Divider (contd.)

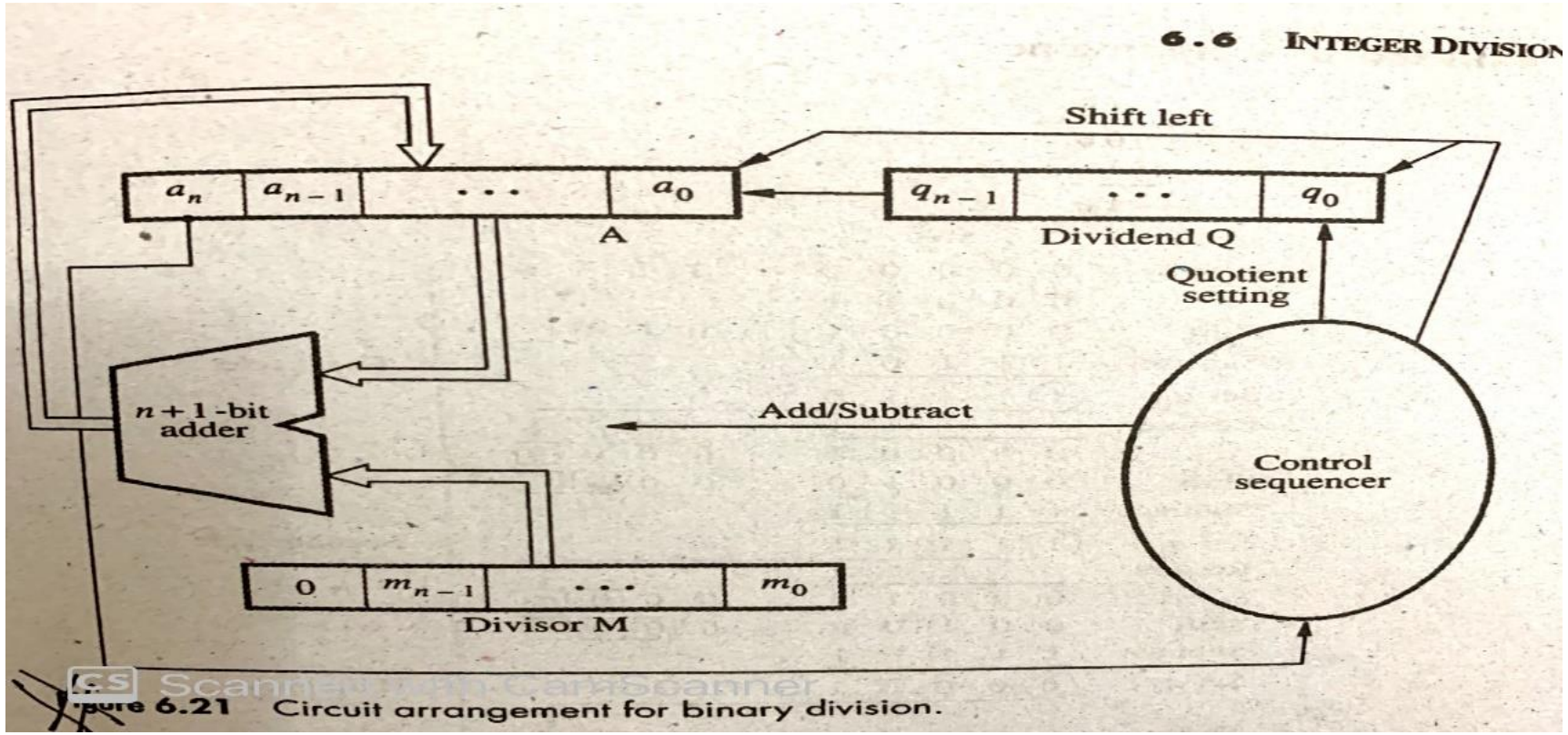


Figure: Circuit for Restoring Division

Divider (contd.)

- The restoring division can be improved by avoiding the restoring process after an unsuccessful subtraction.
- A subtraction is said to be unsuccessful if the result is negative (if the sign bit of A is 1).
- Non-restoring division: The non-restoring algorithm works as follows:

Algorithm:

Step 1: Do the following n times:

1. If the sign of A is 0, shift A and Q left one bit position and subtract M from A; otherwise, Shift A and Q left and add M to A.
2. Now, if the sign of A 0, set q_0 to 1; otherwise, set q_0 to 0.

Step 2: If the sign of A is 1, Add M to A.

- There is no simple algorithm to perform division operation on signed numbers. Operands can be pre-processed to transform them into positive values. After using the above discussed algorithms, the results are transformed to the correct signed values.

Divider (contd.)

- Example: Dividend=8=1000 and Divisor=3=11

Step	M	A	Q	Operation
Step 1: n=4	00011	00000	1000	Initialization
		00001	000_	Left shift A and Q
		11110		A=A-M (the sign of A is 0)
		11110	0000	Set q_0 to 0 (now the sign of A is 1)
n=3		11100	000_	Left shift A and Q
		11111		A=A+M (the sign of A is 1)
		11111	0000	Set q_0 to 0 (now the sign of A is 1)
n=2		11110	000_	Left shift A and Q
		00001		A=A+M (the sign of A is 1)
		00001	0001	Set q_0 to 1 (now the sign of A is 0)
n=1		00010	001_	Left shift A and Q
		11111		A=A-M (the sign of A is 0)
		11111	0010	Set q_0 to 0 (now the sign of A is 1)
Step 2		00010 Remainder	0010 Quotient	A=A+M (the sign of A is 1)

Fixed Point Number Representation

Fixed point number is a way of presenting fractional numbers by storing a fixed number of bits for their integer and fractional parts. It has a fixed position for the binary point.

