

**CISCO SIMULATOR**

**Manual**

**V 2.0**

## **Group 8**

Md Shahjalal  
Tianyou Bao  
Xuzheng Lu

# Contents

<b>1 Introduction.....</b>	<b>1</b>
<b>1.1 Debugging Panel.....</b>	<b>1</b>
1.1.1 Register Indicators Area .....	1
1.1.2 Memory Area .....	2
1.1.3 Controller Area .....	2
<b>2.2 Classic Panel.....</b>	<b>3</b>
<b>2 Operation.....</b>	<b>3</b>
<b>2.1 Writing Values to Registers .....</b>	<b>3</b>
<b>2.2 Writing Values to Memory .....</b>	<b>3</b>
2.2.1 Using Memory Address Register and Memory Buffer Register .....	3
2.2.2 Modifying the Memory Area .....	4
<b>2.3 Executing Instructions.....</b>	<b>4</b>
2.3.1 Executing Instructions Step-by-Step.....	4
2.3.1 Executing Instructions Automatically.....	5
<b>3 Instructions Reference.....</b>	<b>6</b>
<b>3.1 Load/Store Instructions.....</b>	<b>6</b>
3.1.1 LDR.....	6
3.1.2 STR .....	7
3.1.3 LDA .....	7
3.1.4 LDX .....	7
3.1.5 STX.....	7
<b>3.2 Arithmetic and Logical Instructions .....</b>	<b>8</b>
3.2.1 AMR .....	8
3.2.2 SMR.....	8
3.2.3 AIR.....	9
3.2.4 SIR .....	9
3.2.5 MLT .....	9
3.2.6 DVD.....	9
3.2.7 TRR.....	10
3.2.8 AND.....	10
3.2.9 ORR .....	10
3.2.10 NOT .....	10
<b>3.3 Transfer Instructions.....</b>	<b>11</b>
3.3.1 JZ.....	11
3.3.2 JNE.....	11

3.3.3 JCC.....	11
3.3.4 JMA.....	12
3.3.5 JSR.....	12
3.3.6 RFS .....	12
3.3.7 SOB.....	12
3.3.8 JGE.....	13
<b>3.4 Shift/Rotate Instructions .....</b>	<b>13</b>
3.4.1 SRC.....	13
3.4.2 RRC.....	14
<b>3.5 I/O Instructions .....</b>	<b>14</b>
3.5.1 IN .....	14
3.5.2 OUT .....	14
3.5.3 CHK.....	15
<b>3.6 Other Instructions.....</b>	<b>15</b>
3.6.1 HALT.....	15

# 1 Introduction

This simulator is a simulation of a Complex Instruction Set Computer (CISC). Two panels are designed for the simulator.

## 1.1 Debugging Panel

**Debugging Panel** displays all the information about the Registers, Indicators, and Memory in the computer and can be written manually.

**CISC Machine Simulator**

**Indicators**

**Registers**

R0	0000000000000000	x0000	W
R1	0000000000000000	x0000	W
R2	0000000000000000	x0000	W
R3	0000000000000000	x0000	W

**Index Register**

IX1	0000000000000000	x0000	W
IX2	0000000000000000	x0000	W
IX3	0000000000000000	x0000	W

**Memory Register**

MAR	0000000000000000	x0000	W
MBR	0000000000000000	x0000	W

**Basic Indicators**

PC	000000000000	x000	W
IR	0000000000000000	x0000	W
CC	0000	0	W
MFR	0000	0	W

**Memory**

Address	Value	Hex Value	Assemble Code
x0000	0000000000000000	x0000	null
x0001	0000000001100100	x0064	null
x0002	0000000000000000	x0000	null
x0003	0000000000000000	x0000	null
x0004	000000000100010	x0022	null
x0005	0000000000000000	x0000	null
x0006	0000000001100100	x0064	null
x0007	0000000000000000	x0000	null
x0008	0000010001000010	x0442	LDR 0,1,2
x0009	0000000000000000	x0000	null
x000A	0000000000000000	x0000	null
x000B	0000000001001101	x004D	null
x000C	0000000000000000	x0000	null

