COMPUTER ARCHITECTURE (ECT-312)

PROJECT – SANDL PROCESSOR

Submitted to:

Rakesh Bairathi

Submitted by:

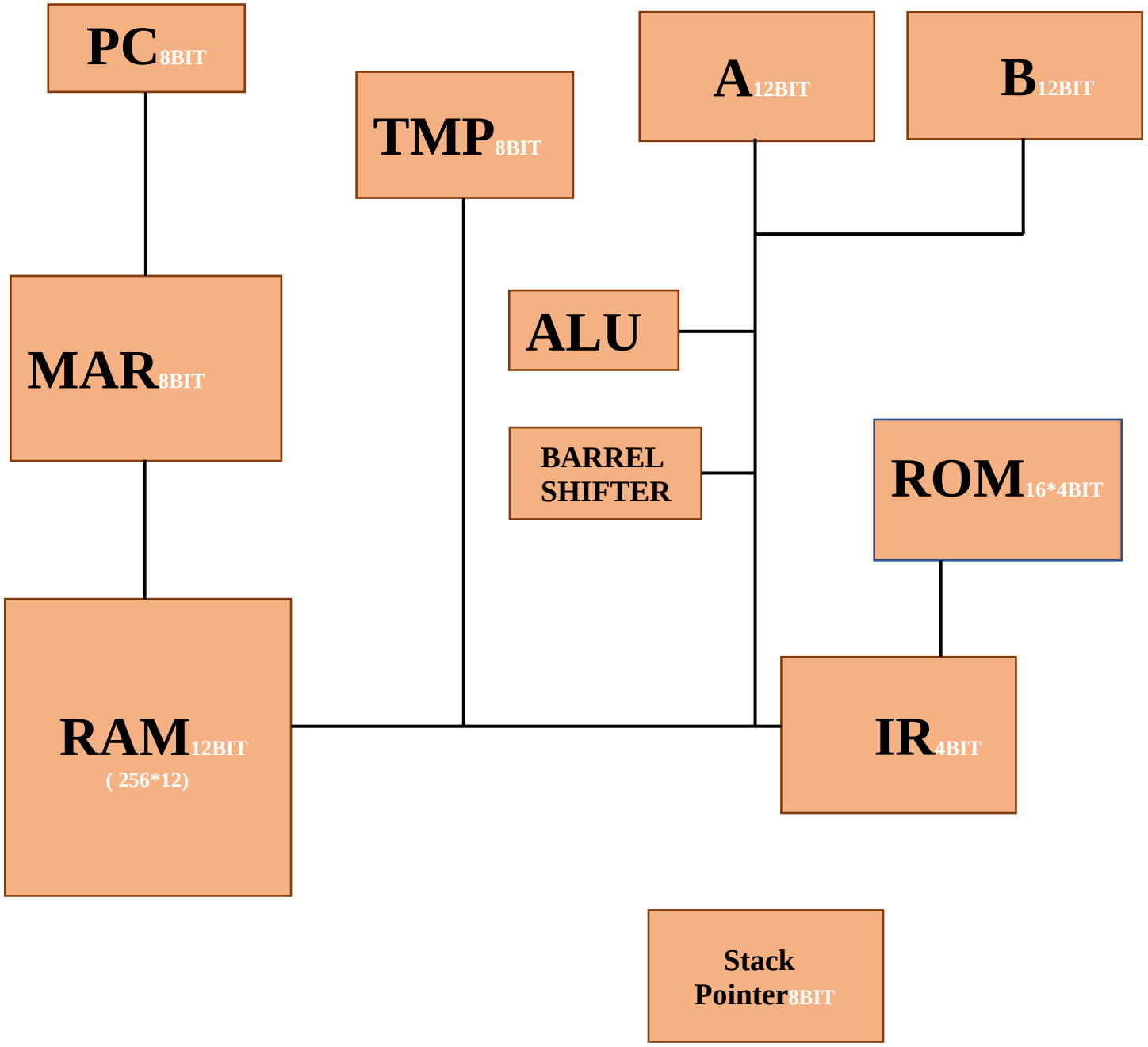
N.Aryan Ruthwik	2020UEC1686
Donabodi Suneel Kumar	2020UEC1654
Neelirothu Dileep Ram Kumar	2020UEC2009
Banoth Likitha	2020UEC1814
Baddam Nithya Reddy	2020UEC1383

Malaviya National Institute of Technology

ELECTRONICS AND COMMUNICATION DEPARTMENT

2022-2023

Processor Architecture Layout:



FLAGS:

Carry Flag (CF)

In SANDL Processor Carry flag is used to set the carry bit when carry generated by various operations between registers.

INSTRUCTION SET:

OPCODE	ADDRESS
4BIT	8BIT

Sandl Processor enhances the user experience by accepting the strings as input. This processor culminates the Memory referencing instructions (MRI) and Non- Memory referencing Instructions (Non-MRI) and IMMEDIATE and REGISTER sets which can be used as per the user requirements. The below table shows the MRI, Non-MRI, and Immediate instructions.

