Module 2:
Data Representation and Arithmetic Algorithms

Prof. Datta Deshmukh

Binary operations: 1's & 2's complements

1's complement:

Each binary digit is subtracted from '1' (simply the bit is inverted)

$(1010)_2 \rightarrow 0101$

$(13.375)_{10} = (1101.011)_2 \rightarrow 0010.100$

$(35.71)_8 = (011101.111001)_2 \rightarrow 100010.000110$

2's complement:

2's complement = 1's complement + 1

('1' added to LSB without consideration of the binary point)

$(1010)_2$: 1's compl.= 0101

2's compl. = 0101 + 1 → 0110

$(13.375)_{10} = (1101.011)_2$

1's complement = 0010.100

2's complement = 0010.100 + 1 → 0010.101

Binary addition rules

0 + 0 : Sum = 0 & Carry = 0

0 + 1 : Sum = 1 & Carry = 0

1 + 0 : Sum = 1 & Carry = 0

1 + 1 : Sum = 0 & Carry = 1

Sometimes,

1 + 1 + 1 : Sum = 1 & Carry = 1

Binary addition example

$(1101.011)_2 + (100.0011)_2$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
\hline
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
\hline
 & & & & & & & & 1 & &
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
\hline
 & & & & & & 0 & 1 & & &
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{cccccccccc}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
 & & & & & & 1 & & & & \\
\hline
 & & & & & & & 0 & 1 & &
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
 & & & & & & 1 & & & & \\
\hline
 & & & & & & 0 & 0 & 1 & &
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rccccccccl}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
 & & & & 1 & 1 & & & & & \\
\hline
 & & & & & & 0 & 0 & 1 & &
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
 & & & & 1 & 1 & & & & & \\
\hline
 & & & & 1 & 0 & 0 & 1 & & &
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rccccccccl}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
 & & & & 1 & 1 & & & & & \\
\hline
 & & & & & . & 1 & 0 & 0 & 1 &
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
 & & & & & & & & & & \\
 & & & 1 & 1 & & & & & & \\
\hline
 & & & & 1 & . & 1 & 0 & 0 & 1 &
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
 & & & & 1 & 1 & & & & & \\
\hline
 & & 0 & 1 & & . & 1 & 0 & 0 & 1 &
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rccccccccl}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
 & & & & 1 & 1 & & & & & \\
\hline
 & & 0 & 0 & 1 & . & 1 & 0 & 0 & 1 &
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{cccccccccc}
  & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & \quad : \text{Number } ① \\
+ &   &   & 1 & 0 & 0 & . & 0 & 0 & 1 \; 1 \quad : \text{Number } ②
\end{array}
$$

1   1 1

0 0 1 . 1 0 0 1

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } 1 \\
+ & & & 1 & 0 & 0 & . & 0 & 0 & 1 \; 1 & : \text{Number } 2 \\
\end{array}
$$

1   1 1

---

0 0 0 1 . 1 0 0 1

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rcccccccccl}
 & & & & & & & & & & \\
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
1 & 1 & & & & 1 & 1 & & & & \\
\hline
 & 0 & 0 & 0 & 1 & . & 1 & 0 & 0 & 1 &
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & & 1 & 0 & 0 & . & 0 & 0 & 1 & 1 & : \text{Number } ② \\
1 \; 1 & & & & & & 1 & 1 & & & \\
\hline
1 & 0 & 0 & 0 & 1 & . & 1 & 0 & 0 & 1 &
\end{array}
$$

Binary addition example

$(1101.011)_2 + (100.0011)_2$

1 1 0 1 . 0 1 1 : Number ①

\+ 1 0 0 . 0 0 1 1 : Number ②

1 1 1 1 : Carry

---

1 0 0 0 1 . 1 0 0 1

Binary addition example

$(1101.011)_2 + (100.0011)_2$

	1	1	0	1	.	0	1	1		: Number 1
+		1	0	0	.	0	0	1	1	: Number 2
1	1			1	1					: Carry
1	0	0	0	1	.	1	0	0	1	

$(1101.011)_2 + (100.0011)_2 = (10001.1001)_2$

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

$$
\begin{array}{rcccccccl}
 & 1 & 0 & 1 & . & 0 & 1 & 1 & : \text{Number } ① \\
+ & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 \; : \text{Number } ②
\end{array}
$$

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

1 0 1 . 0 1 1 : Number ①

\+ 1 0 0 1 . 1 1 1 : Number ②

0

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

$$
\begin{array}{lcccccccccl}
 & 1 & 0 & 1 & . & 0 & 1 & 1 & & & : \text{Number } ① \\
+ & 1 & 0 & 0 & & 1 & . & 1 & 1 & 1 & : \text{Number } ② \\
 & & & & & 1 & & & & & \\
\hline
 & & & & & & 0 & & & &
\end{array}
$$

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

$$\begin{array}{rccccccccl} & & 1 & 0 & 1 & . & 0 & 1 & 1 & : \text{Number } ① \\ + & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } ② \\ & & & & & 1 & & & & \\ \hline & & & & & 1 & 0 & & & \end{array}$$

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

$$
\begin{array}{rcccccccl}
 & 1 & 0 & 1 & . & 0 & 1 & 1 & : \text{Number } ① \\
+ \; 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } ② \\
 & & & 1 & 1 & & & & \\
\hline
 & & & & 1 & 0 & & &
\end{array}
$$

