CSE-331

COMPUTER ORGANIZATION

FINAL PROJECT

Kahraman Arda Kılıç

1901042648

1. Alu Control Functions Table

ALU CONTROL INPUT	FUNCTION	OPERATIONS
000	Add	ADD, ADDI, LW,SW
001	Xor	XOR
010	Sub	SUB, BEQ,BNE
011	Mult	
100	Slt	SLTI
101	Nor	NOR,NORI
110	And	AND,ANDI
111	Or	OR,ORI

- R-Type, LW, SW, BEQ, BNE, Addi, Andi, Ori, Nori, Slti -> 3 bits AluOP & 3 bits Function will be enough to create Alu Control signals which are 3 bits.

2. Truth Table for different Opcodes

Instr.	Reg Des	AluSrc	MemToReg	RegWr	MemRd	MemWr	Branch	BNE	AluOp2	AluOp1	AluOp0
R-Type	1	0	0	1	0	0	0	0	0	1	0
Lw	0	1	1	1	1	0	0	0	0	0	0
Sw	X	1	X	0	0	1	0	0	0	0	0
Beq	X	0	X	0	0	0	1	0	0	0	1
Bne	X	0	X	0	0	0	0	1	0	0	1
Addi	0	1	0	1	0	0	0	0	0	1	1
Andi	0	1	0	1	0	0	0	0	1	0	0
Ori	0	1	0	1	0	0	0	0	1	0	1
Nori	0	1	0	1	0	0	0	0	1	1	0
Slti	0	1	0	1	0	0	0	0	1	1	1

- R Signal is R-Type

a. Main Control Signals

- RegDes = R
- AluSrc = LW + SW + Addi + Andi + Ori + Nori + Slti
- MemToReg = LW
- RegWr = R + LW + Addi + Andi + Ori + Nori + Slti
- MemRd = LW
- MemWr = SW
- Branch = Beq
- BNE = Bne
- AluOp2 = Andi + Ori + Nori + Slti
- AluOp1 = R + Addi + Nori + Slti
- AluOp0 = Beq + Bne + Addi + Ori + Slti

3. Alu Control Signals Table

Instruction Opcode	P2P1P0 AluOp	F2 F1 F0 Function	Desired Alu Action	C2C1C0 Alu Control
LW	000	XXX	Add	000
SW	000	XXX	Add	000
BEQ	001	XXX	Subtract	010
BNE	001	XXX	Subtract	010
R-Type	010	000	Add	000
R-Type	010	001	And	110
R-Type	010	010	Sub	010
R-Type	010	011	Xor	001
R-Type	010	100	Nor	101
R-Type	010	101	Or	111
Addi	011	XXX	Add	000
Andi	100	XXX	And	110
Ori	101	XXX	Or	111
Nori	110	XXX	Nor	101
Slti	111	XXX	Set Less Than	100

Boolean Expressions:

- C2 = P2'P1P0'F2'F1'F0 + P2'P1P0'F2F1'F0' + P2'P1P0'F2F1'F0 + P2P1'P0' + P2P1'P0 + P2P1P0' + P2P1P0
- C1 = P2'P1'P0 + P2'P1P0'F2'F1'F0 + P2'P1P0'F2'F1F0' + P2'P1P0'F2F1'F0 + P2P1'P0' + P2P1'P0
- = P2'P1P0'F1'F0 + P2'P1P0'F2'F1F0' + P2P1' + P1'P0 (Simplified)
- C0 = P2'P1P0'F2'F1F0 + P2'P1P0'F2F1'F0' + P2'P1P0'F2F1'F0 + P2P1'P0 + P2P1P0'
- = P2'P1P0'F2'F1F0+P2'P1P0'F2F1'+P2P1'P0+P2P1P0' (Simplified)