**Controller**

Instruction: 0000000000000000

Buttons: Reload, Save, IPL, Program 1, Program 2, Floating Test, Vector Test, Auto Run, Single Run, Pause, Stop, Restart

The panel is divided into three parts:

### 1.1.1 Register Indicators Area

**Indicators**

**Registers**

R0	0000000000000000	x0000	W
R1	0000000000000000	x0000	W
R2	0000000000000000	x0000	W
R3	0000000000000000	x0000	W

**Index Register**

IX1	0000000000000000	x0000	W
IX2	0000000000000000	x0000	W
IX3	0000000000000000	x0000	W

**Memory Register**

MAR	0000000000000000	x0000	W
MBR	0000000000000000	x0000	W

**Basic Indicators**

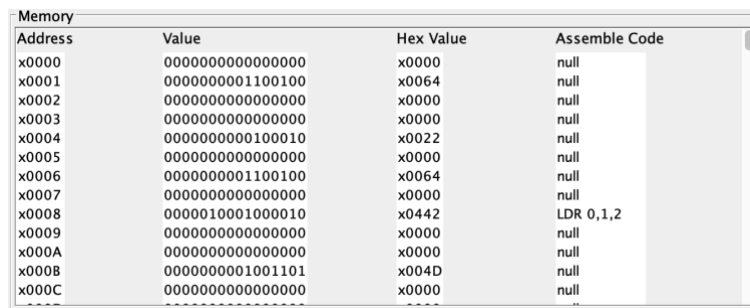
PC	000000000000	x000	W
IR	0000000000000000	x0000	W
CC	0000	0	W
MFR	0000	0	W

The Register Indicators display the values of all kinds of registers.

- Click the 'W' button to manually modify the value of a register.
- Hexadecimal values are shown on the right.

Type	Size(bits)	Number	Description
R0...R3	16	4	General-Purpose Register
IX1...IX3	16	3	Index Register
MAR	16	1	Memory Address Register
MBR	16	1	Memory Buffer Register
PC	12	1	Program Counter
IR	16	1	Instruction Register
CC	4	1	Condition Code
MFR	4	1	Machine Fault Register

### 1.1.2 Memory Area



Address	Value	Hex Value	Assemble Code
x0000	0000000000000000	x0000	null
x0001	0000000001100100	x0064	null
x0002	0000000000000000	x0000	null
x0003	0000000000000000	x0000	null
x0004	0000000000100010	x0022	null
x0005	0000000000000000	x0000	null
x0006	0000000001100100	x0064	null
x0007	0000000000000000	x0000	null
x0008	0000010001000010	x0442	LDR 0,1,2
x0009	0000000000000000	x0000	null
x000A	0000000000000000	x0000	null
x000B	0000000001001101	x004D	null
x000C	0000000000000000	x0000	null

The Memory Area shows the address, the value, the Hexadecimal value, and the Assemble Code of each line on memory.

- The memory address pointed by the Program Counter will be highlighted.
- Double click to manually modify the binary value of a memory row.

### 1.1.3 Controller Area



The Controller Area integrates all function buttons and the instruction input box.

Button	Description
Reload	Initialize the values
Save	Save inputs
IPL	Pre-load a program
Auto Run	Run instructions automatically
Single Run	Run instructions step by step
Pause	Pause the machine
Stop	Stop the machine
Restart	Restart the machine

## 2.2 Classic Panel

The appearance and operational logic of the **Classic Panel** emulate the PDP-8 computer. Users will use switches to input and lights for indication.

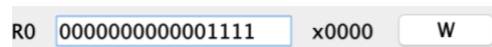
The **Classic Panel** has not been finished yet and will be released in the next version.

## 2 Operation

### 2.1 Writing Values to Registers

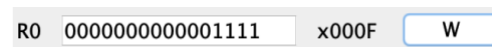
Following the steps below to write a value to a register.

**Step 1:** Input a value into the box



R0 0000000000001111 x0000 W

**Step 2:** Click the 'W' button at right to write the value to the register



R0 0000000000001111 x000F W

**Step 3:** Done! The value will be written to the Register.

#### Error handling:

- Input too long: Remove the excess bits from the left
- Input too short: Add zeros from the left
- Input is not binary: Pop up an Error window

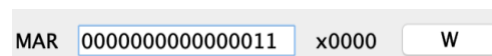


## 2.2 Writing Values to Memory

Two methods are acceptable to write a value to the Memory.