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

		1	0	1	.	0	1	1		: Number ①
+	1	0	0		1	.	1	1	1	: Number ②
					1	1				
					0	1	0			

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

		1	0	1	.	0	1	1	: Number ①
+	1	0	0	1	.	1	1	1	: Number ②
				1	1				
					.	0	1	0	

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

$$
\begin{array}{rcccccccl}
 & & 1 & 0 & 1 & . & 0 & 1 & 1 \quad : \text{Number } ① \\
+ & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 \quad : \text{Number } ② \\
 & & & 1 & 1 & 1 & & & \\
\hline
 & & & & & . & 0 & 1 & 0
\end{array}
$$

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

$$
\begin{array}{rcccccccl}
 & & 1 & 0 & 1 & . & 0 & 1 & 1 \quad : \text{Number } ① \\
+ & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 \quad : \text{Number } ② \\
 & & & & 1 & 1 & 1 & & \\
\hline
 & & & & 1 & . & 0 & 1 & 0
\end{array}
$$

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

$$
\begin{array}{rccccccccc}
 & & 1 & 0 & 1 & . & 0 & 1 & 1 & : \text{Number } ① \\
+ & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } ② \\
 & & & 1 & 1 & 1 & 1 & & & \\
\hline
 & & & & 1 & . & 0 & 1 & 0 &
\end{array}
$$

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

$$
\begin{array}{rcccccccccl}
 & & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & & : \text{Number } ② \\
 & & 1 & 1 & 1 & 1 & & & & & \\
\hline
 & & 1 & 1 & & . & 0 & 1 & 0 & &
\end{array}
$$

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

1 0 1 . 0 1 1 : Number ①

\+ 1 0 0 1 . 1 1 1 : Number ②

1 1 1 1

1 1 1 . 0 1 0

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

$$
\begin{array}{rcccccccl}
 & & 1 & 0 & 1 & . & 0 & 1 & 1 \quad : \text{Number } ① \\
+ & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 \quad : \text{Number } ② \\
 & & 1 & 1 & 1 & 1 & & & \\
\hline
 & 1 & 1 & 1 & 1 & . & 0 & 1 & 0
\end{array}
$$

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

$$
\begin{array}{rcccccccl}
 & & 1 & 0 & 1 & . & 0\ 1\ 1 & : \text{Number } ① \\
+ & 1 & 0 & 0 & 1 & . & 1\ 1\ 1 & : \text{Number } ② \\
 & & 1 & 1 & 1\ 1 & & & :\text{Carry} \\
\hline
 & 1 & 1 & 1 & 1 & . & 0\ 1\ 0 &
\end{array}
$$

Binary addition example-2

$(101.011)_2 + (1001.111)_2$

$$
\begin{array}{rcccccccccl}
 & & 1 & 0 & 1 & . & 0 & 1 & 1 & & : \text{Number } ① \\
+ & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & & : \text{Number } ② \\
 & & 1 & 1 & 1 & 1 & & & & & : \text{Carry} \\
\hline
 & 1 & 1 & 1 & 1 & . & 0 & 1 & 0 & &
\end{array}
$$

$(101.011)_2 + (1001.111)_2 = (1111.01)_2$

Binary Subtraction rules

0 - 0 : Diff. = 0 &Borrow = 0

0 - 1 : Diff. = 1 &Borrow = 1

1 - 0 : Diff. = 1 &Borrow = 0

1 - 1 : Diff. = 0 &Borrow = 0

Sometimes,

1 - 1 - 1 : Diff. = 1 &Borrow = 1 (1-1 = 0) 10-1 = 1

0 - 1 - 1 : Diff. = 0 & Borrow = 1 (0-1 = 1) 1-1 = 0

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

$$
\begin{array}{rccccccccl}
 & 1 & 0 & 1 & 1 & 0 & . & 0 & 1 & 0 & : \text{Number } 1 \\
- & 0 & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } 2 \\
\hline
\end{array}
$$

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

	1	0	1	1	0	.	0	1	0	: Number 1
-	0	1	0	0	1	.	1	1	1	: Number 2
								1		

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

$$\begin{array}{rcccccccccl} & 1 & 0 & 1 & 1 & 0 & . & 0 & 1 & 0 & : \text{Number } 1 \\ - & 0 & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } 2 \end{array}$$

1

1

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

	1	0	1	1	0	.	0	1	0	: Number 1
-	0	1	0	0	1	.	1	1	1	: Number 2
							1			
							1	1		

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

	1	0	1	1	0	.	0	1	0	: Number 1
-	0	1	0	0	1	.	1	1	1	: Number 2
							1	1		
								1	1	

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

$$\begin{array}{rcccccccccl} & 1 & 0 & 1 & 1 & 0 & . & 0 & 1 & 0 & : \text{Number } ① \\ - & 0 & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } ② \\ & & & & & & 1 & 1 & & & \\ \hline & & & & & & 0 & 1 & 1 & & \end{array}$$

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

	1	0	1	1	0	.	0	1	0	: Number 1
-	0	1	0	0	1	.	1	1	1	: Number 2
					1		1	1		
							0	1	1	

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 0 & 1 & 1 & 0 & . & 0 & 1 & 0 & : \text{Number } 1 \\
- & 0 & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } 2 \\
 & & & & & 1 & 1 & 1 & & & \\
\hline
 & & & & & & . & 0 & 1 & 1 &
\end{array}
$$