MRI

- ADM- ADD data present in memory location to A and store in A
- SBM- SUBTRACT data present in memory location from A and store in A
- STA-Store accumulator in Memory Location
- LDA-Load accumulator with Memory Location

Non-MRI

- RSH- Rotate right by 1 bit using barrel shifter.
- BMOV- Move A to B
- INR- Increment accumulator
- AND- AND A and B and store in A
- XOR- XOR A and B and store in A
- CLR- Set value of A to zero
- POP- Remove data to the stack
- LSH- Rotate left by 1 bit using barrel shifter.

IMMEDIATE

- AMVI- Load A with data
- ADI- Add data to accumulator
- SBI- Subtract data from accumulator
- PUSH- Add data to the stack

Unique features of conjoint processor!

- Our processor consists inbuilt assembler which does the work of assembling on its own. By taking the string and decoding it to the opcodes and taking the address as an integer value and dealing the instruction and output is also generated as integer value.
- We are giving the input in the form of the string which is really user friendly.
- If the user inputs a wrong instruction. Our processor identifies it and informs it to the user that wrong instruction is entered.
- Our Processor also has the ability to perform the IMMEDIATE PUSH operation.
- At every output and input stage we can check the values of different registers, thus again we can say it is reliable for user to check the changes after performing the operations.

OPCODES:

NAME	Operation	Opcode
AMVI	Load A with data	0000
BMOV	Move A to B	0001
INR	Increment Accumulator	0010
ADI	Add data to accumulator	0011
ADM	Add data present in memory location	0100
SBI	Subtract data from accumulator	0101
SBM	Subtract data present in memory location from A	0110
AND	And operation	0111
XOR	Xor operation	1000
CLR	Set value of A to 0	1001
RSH	Rotate right	1010
LSH	Rotate left	1011
STA	Store Accumulator	1100
LDA	Load Accumulator	1101
PUSH	Add data to the stack	1110
POP	Remove data to the stack	1111

CODE OUTPUT SAMPLES:

1.INPUT:

```
PUSH 155
LDA 255
BMOV -1
POP -1
STA 24
HLT -1
```

OUTPUT:

```
Valid Instructions are
AMVI---->Load A with data                        AMVI #DATA
BMOV---->Move A to B                             BMOV -1
INR---->Increment Accumulator                    INR -1
ADI---->Add data to Accumulator                  ADI #DATA
SBI---->Subtract data from Accumulator           SBI #DATA
ADM---->Add data present in Memory Location      ADM #ADDRESS
SBM---->Substract data present in Memory Location  SBM #ADDRESS
AND---->Bitwise And operation                    AND -1
XOR---->Bitwise XOR operation                    XOR -1
CLR---->Set value of A to 0                      CLR -1
RSH---->Rotate right                             RSH -1
LSH---->Rotate Left                              LSH -1
STA---->Store Accumulator in Memory Location     STA #ADDRESS
LDA---->Load Accumulator with Memory Location    LDA #ADDRESS
PUSH---->Add data to the stack                   PUSH #DATA
POP---->Remove data from the stack               POP -1
If the operation operand is implicit then write '-1' in place of data/address.

ENTER YOUR PROGRAM :
PUSH 155
LDA 255
BMOV -1
POP -1
STA 24
HLT -1
FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2 -->SP(D)
 T3 -->SP(E) , MAR(E,L)
 T4 -->TEMP(E) , RAM(L), R(reset)
```

```
100 ---> 0    101 ---> 0    102 ---> 0    103 ---> 0    104 ---> 0    105 ---> 0    106 ---> 0    107 ---> 0    108 ---> 0    109 ---> 0
110 ---> 0    111 ---> 0    112 ---> 0    113 ---> 0    114 ---> 0    115 ---> 0    116 ---> 0    117 ---> 0    118 ---> 0    119 ---> 0
120 ---> 0    121 ---> 0    122 ---> 0    123 ---> 0    124 ---> 0    125 ---> 0    126 ---> 0    127 ---> 0    128 ---> 0    129 ---> 0
130 ---> 0    131 ---> 0    132 ---> 0    133 ---> 0    134 ---> 0    135 ---> 0    136 ---> 0    137 ---> 0    138 ---> 0    139 ---> 0
140 ---> 0    141 ---> 0    142 ---> 0    143 ---> 0    144 ---> 0    145 ---> 0    146 ---> 0    147 ---> 0    148 ---> 0    149 ---> 0
150 ---> 0    151 ---> 0    152 ---> 0    153 ---> 0    154 ---> 0    155 ---> 0    156 ---> 0    157 ---> 0    158 ---> 0    159 ---> 0
160 ---> 0    161 ---> 0    162 ---> 0    163 ---> 0    164 ---> 0    165 ---> 0    166 ---> 0    167 ---> 0    168 ---> 0    169 ---> 0
170 ---> 0    171 ---> 0    172 ---> 0    173 ---> 0    174 ---> 0    175 ---> 0    176 ---> 0    177 ---> 0    178 ---> 0    179 ---> 0
180 ---> 0    181 ---> 0    182 ---> 0    183 ---> 0    184 ---> 0    185 ---> 0    186 ---> 0    187 ---> 0    188 ---> 0    189 ---> 0
190 ---> 0    191 ---> 0    192 ---> 0    193 ---> 0    194 ---> 0    195 ---> 0    196 ---> 0    197 ---> 0    198 ---> 0    199 ---> 0
200 ---> 0    201 ---> 0    202 ---> 0    203 ---> 0    204 ---> 0    205 ---> 0    206 ---> 0    207 ---> 0    208 ---> 0    209 ---> 0
210 ---> 0    211 ---> 0    212 ---> 0    213 ---> 0    214 ---> 0    215 ---> 0    216 ---> 0    217 ---> 0    218 ---> 0    219 ---> 0
220 ---> 0    221 ---> 0    222 ---> 0    223 ---> 0    224 ---> 0    225 ---> 0    226 ---> 0    227 ---> 0    228 ---> 0    229 ---> 0
230 ---> 0    231 ---> 0    232 ---> 0    233 ---> 0    234 ---> 0    235 ---> 0    236 ---> 0    237 ---> 0    238 ---> 0    239 ---> 0
240 ---> 0    241 ---> 0    242 ---> 0    243 ---> 0    244 ---> 0    245 ---> 0    246 ---> 0    247 ---> 0    248 ---> 0    249 ---> 0
250 ---> 0    251 ---> 0    252 ---> 0    253 ---> 0    254 ---> 0    255 ---> 155


PC=1 IR=14 TEMP=155 MAR=255 A=0 B=0 CF=0 SP=255


--------------------------------------------------------------------------------------------------------------
```

```
FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2 -->TEMP(E) , MAR(E,L)
 T3 -->RAM(E) , A(L), R(reset)

100 ---> 0      101 ---> 0      102 ---> 0      103 ---> 0      104 ---> 0      105 ---> 0      106 ---> 0      107 ---> 0      108 ---> 0      109 ---> 0
110 ---> 0      111 ---> 0      112 ---> 0      113 ---> 0      114 ---> 0      115 ---> 0      116 ---> 0      117 ---> 0      118 ---> 0      119 ---> 0
120 ---> 0      121 ---> 0      122 ---> 0      123 ---> 0      124 ---> 0      125 ---> 0      126 ---> 0      127 ---> 0      128 ---> 0      129 ---> 0
130 ---> 0      131 ---> 0      132 ---> 0      133 ---> 0      134 ---> 0      135 ---> 0      136 ---> 0      137 ---> 0      138 ---> 0      139 ---> 0
140 ---> 0      141 ---> 0      142 ---> 0      143 ---> 0      144 ---> 0      145 ---> 0      146 ---> 0      147 ---> 0      148 ---> 0      149 ---> 0
150 ---> 0      151 ---> 0      152 ---> 0      153 ---> 0      154 ---> 0      155 ---> 0      156 ---> 0      157 ---> 0      158 ---> 0      159 ---> 0
160 ---> 0      161 ---> 0      162 ---> 0      163 ---> 0      164 ---> 0      165 ---> 0      166 ---> 0      167 ---> 0      168 ---> 0      169 ---> 0