The instructions in instructions.txt

```
0001010010001010 // addi $2 , $2,  10
0001001001000101 // addi $1 ,  $1, 5
0000010001011010// sub $3, $2 , $1
0000110100111010 //sub $7 , $6 , $4
0000110100111000 //and $7, $6, $4
0000101010001000 //and $1,$5,$2
0000101001010001 //add $2, $5 , $1
0000000010001001 //add $1, $0, $2
0010001100000011 //andi $4, $1, 000011
0010111001001100 //andi $1, $7, 001100
0000001100001011 //xor $1, $1, $4
0000001111001011 //xor $1, $1, $7
0101000000000011 //beq $0, $0, (go to next third line PC =  PC + 1
+ 3)
0000000000000000
0000000000000000
0000000000000000
0101001100000001 //beq $1, $4, (go to next first line PC = PC + 1
+ 1)
0011010001110011 //ori $1 , $2 , 110011
0011011010001010 //ori $2 , $3, 001010
1001110101000111 //sw $5 , 6(000111)
1001111111001010 //sw $7 , 7(001010)
0111010011000001 //slti $3, $2, 000001
0111000011000011 //slti $3, $0, 000011
0110001010000010 //bne $1, $2, 000010
0000000000000000
0000000000000000
0110000000001110 //bne $0, $0, 001110
0000010111101100 //nor $5, $7, $2
0000010010100100 //nor $4, $5, $5
0100100110010101 //nori $6, $4, 010101
0100110101101010 //nori $5, $6, 101010
0000110101100101 //or $4, $6, $5
0000100011001101 //or $1, $3, $4
1000011111000111 //lw $7, 3(000111)
1000000010000011 //lw $2, 0(000011)
```

- There are 64 instructions in the instructions memory but only the given program printed here.

4. EXPLANATION

- There is no Shift Left for the sign extended immediate value. PC increases by 1, therefore I did not use Shift Left. In the instruction, the immediate value will be the difference between the desired line to jump and the current line.
- Instruction width is 16 bits, the maximum number of instructions is 64.
- There is a BNE signal (Control's output) which controls the BNE operation.
- All files are located in altera\13.1\Final\simulation\modelsim
- There are 4 files. 3 of them are for input, the other one is for the output of the registers.
- Program Counter changes in the testbench. The MiniMIPS module gets an PC and outputs a new PC. The new PC will be the PC in the testbench.
- Program terminates after 64 instructions.

5. Test Cases

Instruction Memory Test

```
VSIM 53> vsim work.instruction_memory_testbench
# vsim work.instruction_memory_testbench
# Loading work.instruction_memory_testbench
# Loading work.instructions
VSIM 54> run
# instruction=0001010010001010, PC=00000000000000000000000000000000
# instruction=0001001001000101, PC=00000000000000000000000000000001
# instruction=0000010001011010, PC=00000000000000000000000000000010
# instruction=0001001001000101, PC=00000000000000000000000000000001
```

Data Memory Test

```
# vsim work.DataMemoryTestBench
# Loading work.DataMemoryTestBench
# Loading work.DataMemory
VSIM 56> run
# read_data=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, address=00000000000000000000000000000101 , write_data=00000000000011100001010001000000, memRead=0, memWrite=1
# read_data=00000000000000000000000000000110, address=00000000000000000000000000000110 , write_data=11111110011111111111110011111100, memRead=1, memWrite=0
```

Register Test

```
# vsim work.mips_registers_testbench
# Loading work.mips_registers_testbench
# Loading work.mips_registers
VSIM 58> run
# read_data1=00000000000000000000000000000001, read_data2=00000000000000000000000000000010, write_reg=100, signal_reg_write=1
# read_data1=11001110011111111111111111110000, read_data2=00000000000000000000000000000010, write_reg=101, signal_reg_write=1
```