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 0 & 1 & 1 & 0 & . & 0 & 1 & 0 & : \text{Number } ① \\
- & 0 & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } ② \\
 & & & & & 1 & & 1 & 1 & & \\
\hline
 & & & & & 0 & . & 0 & 1 & 1 &
\end{array}
$$

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 0 & 1 & 1 & 0 & . & 0 & 1 & 0 & : \text{Number } ① \\
- & 0 & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } ② \\
 & & & & 1 & 1 & & 1 & 1 & & \\
\hline
 & & & & & 0 & . & 0 & 1 & 1 &
\end{array}
$$

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

$$\begin{array}{rcccccccccl} & 1 & 0 & 1 & 1 & 0 & . & 0 & 1 & 0 & : \text{Number } 1 \\ - & 0 & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } 2 \\ & & & & 1 & 1 & & 1 & 1 & & \\ \hline & & & & 0 & 0 & . & 0 & 1 & 1 & \end{array}$$

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

$$
\begin{array}{rcccccccccl}
 & 1 & 0 & 1 & 1 & 0 & . & 0 & 1 & 0 & : \text{Number } ① \\
- & 0 & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } ② \\
 & & & & 1 & 1 & & 1 & 1 & & \\
\hline
 & & & 1 & 0 & 0 & . & 0 & 1 & 1 &
\end{array}
$$

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

$$
\begin{array}{rccccccccl}
 & 1 & 0 & 1 & 1 & 0 & . & 0 & 1 & 0 & : \text{Number } 1 \\
- & 0 & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } 2 \\
 & & & & 1 & 1 & 1 & 1 & & & \\
\hline
 & & 1 & 1 & 0 & 0 & . & 0 & 1 & 1 &
\end{array}
$$

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

	1	0	1	1	0	.	0	1	0	: Number 1
-	0	1	0	0	1	.	1	1	1	: Number 2
	1			1	1	1	1			
		1	1	0	0	.	0	1	1	

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

$$
\begin{array}{rccccccccl}
 & 1 & 0 & 1 & 1 & 0 & . & 0 & 1 & 0 & : \text{Number } ① \\
- & 0 & 1 & 0 & 0 & 1 & . & 1 & 1 & 1 & : \text{Number } ② \\
 & 1 & & & 1 & 1 & & 1 & 1 & & \\
\hline
 & 0 & 1 & 1 & 0 & 0 & . & 0 & 1 & 1 &
\end{array}
$$

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

	1	0	1	1	0	.	0	1	0	: Number 1
-	0	1	0	0	1	.	1	1	1	: Number 2
	1			1	1		1	1		:Borrow taken
	0	1	1	0	0	.	0	1	1	

Binary Subtraction example without complment

$(10110.01)_2 - (1001.111)_2$

	1	0	1	1	0	.	0	1	0	: Number 1
-	0	1	0	0	1	.	1	1	1	: Number 2
	1			1	1		1	1		:Borrow taken
	0	1	1	0	0	.	0	1	1	

$(10110.01)_2 - (1001.111)_2 = (1100.011)_2$

Binary Subtraction example using 1's complment

$(10110.01)_2 - (1001.111)_2$

No. 2 to be subtracted = $(01001.111)_2$

1's complement of No. 2 = $(10110.000)_2$

1 0 1 1 0 . 0 1 0 : No. 1 as it is

+ 1 0 1 1 0 . 0 0 0 : 1's compl. of No. 2

1 1 1 : Internal carry

1 0 1 1 0 0 . 0 1 0 : Final Cy = 1

1 : Cy add to LSB

0 1 1 0 0 . 0 1 1

$(10110.01)_2 - (1001.111)_2 = (1100.011)_2$

1’s complement based subtraction rules

- Take 1’s complement of second no.and then add that with first no
- After addition if no carry then result is in 1’s complement
- After addition of result is having carry the add carry bit , don’t complement result.

Binary Subtraction example using 2's complment

$(10110.01)_2 - (1001.111)_2$

No. 2 to be subtracted = $(01001.111)_2$

1's complement of No. 2 = $(10110.000)_2$

2's complement of No. 2 = $(10110.001)_2$

1 0 1 1 0 . 0 1 0 : No. 1 as it is

+ 1 0 1 1 0 . 0 0 1 : 2's compl. of No. 2

1 1 1 : Internal carry

1 0 1 1 0 0 . 0 1 1 : Final Cy = 1 (to be discarded)

Hence,

$(10110.01)_2 - (1001.111)_2 = (1100.011)_2$

Binary subtraction using 2’s complement method

- Take 2’s complement of 2’nd binary number
- Then add both the numbers
- If result having carry then discard the carry and its final result without carry
- Of no carry bit generated the result is in 2’s complement format so take 2’s complement of result .