170 ---> 0      171 ---> 0      172 ---> 0      173 ---> 0      174 ---> 0      175 ---> 0      176 ---> 0      177 ---> 0      178 ---> 0      179 ---> 0
180 ---> 0      181 ---> 0      182 ---> 0      183 ---> 0      184 ---> 0      185 ---> 0      186 ---> 0      187 ---> 0      188 ---> 0      189 ---> 0
190 ---> 0      191 ---> 0      192 ---> 0      193 ---> 0      194 ---> 0      195 ---> 0      196 ---> 0      197 ---> 0      198 ---> 0      199 ---> 0
200 ---> 0      201 ---> 0      202 ---> 0      203 ---> 0      204 ---> 0      205 ---> 0      206 ---> 0      207 ---> 0      208 ---> 0      209 ---> 0
210 ---> 0      211 ---> 0      212 ---> 0      213 ---> 0      214 ---> 0      215 ---> 0      216 ---> 0      217 ---> 0      218 ---> 0      219 ---> 0
220 ---> 0      221 ---> 0      222 ---> 0      223 ---> 0      224 ---> 0      225 ---> 0      226 ---> 0      227 ---> 0      228 ---> 0      229 ---> 0
230 ---> 0      231 ---> 0      232 ---> 0      233 ---> 0      234 ---> 0      235 ---> 0      236 ---> 0      237 ---> 0      238 ---> 0      239 ---> 0
240 ---> 0      241 ---> 0      242 ---> 0      243 ---> 0      244 ---> 0      245 ---> 0      246 ---> 0      247 ---> 0      248 ---> 0      249 ---> 0
250 ---> 0      251 ---> 0      252 ---> 0      253 ---> 0      254 ---> 0      255 ---> 155

PC=2 IR=13 TEMP=255 MAR=255 A=155 B=0 CF=0 SP=255


-------------------------------------------------------------------------------------------------------------------

FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2 --> A(E) , B(L), R(reset)


PC=3 IR=1 TEMP=255 MAR=2 A=155 B=155 CF=0 SP=255
```

```
-------------------------------------------------------------------------------------------------------------------

FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2 -->SP(E) , MAR(E,L)
 T4 -->MAR(E) , B(L)
 T5 -->SP(I), R(reset)

100 ---> 0      101 ---> 0      102 ---> 0      103 ---> 0      104 ---> 0      105 ---> 0      106 ---> 0      107 ---> 0      108 ---> 0      109 ---> 0
110 ---> 0      111 ---> 0      112 ---> 0      113 ---> 0      114 ---> 0      115 ---> 0      116 ---> 0      117 ---> 0      118 ---> 0      119 ---> 0
120 ---> 0      121 ---> 0      122 ---> 0      123 ---> 0      124 ---> 0      125 ---> 0      126 ---> 0      127 ---> 0      128 ---> 0      129 ---> 0
130 ---> 0      131 ---> 0      132 ---> 0      133 ---> 0      134 ---> 0      135 ---> 0      136 ---> 0      137 ---> 0      138 ---> 0      139 ---> 0
140 ---> 0      141 ---> 0      142 ---> 0      143 ---> 0      144 ---> 0      145 ---> 0      146 ---> 0      147 ---> 0      148 ---> 0      149 ---> 0
150 ---> 0      151 ---> 0      152 ---> 0      153 ---> 0      154 ---> 0      155 ---> 0      156 ---> 0      157 ---> 0      158 ---> 0      159 ---> 0
160 ---> 0      161 ---> 0      162 ---> 0      163 ---> 0      164 ---> 0      165 ---> 0      166 ---> 0      167 ---> 0      168 ---> 0      169 ---> 0
170 ---> 0      171 ---> 0      172 ---> 0      173 ---> 0      174 ---> 0      175 ---> 0      176 ---> 0      177 ---> 0      178 ---> 0      179 ---> 0
180 ---> 0      181 ---> 0      182 ---> 0      183 ---> 0      184 ---> 0      185 ---> 0      186 ---> 0      187 ---> 0      188 ---> 0      189 ---> 0
190 ---> 0      191 ---> 0      192 ---> 0      193 ---> 0      194 ---> 0      195 ---> 0      196 ---> 0      197 ---> 0      198 ---> 0      199 ---> 0
200 ---> 0      201 ---> 0      202 ---> 0      203 ---> 0      204 ---> 0      205 ---> 0      206 ---> 0      207 ---> 0      208 ---> 0      209 ---> 0
190 ---> 0      191 ---> 0      192 ---> 0      193 ---> 0      194 ---> 0      195 ---> 0      196 ---> 0      197 ---> 0      198 ---> 0      199 ---> 0
200 ---> 0      201 ---> 0      202 ---> 0      203 ---> 0      204 ---> 0      205 ---> 0      206 ---> 0      207 ---> 0      208 ---> 0      209 ---> 0
210 ---> 0      211 ---> 0      212 ---> 0      213 ---> 0      214 ---> 0      215 ---> 0      216 ---> 0      217 ---> 0      218 ---> 0      219 ---> 0
220 ---> 0      221 ---> 0      222 ---> 0      223 ---> 0      224 ---> 0      225 ---> 0      226 ---> 0      227 ---> 0      228 ---> 0      229 ---> 0
230 ---> 0      231 ---> 0      232 ---> 0      233 ---> 0      234 ---> 0      235 ---> 0      236 ---> 0      237 ---> 0      238 ---> 0      239 ---> 0
240 ---> 0      241 ---> 0      242 ---> 0      243 ---> 0      244 ---> 0      245 ---> 0      246 ---> 0      247 ---> 0      248 ---> 0      249 ---> 0
250 ---> 0      251 ---> 0      252 ---> 0      253 ---> 0      254 ---> 0      255 ---> 155

PC=5 IR=12 TEMP=255 MAR=255 A=155 B=155 CF=0 SP=256


-------------------------------------------------------------------------------------------------------------------

<-----------------------------------------------------------------THANK YOU------------------------------------------------------------------>
```