Test For All Instructions

```
# Opcode: xxxx , R[s]: xxx , R[t]: xxx , Imm: xxxxxx (Branch: x or BNE: x and zero: x)
#  AluCtrl: xxx , AluInput1: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx , AluInput2: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#  AluResult: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#  WriteData: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#  PC: 00000000000000000000000000000000
#  newPC = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

The first line prints XXX because the clock finished 1 cycle yet.

ADDI

```
# Opcode: 0001 , R[s]: 010 , R[t]: 010 , Imm: 001010 (Branch: 0 or BNE: 0 and zero: 0)
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000000010 , AluInput2: 00000000000000000000000000001010
#  AluResult: 00000000000000000000000000001100
#  WriteData: 00000000000000000000000000001100
#  PC: 00000000000000000000000000000001
#  newPC = 00000000000000000000000000000010
#  *********
# Opcode: 0001 , R[s]: 001 , R[t]: 001 , Imm: 000101 (Branch: 0 or BNE: 0 and zero: 0)
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000000001 , AluInput2: 00000000000000000000000000000101
#  AluResult: 00000000000000000000000000000110
#  WriteData: 00000000000000000000000000000110
#  PC: 00000000000000000000000000000010
#  newPC = 00000000000000000000000000000011
```

SUB

```
# Opcode: 0000 , R[s]: 010 , R[t]: 001 , R[d]: 011 , Func: 010
#  AluCtrl: 010 , AluInput1: 00000000000000000000000000001100 , AluInput2: 00000000000000000000000000000110
#  AluResult: 00000000000000000000000000000110
#  WriteData: 00000000000000000000000000000110
#  PC: 00000000000000000000000000000011
#  *********
# Opcode: 0000 , R[s]: 110 , R[t]: 100 , R[d]: 111 , Func: 010
#  AluCtrl: 010 , AluInput1: 00000000000000000000000000000110 , AluInput2: 00000000000000000000000000000100
#  AluResult: 00000000000000000000000000000010
#  WriteData: 00000000000000000000000000000010
#  PC: 00000000000000000000000000000100
#  *********
```

AND

```
# Opcode: 0000 , R[s]: 110 , R[t]: 100 , R[d]: 111 , Func: 000
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000000110 , AluInput2: 00000000000000000000000000000100
#  AluResult: 00000000000000000000000000001010
#  WriteData: 00000000000000000000000000001010
#  PC: 00000000000000000000000000000101
#  *********
# Opcode: 0000 , R[s]: 101 , R[t]: 010 , R[d]: 001 , Func: 000
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000000101 , AluInput2: 00000000000000000000000000001100
#  AluResult: 00000000000000000000000000010001
#  WriteData: 00000000000000000000000000010001
#  PC: 00000000000000000000000000000110
#  *********
```

ADD

```
# Opcode: 0000 , R[s]: 101 , R[t]: 001 , R[d]: 010 , Func: 001
#  AluCtrl: 110 , AluInput1: 00000000000000000000000000000101 , AluInput2: 00000000000000000000000000010001
#  AluResult: 00000000000000000000000000000001
#  WriteData: 00000000000000000000000000000001
#  PC: 00000000000000000000000000000111
#  *********
# Opcode: 0000 , R[s]: 000 , R[t]: 010 , R[d]: 001 , Func: 001
#  AluCtrl: 110 , AluInput1: 00000000000000000000000000000000 , AluInput2: 00000000000000000000000000000001
#  AluResult: 00000000000000000000000000000000
#  WriteData: 00000000000000000000000000000000
#  PC: 00000000000000000000000000001000
#  *********
```

ANDİ