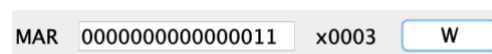
### 2.2.1 Using Memory Address Register and Memory Buffer Register

**Step 1:** Input a value into the MAR box



MAR 0000000000000011 x0000 W

**Step 2:** Click the 'W' button of MAR



MAR 0000000000000011 x0003 W

**Step 3:** Input a value into the MBR box

MBR 0000100010000100 x0000 W

**Step 4:** Click the 'W' button of MBR

MBR 0000000000000000 x0000 W

**Step 5:** Done! The value of MAR will be written to the Memory, and the MAR will automatically change to the next address.

MAR 0000000000000100 x0004 W

MBR 0000000000000000 x0000 W

Memory			
Address	Value	Hex Value	Assemble Code
x0000	0000010010000100	x0484	LDR 0,2,4
x0001	0000000001100100	x0064	null
x0002	0000000000000000	x0000	null
x0003	0000100010000100	x0884	STR 0,2,4

## 2.2.2 Modifying the Memory Area

**Step 1:** Double click the memory row that needs to modify

Memory			
Address	Value	Hex Value	Assemble Code
x0000	0000010010000100	x0484	LDR 0,2,4
x0001	0000000001100100	x0064	null
x0002	0000000000000000	x0000	null
x0003	0000100010000100	x0884	STR 0,2,4
x0004	0000000000100010	x0022	null
x0005	0000000000000000	x0000	null
x0006	0000000001100100	x0064	null

**Step 2:** An input window as the following will pop up. Input the value that needs to write to the memory

Input

Input binary value

0000110101100011

Cancel OK

**Step 3:** Click the 'OK' button, and then the value will be written to the Memory.

Memory			
Address	Value	Hex Value	Assemble Code
x0000	0000010010000100	x0484	LDR 0,2,4
x0001	0000000001100100	x0064	null
x0002	0000000000000000	x0000	null
x0003	0000100010000100	x0884	STR 0,2,4
x0004	0000000000100010	x0022	null
x0005	0000110101100011	x0D63	LDA 1,1,3,1
x0006	0000000001100100	x0064	null

## 2.3 Executing Instructions

Instruction can be executed step-by-step or automatically.

### 2.3.1 Executing Instructions Step-by-Step



**Step 1: Store an instruction to the Memory**

Memory			
x001A	0000000000000000	x0000	null
x001B	0000000001100100	x0064	null
x001C	0000000000000000	x0000	null
x001D	0000000000000000	x0000	null
x001E	1010010001010100	xA454	LDX 1,20
x001F	1010010010010110	xA496	LDX 2,22

**Step 2: Write the address of the instruction to the Program Counter (PC)**

PC	000000011110	x01E	W
----	--------------	------	---

**Step 3: Click the 'Single Run' button, and then the instruction will be executed.**

- The Program Counter will automatically point to the next address of Memory.
- The Instruction Register will store the last executed instruction.

Memory Register		Basic Indicators	
MAR	000000000010100 x0014 W	PC	000000011111 x01F W
MBR	0000000001010000 x0050 W	IR	1010010001010100 xA454 W
		CC	0000 0 W
		MFR	0000 0 W
Memory			
x001B	0000000001100100	x0064	null
x001C	0000000000000000	x0000	null
x001D	0000000000000000	x0000	null
x001E	1010010001010100	xA454	LDX 1,20
x001F	1010010010010110	xA496	LDX 2,22
x0020	1010010011110000	xA4F8	LDX 3,24,1