BCD Addition Example 1: $(35)_{10} + (47)_{10}$

Step1: Express each number in BCD form & add them using Binary addition rules

First Number = $(35)_{10}$ = 0011 0101 binary code is 4 bit code

Second Number = $(47)_{10}$ = 0100 0111

Sum generated = 0111 1100

Step2: Sum generated is checked for both the nibbles: (0111) & (1100)

Step3: Add the correction $(6)_{10}$ to the nibble exceeding $(9)_{10}$ i.e. to lower nibble

Sum generated = 0111 1100

Plus Correction $(6)_{10}$ 0110

Step4: BCD Sum = 1000 0010

Step5: Decimal equivalent of the BCD sum = $(82)_{10}$

BCD Addition Example 2: $(53)_{10} + (64)_{10}$

Step1: Express each number in BCD form & add them using Binary addition rules

First Number = $(53)_{10}$ = 0101 0011

Second Number = $(64)_{10}$ = 0110 0100

Sum generated = 1011 0111

Step2: Sum generated is checked for both the nibbles: (1011) & (0111)

Step3: Add the correction $(6)_{10}$ to the nibble exceeding $(9)_{10}$ i.e. to upper nibble

Sum generated = 1011 0111

Plus Correction $(6)_{10}$ 0110

Step4: BCD Sum = 1 0001 0111

Step5: Decimal equivalent of the BCD sum = $(117)_{10}$

BCD Addition Example 3: $(57)_{10}$ + $(46)_{10}$

Step1: Express each number in BCD form & add them using Binary addition rules

First Number = $(57)_{10}$ = 0101 0111

Second Number= $(46)_{10}$ = 0100 0110

Sum generated = 1001 1101

Step2: Sum generated is checked for both the nibbles: (1001) & (1101)

Step3: Add the correction $(6)_{10}$ to the nibble exceeding $(9)_{10}$ i.e. to lower nibble

Sum generated = 1001 1101

Plus Correction $(6)_{10}$ 0110

Step4: Now the sum = 1010 0011

Step5: Now the upper nibble exceeding $(9)_{10}$, hence add the correction $(6)_{10}$ to the upper nibble also

Sum generated = 1010 0011

Plus Correction $(6)_{10}$ 0110

Step6: Now the sum = 1 0000 0011

Step7: **Decimal equivalent of the BCD sum = $(103)_{10}$**

BCD Subtraction Example : $(63)_{10}$ - $(37)_{10}$

Step1: Express each number in BCD form & subtract them using Binary subtraction

First Number = $(63)_{10}$ = 0110 0011

Second Number = $(37)_{10}$ = 0011 0111

Difference generated = 0010 1100

Step2: Difference is checked for both the nibbles: (0010) & (1100)

Step3: Subtract correction $(6)_{10}$ from the nibble exceeding $(9)_{10}$ i.e. lower nibble

Difference generated = 0010 1100

Minus Correction $(6)_{10}$ 0110

Step4: BCD difference = 0010 0110

Step5: Decimal equivalent of the BCD difference = $(26)_{10}$

BCD Subtraction using 10's complement: $(63)_{10}$ - $(37)_{10}$

First Number = $(63)_{10}$ = 0110 0011

Second Number = $(37)_{10}$ = 0011 0111

9's complement of $(37)_{10}$ = $(99 - 37)_{10}$ = $(62)_{10}$ = 0110 0010

10's complement of $(37)_{10}$ = 0110 0010 + 1 = 0110 0011

BCD difference = $(63)_{10}$ + 10's complement of $(37)_{10}$

= 0110 0011 + 0110 0011

= 1100 0110

Now, correct the upper nibble (exceeding '9') by addition of '6' i.e. 0110

Hence, difference = 1 0010 0110

= 0010 0110 ..... (carry discarded in 10's complement method)

Decimal equivalent of the BCD difference = $(26)_{10}$

BCD Subtraction using 10's complement: $(63)_{10} - (97)_{10}$

First Number = $(63)_{10}$ = 0110 0011

Second Number = $(97)_{10}$ = 1001 0111

9's complement of $(97)_{10}$ = $(99 - 97)_{10} = (02)_{10}$ = 0000 0010

10's complement of $(97)_{10}$ = 0000 0010 + 1 = 0000 0011

BCD difference = $(63)_{10}$ + 10's complement of $(97)_{10}$

= 0110 0011 + 0000 0011

= 0110 0110

Here, the result is negative, get the 10's complement of the this BCD value

Hence, difference = - (0011 0100)

Decimal equivalent of the BCD difference = - $(34)_{10}$

Hexadecimal Addition Example 1: $(DADA)_H + (BABA)_H$

1st Number:	D	A	D	A
2nd Number:	B	A	B	A

Hexadecimal Addition in step by step manner

1st Number:		D	A	D	A
2nd Number:		B	A	B	A
	1	1	1	1	
		$(D + B + 1)_H$	$(A + A + 1)_H$	$(D + B + 1)_H$	$(A + A)_H$
		$(13+11+1)_{10}$	$(10+10+1)_{10}$	$(13+11+1)_{10}$	$(10 + 10)_{10}$
		$=(25)_{10}$	$= (21)_{10}$	$= (25)_{10}$	$= (20)_{10}$
		$- (16)_{10}$	$-(16)_{10}$	$-(16)_{10}$	$- (16)_{10}$
		$= (9)_{10}$	$= (5)_{10}$	$= (9)_{10}$	$= (4)_{10}$
	1	9	5	9	4

Hence, $(DADA)_H + (BABA)_H = (19594)_H$

Hexadecimal Addition Example 2: $(1DE.7)_H + (76A.5)_H$

1st Number:	1	D	E	.	7
2nd Number:	7	6	A	.	5

Hexadecimal Addition in step by step manner

1st Number:	1	D	E	.	7
2nd Number:	7	6	A	.	5
	1	1			
	$(1+7+1)_H$	$(D+6+1)_H$	$(E+A)_H$		$(7+5)_H$
	$(1+7+1)_{10}$ $= (9)_{10}$	$(13+6+1)_{10}$ $=(20)_{10}$ $-(16)_{10}$ $=(4)_{10}$	$(14+10)_{10}$ $=(24)_{10}$ $-(16)_{10}$ $=(8)_{10}$		$(7+5)_{10}$ $=(12)_{10}$
	9	4	8	.	C

