Indian Institute of Technology Bombay

EE309 - Course Project

IITB - RISC

Anmol Saraf, 200070007
Shashank Balaji, 200070043
Mudit Goyal, 200070045
Tanaya Sonkul, 200070079

Preface

This project has been made by the cumulative effort of Anmol Saraf, Shashank Balaji, Mudit Goyal and Tanaya Sonkul. This project has been made using the knowledge gained through the course EE309: Microprocessors. In this project we have implemented a multi-cycle RISC microprocessor in VHDL language. The microprocessor is an 8 register, 16 bit computer system with 17 different instructions.

Acknowledgement

We are really grateful for this project opportunity and would sincerely thank Prof. Virendra Singh for guiding us through the course and the project as well. We would also like to show our gratitude to the teaching assistants for helping through out the course.

Abstract

The task of creating a RISC microprocessor entails creating a data path for the flow of the data around the processor. Furthermore, we designed all the logic for each instruction the microprocessor can process. Also, we made hardware circuitry such as ALUs, Register Bank, Incrementer, MUXes De-MUXs, Instruction Decoder, Sign Extenders for 9 to 16 bit and 6 to 16 bit, Control Word, which are all needed for the microprocessor to work. Lastly, we have made a controller which will send the signals to each hardware, telling it what to do in each instruction. And, to complete the microprocessor, we have also made a RAM which stores all the code for the processor and can all store data for it. Although, a 64kB RAM cannot be synthesized in Quartus due to its limitations, so we have used a 256B RAM instead.

Contents

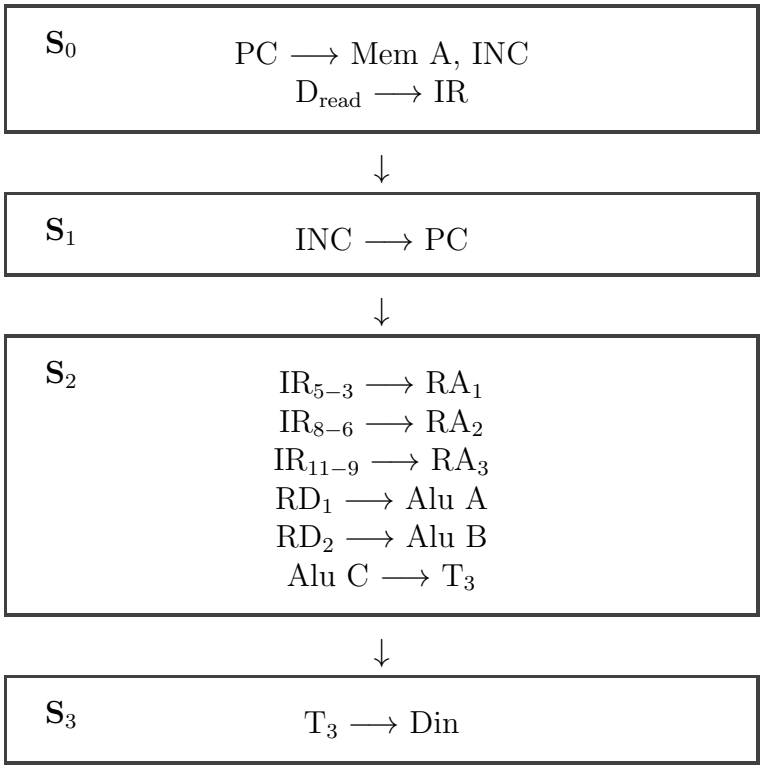
1 Introduction

The project's task was to create a multi-cycle RISC microprocessor. Thus, we were able to simulate our microprocessor on VHDL by implementing various hardwares such as ALUs, Register Bank, RAM, etc. And using the given Instruction Set Architecture(ISA) to write Hardware Flowcharts, desgin a data path and control word for the microprocessor.

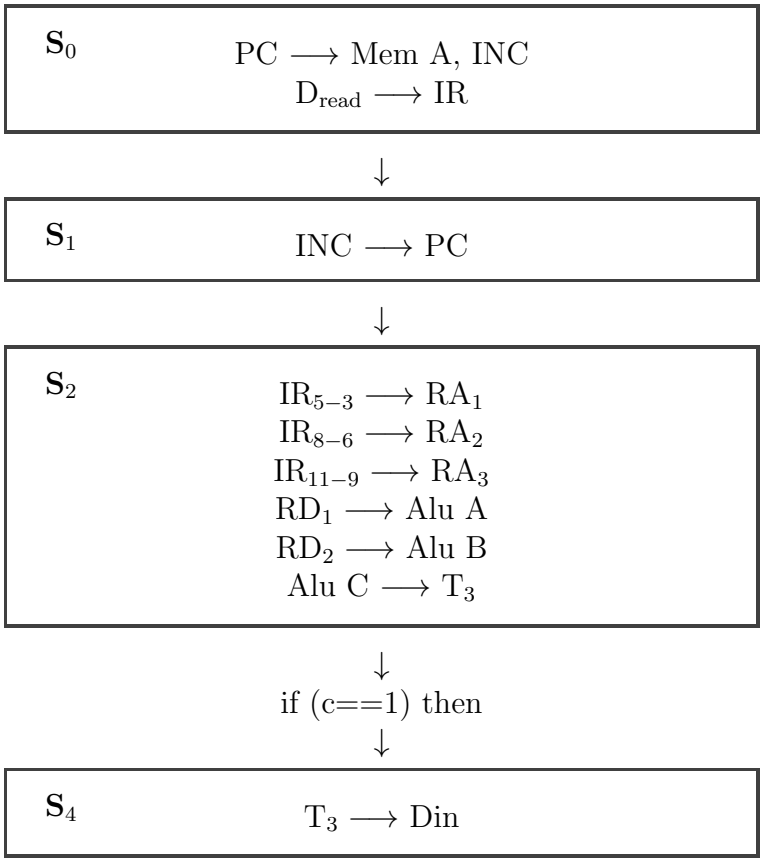
2 Hardware Flowcharts

Hardware flowcharts are constructed for each instruction given in the ISA. The hardware flowchart is made keeping in mind the data path of the microprocessor and simultaneously the data path is updated to account for all the hardware used in a instruction. The Hardware flowcharts for all the 17 instructions are as given as below,