**2.3.1 Executing Instructions Automatically****Step 1: Store instructions to the Memory**

Memory			
x001E	1010010001010100	xA454	LDX 1,20
x001F	1010010010010110	xA496	LDX 2,22
x0020	1010010011110000	xA4F8	LDX 3,24,1
x0021	0000011100001011	x070B	LDR 3,0,11
x0022	0000010000101011	x042B	LDR 0,0,11,1
x0023	0000010111000011	x05C3	LDR 1,3,3
x0024	0000011011100011	x06E3	LDR 2,3,3,1
x0025	0000101000000001	x0A01	STR 2,0,1
x0026	1010100011010000	xA8D0	STX 3,16
x0027	0000110100000100	x0D04	LDA 1,0,4
x0028	0000000000000000	x0000	null

**Step 2: Write the address of the **starting** instruction to the Program Counter (PC)**

PC	000000011110	x01E	W
----	--------------	------	---

**Step 3: Click the 'Auto Run' button, and then the instructions will be executed automatically.**

- The Program Counter will automatically point to the next address of Memory after an instruction being executed.
- All the indicators will be continuously updated while the program is running.

The screenshot shows the SISC Simulator interface with the following components:

- Indicators**
  - Registers:** R0 (000000000110101, x0035, W), R1 (0000000000000100, x0004, W), R2 (0000000001100100, x0064, W), R3 (0000000001001101, x004D, W).
  - Index Register:** IX1 (0000000001010000, x0050, W), IX2 (0000000001011111, x005F, W), IX3 (0000000001100100, x0064, W).
  - Memory Register:** MAR (0000000000000100, x0004, W), MBR (000000000100010, x0022, W).
  - Basic Indicators:** PC (000000101000, x028, W), IR (0000110100000100, x0D04, W), CC (0000, 0, W), MFR (0000, 0, W).
- Memory:** A list of memory addresses and their contents. Address x0024 is highlighted with the value 0000011011100011. Address x06E3 is highlighted with the value LDR 2,3,1.
- Controller:** Includes an Instruction field (0000000000000000) and buttons for Reload, Save, IPL, Program 1, Program 2, Floating Test, Vector Test, Auto Run, Single Run, Pause, Stop, and Restart.

**Step 4:** Click the 'Pause' button or the 'Stop' button to stop the program.

## 3 Instructions Reference

### 3.1 Load/Store Instructions

The instructions to load/store values from/to Registers or Memory. The binary instruction code format of Load/Store Instructions is as follows:

Opcode	R	IX	I	Address
0	5	6 7	8 9	1 1
			0 1	5

- Opcode:** 6 bits Specifies the instruction
- R:** 2 bits Specifies the General-Purpose Register
- IX:** 2 bits Specifies the Index Register
- I:** 1 bit Specifies Indirect Addressing  
If I = 1, indirect addressing; otherwise, no indirect addressing.
- Address:** 5 bits Specifies the location

#### 3.1.1 LDR

000001	R	IX	I	Address
0	5	6 7	8 9	1 1
			0 1	5

Instruction: LDR r, x, address[, I]

Octal-Opcode: 01  
 Binary-Opcode: 000001  
 Function: Loads Register from Memory

### 3.1.2 STR

000010	R	IX	I	Address
0 5	6 7	8 9	1 1	1
		0 1		5

Instruction: STR r, x, address[, I]  
 Octal-Opcode: 02  
 Binary-Opcode: 000010  
 Function: Stores Register to Memory

### 3.1.3 LDA

000011	R	IX	I	Address
0 5	6 7	8 9	1 1	1
		0 1		5

Instruction: LDA r, x, address[, I]  
 Octal-Opcode: 03  
 Binary-Opcode: 000011  
 Function: Loads Register with Address

### 3.1.4 LDX

101001	R	IX	I	Address
0 5	6 7	8 9	1 1	1
		0 1		5

Instruction: LDX x, address[, I]  
 Octal-Opcode: 41  
 Binary-Opcode: 101001  
 Function: Loads Index Register from Memory

### 3.1.5 STX

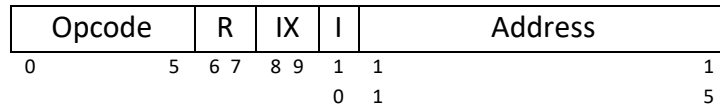
101010	R	IX	I	Address
0 5	6 7	8 9	1 1	1
		0 1		5