Hence, $(1DE.\&)_H + (76A.5)_H = (948.C)_H$

Hexadecimal Subtraction using 16's complement

$(3BE61.86)_{16} - (F92.4AB)_{16}$

No. 2 to be subtracted = $(F92.4AB)_{16}$ = $(00F92.4AB)_{16}$

15's complement of No. 2 = (FF06D.B54) .... Each digit subtracted from $(15)_{10}$

16's complement of No. 2 = (FF06D.B55) ..... 15's compl. + 1

	3	B	E	6	1 .	8	6	0	:	No. 1 as it is
+	F	F	0	6	D .	B	5	5	:	16's compl. of No. 2
	1				1				:	Internal carry
1	3	A	E	C	F .	3	B	5	:	Final Cy = 1 (to be discarded)

Hence,

$$(3BE61.86)_H - (F92.4AB)_H = (3AECF.3B5)_H$$

Booth's Algorithm for multiplication of Binary numbers

- Used for signed/unsigned multiplication
- In signed multiplication, negative operand is taken in 2's complement form
- Contains 3 registers each of n-bit:
  - Multiplicand (M): M_0 to M_{n-1}
  - Multiplier (Q): Q_0 to Q_{n-1}
  - Register (A) : A_0 to A_{n-1}
- 1 bit Q_{-1} is placed on the right of Q_0
- Addition or subtraction performed based on Q_0Q_{-1} bits
- Arithmetic shift operation performed in each iteration (sign is maintained after shift)
- After n-iterations, product of 2n bits is in A.Q register

Performing Arithmetic Shift Right (ASR) operation

- Performs shift right and also maintains sign

Examples:

4 bit Data = 1 0 1 0

After ASR = 1 1 0 1 (outgoing bit '0')

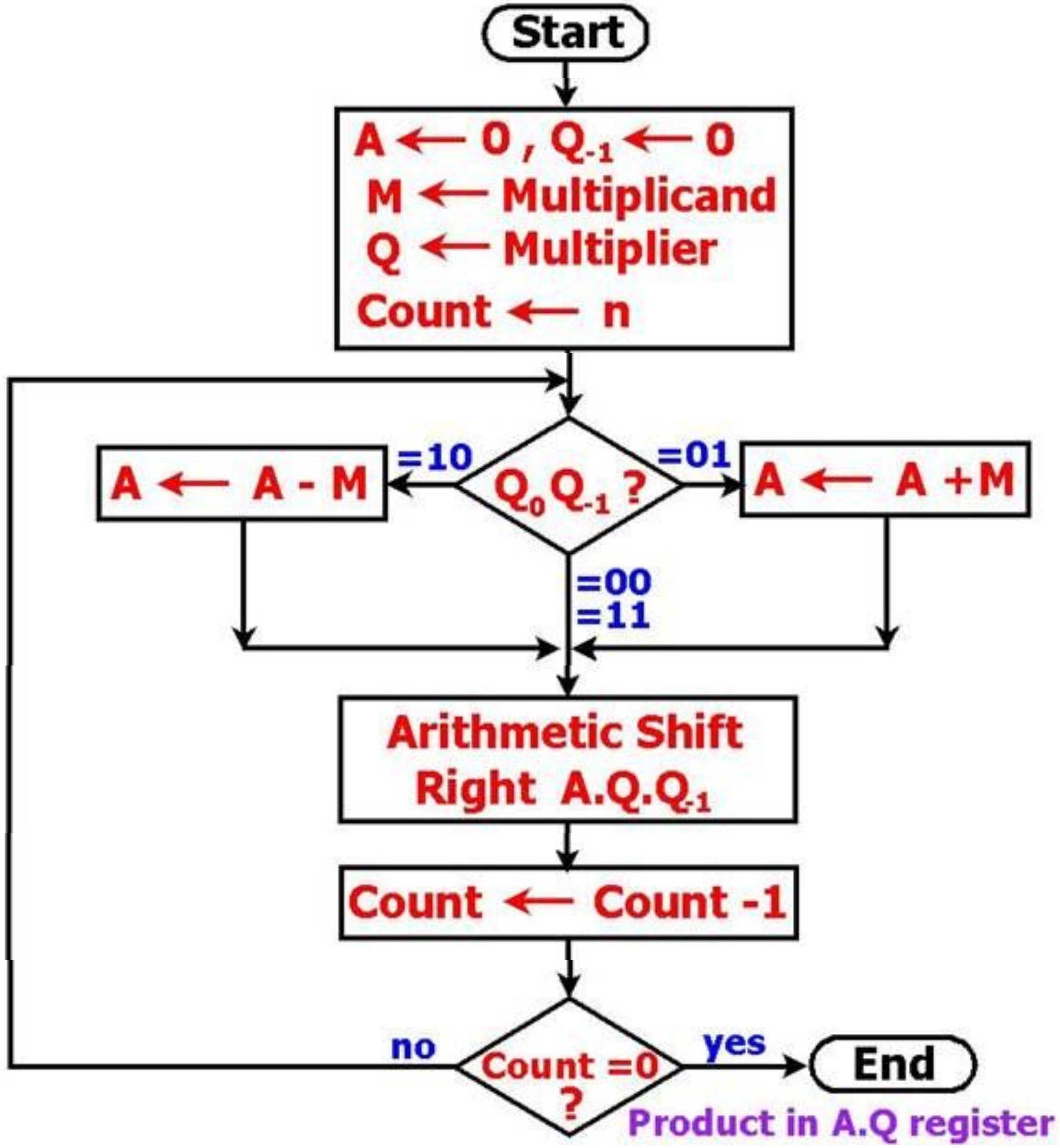
6 bit Data = 1 0 1 1 0 1

After ASR = 1 1 0 1 1 0 (outgoing bit '1')

