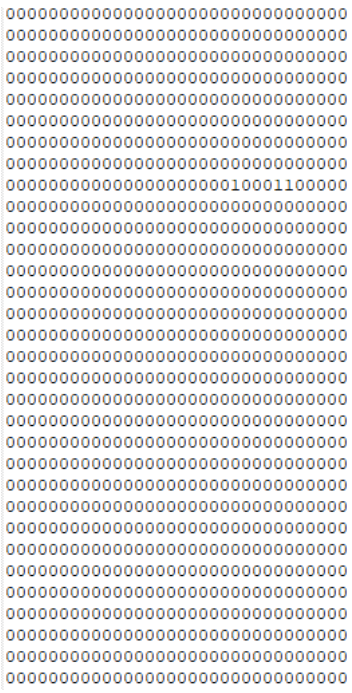
SW: 101011_01001_01010_0000000000000100 (sw $10,13($0))



register.txt

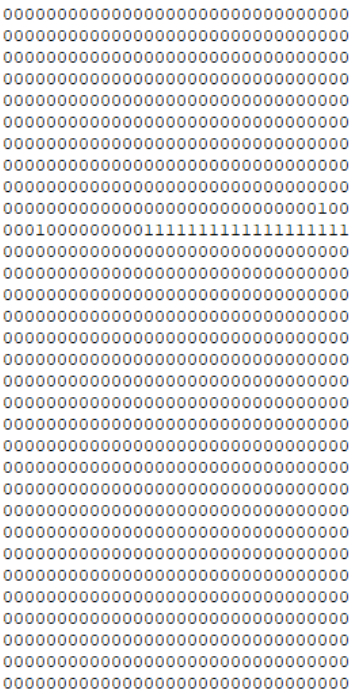


data.mem.txt

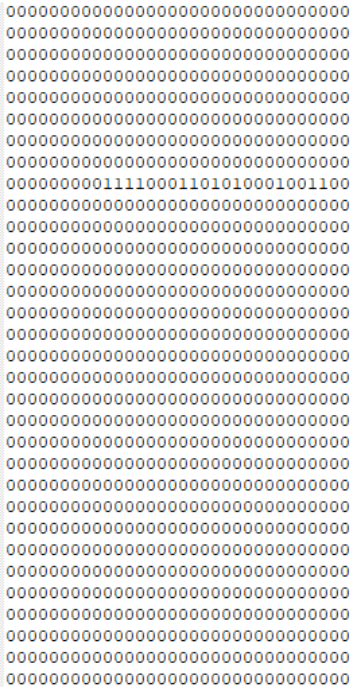


data.mem.out.txt

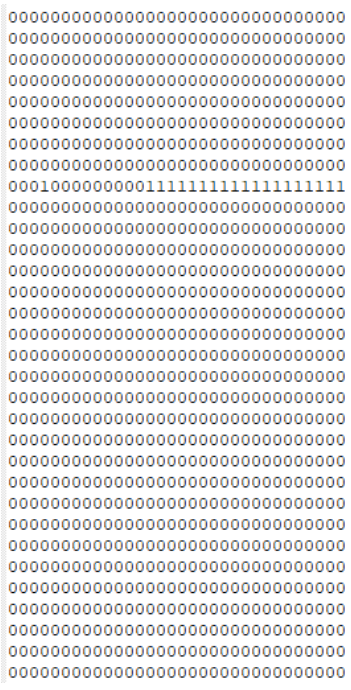
SW: 101011_01001_01010_0000000000000100 (sw $10,13($0))



register.txt



data.mem.txt



data.mem.out.txt

ORİ: 001101_01001_01010_0000000110011000 (ori $10,$9, 408)

register.txt

register.out.txt

ORİ: 001101_01001_01010_0000000000000000 (ori $10,$9,0)

```
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000001110001010010000101010
00000000000000000000000000100000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
```

register.txt

```
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000001110001010010000101010
00000000001110001010010000101010
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
```

register.out.txt

LUİ: 001111_01001_01010_0001000001101110(lui $t10, 4206)

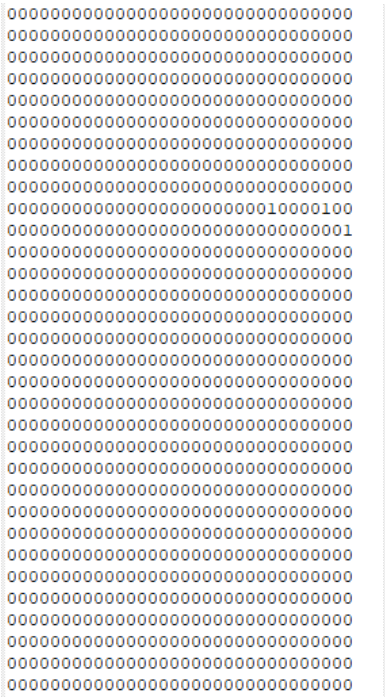
```
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000011001111001010
00000000000000000000000000100000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
```

register.txt

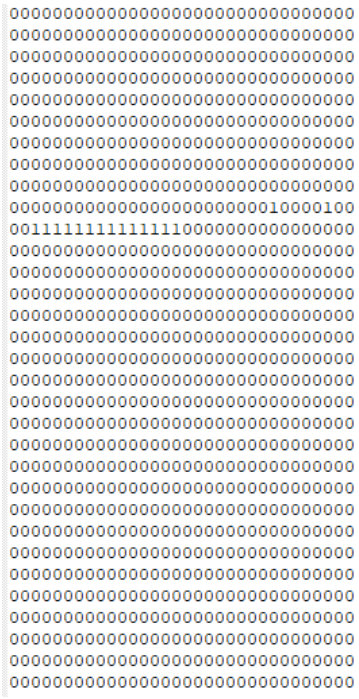
```
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000011001111001010
00010000011011100000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
```

register.out.txt

LUİ: 001111_01001_01010_0001000001101110(lui $t10, 4206)



register.txt



register.out.txt

```
# time =  0, inst=00000001000010010101000000100000
# time = 10, inst=00000001000010010101000000100010
# time = 20, inst=00000001000010010101000000100100
# time = 30, inst=00000001000010010101000000100101
# time = 40, inst=00000001000010010101000000100100
# time = 50, inst=00000001000010010101000000100110
# time = 60, inst=10001101001010100000000000000100
# time = 70, inst=00110101001010100000000000000100
# time = 80, inst=00110101001010100000000110011000
# time = 90, inst=00110101001010100000000000000000
# time = 100, inst=00111101001010100001000001101110
```

This instruction set tests R type, , lw, sw, ori,lui.

In this project, I used Altera Quartus II with Verilog. I will implement a different version of 32-bit MIPS processor. The block that I designed get no inputs from outside. I had two memories: Data Memory and Instruction Memory. The instructions must be loaded to the instruction memory and the data must be put in data memory. My processor supports lw, sw, , addn, subn, xorn, andn, orn, ori and lui instructions.

!!!!

My own computer file path is available for file read and write. You can change them as you run .

ASSIGNMENT 4 REPORT