LOGIC MICROOPERATIONS

- Logic micro operations specify binary operations for strings of bits stored in registers.

- These operations consider each bit of the register separately and treat them as binary variables.

- For example, the exclusive-OR micro operation with the contents of two registers R1 and R2 is symbolized by the statement

$$P: R1 \leftarrow R1 \oplus R2$$

- Provided that the control variable P = 1. Let the content of R1 be 1010 and the content of R2 be 1100. The exclusive-OR micro operation stated above symbolizes the following logic computation:

```
 1010 Content of R1
 1100 Content of R2
 ----
 0110 Content of R1 after P = 1
```

Special symbols

- The **Symbol** $\vee$ will be used to denote an **OR micro operation** and the **symbol** $\wedge$ to denote an **AND micro operation.**

- **The complement** micro operation is the same as the 1's complement and **uses a bar on top of the symbol** that denotes the register name.

LIST OF LOGIC MICROOPERATIONS

- There are 16 different logic operations that can be performed with two binary variables. They can be determined from all possible truth tables obtained with two binary variables as shown below.

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- Each of the 16 columns F0 through F15 represents a truth table of one possible Boolean function for the two variables x and y

- The 16 logic micro operations are derived from these functions by replacing variable x by the binary content of register A and variable y by the binary content of register B.

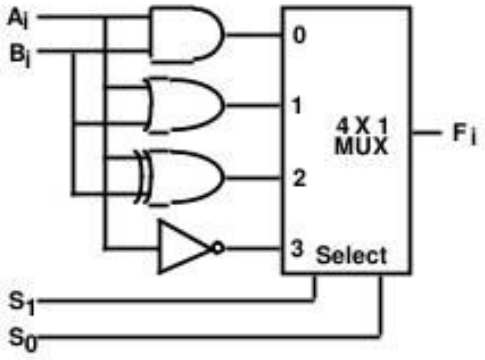
Boolean Function	Micro-Operations	Name
F0 = 0	F ← 0	Clear
F1 = xy	F ← A ∧ B	AND
F2 = xy'	F ← A ∧ B'	
F3 = x	F ← A	Transfer A
F4 = x'y	F ← A'∧ B	
F5 = y	F ← B	Transfer B
F6 = x ⊕ y	F ← A ⊕ B	Exclusive-OR
F7 = x + y	F ← A ∨ B	OR
F8 = (x + y)'	F ← (A ∨ B)'	NOR
F9 = (x ⊕ y)'	F ← (A ⊕ B)'	Exclusive-NOR
F10 = y'	F ← B'	Complement B
F11 = x + y'	F ← A ∨ B	
F12 = x'	F ← A'	Complement A
F13 = x' + y	F ← A'∨ B	
F14 = (xy)'	F ← (A ∧ B)'	NAND
F15 = 1	F ← all 1's	Set to all 1's

HARDWARE IMPLEMENTATION

- The hardware implementation of logic micro operations requires that logic gates be inserted for each bit or pair of bits in the registers to perform the required logic function.

- Although there are 16 logic micro operations, most computers use only four --- AND, OR, XOR (exclusive-OR), and complement by which all others can be derived.

HARDWARE IMPLEMENTATION OF LOGIC MICROOPERATIONS



Function table

S_1	S_0	Output	μ-operation
0	0	$F = A \wedge B$	AND
0	1	$F = A \vee B$	OR
1	0	$F = A \oplus B$	XOR
1	1	$F = A'$	Complement

- It consists of four gates and a multiplexer. Each of the four logic operations is generated through a gate that performs the required logic. The outputs of the gates are applied to the data inputs of the multiplexer.

- The two selection inputs S1 and S0 choose one of the data inputs of the multiplexer and direct its value to the output.

SHIFT MICRO OPERATIONS

- Shift micro operations are used for serial transfer of data. The contents of a register can be shifted to the left or the right.

Logical shift

- A logical shift is one that transfers 0 through the serial input. **The symbols shl and shr for logical shift-left and shift-right micro operations.**

- For example:

  R1 ← Shl R1
  R2 ← shr R2

Circular shift

- The circular shift (also known as a rotate operation) circulates the bits of the register around the two ends without loss of information.

- This is accomplished by connecting the serial output of the shift register to its serial input. We will use the symbols **cil and cir for the circular shift left and right,** respectively.