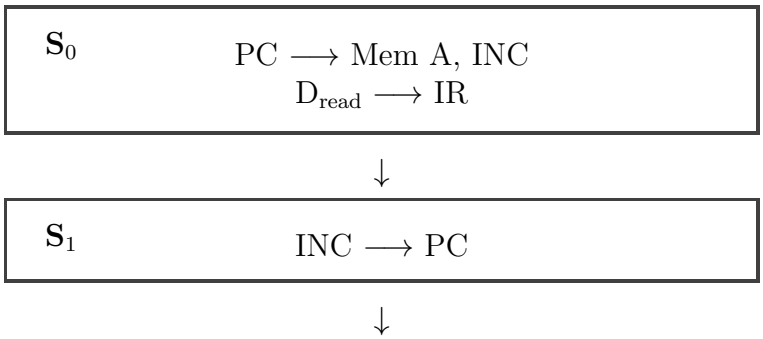
2.1 ADD

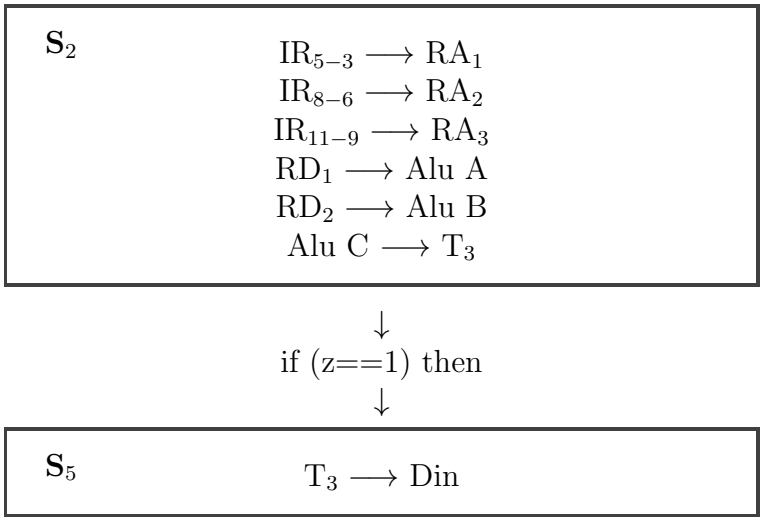


2.2 ADC

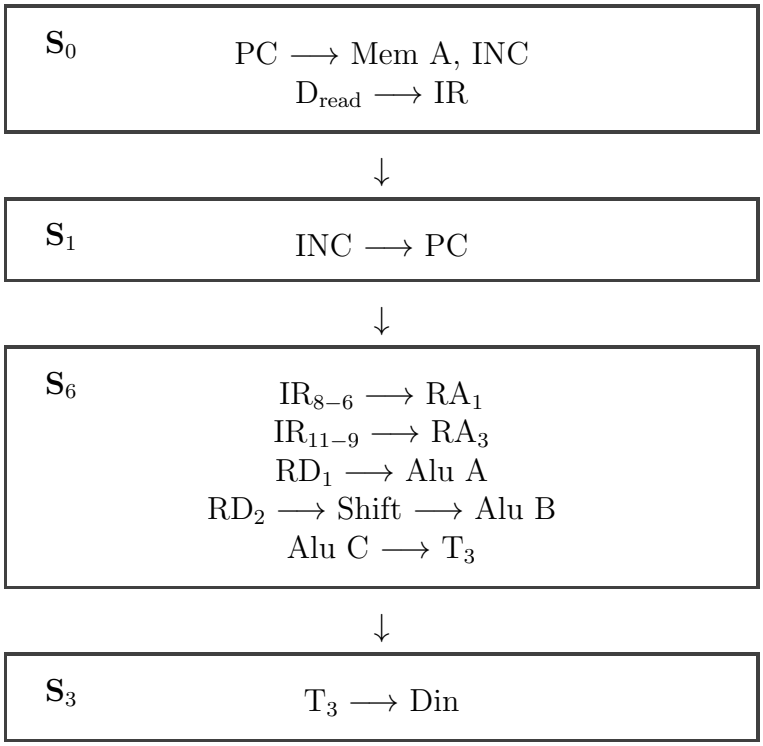

S0
PC ⟶ Mem A, INC
Dread ⟶ IR
S1
INC ⟶ PC
S2
IR5−3 ⟶ RA1
IR8−6 ⟶ RA2
IR11−9 ⟶ RA3
RD1 ⟶ Alu A
RD2 ⟶ Alu B
Alu C ⟶ T3
if (c==1) then
S4
T3 ⟶ Din

2.3 ADZ

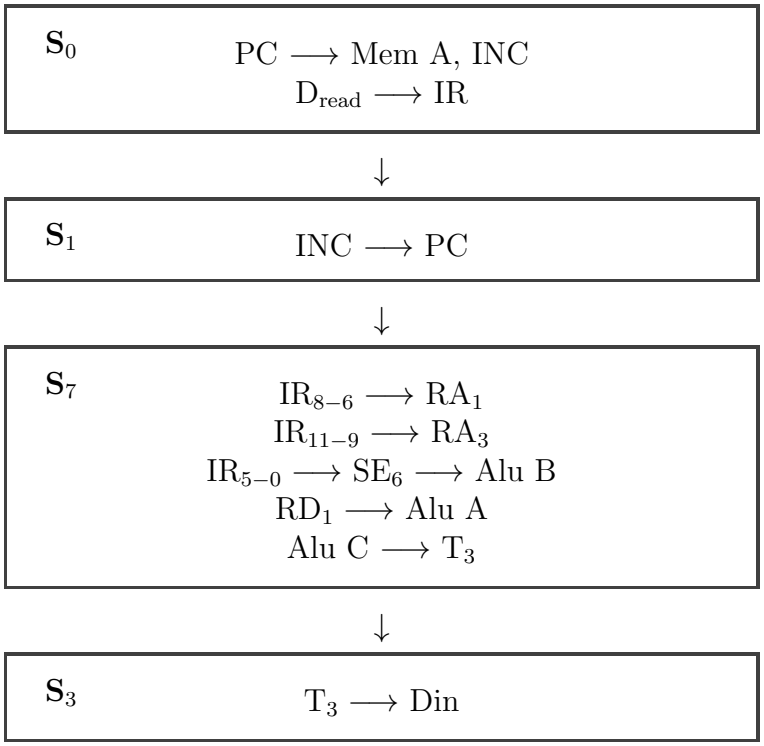

S0
PC ⟶ Mem A, INC
Dread ⟶ IR
S1
INC ⟶ PC


S2
IR5−3 ⟶ RA1
IR8−6 ⟶ RA2
IR11−9 ⟶ RA3
RD1 ⟶ Alu A
RD2 ⟶ Alu B
Alu C ⟶ T3
if (z==1) then
S5
T3 ⟶ Din

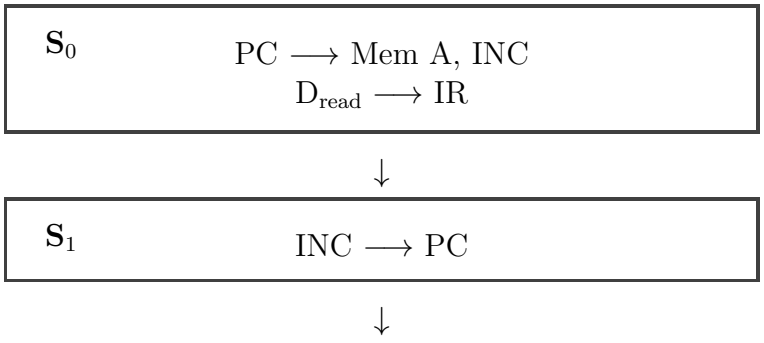
2.4 ADL

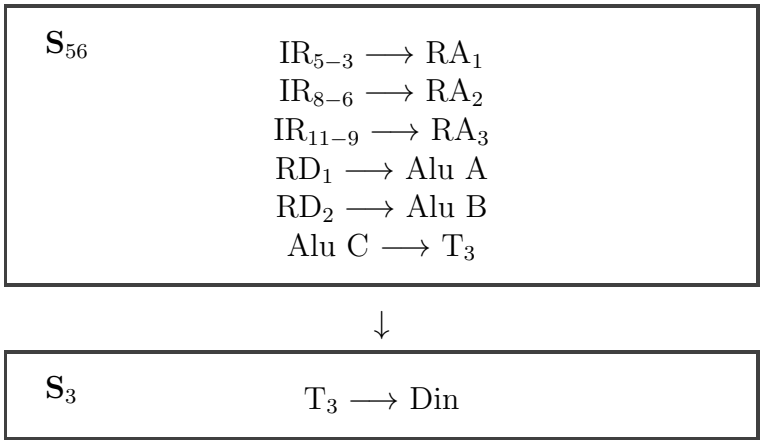

S0
PC ⟶ Mem A, INC
Dread ⟶ IR
S1
INC ⟶ PC
S6
IR8−6 ⟶ RA1
IR11−9 ⟶ RA3
RD1 ⟶ Alu A
RD2 ⟶ Shift ⟶ Alu B
Alu C ⟶ T3
S3
T3 ⟶ Din

