

# **Malaviya National Institute of Technology**

Department of Electronics & Communication Engineering  
B.Tech VI Semester Computer Architecture (ECT-312)  
**Term Project Basic Processor**



**Under the supervision of Prof. Rakesh Bairathi**

Submitted By:-  
Group No. 15

Nitin Singh (2020UEC1356)  
Durgesh Saini (2020UEC1600)  
Ankit Meena (2020UEC1793)  
Neal Gautam (2020UEC1741)  
Gautam Khatik (2020UEC1758)

# INDEX

1) Key points:	
i) Configurable.....	3
ii) Harvard Architecture.....	3
iii) Barrel Shifter.....	3
2) Block Diagram.....	4
3) Register Size.....	5
4) Instruction Register Format:	
i) FETCH.....	9
ii) DECODE.....	9
iii) EXECUTE.....	9-12
5) Steps for Simulation.....	13
6) Command for run & compile.....	14-18

# Key Points:

## **i)Registers are Configurable:**

In this processor we can initialize the number of register used as well as memory

Ex:

```
2          // number of registers used
R0 #3F     //initialize R0 with content
R1 #40     //initialize R1 with content
1          //number of memory used
#3F #A1    ..initialize memory with content
```

## **ii) Harvard Architecture:**

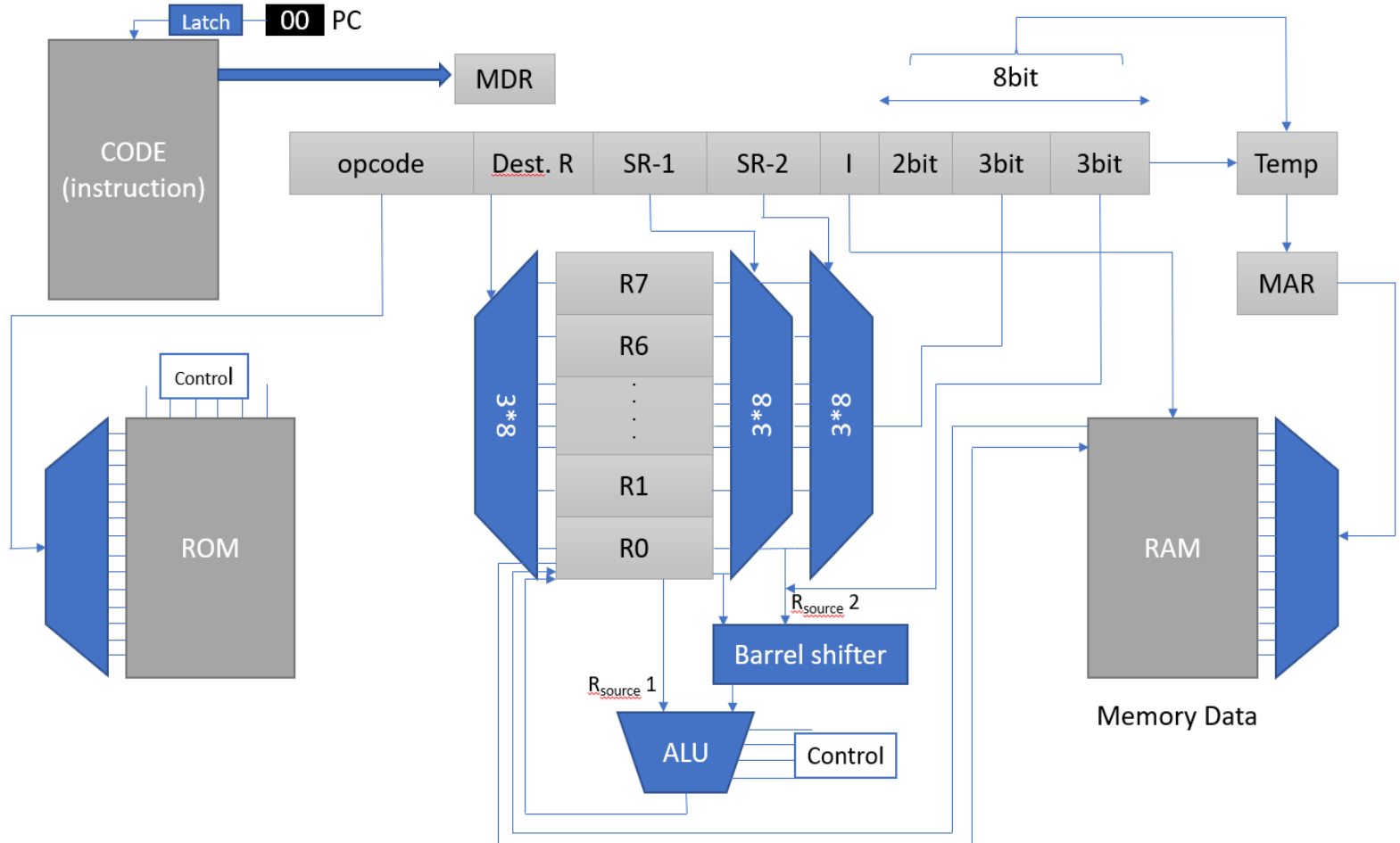
In this processor Harvard Architecture cpu uses separate storage/memory and separate bus for data and microinstruction .