Symbolic designation	Description
$R \leftarrow \text{shl } R$	Shift-left register R
$R \leftarrow \text{shr } R$	Shift-right register R
$R \leftarrow \text{cil } R$	Circular shift-left register R
$R \leftarrow \text{cir } R$	Circular shift-right register R
$R \leftarrow \text{ashl } R$	Arithmetic shift-left R
$R \leftarrow \text{ashr } R$	Arithmetic shift-right R

Arithmetic shift

- An arithmetic shift is a micro operation that shifts a signed binary number to the left or right.
- An arithmetic shift-left multiplies a signed binary number by 2. An arithmetic shift-right divides the number by 2. Arithmetic shifts leave the sign bit unchanged.
- The leftmost bit in a register holds the sign bit, and the remaining bits hold the number. **The sign bit is 0 for positive and 1 for negative.**

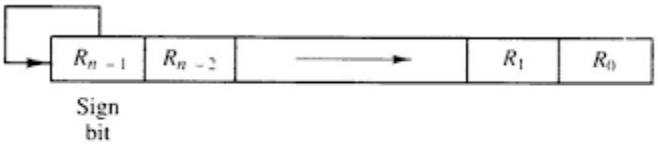
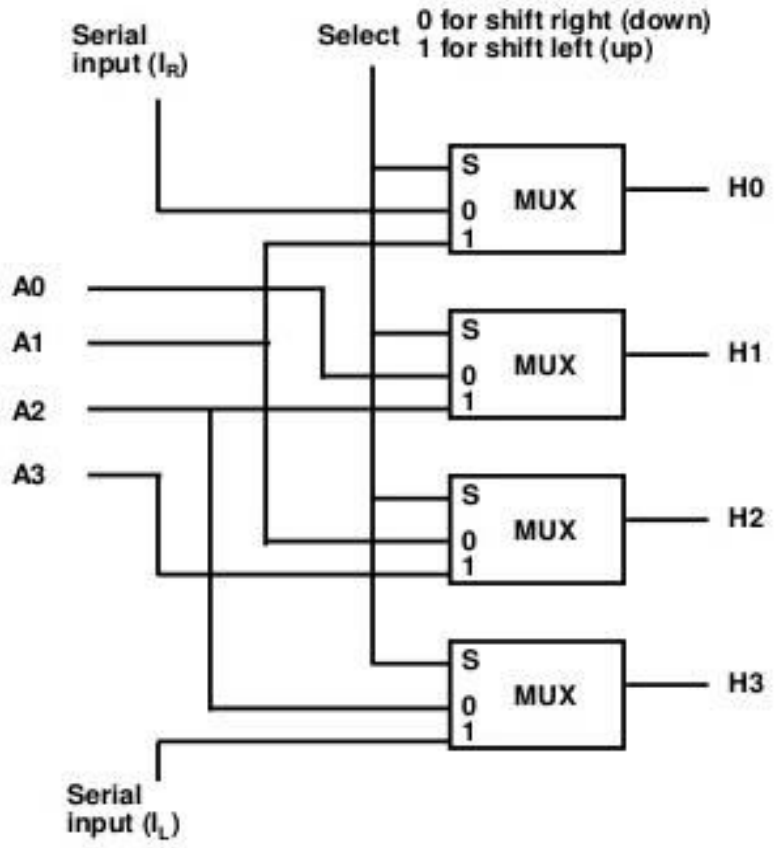


Figure 4-11 Arithmetic shift right.

- Bit R_{n-1} in the leftmost position holds the sign bit. R_{n-2} is the most significant bit of the number and R0 is the least significant bit.

- R_{n-1} remains the same, R_{n-2} receive the bit from R_{n-1}, and so on for the other bits in the register. The bits in R0 are lost.

Hardware Implementation



Truth table

FUNCTIONAL TABLE				
SELECT	OUTPUT			
S	H0	H1	H2	H3
0	IR	A0	A1	A2
1	A1	A2	A3	IL

- The 4 bit shifter has four data inputs A0 to A3 and four data outputs H0 to H3.