2.5 ADI

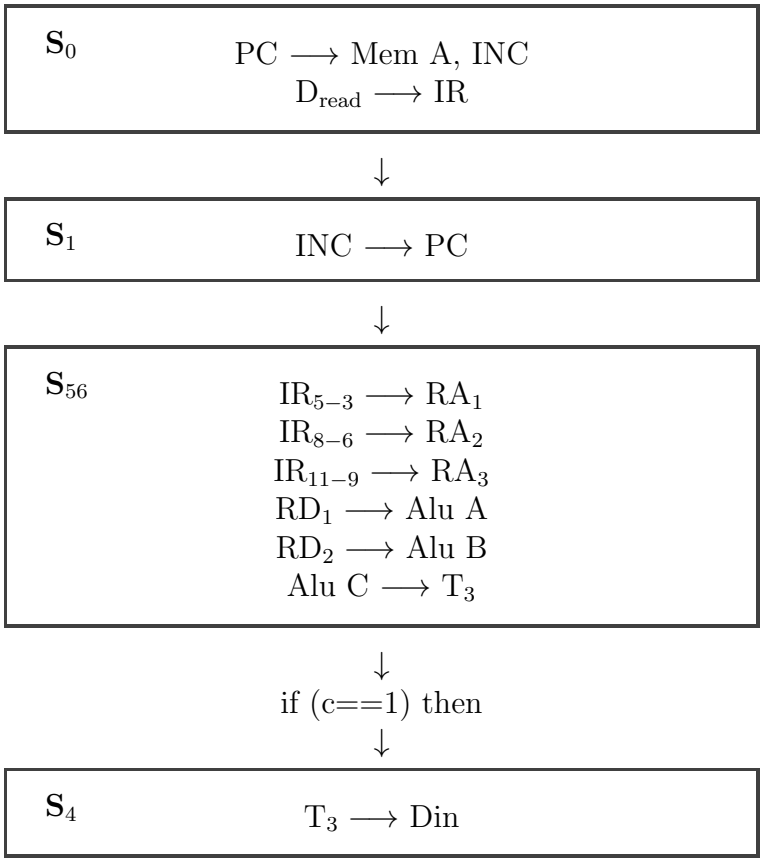

$\mathbf{S}_0$
$PC \longrightarrow$ Mem A, INC
$D_{read} \longrightarrow IR$
$\mathbf{S}_1$
$INC \longrightarrow PC$
$\mathbf{S}_7$
$IR_{8-6} \longrightarrow RA_1$
$IR_{11-9} \longrightarrow RA_3$
$IR_{5-0} \longrightarrow SE_6 \longrightarrow$ Alu B
$RD_1 \longrightarrow$ Alu A
Alu C $\longrightarrow T_3$
$\mathbf{S}_3$
$T_3 \longrightarrow$ Din

2.6 NDU

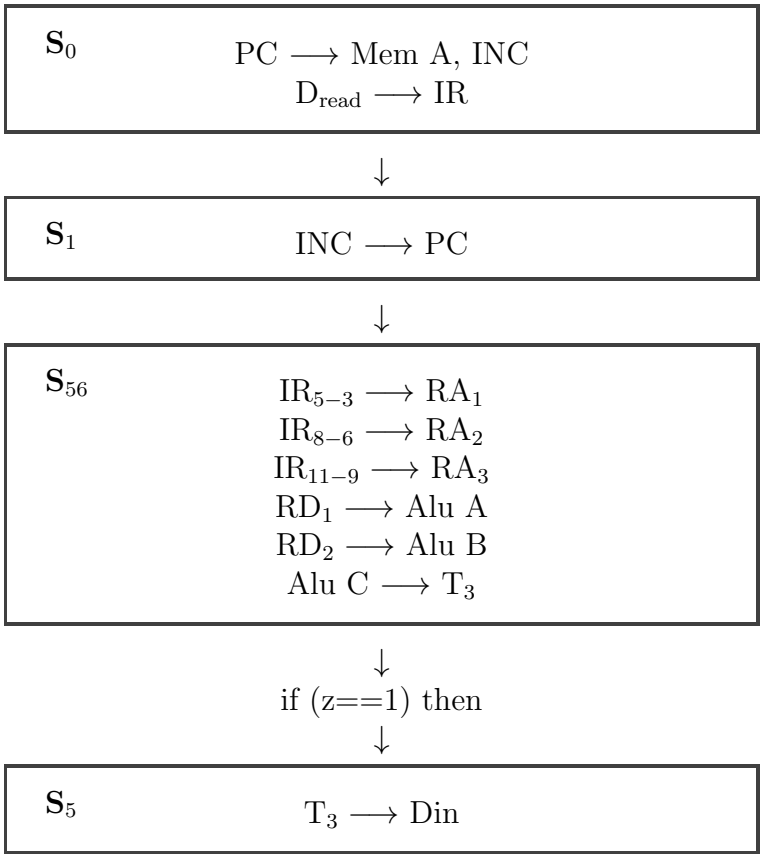

$\mathbf{S}_0$
$PC \longrightarrow$ Mem A, INC
$D_{read} \longrightarrow IR$
$\mathbf{S}_1$
$INC \longrightarrow PC$


S56
IR5−3 ⟶ RA1
IR8−6 ⟶ RA2
IR11−9 ⟶ RA3
RD1 ⟶ Alu A
RD2 ⟶ Alu B
Alu C ⟶ T3
S3
T3 ⟶ Din

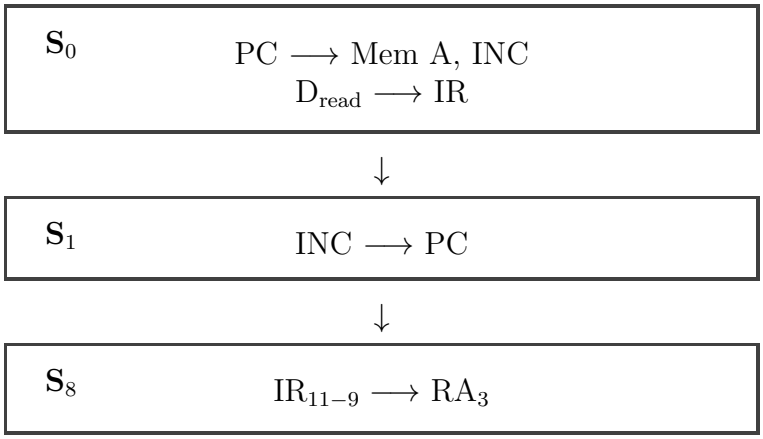
2.7 NDC


S0
PC ⟶ Mem A, INC
Dread ⟶ IR
S1
INC ⟶ PC
S56
IR5−3 ⟶ RA1
IR8−6 ⟶ RA2
IR11−9 ⟶ RA3
RD1 ⟶ Alu A
RD2 ⟶ Alu B
Alu C ⟶ T3
if (c==1) then
S4
T3 ⟶ Din

2.8 NDZ


S0
PC ⟶ Mem A, INC
Dread ⟶ IR
S1
INC ⟶ PC
S56
IR5−3 ⟶ RA1
IR8−6 ⟶ RA2
IR11−9 ⟶ RA3
RD1 ⟶ Alu A
RD2 ⟶ Alu B
Alu C ⟶ T3
if (z==1) then
S5
T3 ⟶ Din

2.9 LHI


S0
PC ⟶ Mem A, INC
Dread ⟶ IR
S1
INC ⟶ PC
S8
IR11−9 ⟶ RA3

↓

$\mathbf{S}_9$	$\mathrm{IR}_{8-0} \longrightarrow \mathrm{Shift} \longrightarrow \mathrm{Din}$

2.10 LW

$\mathbf{S}_0$	$\mathrm{PC} \longrightarrow \mathrm{Mem\ A,\ INC}$ $\mathrm{D_{read}} \longrightarrow \mathrm{IR}$

↓

$\mathbf{S}_1$	$\mathrm{INC} \longrightarrow \mathrm{PC}$

↓

$\mathbf{S}_{20}$	$\mathrm{IR}_{8-6} \longrightarrow \mathrm{RA}_1$ $\mathrm{IR}_{11-9} \longrightarrow \mathrm{RA}_3$ $\mathrm{IR}_{5-0} \longrightarrow \mathrm{Alu\ B}$ $\mathrm{RD}_1 \longrightarrow \mathrm{Alu\ A}$ $\mathrm{Alu\ C} \longrightarrow \mathrm{T}_3$

↓

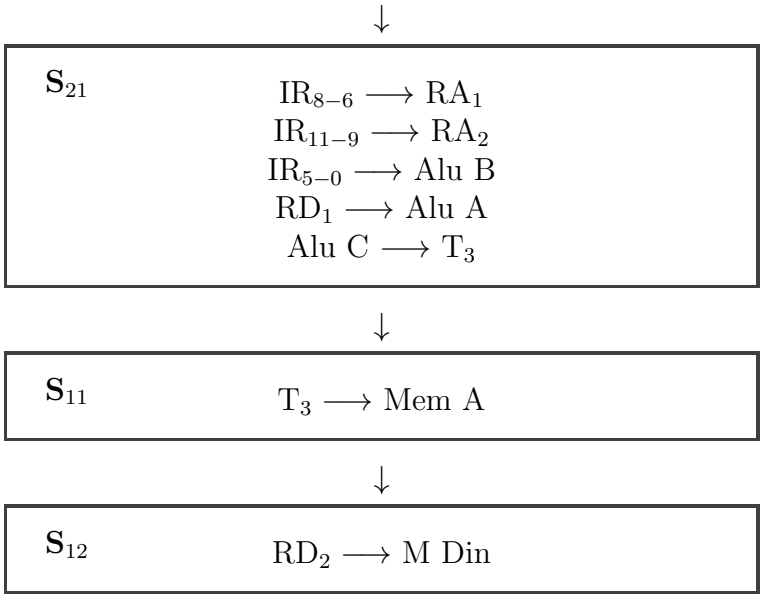
$\mathbf{S}_{10}$	$\mathrm{T}_3 \longrightarrow \mathrm{Mem\ A}$ $\mathrm{D_{read}} \longrightarrow \mathrm{Din}$