```
# Opcode: 0010 , R[s]: 001 , R[t]: 100 , Imm: 000011 (Branch: 0 or BNE: 0 and zero: 0)
#  AluCtrl: 110 , AluInput1: 00000000000000000000000000000000 , AluInput2: 00000000000000000000000000000011
#  AluResult: 00000000000000000000000000000000
#  WriteData: 00000000000000000000000000000000
#  PC: 00000000000000000000000000001001
#  newPC = 00000000000000000000000000001010
#  *********
# Opcode: 0010 , R[s]: 111 , R[t]: 001 , Imm: 001100 (Branch: 0 or BNE: 0 and zero: 0)
#  AluCtrl: 110 , AluInput1: 00000000000000000000000000001010 , AluInput2: 00000000000000000000000000001100
#  AluResult: 00000000000000000000000000001000
#  WriteData: 00000000000000000000000000001000
#  PC: 00000000000000000000000000001010
#  newPC = 00000000000000000000000000001011
```

XOR

```
# Opcode: 0000 , R[s]: 001 , R[t]: 100 , R[d]: 001 , Func: 011
#  AluCtrl: 001 , AluInput1: 00000000000000000000000000001000 , AluInput2: 00000000000000000000000000000000
#  AluResult: 00000000000000000000000000001000
#  WriteData: 00000000000000000000000000001000
#  PC: 00000000000000000000000000001011
#  *********
# Opcode: 0000 , R[s]: 001 , R[t]: 111 , R[d]: 001 , Func: 011
#  AluCtrl: 001 , AluInput1: 00000000000000000000000000001000 , AluInput2: 00000000000000000000000000001010
#  AluResult: 00000000000000000000000000000010
#  WriteData: 00000000000000000000000000000010
#  PC: 00000000000000000000000000001100
```

BEQ (There are 3 instructions between these 2 instructions. First beq instructions jumps to the second beq)

```
# Opcode: 0101 , R[s]: 000 , R[t]: 000 , Imm: 000011 (Branch: 1 or BNE: 0 and zero: 1)
#  AluCtrl: 010 , AluInput1: 00000000000000000000000000000000 , AluInput2: 00000000000000000000000000000000
#  AluResult: 00000000000000000000000000000000
#  WriteData: 00000000000000000000000000000000
#  PC: 00000000000000000000000000010000
#  newPC = 00000000000000000000000000010100
#  *********
# Opcode: 0101 , R[s]: 001 , R[t]: 100 , Imm: 000001 (Branch: 1 or BNE: 0 and zero: 0)
#  AluCtrl: 010 , AluInput1: 00000000000000000000000000000010 , AluInput2: 00000000000000000000000000000000
#  AluResult: 00000000000000000000000000000010
#  WriteData: 00000000000000000000000000000010
#  PC: 00000000000000000000000000010001
#  newPC = 00000000000000000000000000010010
#  *********
```

ORİ

```
# Opcode: 0011 , R[s]: 010 , R[t]: 001 , Imm: 110011 (Branch: 0 or BNE: 0 and zero: 0)
#  AluCtrl: 101 , AluInput1: 00000000000000000000000000000001 , AluInput2: 11111111111111111111111111110011
#  AluResult: 00000000000000000000000000001100
#  WriteData: 00000000000000000000000000001100
#  PC: 00000000000000000000000000010010
#  newPC = 00000000000000000000000000010011
#  *********
# Opcode: 0011 , R[s]: 011 , R[t]: 010 , Imm: 001010 (Branch: 0 or BNE: 0 and zero: 0)
#  AluCtrl: 101 , AluInput1: 00000000000000000000000000000110 , AluInput2: 00000000000000000000000000001010
#  AluResult: 11111111111111111111111111110001
#  WriteData: 11111111111111111111111111110001
#  PC: 00000000000000000000000000010011
#  newPC = 00000000000000000000000000010100
#  *********
```

SW

```
# Opcode: 1001 , R[s]: 110 , R[t]: 101 , Imm: 000111 (Branch: 0 or BNE: 0 and zero: 0)
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000000110 , AluInput2: 00000000000000000000000000000111
#  AluResult: 00000000000000000000000000001101
#  WriteData: 00000000000000000000000000001101
#  PC: 00000000000000000000000000010100
#  newPC = 00000000000000000000000000010101
#  *********
# Opcode: 1001 , R[s]: 111 , R[t]: 111 , Imm: 001010 (Branch: 0 or BNE: 0 and zero: 1)
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000001010 , AluInput2: 00000000000000000000000000001010
#  AluResult: 00000000000000000000000000010100
#  WriteData: 00000000000000000000000000010100
#  PC: 00000000000000000000000000010101
#  newPC = 00000000000000000000000000010110
```

SLTI