6 bit Data = 0 0 1 0 0 0

After ASR = 0 0 0 1 0 0 (outgoing bit '0')

Booth's Multiplication Algorithm



Example 1 of Booth's Algorithm

Multiplicand (M) $= (7)_{10} = (0111)_2$

Multiplier (Q) $= (6)_{10} = (0110)_2$

2's complement of M = M~ + 1 = 1000 + 1 = 1001

A	Q	Q_{-1}	M	Operation
0000	0110	0	0111	:Initial values
0000	0011	0	0111	:Shift right
1001	0011	0	0111	:A ← A - M
1100	1001	1	0111	:Shift right
1110	0100	1	0111	:Shift right
0101	0100	1	0111	:A ← A + M
0010	1010	0	0111	:Shift right

Product (P) $= (0010\ 1010)_2 = (42)_{10}$

0000
+1001
1001

1110
+ 0111
0101

Example 2 of Booth's Algorithm

Multiplicand (M) = $(-7)_{10}$ = $(1001)_2$ ......2's compli.

7 = 0111

Multiplier (Q) = $(6)_{10}$ = $(0110)_2$

2's complement of M = M~ + 1 = 0110 + 1 = 0111

A	Q	Q_{-1}	M	Operation
0000	0110	0	1001	:Initial values
0000	0011	0	1001	:Shift right
0111	0011	0	1001	:A ← A - M
0011	1001	1	1001	:Shift right
0001	1100	1	1001	:Shift right
1010	1100	1	1001	:A ← A + M
1101	0110	0	1001	:Shift right

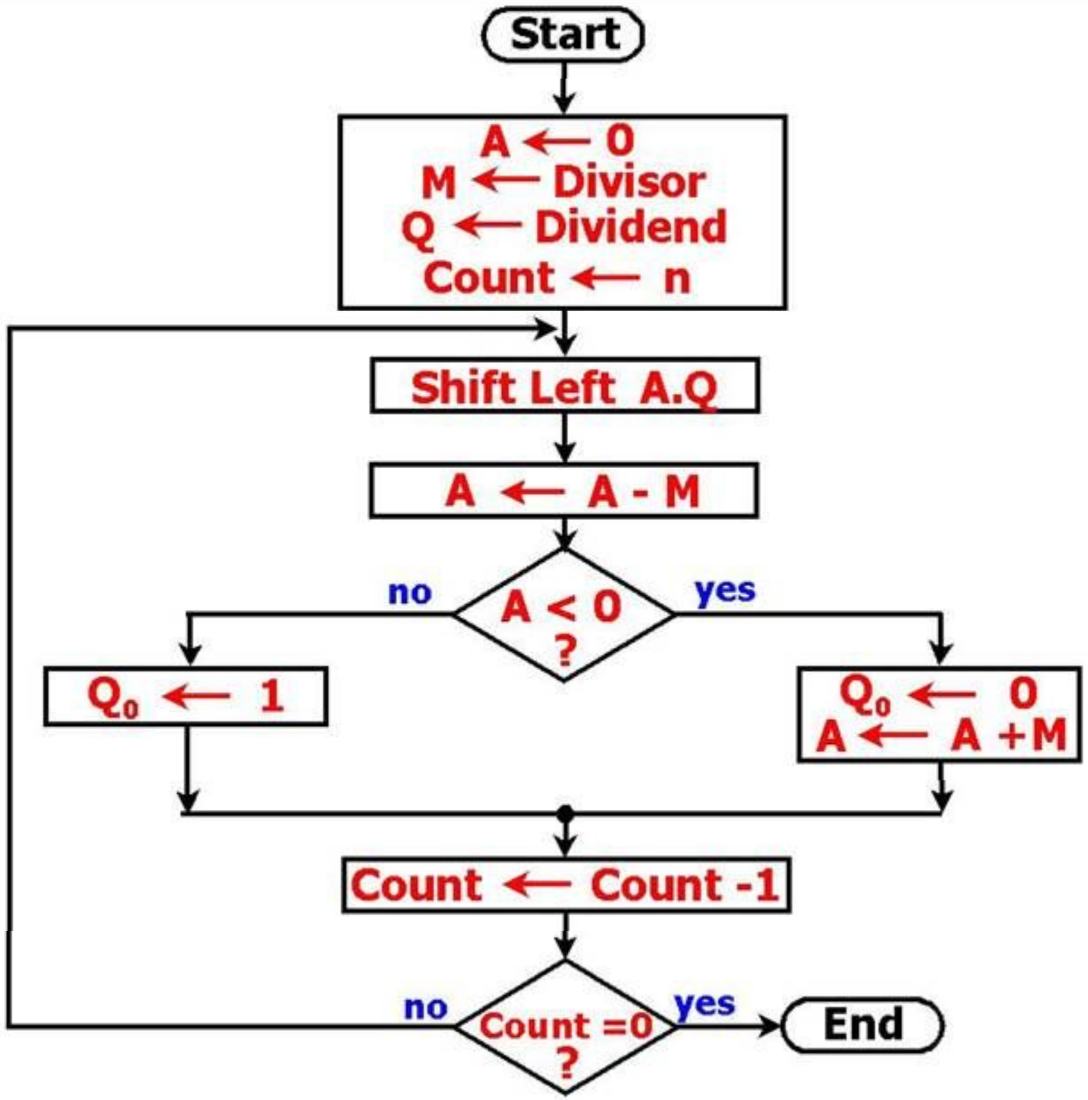
Product (P) = $(1101\ 0110)_2$ is negative