2.11 SW

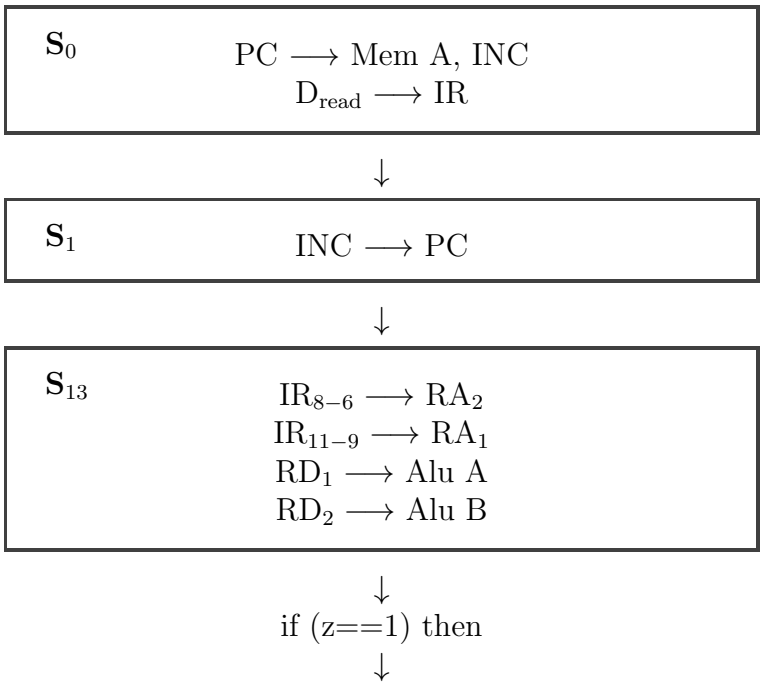
$\mathbf{S}_0$	$\mathrm{PC} \longrightarrow \mathrm{Mem\ A,\ INC}$ $\mathrm{D_{read}} \longrightarrow \mathrm{IR}$

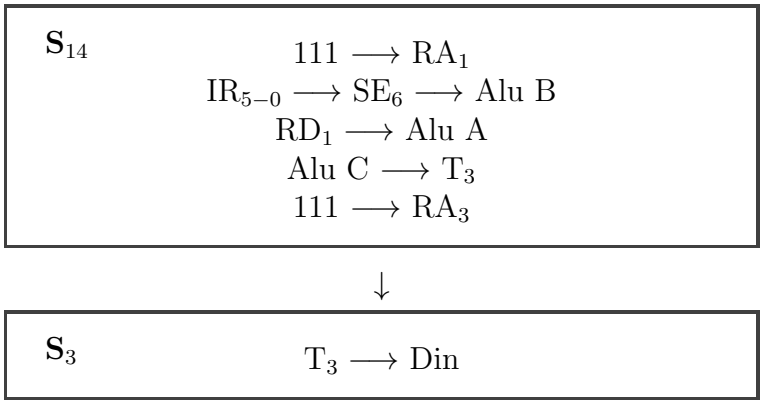
↓

$\mathbf{S}_1$	$\mathrm{INC} \longrightarrow \mathrm{PC}$

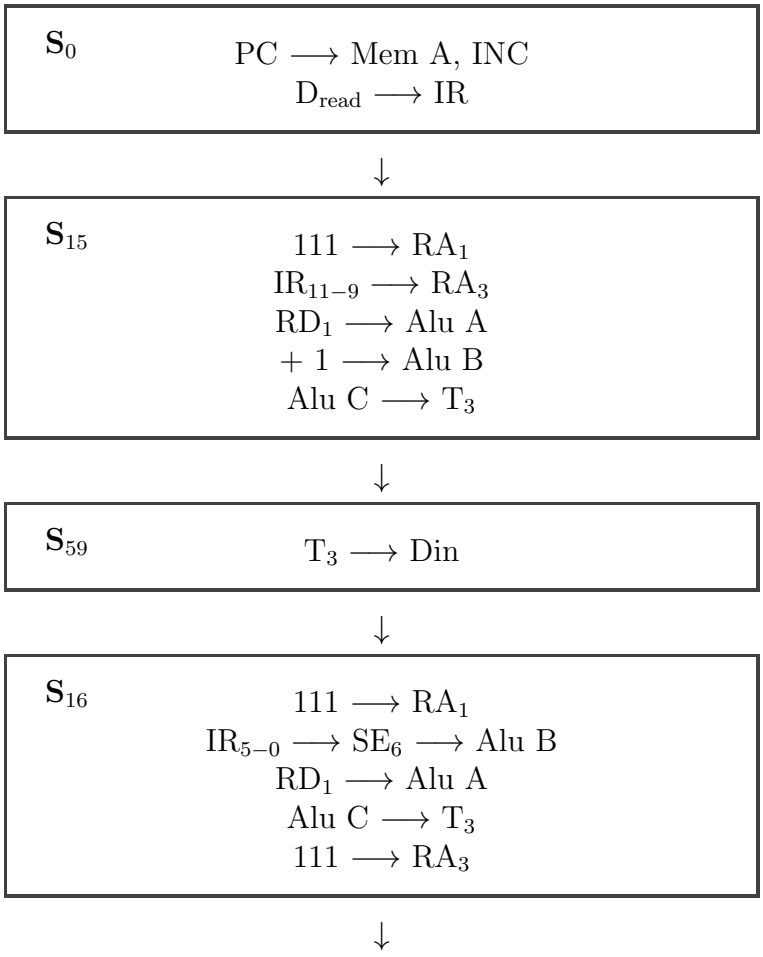

S_{21}
$IR_{8-6} \longrightarrow RA_1$
$IR_{11-9} \longrightarrow RA_2$
$IR_{5-0} \longrightarrow$ Alu B
$RD_1 \longrightarrow$ Alu A
Alu C $\longrightarrow T_3$
S_{11}
$T_3 \longrightarrow$ Mem A
S_{12}
$RD_2 \longrightarrow$ M Din

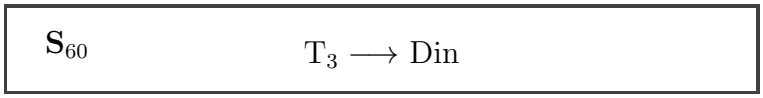
2.12 BEQ


S_0
PC $\longrightarrow$ Mem A, INC
$D_{read} \longrightarrow$ IR
S_1
INC $\longrightarrow$ PC
S_{13}
$IR_{8-6} \longrightarrow RA_2$
$IR_{11-9} \longrightarrow RA_1$
$RD_1 \longrightarrow$ Alu A
$RD_2 \longrightarrow$ Alu B
if (z==1) then