## **iii)Barrel Shifter:**

Barrel shifter is used to shift a data word by a specified number of bits in one clock cycle without affecting registers data during arithmetic and logical operation .

# Block Diagram:-



# Register Size:

- i) RAM(Random access memory) -8 bit address
- ii) PC(Program Counter)–8 bits
- iii) MDR(Memory Data Register) –22 bits
- iv) TEMP–8 bits
- v) ROM(Random access memory)–4 bit address
- vi) MAR–Memory address register–8 bits
- vii) Instruction register –22 bits

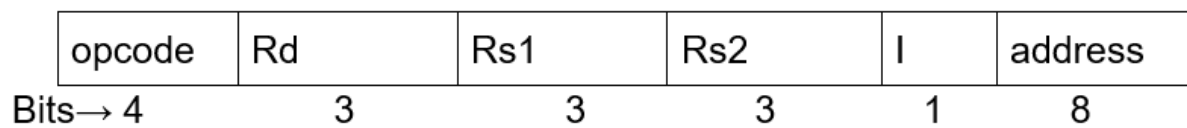
Rd⇒Destination register

Rs1⇒Source Register 1

Rs2⇒Source Register 2

I⇒can be used as MRI or barrel shifter

# Instruction Register Format:



Case -1



⇒ 8 bits

Memory Instruction

I=0 treated as Immediate

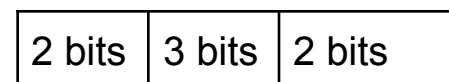
I=1 treated as address

eg.(inst.)

LDA

SDM

Case-2



I=0 treated as Imd.val.

I=1 used as barrel shifter

## **Barrel Shifter (I=1)**

Type of Rotation		Register Number			Immediate		
X	X	X	X	X	X	X	X

### **# Type of Rotation**

S.No.	Type of Rotation	Opcode
1.	LSL	00
2.	LSR	01
3.	ASR	10
4.	ROR	11

### **#Amount of rotation**

#### **i)Via Register**

⇒ We specify the register number (0-7) and last three bits of value stored at that register is taken as amount of rotation

Eg.

ADD R1 R2 R3 I LSL R5

R5[0:2] ⇒ last three bits of R5 is taken as amount of rotation

LSL ⇒ type of rotation

#### **ii)Via Immediate value**

⇒ Immediate values taken as the amount of rotation.

Eg.

ADD R1 R2 R3 I LSL #Imd

Imd ⇒ is used as amount of rotation

LSL ⇒ type of rotation

S.No.	Type of Instruction	Opcode
1.	LDM	0000
2.	SDM	0001
3.	MOV	0010
4.	ADD	0011
5.	SUB	0100
6.	AND	0101
7.	OR	0110
8.	XOR	0111
9.	CSB	1000

For amount of Rotation two options are there:

- i) **Register**: In case of register value of register is loaded in that column
- ii) **Immediate** :In case of Immediate value is loaded in Immediate column

Eg.