2.INPUT:

AMVI 126
ADI 37
INR -1
PUSH 22
ADM 255
HLT -1

OUTPUT:

```
Valid Instructions are
AMVI---->Load A with data                          AMVI #DATA
BMOV---->Move A to B                               BMOV -1
INR---->Increment Accumulator                      INR -1
ADI---->Add data to Accumulator                    ADI #DATA
SBI---->Subtract data from Accumulator             SBI #DATA
ADM---->Add data present in Memory Location        ADM #ADDRESS
SBM---->Substract data present in Memory Location  SBM #ADDRESS
AND---->Bitwise And operation                      AND -1
XOR---->Bitwise XOR operation                      XOR -1
CLR---->Set value of A to 0                        CLR -1
RSH---->Rotate right                               RSH -1
LSH---->Rotate Left                                LSH -1
STA---->Store Accumulator in Memory Location       STA #ADDRESS
LDA---->Load Accumulator with Memory Location      LDA #ADDRESS
PUSH---->Add data to the stack                     PUSH #DATA
POP---->Remove data from the stack                 POP -1
If the operation operand is implicit then write '-1' in place of data/address.

ENTER YOUR PROGRAM :
AMVI 126
ADI 37
INR -1
PUSH 22
ADM 255
HLT -1
FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2 --> TMP(E) , A(L) ,R(reset)


PC=1 IR=0 TEMP=126 MAR=0 A=126 B=0 CF=0 SP=256
```

```
-----------------------------------------------------------------------------------------------------------------------

FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2 --> TEMP(E) ,A(E) , ALU(L)
 T3 -->ALU(E) , A(L), R(reset)


PC=2 IR=3 TEMP=37 MAR=1 A=163 B=0 CF=0 SP=256


-----------------------------------------------------------------------------------------------------------------------

FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2-->A(E) , ALU(L)
 T3-->ALU(E) , A(L), R(reset)


PC=3 IR=2 TEMP=37 MAR=2 A=164 B=0 CF=0 SP=256


-----------------------------------------------------------------------------------------------------------------------

FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2 -->SP(D)
 T3 -->SP(E) , MAR(E,L)
 T4 -->TEMP(E) , RAM(L), R(reset)

100 ---> 0      101 ---> 0      102 ---> 0      103 ---> 0      104 ---> 0      105 ---> 0      106 ---> 0      107 ---> 0      108 ---> 0      109 ---> 0
110 ---> 0      111 ---> 0      112 ---> 0      113 ---> 0      114 ---> 0      115 ---> 0      116 ---> 0      117 ---> 0      118 ---> 0      119 ---> 0
120 ---> 0      121 ---> 0      122 ---> 0      123 ---> 0      124 ---> 0      125 ---> 0      126 ---> 0      127 ---> 0      128 ---> 0      129 ---> 0
130 ---> 0      131 ---> 0      132 ---> 0      133 ---> 0      134 ---> 0      135 ---> 0      136 ---> 0      137 ---> 0      138 ---> 0      139 ---> 0
140 ---> 0      141 ---> 0      142 ---> 0      143 ---> 0      144 ---> 0      145 ---> 0      146 ---> 0      147 ---> 0      148 ---> 0      149 ---> 0
150 ---> 0      151 ---> 0      152 ---> 0      153 ---> 0      154 ---> 0      155 ---> 0      156 ---> 0      157 ---> 0      158 ---> 0      159 ---> 0
160 ---> 0      161 ---> 0      162 ---> 0      163 ---> 0      164 ---> 0      165 ---> 0      166 ---> 0      167 ---> 0      168 ---> 0      169 ---> 0
170 ---> 0      171 ---> 0      172 ---> 0      173 ---> 0      174 ---> 0      175 ---> 0      176 ---> 0      177 ---> 0      178 ---> 0      179 ---> 0
180 ---> 0      181 ---> 0      182 ---> 0      183 ---> 0      184 ---> 0      185 ---> 0      186 ---> 0      187 ---> 0      188 ---> 0      189 ---> 0
190 ---> 0      191 ---> 0      192 ---> 0      193 ---> 0      194 ---> 0      195 ---> 0      196 ---> 0      197 ---> 0      198 ---> 0      199 ---> 0
200 ---> 0      201 ---> 0      202 ---> 0      203 ---> 0      204 ---> 0      205 ---> 0      206 ---> 0      207 ---> 0      208 ---> 0      209 ---> 0
210 ---> 0      211 ---> 0      212 ---> 0      213 ---> 0      214 ---> 0      215 ---> 0      216 ---> 0      217 ---> 0      218 ---> 0      219 ---> 0
220 ---> 0      221 ---> 0      222 ---> 0      223 ---> 0      224 ---> 0      225 ---> 0      226 ---> 0      227 ---> 0      228 ---> 0      229 ---> 0
230 ---> 0      231 ---> 0      232 ---> 0      233 ---> 0      234 ---> 0      235 ---> 0      236 ---> 0      237 ---> 0      238 ---> 0      239 ---> 0
240 ---> 0      241 ---> 0      242 ---> 0      243 ---> 0      244 ---> 0      245 ---> 0      246 ---> 0      247 ---> 0      248 ---> 0      249 ---> 0
250 ---> 0      251 ---> 0      252 ---> 0      253 ---> 0      254 ---> 0      255 ---> 22


PC=4 IR=14 TEMP=22 MAR=255 A=164 B=0 CF=0 SP=255


-----------------------------------------------------------------------------------------------------------------------

FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2 -->TEMP(E) , MAR(E,L)
 T3 -->RAM(E) , TEMP(L)
 T4 --> TEMP(E) ,A(E) , ALU(L)
 T5 -->ALU(E) , A(L), R(reset)


PC=5 IR=4 TEMP=22 MAR=255 A=186 B=0 CF=0 SP=255


-----------------------------------------------------------------------------------------------------------------------

<----------------------------------------------------------THANK YOU---------------------------------------------------------->
```