Instruction: STX x, address[, I]  
 Octal-Opcode: 42

Binary-Opcode: 101010  
 Function: Stores Index Register to Memory

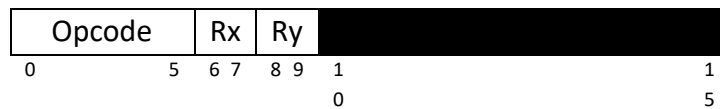
### 3.2 Arithmetic and Logical Instructions

The instructions to perform most of the computational works in the machine. The binary instruction code format of basic Arithmetic and Logical Instructions is as follows:



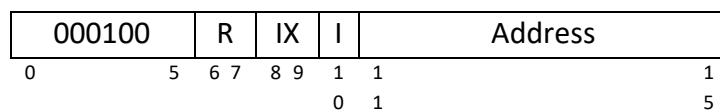
**Opcode:** 6 bits Specifies the instruction  
**R:** 2 bits Specifies the General-Purpose Register  
**IX:** 2 bits Specifies the Index Register  
**I:** 1 bit Specifies Indirect Addressing  
 If I =1, indirect addressing; otherwise, no indirect addressing.  
**Address:** 5 bits Specifies the location

The binary instruction code format of register-to-register Arithmetic and Logical Instructions is as follows:



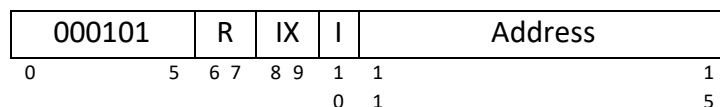
**Opcode:** 6 bits Specifies the instruction  
**Rx:** 2 bits Specifies the General-Purpose Register x  
**Ry:** 2 bits Specifies the General-Purpose Register y

#### 3.2.1 AMR



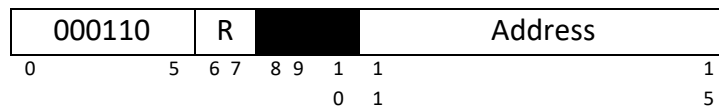
Instruction: AMR r, x, address[, I]  
 Octal-Opcode: 04  
 Binary-Opcode: 000100  
 Function: Add Memory to Register

#### 3.2.2 SMR



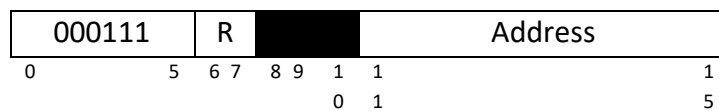
Instruction: SMR r, x, address[, I]  
 Octal-Opcode: 05  
 Binary-Opcode: 000101  
 Function: Subtract Memory from Register

### 3.2.3 AIR



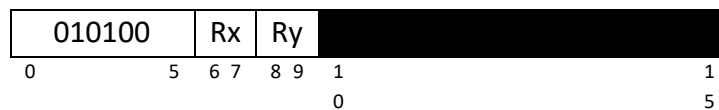
Instruction: AIR r, immed  
 Octal-Opcode: 06  
 Binary-Opcode: 000110  
 Function: Add Immediate to Register

### 3.2.4 SIR



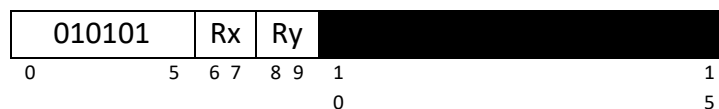
Instruction: SIR r, immed  
 Octal-Opcode: 07  
 Binary-Opcode: 000111  
 Function: Subtract Immediate from Register

### 3.2.5 MLT



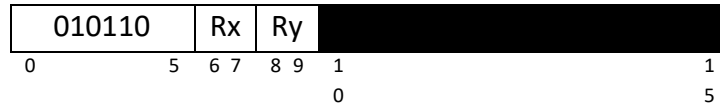
Instruction: MLT rx, ry  
 Octal-Opcode: 20  
 Binary-Opcode: 010100  
 Function: Multiply Register by Register

### 3.2.6 DVD



Instruction: DVD rx, ry  
 Octal-Opcode: 21  
 Binary-Opcode: 010101  
 Function: Divide Register by Register