I =1 LSL #3F

I=1 LSL R2 (R2=8)



(Initialise PC=00)

PC(E) MDR(E,L)  
PC(I)  
PC $\leftarrow$ PC+1

IR: XXXX	XXX	XXX	XXX	X	XXXXXXXXX
Opcode	Rd	Rs1	Rs2	I	Address/Immediate

### i) LDM

I=0	Rd $\leftarrow$ #Imd.	Imd $\Rightarrow$ Immediate value
I=1	Rd $\leftarrow$ Mem[Imd]	

```

I=0   Rd(L)   temp(E)
I=1   temp(E) MAR(E,L)
      RAM(E) Rd(L)
flags=[Z,N,C,V]  {affected flags}

```

Eg . LDM Rd #3F flags=[0,0,0,0]  
LDM Rd I #8F flags=[0,0,1,0]

ii)SDM

Mem[lmd]  $\leftarrow$  Rd

---

temp(E)      MAR(E,L)

RAM(L)      Rd(E)

flags=[Z,N,C,V]    {affected flags}

---

Eg. SDM Rd I #3F

Mem[3F]  $\leftarrow$  Rd

iii) MOV Rd Rs

I=0 Rd  $\leftarrow$  Rs

I=1 Rd  $\leftarrow$  Rs <<(rotate by some bits)

---

I=0 Rs(E) Rd(L)

I=1 Rs(E) Bar(L) temp(E)

---

Eg. MOV Rd Rs I LSL #3

flags=[Z,N,C,V]      {affected flags}

iv) ADD

I=0  $Rd \leftarrow Rs1 + Rs2$

$Rd \leftarrow Rs1 + Im$

I=1  $Rd \leftarrow Rs1 + Rs2 \ll \#Imd$

---

I=0 case-1      Rs1(E) Rs2(E) ALU(L)  
                    Rd(L) ALU(E)

case-2      Rs1(E) temp(E) ALU(L)  
              Rd(L) ALU(E)

I=1

Rs2(E)	temp(E)	Barrel (L)
(Data)	(type of rotation )	( amount of rotation)

Rs1(E) Barrel (E) ALU (L)

Rd(L) ALU (E)

//flags will be affected

---

ex.

I=0 ADD Rd Rs1 Rs2

ADD Rd Rs1 #Imd

I=1 ADD R1 R2 R3 I LSL #Imd

---

v) SUB,AND,XOR,OR → same as ADD instruction

vii) CSB (count set bit)

I=0 CSB Rd Rs1  
Rs1(E) ALU(L)  
ALU(E) Rd(L)

I=0 CSB Rd \_ #lmd  
temp(E) ALU(L)  
ALU(E) Rd(L)

I=1 CSB Rd I #lmd  
temp(E) MAR(E,L)  
RAM(E) ALU(L)  
ALU(E) Rd(L)

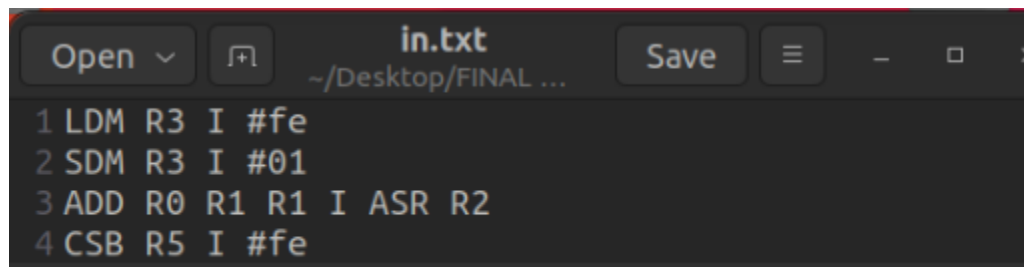
# Steps for Simulation

i) Open the file the config.txt file and set the registers and memory as per requirement and close after configure file

A screenshot of a code editor window titled 'config.txt' with a path of '~/.Desktop/FINAL P...'. The editor contains six lines of text: '1 2', '2 1 #cd', '3 2 #01', '4 2', '5 #00 #fe', and '6 #fe #3f'. The fifth line is highlighted in a light gray background.