3.INPUT:

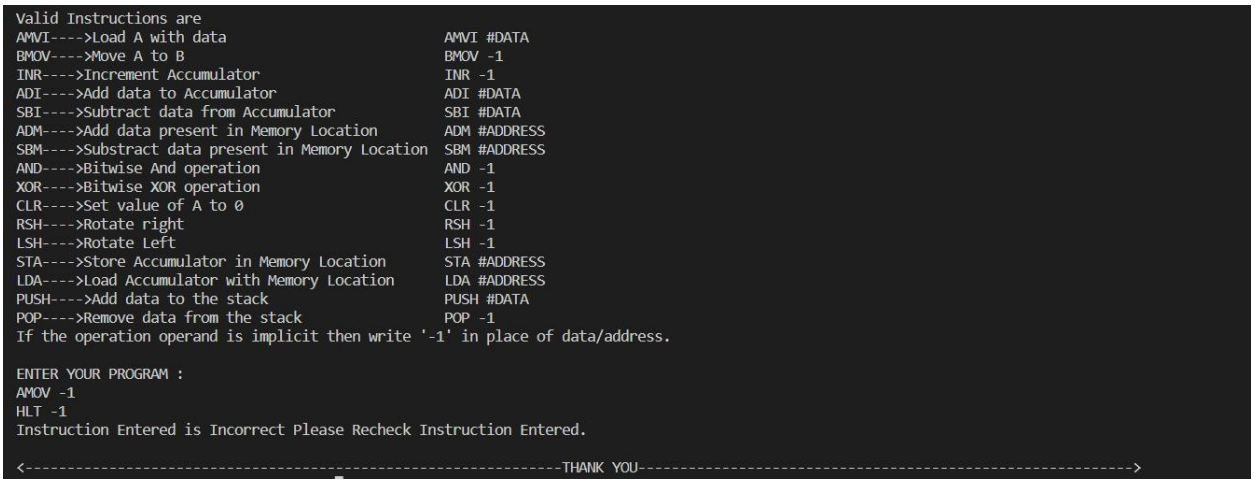
AMVI 52
BMOV -1
CLR -1
AMVI 24
AND -1
LSH -1
HLT -1

OUTPUT:

```
Valid Instructions are
AMVI---->Load A with data                       AMVI #DATA
BMOV---->Move A to B                            BMOV -1
INR---->Increment Accumulator                   INR -1
ADI---->Add data to Accumulator                 ADI #DATA
SBI---->Subtract data from Accumulator          SBI #DATA
ADM---->Add data present in Memory Location     ADM #ADDRESS
SBM---->Substract data present in Memory Location  SBM #ADDRESS
AND---->Bitwise And operation                   AND -1
XOR---->Bitwise XOR operation                   XOR -1
CLR---->Set value of A to 0                     CLR -1
RSH---->Rotate right                            RSH -1
LSH---->Rotate Left                             LSH -1
STA---->Store Accumulator in Memory Location    STA #ADDRESS
LDA---->Load Accumulator with Memory Location   LDA #ADDRESS
PUSH---->Add data to the stack                  PUSH #DATA
POP---->Remove data from the stack              POP -1
If the operation operand is implicit then write '-1' in place of data/address.

ENTER YOUR PROGRAM :
AMVI 52
BMOV -1
CLR -1
AMVI 24
AND -1
LSH -1
HLT -1
FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2 --> TMP(E) , A(L) ,R(reset)


PC=1 IR=0 TEMP=52 MAR=0 A=52 B=0 CF=0 SP=256


--------------------------------------------------------------------------------------------------------------------------
```

```
FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2 --> A(E) , B(L), R(reset)


PC=2 IR=1 TEMP=52 MAR=1 A=52 B=52 CF=0 SP=256


-------------------------------------------------------------------------------------------------------------------------

FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2-->A(E) , ALU(L)
 T3-->ALU(E) , A(L), R(reset)


PC=3 IR=9 TEMP=52 MAR=2 A=0 B=52 CF=0 SP=256


-------------------------------------------------------------------------------------------------------------------------

FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2 --> TMP(E) , A(L) ,R(reset)


PC=4 IR=0 TEMP=24 MAR=3 A=24 B=52 CF=0 SP=256

FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2 --> B(E) ,A(E) , ALU(L)
 T3 -->ALU(E) , A(L), R(reset)


PC=5 IR=7 TEMP=24 MAR=4 A=16 B=52 CF=0 SP=256


-------------------------------------------------------------------------------------------------------------------------

FETCH CYCLES :-

 T0 --> PC(E) , MAR(E,L)
 T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)

EXECUTION CYCLES :-

 T2-->A(E) , BS(L)
 T3-->BS(E) , A(L), R(reset)


PC=6 IR=11 TEMP=24 MAR=5 A=32 B=52 CF=0 SP=256


-------------------------------------------------------------------------------------------------------------------------

<----------------------------------------------------------THANK YOU---------------------------------------------------------->
```