### 3.2.7 TRR



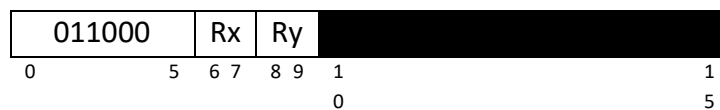
Instruction: TRR rx, ry  
 Octal-Opcode: 22  
 Binary-Opcode: 010110  
 Function: Test the Equality of Register and Register

### 3.2.8 AND



Instruction: AND rx, ry  
 Octal-Opcode: 23  
 Binary-Opcode: 010111  
 Function: Logical AND of Register and Register

### 3.2.9 ORR



Instruction: ORR rx, ry  
 Octal-Opcode: 24  
 Binary-Opcode: 011000  
 Function: Logical OR of Register and Register

### 3.2.10 NOT



Instruction: NOT rx

Octal-Opcode: 25  
 Binary-Opcode: 011001  
 Function: Logical NOT of Register to Register

### 3.3 Transfer Instructions

The instructions to check the value of a register and then change the control of program execution.

The binary instruction code format of Transfer Instructions is as follows:

Opcode	R	IX	I	Address
0 5	6 7	8 9	1 1	1 5
		0 1		

**Opcode:** 6 bits Specifies the instruction  
**R:** 2 bits Specifies the General-Purpose Register  
**IX:** 2 bits Specifies the Index Register  
**I:** 1 bit Specifies Indirect Addressing  
 If I = 1, indirect addressing; otherwise, no indirect addressing.  
**Address:** 5 bits Specifies the location

#### 3.3.1 JZ

001010	R	IX	I	Address
0 5	6 7	8 9	1 1	1 5
		0 1		

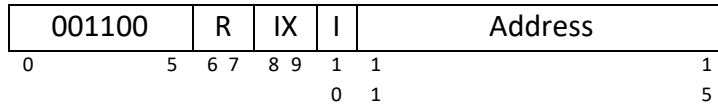
Instruction: JZ r, x, address[, I]  
 Octal-Opcode: 10  
 Binary-Opcode: 001010  
 Function: Jump if Zero

#### 3.3.2 JNE

001011	R	IX	I	Address
0 5	6 7	8 9	1 1	1 5
		0 1		

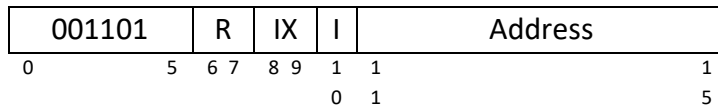
Instruction: JNE r, x, address[, I]  
 Octal-Opcode: 11  
 Binary-Opcode: 001011  
 Function: Jump if Not Equal

#### 3.3.3 JCC



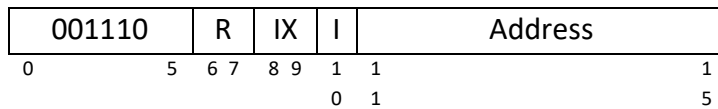
Instruction: JCC cc, x, address[, I]  
 Octal-Opcode: 12  
 Binary-Opcode: 001100  
 Function: Jump if Condition Code

### 3.3.4 JMA



Instruction: JMA x, address[, I]  
 Octal-Opcode: 13  
 Binary-Opcode: 001101  
 Function: Unconditional Jump to Address

### 3.3.5 JSR



Instruction: JSR x, address[, I]  
 Octal-Opcode: 14  
 Binary-Opcode: 001110  
 Function: Jump and Save Return Address

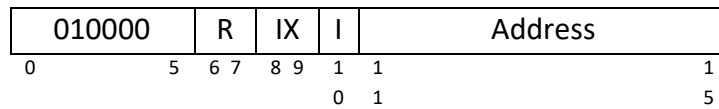
### 3.3.6 RFS



Instruction: RFS immed  
 Octal-Opcode: 15  
 Binary-Opcode: 001111  
 Function: Return from Subroutine with Return Code as Immediate Portion (optional) Stored in the Instruction's Address Field