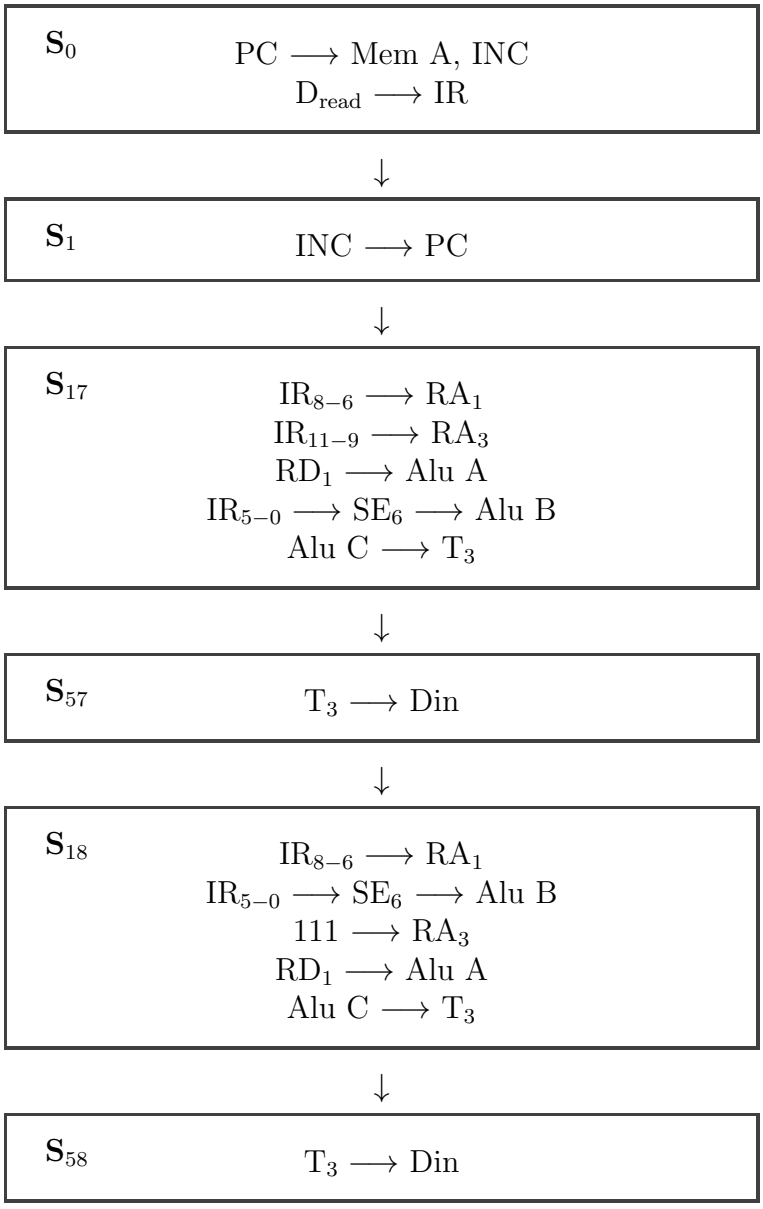

S14
111 ⟶ RA1
IR5−0 ⟶ SE6 ⟶ Alu B
RD1 ⟶ Alu A
Alu C ⟶ T3
111 ⟶ RA3
S3
T3 ⟶ Din

2.13 JAL

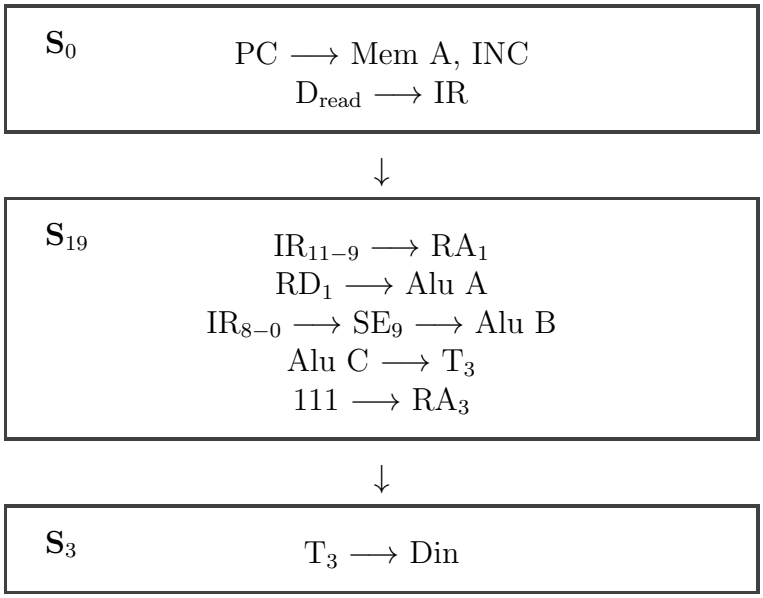

S0
PC ⟶ Mem A, INC
Dread ⟶ IR
S15
111 ⟶ RA1
IR11−9 ⟶ RA3
RD1 ⟶ Alu A
+ 1 ⟶ Alu B
Alu C ⟶ T3
S59
T3 ⟶ Din
S16
111 ⟶ RA1
IR5−0 ⟶ SE6 ⟶ Alu B
RD1 ⟶ Alu A
Alu C ⟶ T3
111 ⟶ RA3