SANDL processor also checks if the input given by user is valid or invalid as it accepts string as input.



MRI

Memory reference instructions are those commands or instructions which are in the custom to generate a reference to the memory and approval to a program to have an approach to the commanded information and that states as to from where the data is cache continually. These instructions are known as Memory Reference Instructions.

- For ADM,opcode-0100 -- ADM #ADDRESS

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2 -->TEMP(E) , MAR(E,L)
  T3 -->RAM(E) , TEMP(L)
  T4 --> TEMP(E) ,A(E) , ALU(L)
  T5 -->ALU(E) , A(L) ,R(reset)

- For SBM,opcode-0110 -- SBM #ADDRESS

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2 -->TEMP(E) , MAR(E,L)
  T3 -->RAM(E) , TEMP(L)
  T4 --> TEMP(E) ,A(E) , ALU(L)
  T5 -->ALU(E) , A(L) ,R(reset)

- For STA,opcode-1100 -- STA #ADDRESS

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2 -->TEMP(E) , MAR(E,L)
  T3 -->RAM(L) , A(E) ,R(reset)

- For LDA,opcode-1101 -- LDA #ADDRESS

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2 -->TEMP(E) , MAR(E,L)
  T3 -->RAM(E) , A(L) ,R(reset)

Non-MRI

Non memory reference instruction, an instruction that can be carried out without having to obtain an operand from, or return a result to, memory. Immediate instructions and some branch instructions are examples. See also zero-address instruction.

In SANDL Processor we use "-1" where there is no data or address as input. Non-MRI instructions , If the operation operand is implicit then "-1" is written so as to know that there is no data or address.

➢ For BMOV,opcode-0001 -- BMOV -1

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2 --> A(E) , B(L) ,R(reset)

➢ For INR,opcode-0010 -- INR -1

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2-->A(E) , ALU(L)
  T3-->ALU(E) , A(L) ,R(reset)

➢ For AND,opcode-0111 -- AND -1

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2 --> B(E) ,A(E) , ALU(L)
  T3 -->ALU(E) , A(L) ,R(reset)

➢ For XOR,opcode-1000 -- XOR -1

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2 --> B(E) ,A(E) , ALU(L)
  T3 -->ALU(E) , A(L) ,R(reset)

➢ For CLR,opcode-1001 -- CLR -1

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2-->A(E) , ALU(L)
  T3-->ALU(E) , A(L) ,R(reset)

- For RSH,opcode-1010 -- RSH -1

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2-->A(E) , BS(L)
  T3-->BS(E) , A(L) ,R(reset)

- For LSH,opcode-1011 -- LSH -1

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2-->A(E) , BS(L)
  T3-->BS(E) , A(L) ,R(reset)

- For POP,opcode-1111 -- POP -1

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2 -->SP(E) , MAR(E,L)
  T3 -->MAR(E) , B(L)
  T4 -->SP(I) ,R(reset)

IMMEDIATE AND REGISTER ADDRESSING

Register and immediate addressing means that the operand is either a register or specified as a constant within the assembly language instruction itself. In register addressing, an operand is fetched from, or written to, a register.

