Arithmetic and Logic Micro Operations

(Core Course Computer System Architecture unit-2)

E- content for B.A./B.Sc. in Computer Applications Course, Semester-II

By
Sabitri Sharma
Visiting faculty, Magadh Mahila College
E-mail sabitrisharma@gmail.com

In computer central processing units, micro-operations are detailed low-level instructions used in some designs to implement complex machine instructions .

Usually, micro-operations perform basic operations on data stored in one or more registers. It also transfers data between registers or between registers and external buses of the central processing unit, and performs arithmetic or logical operations on registers. In a typical fetch decode and execute cycle, each step of a macro-instruction is decomposed during its execution so the CPU determines and steps through a series of micro-operations. The execution of micro-operations is performed under control of the CPU's control unit, which decides on their execution while performing various optimizations such as reordering, fusion and caching

Types of Micro Operation

- Register Transfer Micro Operation
- Arithmetic (Addition, subtraction etc..) Micro Operation
  Data is numeric, and bits with a word are interdependent.
- Logic (AND, OR etc.) Micro Operation
  Data is not numeric, and bits are independent of each other. The same logical operation is applied to each bit in a word in parallel.
- Shift. Micro Operation

  Data may or may not be numeric. All bits are moved the same number of positions left or right.

Table-1 Arithmetic Microoperations

Example	Description
R3 ← R1 + R2	Addition contents of R1 plus R2 transferred to R3
R3 ← R1 - R2 , R3← R1 + R2' + 1	Subtraction contents of R1 minus R2 transferred to R3
R2 ← R2'	Complement (really a logic operation)
R2 ← R2' + 1)	Negation
R1 ← R1 + 1	Increment the content of R1 by one
R1 ← R1 - 1	Decrement Increment the content of R1 by one

Increment and decrement can be done with combinational increments and decrements, counter registers, or by adding a 1.

Multiply and divide are not often implemented as microoperations due to the amount of time they require. They are usually implemented as a multi-clock-cycle routine of shifts and adds.

Logic Microoperations
Logic Microoperations specify binary operations performed for strings of bits in registers. These operations consider each bit of the register separately and treat them as binary variables.

Example

R3←R1 ⊕ R2

```
                R1     1 0 1 0

                 R2 ⊕ 1 1 0 0

    R1 after P=1.           0 1 1 0
```

Where P is a control function

Table -2 Logic Micro Operations

Boolean Function	**Microoperation**	**Name**
$F_0 = 0$	$F \leftarrow 0$	Clear
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge B'$	
$F_3 = x$	$F \leftarrow A$	Transfer A
$F_4 = x'y$	$F \leftarrow A' \wedge B$	
$F_5 = y$	$F \leftarrow B$	Transfer B
$F_6 = x \oplus y$	$F \leftarrow$ Å $\oplus$ B	Exclusive-OR
$F_7 = x + y$	$F \leftarrow A \vee B$	OR
$F_8 = (x + y)'$	$F \leftarrow (A \vee B)'$	NOR
$F_9 = (x \oplus y)'$	$F \leftarrow (A \oplus B)'$	Exclusive-NOR
$F_{10} = y'$	$F \leftarrow B'$	Complement B
$F_{11} = x + y'$	$F \leftarrow A \vee B'$	A OR Complement B
$F_{12} = x'$	$F \leftarrow A'$	Complement A
$F_{13} = x' + y$	$F. \leftarrow A' \vee B$	
$F_{14} = (xy)'$	$F \leftarrow (A \wedge B)'$	NAND
$F_{15} = 1$	$F \leftarrow$ 1`s	Set to all 1`s

Thus, there are 16 different logic operations with 2 binary variables.

Truth Table Logic Micro Operations

X	Y	F0	F1	F2	F3	F4	F5	F6	F7
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1
X	**Y**	**F8**	**F9**	**F10**	**F11**	**F12**	**F13**	**F14**	**F15**
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

Consider the following logic micro operation

F0= 0000 from truth table and also F0=0 in logic micro operation table

For F1= XY logic micro operation

F1 =0001 from the truth table. Now look at the truth table, 1 in F1 corresponds to the variables X=1 and Y=1

Therefore, for F1= XY AND operation, here X=1 and Y=1 after performing AND operation the result is stored in F1.