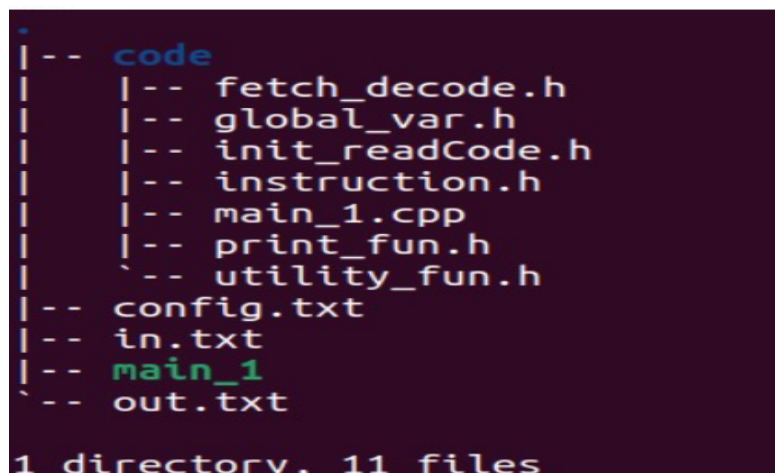
```
1 2
2 1 #cd
3 2 #01
4 2
5 #00 #fe
6 #fe #3f
```

ii) Open the in.txt file write Instructions

A screenshot of a code editor window titled 'in.txt' with a path of '~/.Desktop/FINAL ...'. The editor contains four lines of assembly instructions: '1 LDM R3 I #fe', '2 SDM R3 I #01', '3 ADD R0 R1 R1 I ASR R2', and '4 CSB R5 I #fe'. The window has standard buttons for 'Open', 'Save', and window controls.

```
1 LDM R3 I #fe
2 SDM R3 I #01
3 ADD R0 R1 R1 I ASR R2
4 CSB R5 I #fe
```

iii) Open terminal in Code Directory and compile and Run the main\_1.cpp file

A screenshot of a terminal window with a dark purple background. It shows a directory listing for a 'code' directory. The listing includes several header files, a main\_1.cpp file, and utility files. At the bottom, it states '1 directory, 11 files'.

```
-- code
|-- fetch_decode.h
|-- global_var.h
|-- init_readCode.h
|-- instruction.h
|-- main_1.cpp
|-- print_fun.h
|-- utility_fun.h
-- config.txt
-- in.txt
-- main_1
-- out.txt

1 directory, 11 files
```

## Command for run and compile

[g++ main\_1.pp -o app && ./app]

```
nitin@nsp-1t:~/Desktop/FINAL PRO/working/code$ g++ main_1.cpp -o app && ./app
[line] = [{"LDM","R3","I","#fe"}]
[line] = [{"SDM","R3","I","#01"}]
[line] = [{"ADD","R0","R1","R1","I","ASR","R2"}]
[line] = [{"CSB","R5","I","#fe"}]
nitin@nsp-1t:~/Desktop/FINAL PRO/working/code$
```

4) we get output in out.txt file

### # example

Config.txt	In.txt
2 1 #cd 2 #01 2 #00 #fe #fe #3f	LDM R3 I #fe SDM R3 I #01 ADD R0 R1 R1 I ASR R2 CSB R5 I #fe

out.txt

set Register(index , value(Hex))

Register Bank [8]:

0x00 0xcd 0x01 0x00 0x00 0x00 0x00 0x00

Memory:

0x00 0xfe

0xfe 0x3f

---

Fetch:

PC 0

PC(E) MDR(E,L)

PC(I)

MDR: LDM R3 I #fe

Decode:

IR : 0000 011 xxx xxx 1 11111110

Execute :

temp(E) MAR(E,L)

RAM(E) R3(L)

Flags: Z V C N ⇒0010

Register Bank [8]:

0x00 0xcd 0x01 0x3f 0x00 0x00 0x00 0x00

Memory:

0x00 0xfe

0xfe 0x3f

---

Fetch:

PC 1

PC(E) MDR(E,L)

PC(I)