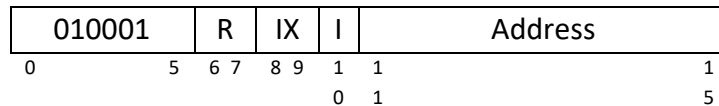
### 3.3.7 SOB





Instruction: SOB r, x, address[, I]  
 Octal-Opcode: 16  
 Binary-Opcode: 010000  
 Function: Subtract One and Branch

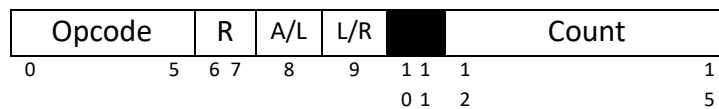
### 3.3.8 JGE



Instruction: JGE r, x, address[, I]  
 Octal-Opcode: 17  
 Binary-Opcode: 010001  
 Function: Jump Greater than or Equal to

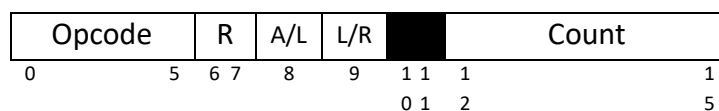
## 3.4 Shift/Rotate Instructions

The instructions to manipulate a datum in a register. The binary instruction code format of Shift and Rotate Instructions is as follows:



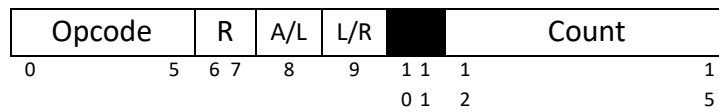
**Opcode:** 6 bits Specifies the instruction  
**R:** 2 bits Specifies the General-Purpose Register  
**A/L:** 2 bits Arithmetic Shift (A/L = 0); Logical Shift (A/L = 1)  
**L/R:** 2 bits Logical Rotate (L/R = 1)  
**Count:** 4 bits Specifies the Count for Operation

### 3.4.1 SRC



Instruction: SRC r, count, L/R, A/L  
 Octal-Opcode: 31  
 Binary-Opcode: 011111  
 Function: Shift Register by Count

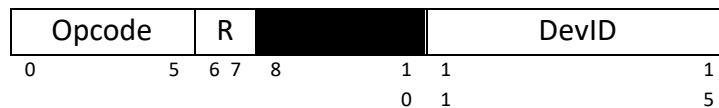
### 3.4.2 RRC



Instruction: RRC r, count, L/R, A/L  
 Octal-Opcode: 32  
 Binary-Opcode: 100000  
 Function: Rotate Register by Count

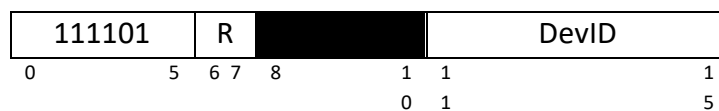
### 3.5 I/O Instructions

The instructions to communicate with the peripherals attached to the computer system. The binary instruction code format of I/O Instructions is as follows:



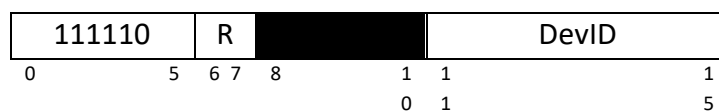
**Opcode:** 6 bits Specifies the instruction  
**R:** 2 bits Specifies the General-Purpose Register  
**DevID:** 5 bits Device ID:  
           0 Console Keyboard  
           1 Console Printer  
           2 Card Reader  
           3-31 Console Registers, Switches, etc.

#### 3.5.1 IN

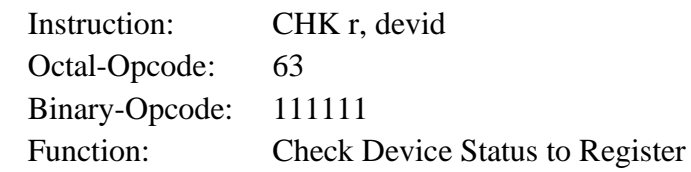


Instruction: IN r, devid  
 Octal-Opcode: 61  
 Binary-Opcode: 111101  
 Function: Input Character to Register from Device

#### 3.5.2 OUT



### 3.5.3 CHK



### 3.6.1 HALT