- **Unsigned fixed point number:** The format is as follows:

  integer part+fractional part

- In **signed fixed point number**, the first bit is used as a sign bit.

  The integral and fractional parts are of different lengths based on the size of the register.

  - In a 8 bit register-----1st bit as sign bit+4 bits integer part+3 bits fractional part
  - In a 16 bit register--- 1st bit as sign bit+9 bits integer part+6 bits fractional part
  - In a 32 bit register---1st bit as sign bit+15 bits integer part+16 bits fractional part

Fixed Point Number Representation(contd.)

- In a 8 bit register, the lowest positive number is 00000001
- In a 8 bit register, the largest positive number is 01111111
- For example -14.50 is represented as 11110100

- Fixed point number is faster than floating point number. So, the advantage is performance.
- A very limited range of numbers can be represented using fixed point representation; disadvantage.
- For numerical analysis, scientific calculations, etc., this representation is not sufficient.

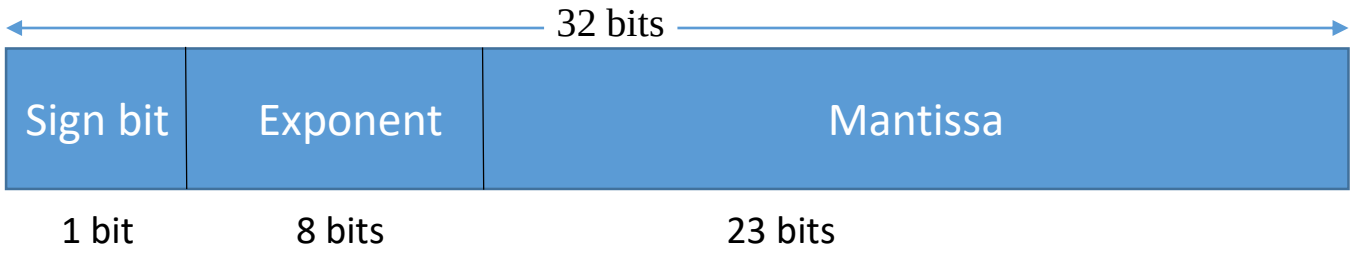
Floating Point Number Representation

- A computer must be able to represent numbers in a way that the position of the binary point is variable (not fixed) and is adjusted automatically as the computation operation proceeds.
- The numbers are called floating point as the binary point is said to float.
- A floating point number does not have a fixed number of bits for their integer and fractional parts. Rather it has total number of bits reserved for representing a number.
- A floating point number is represented by a **sign bit**, a string of significant numbers (**mantissa**) and an **exponent**.
- The exponent determines the location of the binary point.
- Only the mantissa and the exponent are represented in register.
- A floating-point number is said to be **normalized** if the most significant digit of the mantissa is 1.
- Floating point number is represented in the following form: Mxb^e.

M—Mantissa, b—base, e—exponent

Floating Point Number (IEEE754)

- The Institute of Electrical and Electronic Engineers (IEEE) has specified a standard to represent floating point numbers in 32 (single precision) and 64 (double precision) bits (IEEE754).
- This standard describes both the representation of the numbers and the way in which the basic four arithmetic operations are performed.
- 32 bit representation (single precision) is as follows:

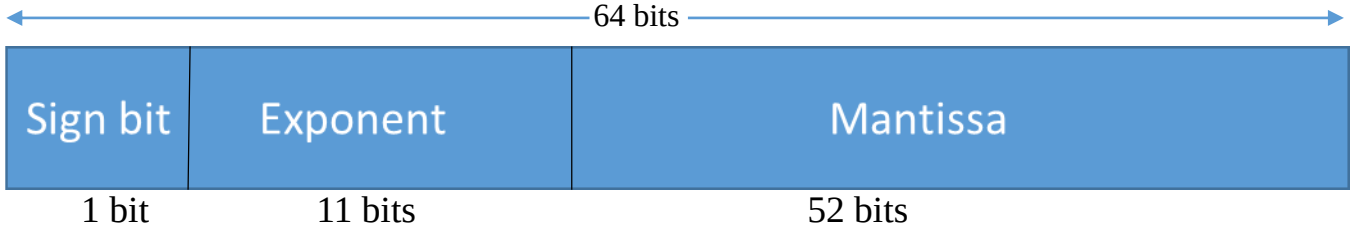


- For example, -53.5=110101.1=1.101011x2^5 would be represented as

- The leading 1 (hidden bit) is not stored in the mantissa as it is always 1 in the normalized form. So, the bits stored in mantissa are actually the bits right to the binary point.

Floating Point Number (IEEE754)$_{(contd.)}$

- The 8 bit exponent is in the range -126≤E≤+127.
- **Precision** is the number of positions reserved for the bits plus one for hidden bit. In single precision it is 24 bit (23 bit mantissa+1 hidden bit).
- The representation is called **single precision** because, it occupies a single 32 bit word.
- 64 bit representation (double precision) is as follows:



- The 11 bit exponent is in the range -1022≤E≤+1023.
- The 53 bit mantissa provides a precision equivalent 16 decimal digits.

Floating Point Number (IEEE754)(contd.)

- A computer must have at least 32 bit representation to confirm IEEE standard. Double precision is optional.
- Special values:
  - If all the bits of exponent and mantissa are 0 (E=0 and M=0), then it would represent ±0 (± depends on the sign bit).
  - If all the bits of exponent are 1 and mantissa are 0 (E=1 and M=0), then the value is ±∞ (± depends on the sign bit).
  - If all the bits of exponent are 0 and mantissa are non-zero (E=0 and M=non-zero), then the value is denormal number.
  - If all the bits of exponent are 1 and mantissa are non-zero (E=1 and M=non-zero), then the value is called Not a Number (NaN).

See Zaky (Chapter 6)