- For ADI,opcode-0011 -- ADI #DATA

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2 --> TEMP(E) ,A(E) , ALU(L)
  T3 -->ALU(E) , A(L) ,R(reset)

- For SBI,opcode-0101 -- SBI #DATA

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2 --> TEMP(E) ,A(E) , ALU(L)
  T3 -->ALU(E) , A(L) ,R(reset)

- For AMVI,opcode-0000 -- AMVI #DATA

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2 --> TMP(E) , A(L) ,R(reset)

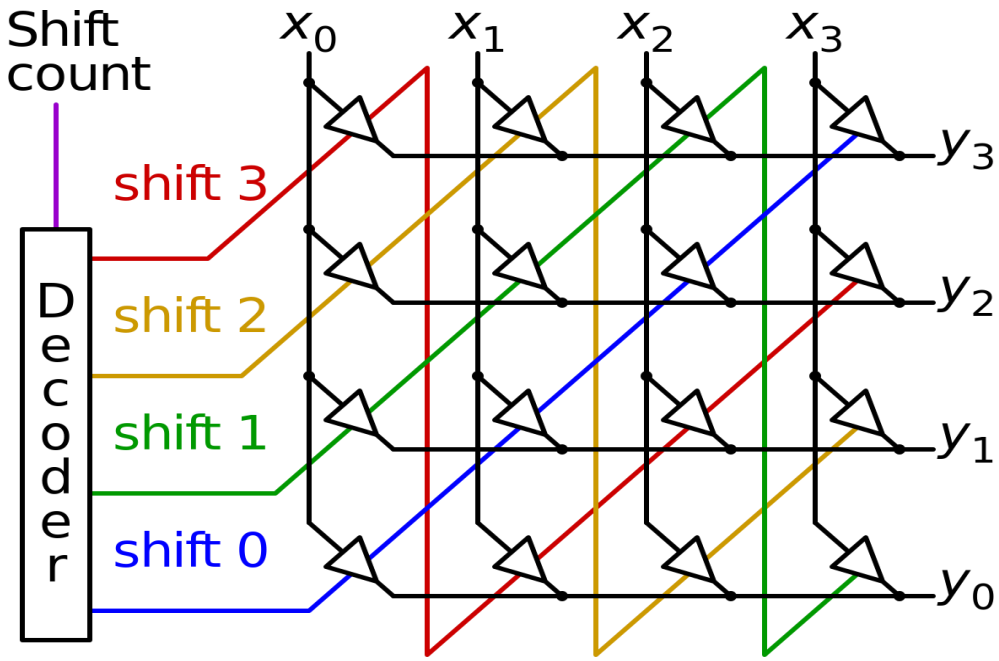
- For PUSH,opcode-1110 -- PUSH #DATA

- T0 --> PC(E) , MAR(E,L)
  T1 --> PC(I) , RAM(E) , IR(L) , TMP(L)
  T2 -->SP(D)
  T3 -->SP(E) , MAR(E,L)
  T4 -->TEMP(E) , RAM(L) ,R(reset)

BARREL SHIFTER

A Barrel shifter is a digital circuit that can shift a data word by a specified number of bits without the use of any sequential logic, only pure combinational logic, i.e., it inherently provides a binary operation. The simplest way of achieving this is by using a series of multiplexers where one output is connected to the input of the next multiplexer in the chain, in a specific manner that depends on the amount of shift specified.

A barrel shifter is often used to shift and rotate n-bits in modern microprocessors, typically within a single clock cycle.



Clear operation : This operation compares the bits in A and B and produces an all 0's result if the two number are equal.

In SANDL Processor Clear operation used to clear the given register that is to make the value of the register to zero.

Stack operations

The base register is usually the stack pointer, SP. This means that you can use these instructions to implement push and pop operations for any number of registers in a single instruction. The load and store multiple instructions can be used with several types of stacks: Descending or ascending.

Here we use PUSH and POP operations.

Push which adds an element to the collection.

Pop which removes the most recently added element that was not yet removed.

In SANDL Processor we also have Load Store operations such as LDA and STA which Loads the register and stores the register respectively.

RESET CLOCK

A reset clears any pending errors or events and brings a system to normal condition or an initial state, usually in a controlled manner. In SANDL Processor Reset starts the counter or clock from beginning to use the clock cycles efficiently.

CONCLUSION

SANDL Processor deals with instruction set of size 12-bit using RAM 256*12bit.For one instruction the given instruction set is decoded as 4-bit opcode and 8-bit data or address. The 4 bits of opcode is used to identify the type of instruction we are executing. Then the last 8-bit is used for address handling while dealing with MRI instructions and in the case of non-MRI instructions it signifies the type of instructions we are dealing and in the case of immediate these 8 bits take 8-bit immediate value and we execute the instruction. Immediate and Register Instructions takes the 8-bit and performs the operations.

SANDL Processor satisfies the user requirements by allowing different types of instructions to work with. It simply works on the instructions based on the types of operations.

SANDL Processor also deals with Arithmetic Operations like ADD, SUBTRACT and Logical Operations like AND,XOR(exclusive-or) and it also enables us to perform Shifting operations like left and right shifts.

Hence SANDL Processor takes basic computer instructions and converts them into a pattern of bits used to perform basic operations based on one-to-one mapping using vector.

(SANDL Processor is named after the members of the team - **S**uneel , **A**ryan , **N**ithya , **D**ileep , **L**ikitha)

THANK TOU!