```
# Opcode: 0111 , R[s]: 010 , R[t]: 011 , Imm: 000001 (Branch: 0 or BNE: 0 and zero: 0)
#  AluCtrl: 100 , AluInput1: 11111111111111111111111111110001 , AluInput2: 00000000000000000000000000000001
#  AluResult: 00000000000000000000000000000001
#  WriteData: 00000000000000000000000000000001
#  PC: 00000000000000000000000000010110
#  newPC = 00000000000000000000000000010111
#  *********
# Opcode: 0111 , R[s]: 000 , R[t]: 011 , Imm: 000011 (Branch: 0 or BNE: 0 and zero: 0)
#  AluCtrl: 100 , AluInput1: 00000000000000000000000000000000 , AluInput2: 00000000000000000000000000000011
#  AluResult: 00000000000000000000000000000001
#  WriteData: 00000000000000000000000000000001
#  PC: 00000000000000000000000000010111
#  newPC = 00000000000000000000000000011000
#  *********
```

BNE

```
# Opcode: 0110 , R[s]: 001 , R[t]: 010 , Imm: 000010 (Branch: 0 or BNE: 1 and zero: 0)
#  AluCtrl: 010 , AluInput1: 00000000000000000000000000001100 , AluInput2: 11111111111111111111111111110001
#  AluResult: 00000000000000000000000000011011
#  WriteData: 00000000000000000000000000011011
#  PC: 00000000000000000000000000011010
#  newPC = 00000000000000000000000000011101
#  *********
# Opcode: 0110 , R[s]: 000 , R[t]: 000 , Imm: 001110 (Branch: 0 or BNE: 1 and zero: 1)
#  AluCtrl: 010 , AluInput1: 00000000000000000000000000000000 , AluInput2: 00000000000000000000000000000000
#  AluResult: 00000000000000000000000000000000
#  WriteData: 00000000000000000000000000000000
#  PC: 00000000000000000000000000011011
#  newPC = 00000000000000000000000000011100
```

NOR

```
# Opcode: 0000 , R[s]: 010 , R[t]: 111 , R[d]: 101 , Func: 100
#  AluCtrl: 001 , AluInput1: 11111111111111111111111111110001 , AluInput2: 00000000000000000000000000001010
#  AluResult: 11111111111111111111111111111011
#  WriteData: 11111111111111111111111111111011
#  PC: 00000000000000000000000000011100
#  *********
# Opcode: 0000 , R[s]: 010 , R[t]: 010 , R[d]: 100 , Func: 100
#  AluCtrl: 001 , AluInput1: 11111111111111111111111111110001 , AluInput2: 11111111111111111111111111110001
#  AluResult: 00000000000000000000000000000000
#  WriteData: 00000000000000000000000000000000
#  PC: 00000000000000000000000000011101
#  *********
```

NORİ

```
# Opcode: 0100 , R[s]: 100 , R[t]: 110 , Imm: 010101 (Branch: 0 or BNE: 0 and zero: 0)
#  AluCtrl: 101 , AluInput1: 00000000000000000000000000000000 , AluInput2: 00000000000000000000000000010101
#  AluResult: 11111111111111111111111111101010
#  WriteData: 11111111111111111111111111101010
#  PC: 00000000000000000000000000011110
#  newPC = 00000000000000000000000000011111
#  *********
# Opcode: 0100 , R[s]: 110 , R[t]: 101 , Imm: 101010 (Branch: 0 or BNE: 0 and zero: 1)
#  AluCtrl: 101 , AluInput1: 11111111111111111111111111101010 , AluInput2: 11111111111111111111111111101010
#  AluResult: 00000000000000000000000000010101
#  WriteData: 00000000000000000000000000010101
#  PC: 00000000000000000000000000011111
#  newPC = 00000000000000000000000000100000
#  *********
```

OR