2's complement of P = $(0010\ 1010)_2 = (-42)_{10}$

0001
+1001
1010

1101 0110
0010 1001
+1
00101010

Binary Division using Restoring Division Algorithm



Mandatory steps in each iterations:

Shift Left 'A.Q' contents

Subtraction 'A - M'

After subtraction:

If (A < 0) = yes: 'A' is -ve, then RESTORE earlier value of 'A' and 'Q_0 = 0'

(RESTORE means: A = A + M)

OR

If (A < 0) = no: 'A' is +ve or zero, then no restore, but only 'Q0 = 1'

Example of Restoring Division: 13/3 (Divisor:M=3=0011, Dividend: Q=13=1101)

2's complement of M = M~ + 1 = 1100 + 1 = 1101

A	Q	Operation
0000	1101	:Initial values
0001	1010	:Shift left
1110	1010	:A ← A - M
0001	1010	:Restore (A is -ve)
0011	0100	:Shift left
0000	0100	:A ← A - M
0000	0101	(No restore, Q_0=1)
0000	1010	:Shift left
1101	1010	:A ← A - M
0000	1010	:Restore (A is -ve)
0001	0100	:Shift left
1110	0100	:A ← A - M
0001	0100	:Restore (A is -ve)

Reainder in A=0001 & Quotient in Q=0100

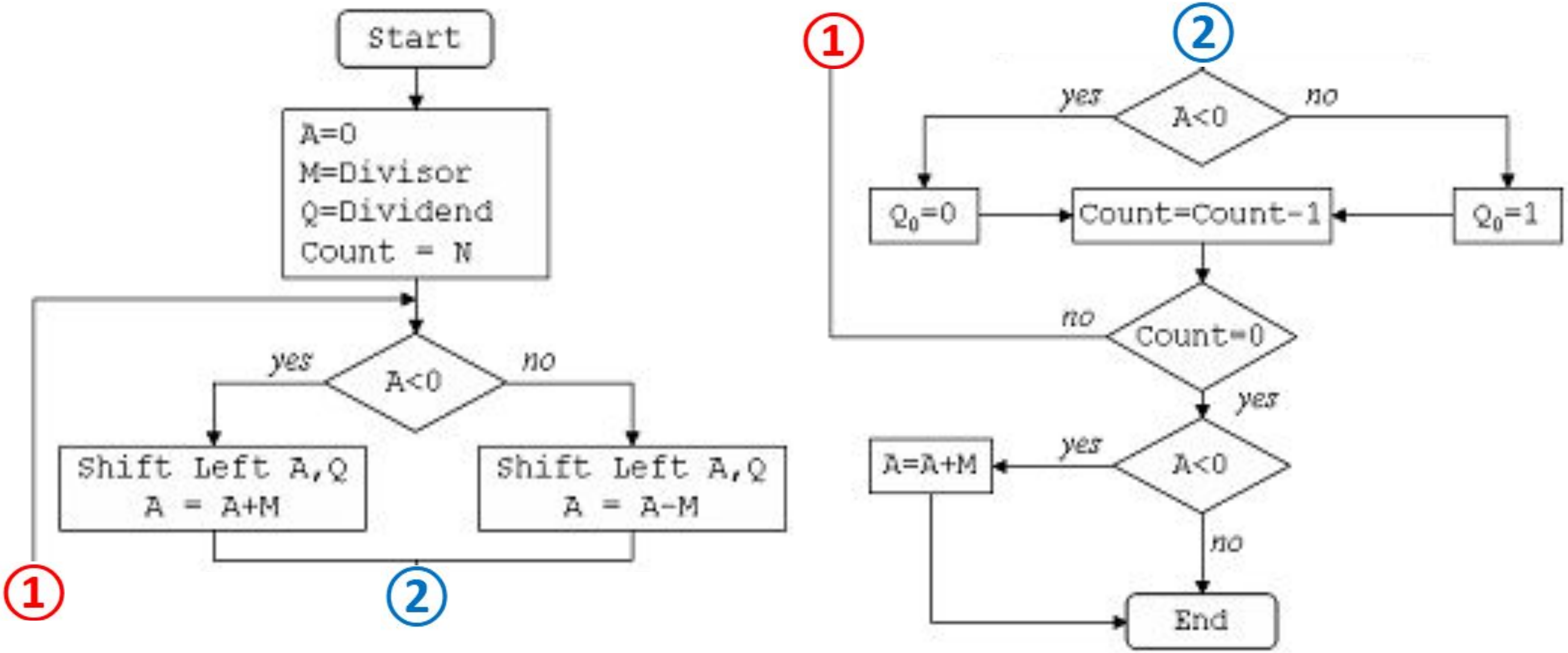
A = A + 2's compl. of M = 0001 + 1101 = 1110