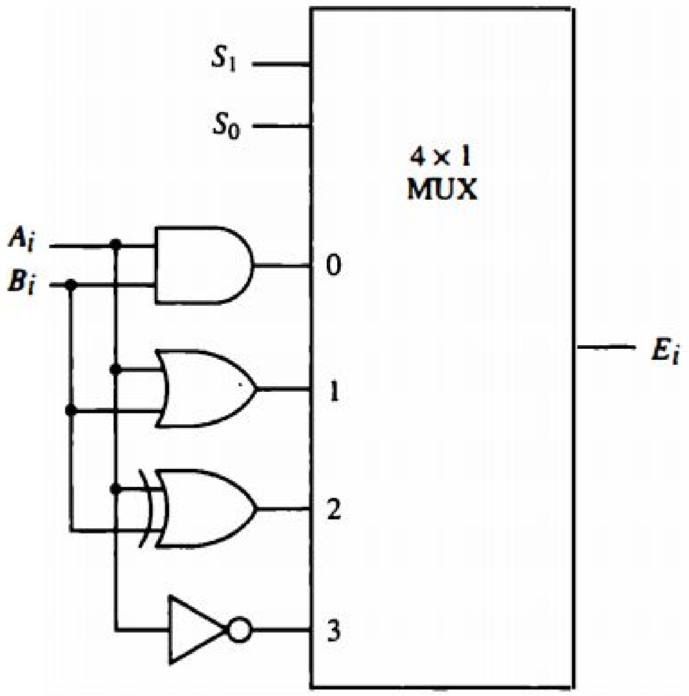
Hence F1= 0001

Similarly, F3= XY' + XY= X(Y' + Y)=X

Similarly it can be checked for all fifteen logic operations.

Logic diagram of Hardware Implementation of Logic Circuit

Figure One stage of logic circuit.



(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \overline{A}$	Complement

(b) Function table

Here only one stage is defined for 1 bit. If there are n number of bits then n such stages has to be defined.

Applications of Logic Microoperations

1. **Selective Set Operation** - It sets 1 to the bits in register A where there are corresponding 1's in register B.It does not affect bit positions that have 0's in B. The OR microoperation can be used to selectively set bits of a register.

   Example:

   consider register A contains 1010 and register B contains 1100. The bits in A corresponding to the bit 1 in the register B will be changed to 1. Therefore, bits 1 and 0 in the register A corresponding to the bits 1 and 1 in the register B will be changed to 1 and 1.

   1 0 1 0. A before
   1 1 0 0 B(logic operand)
   1 1 1 0 A after

2. **Selective Complement Operation**

   The selective Complement operation complements bits in A where there are corresponding 1's in B. It does not affect bit positions that have 0's in B. The exclusive OR microoperation can be used to selectively set bits of a register.

   Example:

1 0 1 0.	A before
1 1 0 0	B(logic operand)
0 1 1 0	A after

3. **Selective Clear Operation**

   The selective clear operation clears to 0 the bits in A where there are corresponding 1's in B. It does not affect bit positions that have 0's in B. The selective clear operation can be achieved by the microoperation $A \leftarrow A \wedge B'$

   Example:

1 0 1 0.	A before
1 1 0 0	B(logic operand)
0 0 1 0	A after

4. **Mask operation** - It is similar to selective clear operation except that the bits of A are cleared only where there are corresponding 0's in B.

   The mask operation is an AND Micro Operation.

   Example:

1 0 1 0.	A before
1 1 0 0	B(logic operand)
1 0 0 0	A after

5. **Insert operation** - The insert operation inserts a new value into a group of bits. It incorporates two operations that are masking and then ORing them with required value. The mask operation is an AND microoperation and the insert operation is an OR microoperation.

   Example:

   Suppose the binary number 1001 is to be inserted in place of 0110

   Let the register A contain the bit 0110 1010.

   0 1 1 0 is replaced by 0 0 0 0 and remaining bits of A will be 1 1 1 1

Mask

A   0 1 1 0   1 0 1 0

B.   0 0 0 0   1 1 1 1.   (In register B the position of bits, wherever the new bits are to be inserted in the register A are filled with zeros and remaining bits are 1)

After performing AND operation between the contents of the registers A and B, the output will be as follows

Output A   0 0 0 0   1 01 0.   AND microoperation

Now, the new bits to be inserted in register A are placed in register B at the masked bit position and remaining bits of register B are zero. Then, OR operation is performed between the contents of register A and register B.

```
A     0 0 0 0   1 0 1 0
B.    1 0 0 1.  0 0 0 0.  OR microoperation
      1 0 0 1   1 0 1 0
```

6. **Clear operation** - The clear operation compares the words in A and B and produces an all 0's result if the two numbers are equal. It is achieved by Exclusive OR operation.

Example:

```
1 0 1 0.   A
1 0 1 0.   B
0 0 0 0.
```