CSE 3015 – DIGITAL LOGIC DESIGN

TERM PROJECT

Iteration 3

Authors:

1) 150117905 Erhan Yalnız

2) 150118013 Gökberk Çelikmasat

3) 150118065 Şeref Kutay Akgün

4) 150117005 Kadir Acun

INSTRUCTION SET ARCHITECTURE

Instructure	[15]	[14:12]	[11:9]	[8:6]	[5:3]	[2:0] (opcode)
AND	0	000	SR2	SR1	DR	000
ADD	1	000	SR2	SR1	DR	000
ANDI	ımm7			SR1	DR	001
ADDI	ımm7			SR1	DR	010
CMP	0	000	000	SR1	SR2	011
LD	Address10				DR	100
ST	Address10				SR	101
JUMP	Address10				000	110
JA	Address10				100	111
JB	Address10				101	111
JE	Address10				110	111
JBE	Address10				111	111
JAE	Address10				000	111

The bits from [4:2] reserved for CF and ZF in JA, JB, JE, JBE and JAE.

IMPLEMENTATION DETAILS

Firstly, we arrange more meetings at the beginning of the project and we talk many times about analysis of requirements. After we decided what we will do, how we will design desired project correctly, we created out instruction set architecture table and determined required components on this table. Furthermore, we prepared an assembler by using Java Programming Language which converts an assembly code.

Secondly, we switched our design to the Logisim and we implement our instructure set architecture and datapath in this program. This datapath takes the given instructions from the instruction memory and by the signals provided from our control unit provides the right output to either data memory or the registers. To implement our datapath to Logisim, we have to use 5 core components which desired from us and also required components which we decided when doing implementation. These 5 core components are Register File, ALU, Jump Condition Checker, Comparator and Control Unit. We will explain them and other components we had defined to the system:

1) Full Adder

   Full adder takes 3 inputs and two of them are standard inputs and one of them is carry_in input.
   Gives the addition of these inputs as a result and gives a carry out if there is one.

2) Adder

   Adders take two 16 bits inputs. For this 2 input's each bit, adder connects 16 full adders to each other. After then, this component gives the addition of these 2 inputs as a result.

3) ALU

   ALU is an arithmetic logic unit. Arithmetic Logic Unit takes three input. 2 of these inputs comes from the registers and third input is an immediate value. Depending on the opcode of the instruction memory, we decide which one it choices from ADD, ADDI, AND and ANDI operation according to instruction set architecture table which we created before. Furthermore, there was an Alu_Control. Alu_control decides on what to choice between source register1 and immediate value. 15th bit decides which operation used, ADD or AND.

4) 1-Bit Register

We created this component for carry flag and zero flag control. In this register, there is a D flip flop and enable, clock, preset, clear inputs.

5) 16-Bit Register

This component has one 16-bit input and gives 16-bit output. In this component there is 16 D flip flop for each bit. Furthermore, it has enabled, clock, preset, clear inputs.

6) Register File

Register File, include 8 x 16Bit Register. It takes DR input, write register signal, input, Source 1, Source 2 as input.

If write register is enabled, given input is written in destination register. If write register is not enabled,

The value of Source Register 1 and Source Register 2 will be written in output.

7) Comparator

Comparator takes two 16 Bit input, and compare them.

If these two inputs are equal; Zero Flag is set to 1.

If Number1 is less than Number2; Carry Flag is set to 1.

If isCompare signal is 1; comparator does the operations above, otherwise it does not.