$\mathbf{S}_{60}$
$T_3 \longrightarrow$ Din

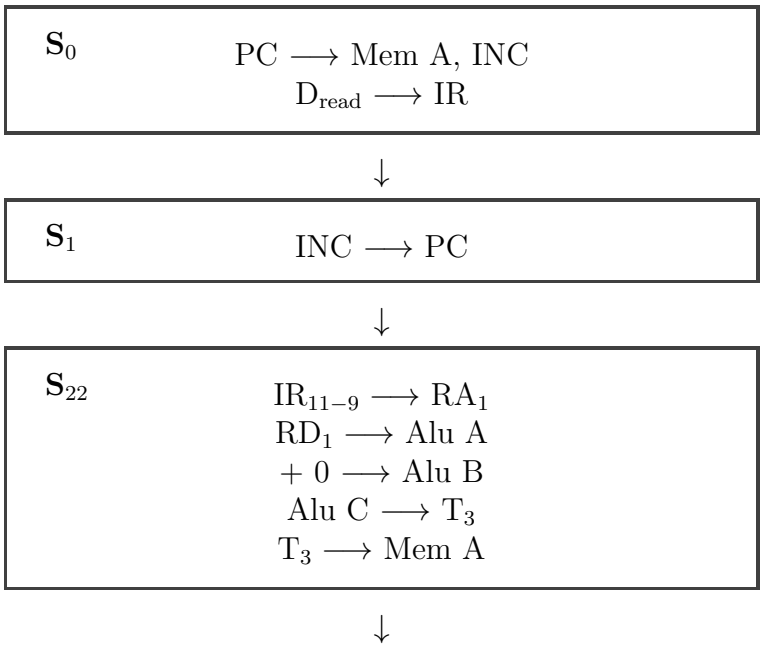
2.14 JLR

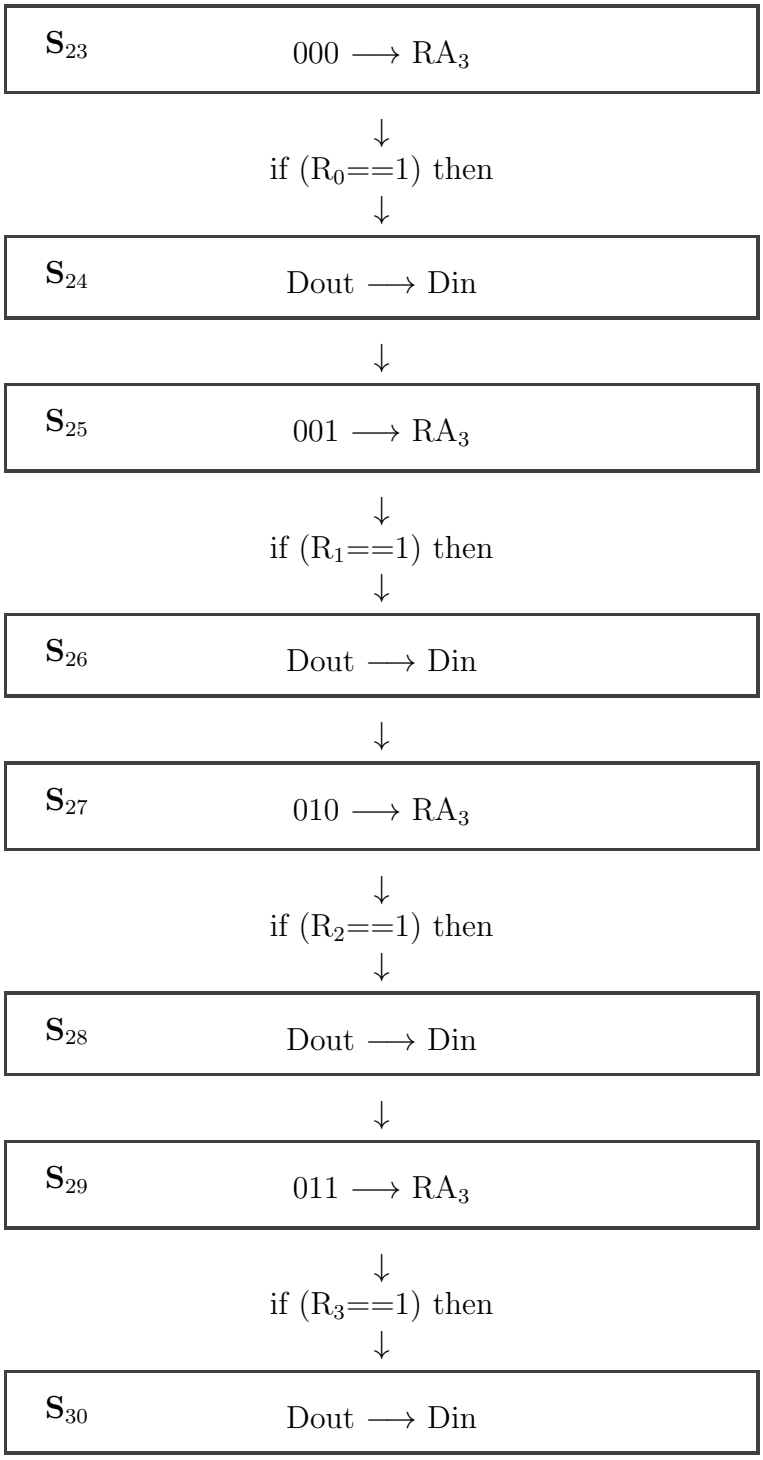

$\mathbf{S}_0$
PC $\longrightarrow$ Mem A, INC
$D_{read} \longrightarrow$ IR
$\downarrow$
$\mathbf{S}_1$
INC $\longrightarrow$ PC
$\downarrow$
$\mathbf{S}_{17}$
$IR_{8-6} \longrightarrow RA_1$
$IR_{11-9} \longrightarrow RA_3$
$RD_1 \longrightarrow$ Alu A
$IR_{5-0} \longrightarrow SE_6 \longrightarrow$ Alu B
Alu C $\longrightarrow T_3$
$\downarrow$
$\mathbf{S}_{57}$
$T_3 \longrightarrow$ Din
$\downarrow$
$\mathbf{S}_{18}$
$IR_{8-6} \longrightarrow RA_1$
$IR_{5-0} \longrightarrow SE_6 \longrightarrow$ Alu B
$111 \longrightarrow RA_3$
$RD_1 \longrightarrow$ Alu A
Alu C $\longrightarrow T_3$
$\downarrow$
$\mathbf{S}_{58}$
$T_3 \longrightarrow$ Din