A = A + 2's compl. of M = 0011 + 1101 = 10000
means, A = 0000 and Cy bit discarded

A = 0000 + 1101 = 1101

A= 0001 + 1101 = 1110

Binary Division using Non-Restoring Division Algorithm



Example of Non-restoring Division: 13/3 (Divisor:M=3=0011, Dividend: Q=13=1101)

M = 0011, Q = 1101
2's Complement of M = 1101

A	Q	Operation
0000	1101	: Initial values
0001	1010	: Shift Left A.Q
1110	1010	: A = A - M
1110	1010	: $Q_0 = 0$
1101	0100	: Shift Left A.Q
0000	0100	: A = A + M
0000	0101	: $Q_0 = 1$
0000	1010	: Shift Left A.Q
1101	1010	: A = A - M
1101	1010	: $Q_0 = 0$
1011	0100	: Shift Left A.Q
1110	0100	: A = A + M
1110	0100	: $Q_0 = 0$
0001	0100	: A = A + M

Quotient (Q) = 0100 = 4
Remainder (A) = 0001 = 1

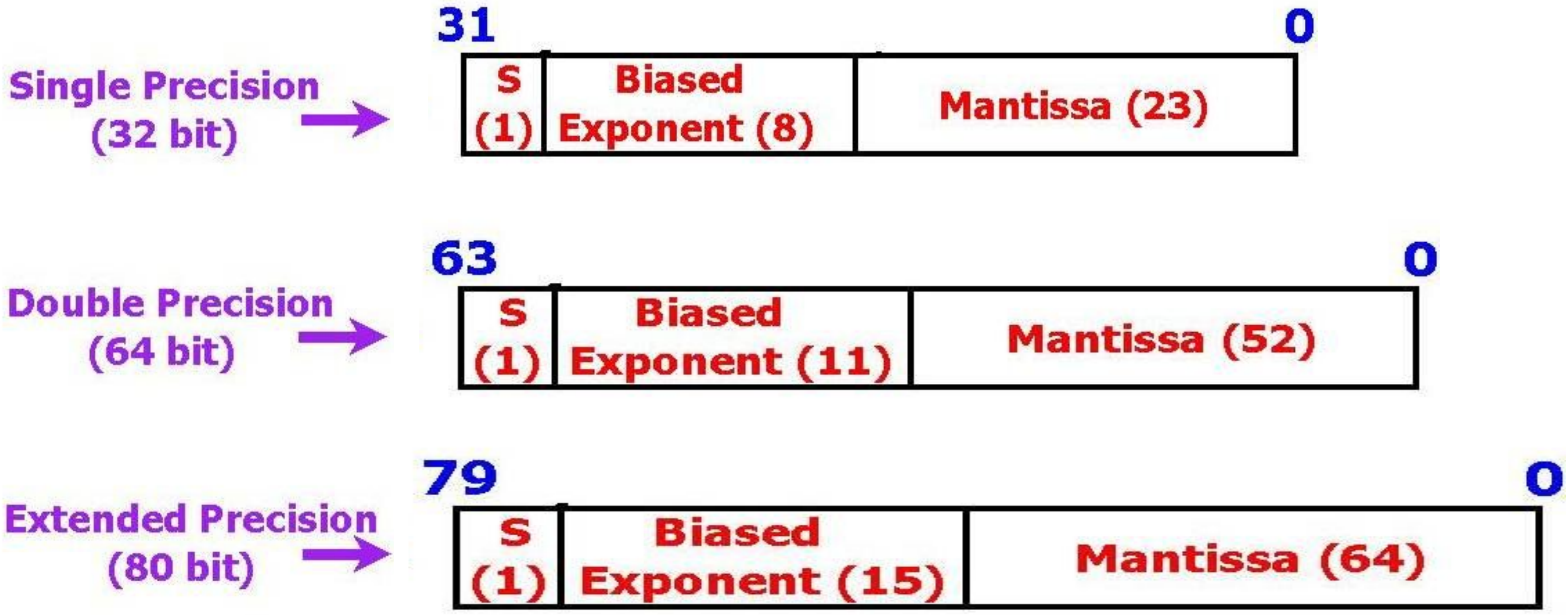
'A - M' means 'A + 2's compl. of M' = 0001 + 1101 = 1110

'A + M' = 1101 + 0011 = 0000 (Cy = 1 to be discarded)

'A - M' means 'A + 2's compl. of M' = 0000 + 1101 = 1101

'A + M' = 1011 + 0011 = 1110

IEEE 754 Floating Point Representation



Convert $(12.3125)_{10}$ into Single Precision format of IEEE 754

1) Convert number into binary

   $(12.3125)_{10} = (1100.0101)_2$

2) Represent the binary into scientific notation (only one significant digit to the left of binary point)

   $(1100.0101)_2 = (1.1000101 \times 2^3)$

3) Define the exponent into biased exponent form

   For 8 bit biased exponent, Bias value = $(2^7 - 1) = 127$

   Biased exponent = Exponent + Bias = 3 + 127 = 130

   $=(1000\ 0010)_2$

4) Define mantissa in appropriate number of bits

   23 bit mantissa = 10001010000000000000000

5) Represent number in proper 32 bit format

   = 0 10000010 10001010000000000000000

   $= (4145\ 0000)_H$