MDR: SDM R3 I #01

Decode:

IR : 0001 011 xxx xxx 1 00000001

Execute :

temp(E),MAR(E,L)

RAM(L) R3(E)

Flags: Z V C N  $\Rightarrow$ 0010

Memory:

0x00 0xfe

0x01 0x3f

0xfe 0x3f

---

Fetch:

PC 2

PC(E) MDR(E,L)

PC(I)

MDR: ADD R0 R1 R1 I ASR R2

Decode:

IR : 0011 000 001 001 1 10010xxx

Execute :

R1(E) temp(E) Barrel(E)

R1(E) Barrel(E) ALU(L)

R0(L) ALU(E)

Flags: Z V C N  $\Rightarrow$ 0111

Register Bank [8]:

0xb3 0xcd 0x01 0x3f 0x00 0x00 0x00 0x00

Memory:

0x00 0xfe

0x01 0x3f

0xfe 0x3f



---

Fetch:

PC 3

PC(E) MDR(E,L)

PC(I)

MDR: CSB R5 I #fe

Decode:

IR : 1000 101 xxx xxx 1 xxxxxxxx

Execute :

temp(E) MAR(E,L)

RAM(E) ALU(L)

ALU(E) R5(L)

Flags: Z V C N  $\Rightarrow$ 0111

Register Bank [8]:

0xb3 0xcd 0x01 0x3f 0x00 0x06 0x00 0x00

Memory:

0x00 0xfe

0x01 0x3f

0xfe 0x3f

---

Activities

Terminal

Tue Apr 18 10:38 AM

100 %

Open

out.txt

~/Desktop/FINAL PRO/working

```

1 set Register(index , value(Hex))
2 Register Bank [8]:
3     0x00 0xcd 0x01 0x00 0x00 0x00 0x00 0x00
4 Memory:
5     0x00 0xfe
6 0xfe 0x3f
7
8 -----
9 Fetch:
10    PC 0
11    PC(E) MDR(E,L)
12    PC(I)
13    MDR: LDM R3 I #fe
14 Decode:
15    IR : 0000 011 xxx xxx 1 11111110
16 Excute :
17    temp(E) MAR(E,L)
18 RAM(E) R3(L)
19 Flags: Z V C N =>0010
20 Register Bank [8]:
21    0x00 0xcd 0x01 0x3f 0x00 0x00 0x00 0x00
22 Memory:
23    0x00 0xfe
24 0xfe 0x3f
25
26 -----
27 Fetch:
28    PC 1
29    PC(E) MDR(E,L)
30    PC(I)
31    MDR: SDM R3 I #01
32 Decode:
33    IR : 0001 011 xxx xxx 1 00000001
34 Excute :
35    temp(E),MAR(E,L)
36 RAM(L) R3(E)
37 Flags: Z V C N =>0010
38 Memory:
39    0x00 0xfe
40 0x01 0x3f
41 0xfe 0x3f
42
43 -----
44 Fetch:

```

nitin@nsp-lt: ~/Desktop/FINAL PRO/working/code

Q

```

nitin@nsp-lt:~/Desktop/FINAL PRO/working/code$ g++ main_1.cpp -o app && ./app
[line] = [{"LDM","R3","I","#fe"}]
[line] = [{"SDM","R3","I","#01"}]
[line] = [{"ADD","R0","R1","R1","I","ASR","R2"}]
[line] = [{"CSB","R5","I","#fe"}]
nitin@nsp-lt:~/Desktop/FINAL PRO/working/code$

```

Open

config.txt

~/Desktop/FINAL P...

Save

```

1 2
2 1 #cd
3 2 #01
4 2
5 #00 #fe
6 #fe #3f

```

Plain Text

Tab Width: 8

Ln 5, Col 8

INS

Open

in.txt

~/Desktop/FINAL ...

Save

```

1 LDM R3 I #fe
2 SDM R3 I #01
3 ADD R0 R1 R1 I ASR R2
4 CSB R5 I #fe
5

```

Plain Text

Tab Width: 8

Ln 5, Col 1

INS

Plain Text

Tab Width: 8

Ln 14, Col 7

INS