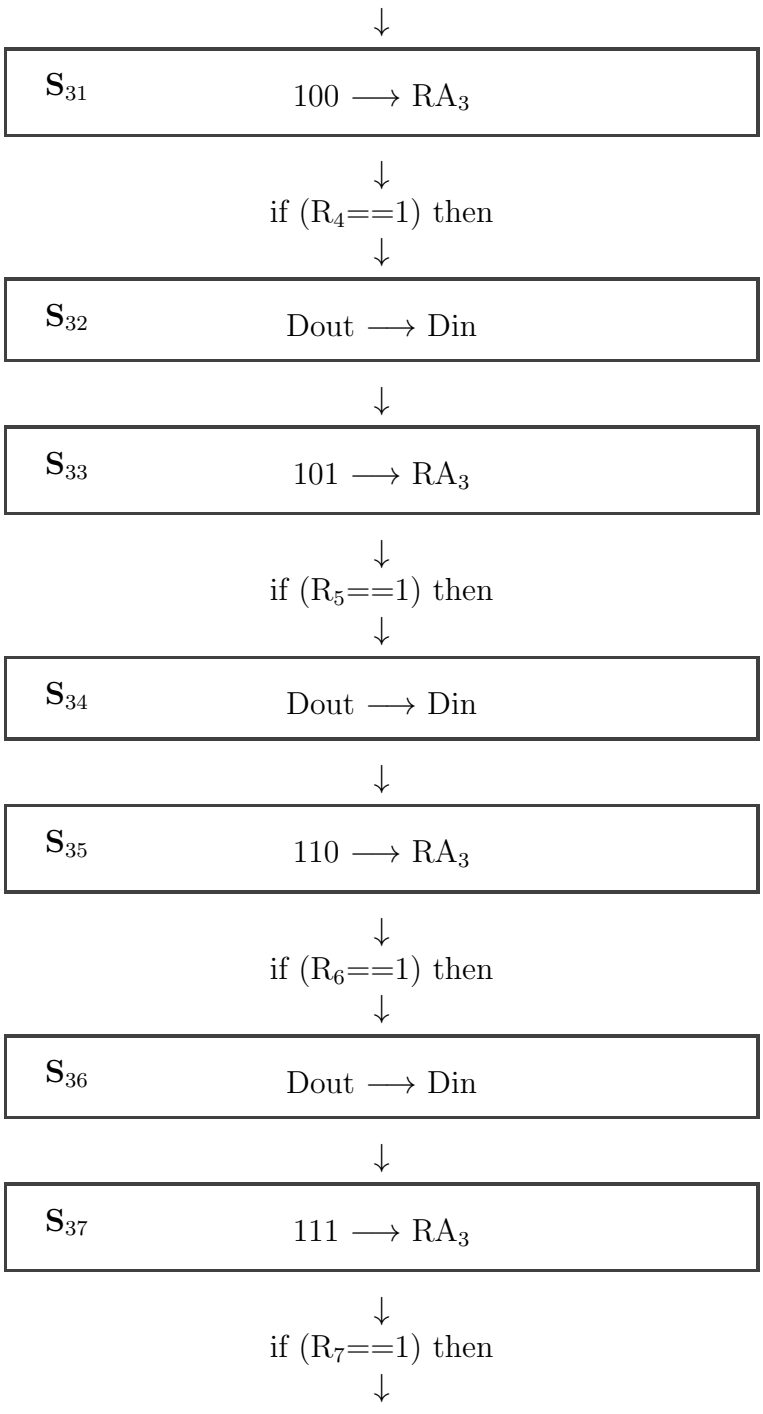
2.15 JRI

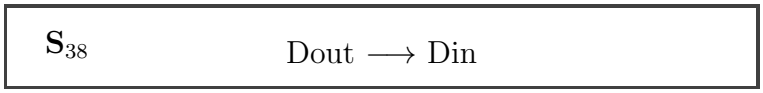


2.16 LM

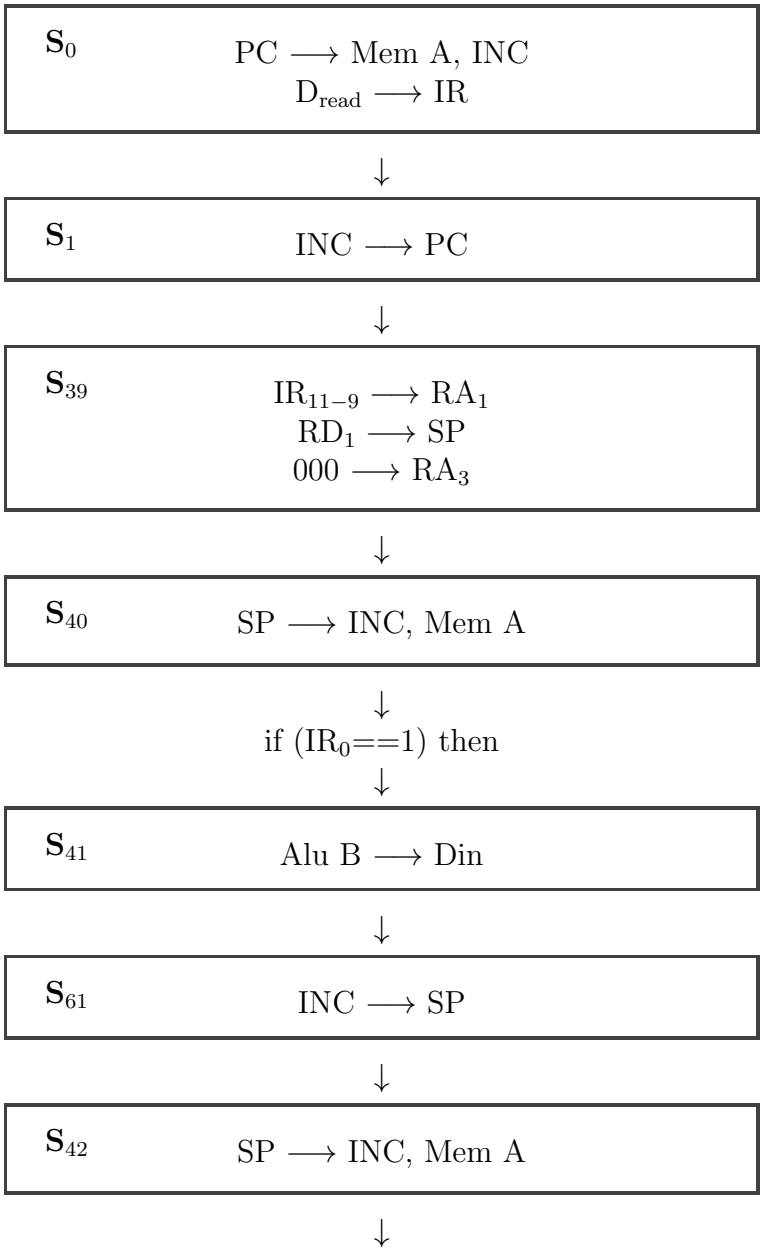


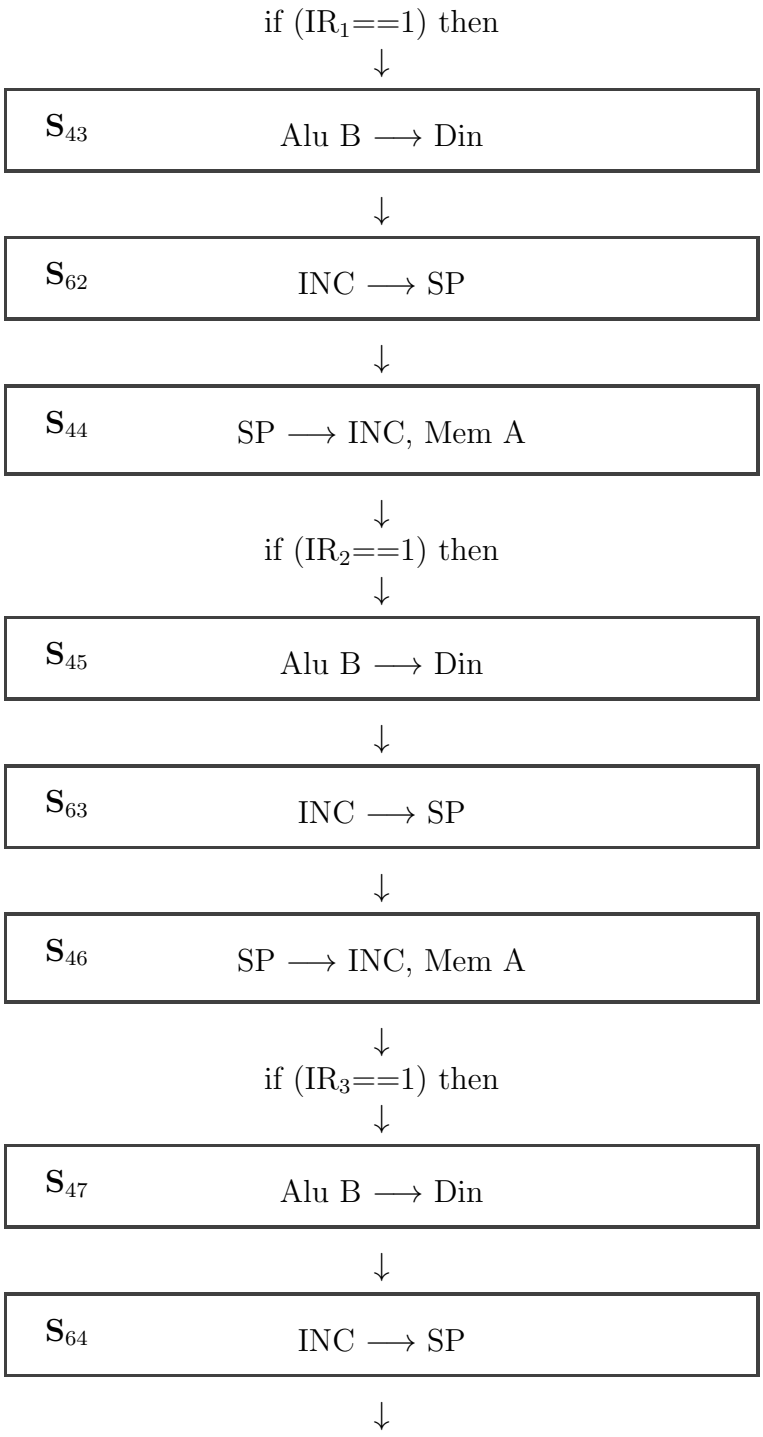

S23
000 ⟶ RA3
if (R0==1) then
S24
Dout ⟶ Din
S25
001 ⟶ RA3
if (R1==1) then
S26
Dout ⟶ Din
S27
010 ⟶ RA3
if (R2==1) then
S28
Dout ⟶ Din
S29
011 ⟶ RA3
if (R3==1) then
S30
Dout ⟶ Din

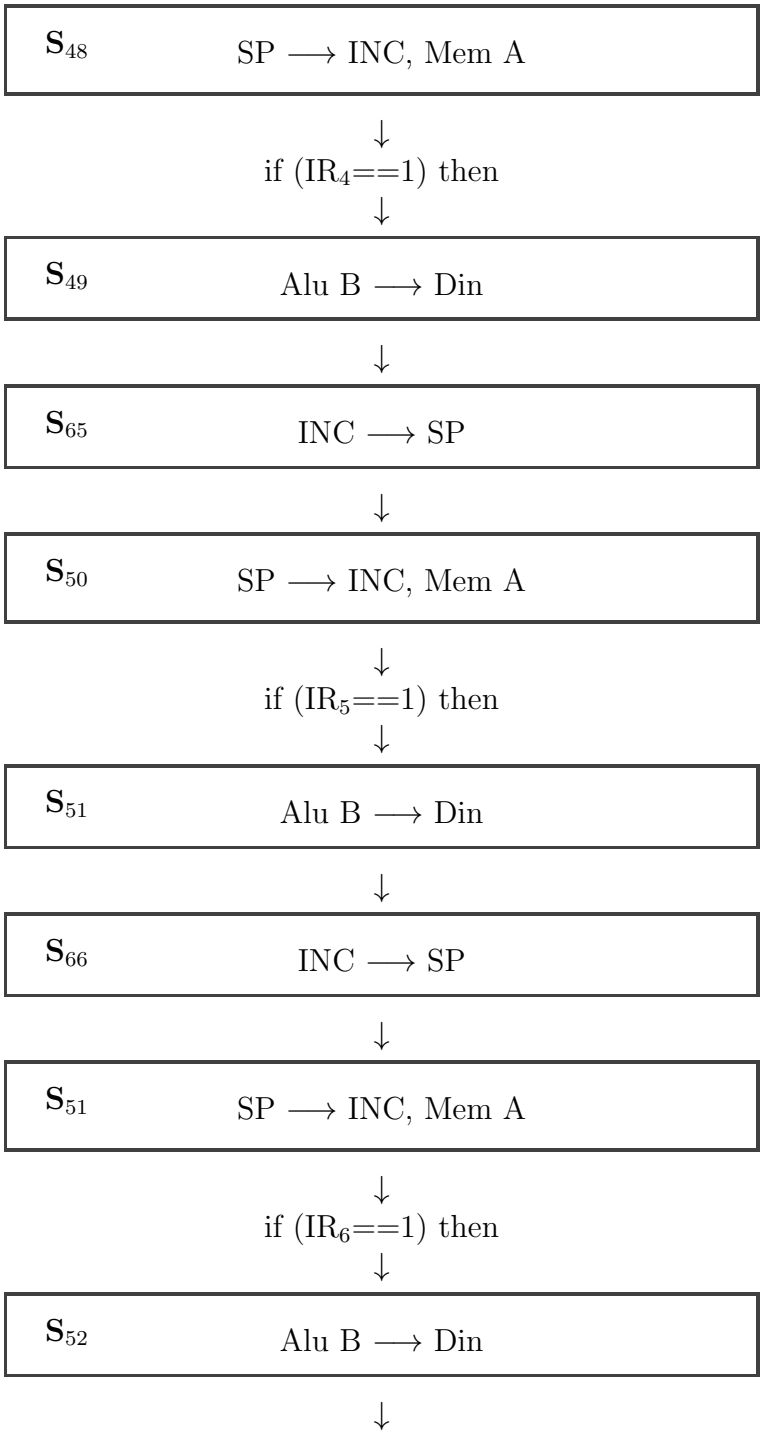

$\mathbf{S}_{31}$ 100 ⟶ RA_3
if (R_4==1) then
$\mathbf{S}_{32}$ Dout ⟶ Din
$\mathbf{S}_{33}$ 101 ⟶ RA_3
if (R_5==1) then
$\mathbf{S}_{34}$ Dout ⟶ Din
$\mathbf{S}_{35}$ 110 ⟶ RA_3
if (R_6==1) then
$\mathbf{S}_{36}$ Dout ⟶ Din
$\mathbf{S}_{37}$ 111 ⟶ RA_3
if (R_7==1) then