8) Jump Condition Checker

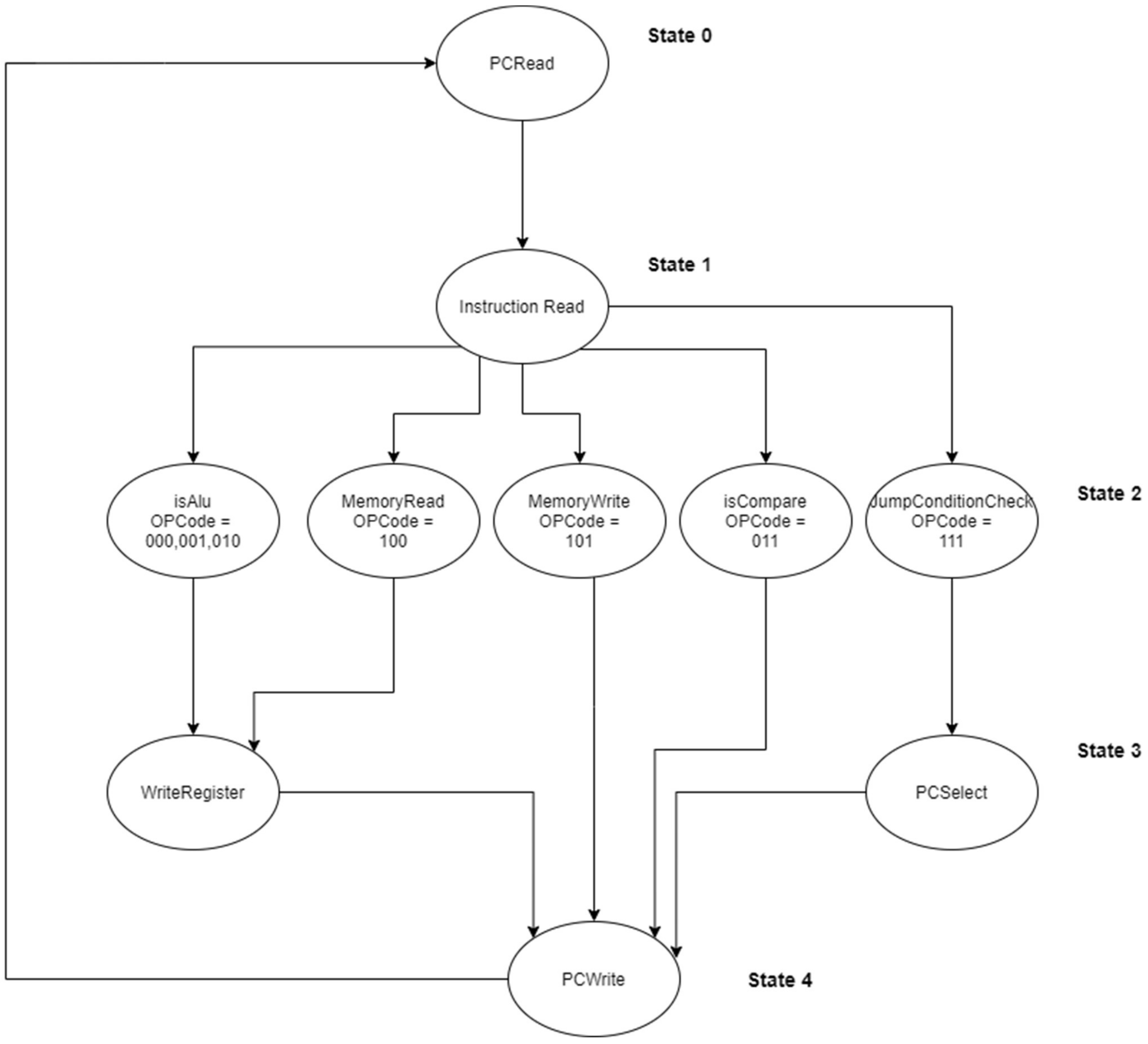
According to OpCode;

if OpCode equals “110”, it set to Jump Condition as 1,

if OpCode equals “111”, it decides which jump operation will be executed according to CondCheck input.

After that, according to ZF, CF, JCC inputs and corresponding ISA command, the jump condition is set to 1.

9) Control Unit



In control unit according to FSM table which is given above, states are implemented by counter. At the end, required signals are assigned accordingly.

10) CPU

By using all components, we defined before for the CPU and tunnels corresponding to these components, we created our datapath in CPU part. In addition, we added sections required for a control process unit. These extra sections are Instruction Memory, Data Memory, PC counter event. By using Instruction Memory, the operations which will be executed are sent to the Control Unit. The signals given by Control Unit, decides which instruction to execute and writes the corresponding output and pc is incremented accordingly.