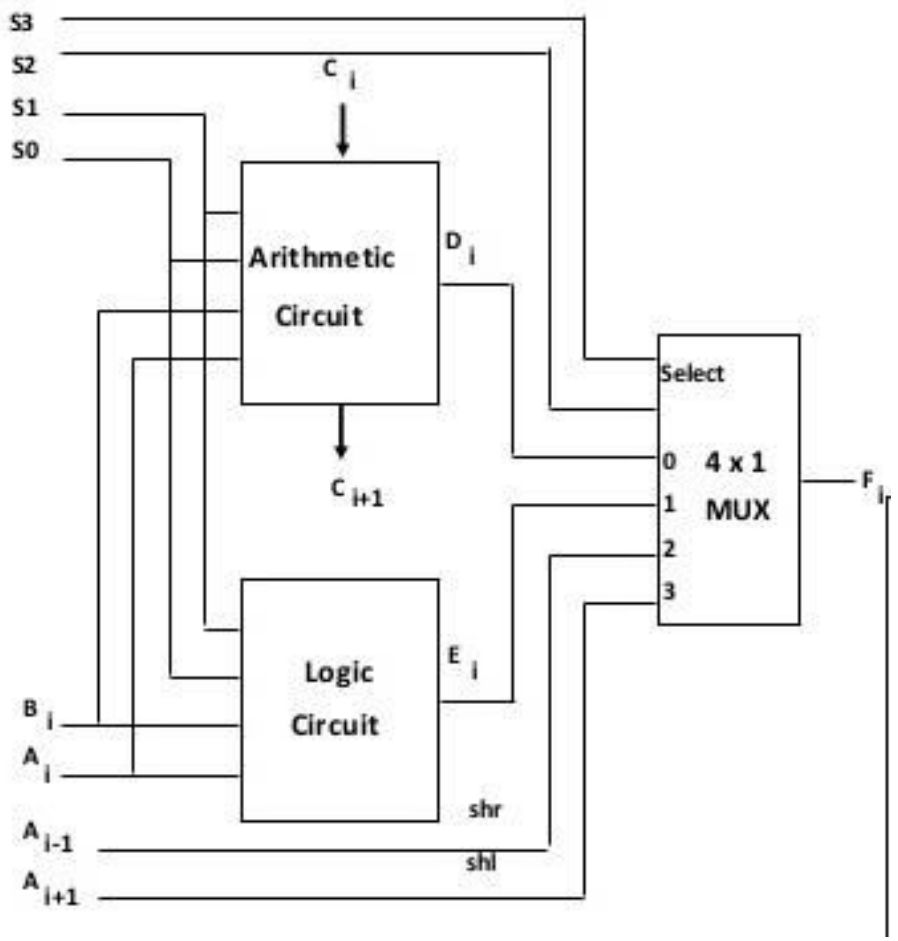
- There are two serial inputs one for shift left(IL) and other for shift right(IR).

- When selection input S=0,the input data are shifted right(down). When S=1,the input data are shifted left(up).

Arithmetic Logic Shift Unit

- Instead of having individual registers performing the micro operations directly, computer systems employ a number of storage registers connected to a common operational unit called an arithmetic logic unit (ALU).

- The ALU is a combinational circuit so that the entire register transfer operation from the source registers through the ALU and into the destination register can perform during one clock pulse period.



- The subscript i designates a typical stage. Inputs Ai and Bi applied to both the arithmetic and logic units.

- A particular micro operation selected with inputs S1 and S0.

- A 4 x 1 multiplexer at the output chooses between an arithmetic output in Di and a logic output in Ei .

- The data in the multiplexer selected with inputs S3 and S2.

- The other two data inputs to the multiplexer receive inputs A_{i-1} for the shift-right operation and A_{i+1} for the shift-left operation.

- The outputs carry C_{i+1} of a given arithmetic stage must connect to the input carry C_{in} of the next stage in sequence.

Operation Select					Operation	Function
S_3	S_2	S_1	S_0	C_{in}		
0	0	0	0	0	F = A	Transfer A
0	0	0	0	1	F = A + 1	Increment A
0	0	0	1	0	F = A + B	Addition
0	0	0	1	1	F = A + B + 1	Add with carry
0	0	1	0	0	$F = A + \overline{B}$	Subtract with borrow
0	0	1	0	1	$F = A + \overline{B} + 1$	Subtraction
0	0	1	1	0	F = A - 1	Decrement A
0	0	1	1	1	F = A	Transfer A
0	1	0	0	X	$F = A \wedge B$	AND
0	1	0	1	X	$F = A \vee B$	OR
0	1	1	0	X	$F = A \oplus B$	XOR
0	1	1	1	X	$F = \overline{A}$	Complement A
1	0	X	X	X	F = shr A	Shift right A into F
1	1	X	X	X	F = shl A	Shift left A into F

- The first eight are arithmetic operations and selected with S3S2= 00.
- Also, the next four logic operations selected with S3S2 = 01. The input carry has no effect on the logic operations and marked with don't-care X's.
- Moreover, the last two operations are shift operations and selected with S3S2 = 10 and 11.
- The other three selection inputs have no effect on the shift.