```
# Opcode: 0000 , R[s]: 110 , R[t]: 101 , R[d]: 100 , Func: 101
#  AluCtrl: 111 , AluInput1: 11111111111111111111111111101010 , AluInput2: 11111111111111111111111111111011
#  AluResult: 11111111111111111111111111111011
#  WriteData: 11111111111111111111111111111011
#  PC: 00000000000000000000000000100000
#  *********
# Opcode: 0000 , R[s]: 100 , R[t]: 011 , R[d]: 001 , Func: 101
#  AluCtrl: 111 , AluInput1: 11111111111111111111111111111011 , AluInput2: 00000000000000000000000000000001
#  AluResult: 11111111111111111111111111111011
#  WriteData: 11111111111111111111111111111011
#  PC: 00000000000000000000000000100001
#  *********
```

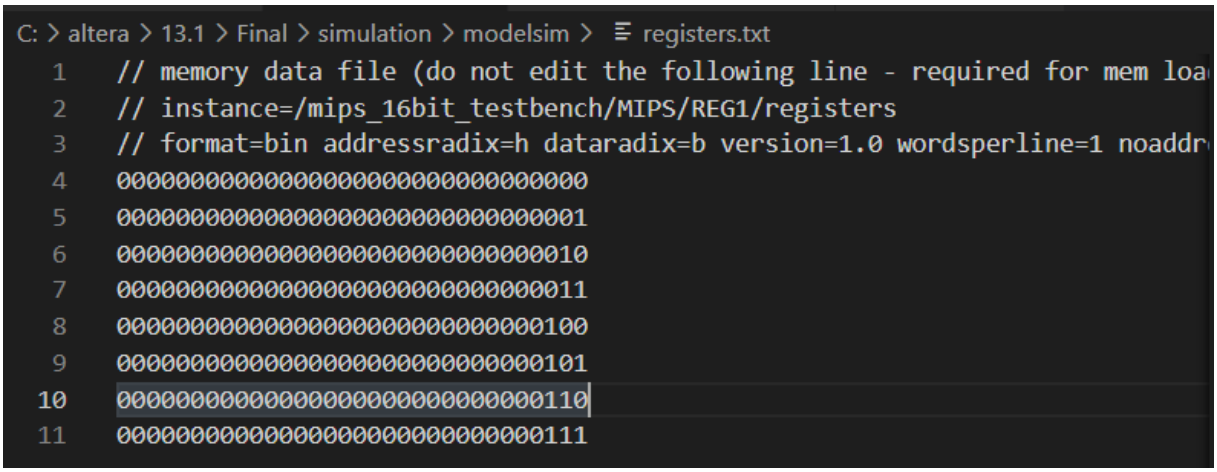
LW