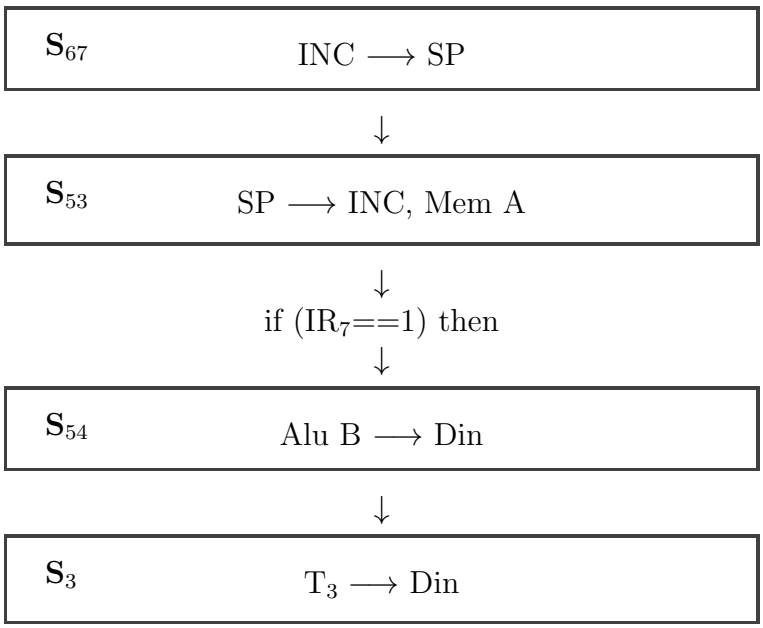

$\mathbf{S}_{38}$
Dout $\longrightarrow$ Din

2.17 SM


$\mathbf{S}_0$
PC $\longrightarrow$ Mem A, INC
$D_{read} \longrightarrow$ IR
$\mathbf{S}_1$
INC $\longrightarrow$ PC
$\mathbf{S}_{39}$
$IR_{11-9} \longrightarrow RA_1$
$RD_1 \longrightarrow$ SP
000 $\longrightarrow RA_3$
$\mathbf{S}_{40}$
SP $\longrightarrow$ INC, Mem A
if (IR_0==1) then
$\mathbf{S}_{41}$
Alu B $\longrightarrow$ Din
$\mathbf{S}_{61}$
INC $\longrightarrow$ SP
$\mathbf{S}_{42}$
SP $\longrightarrow$ INC, Mem A


if (IR1==1) then
S43
Alu B ⟶ Din
S62
INC ⟶ SP
S44
SP ⟶ INC, Mem A
if (IR2==1) then
S45
Alu B ⟶ Din
S63
INC ⟶ SP
S46
SP ⟶ INC, Mem A
if (IR3==1) then
S47
Alu B ⟶ Din
S64
INC ⟶ SP


S48
SP ⟶ INC, Mem A
if (IR4==1) then
S49
Alu B ⟶ Din
S65
INC ⟶ SP
S50
SP ⟶ INC, Mem A
if (IR5==1) then
S51
Alu B ⟶ Din
S66
INC ⟶ SP
S51
SP ⟶ INC, Mem A
if (IR6==1) then
S52
Alu B ⟶ Din



3 Datapath Design

Using the Hardware flowcharts and the given instructions we build a datapath for the microprocessor. This datapath connects all the hardware and enables data flow through out the microprocessor. The datapath entails of,

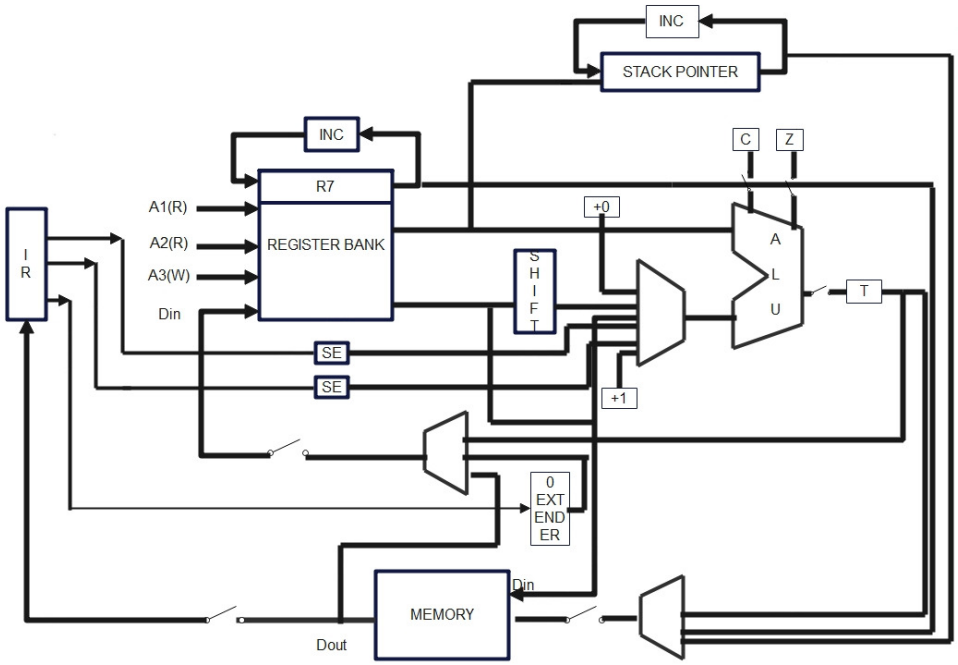
- **Register Bank** It consists of 8 programmers registers which can be used using the instructions. Furthermore, R_7 register is also the program counter of the microprocessor.
- **Incrementer** It is directly connected to the program counter and is used to increase the program counter after each instruction is executed. mm 4nd55e
- **Shifters** They are connected in various places extending the immediate data extracted from the instructions and converting them to 16-bit numbers so that they can be used elsewhere.
- **Arithmetic Logical Unit** The ALU of the microprocessor has 3 different modes of addition, bitwise and and bitwise xor(to compare). These can be accessed using the control signals.

- **Stack Pointer** The stack pointer is provided for SM instruction as we need to increase the address after each iteration.

- **Memory** The memory is of RAM type. Although the RAM is not of 64kB size as it was computationally difficult to simulate it in Quartus. We have used a 256B RAM instead. The memory also contains the code in the start and the rest can be used to store data.

- **Instruction Register** The instruction register stores the current instruction extracted from the memory at the location stored in the program counter and is set in the start of each instruction.

Thus, using these hardware elements we have created a data path accounting for all the instructions and the required states. The datapath is as given as below,



4 Conclusion

Therefore, using the Hardware flowcharts and the datapath we have defined earlier, we constructed the control word for the microprocessor. The control word consists of all the bits needed to control all the hardwares and the flow of the data in the microprocessor. Therefore, using the control word we are able to construct the microprocessor. We have added an external clock to the microprocessor so that the write statements are synchronous and do not interfere with other processes.

Using the control word the states of all the instructions can be made as a finite state machine. Each next state of the instruction only depends on the current state, thus the finite state machine is of type Moore. The finite state machine is depicted as below,