```
# Opcode: 1000 , R[s]: 011 , R[t]: 111 , Imm: 000111 (Branch: 0 or BNE: 0 and zero: 0)
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000000001 , AluInput2: 00000000000000000000000000000111
#  AluResult: 00000000000000000000000000001000
#  WriteData: 00000000000000000000000000001000
#  PC: 00000000000000000000000000100010
#  newPC = 00000000000000000000000000100011
#  *********
# Opcode: 1000 , R[s]: 000 , R[t]: 010 , Imm: 000011 (Branch: 0 or BNE: 0 and zero: 0)
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000000000 , AluInput2: 00000000000000000000000000000011
#  AluResult: 00000000000000000000000000000011
#  WriteData: 00000000000000000000000000000011
#  PC: 00000000000000000000000000100011
#  newPC = 00000000000000000000000000100100
#  *********
```

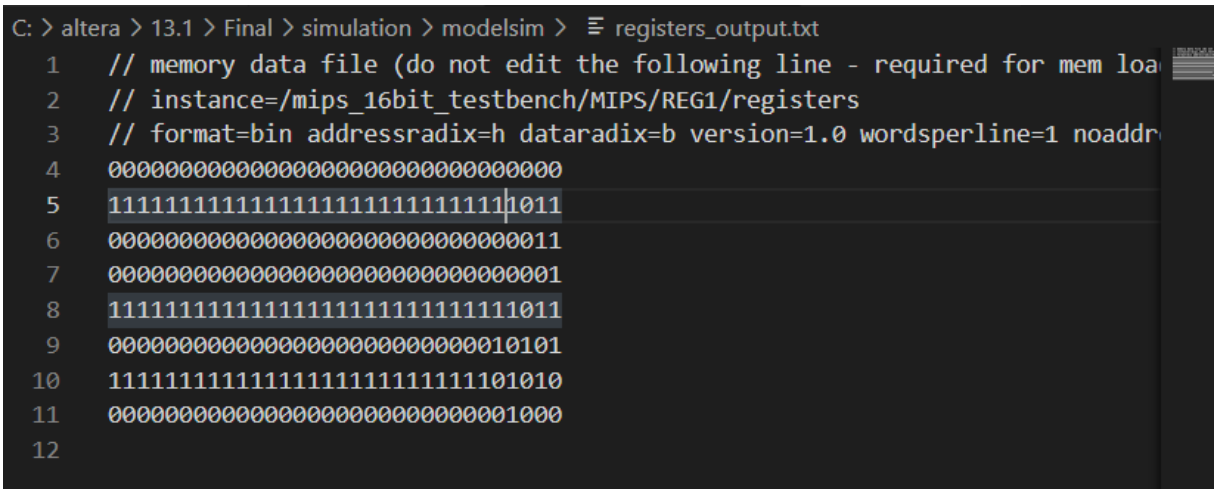
These nop instructions are the empty slots in the instructions memory.

```
# Opcode: 0000 , R[s]: 000 , R[t]: 000 , R[d]: 000 , Func: 000
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000000000 , AluInput2: 00000000000000000000000000000000
#  AluResult: 00000000000000000000000000000000
#  WriteData: 00000000000000000000000000000000
#  PC: 00000000000000000000000000100100
#  *********
# Opcode: 0000 , R[s]: 000 , R[t]: 000 , R[d]: 000 , Func: 000
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000000000 , AluInput2: 00000000000000000000000000000000
#  AluResult: 00000000000000000000000000000000
#  WriteData: 00000000000000000000000000000000
#  PC: 00000000000000000000000000100101
#  *********
# Opcode: 0000 , R[s]: 000 , R[t]: 000 , R[d]: 000 , Func: 000
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000000000 , AluInput2: 00000000000000000000000000000000
#  AluResult: 00000000000000000000000000000000
#  WriteData: 00000000000000000000000000000000
#  PC: 00000000000000000000000000100110
#  *********
# Opcode: 0000 , R[s]: 000 , R[t]: 000 , R[d]: 000 , Func: 000
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000000000 , AluInput2: 00000000000000000000000000000000
#  AluResult: 00000000000000000000000000000000
#  WriteData: 00000000000000000000000000000000
#  PC: 00000000000000000000000000100111
#  *********
# Opcode: 0000 , R[s]: 000 , R[t]: 000 , R[d]: 000 , Func: 000
#  AluCtrl: 000 , AluInput1: 00000000000000000000000000000000 , AluInput2: 00000000000000000000000000000000
#  AluResult: 00000000000000000000000000000000
#  WriteData: 00000000000000000000000000000000
#  PC: 00000000000000000000000000101000
#  *********
```

First Version of Registers



Final Version of Registers



- The other modules which are used in the previous homework are tested before.