14/10/2020

CO Assignment-3

S.FIROZA
19131A05M3
CSE-4

1) Represent the following decimal numbers in both binary sign/magnitude and two's complement using 16 bits: +512, -29.

Sol:- +512 , since it is positive, MSB bit = 0.

Signed magnitude representation :- 0000 0010 0000 0000

Since the given number is positive, it is same in the signed 1's and or 2's complement representation. ~~is also same~~.

-29 , since it is negative, MSB bit = 1.

Signed magnitude representation :- 1000 0000 0001 1101.

Signed 2's complement :- 2's complement of +29 (0000 0000 0001 1101)

$\Rightarrow$ 1111 1111 1110 0010
+ 1
1111 1111 1110 0011

∴ Signed 2's complement representation of (-29).

→ 1111 1111 1110 0011.

---

(2). Represent the following two's complement values in decimal: 1101011 ; 0101101.

Ans:-

1101011.

Here MSB bit = 1, so it is a negative number. So find 2's complement of 1101011 (signed) $\Rightarrow$ 0010100
+1
0010101

magnitude :- 0010101 ($2^0 2^2 2^4$) = 16+4+1 = 21.

Since, it is a negative no. ; magnitude = -21.

0101101. ; Here MSB=0, so, it is a positive number. So, compute the magnitude as an unsigned binary number.

0101101 ($2^0 2^2 2^3 2^5$) = 32+8+4+1 = 45 ; ∴ magnitude = 45.

(3) Calculate (72530 − 13250) using tens complement arithmetic.

Ans:

72530 − 13250.

minuend = 72530, subtrahend = 13250.

(i) 10's complement of subtrahend (13250):-

$$\begin{array}{r} 99999 \\ -\ 13250 \\ \hline 86749 \\ +\ 1 \\ \hline 86750 \end{array}$$

(ii) Add this to minuend (72530)

$$\begin{array}{r} 72530 \\ +\ 86750 \\ \hline (1)59280 \end{array}$$

∴ Result = 59280.

↙ Omit the carry

(iii) ∴ 72530 − 13250 = 59280.

(4). Assume numbers are represented in 8-bit two's complement representation. Show the calculation of following:

(a) 6+13 (b) −6+13 (c) 6−13 (d) −6−13.

Ans: Add the two numbers, including their sign bits and discard any carry out of the signed position.

Note that −ve no.'s must initially be in 2's complement so that if the sum obtained after the addition is −ve, it is in 2's complement form.

(a) 6 + 13

+6 : 0000 0110

+13 : 0000 1101

+19 ⇐ (0)0001 0011

↓ MSB is 0, so +ve no.

∴ 6+13 = 19

(b) −6 + 13.

−6 : 1111 1010 (in 2's comp. form)

+13 : 0000 1101

+7 ⇐ (1)0000 0111

↓ discard carry

0000 0111 ⇒ MSB = 0, So +ve no.

+7

−6+13 = 7.

(c) 6 − 13

+6 : 0000 0110

−13 : 1111 0011 (in 2's comp form)

−7 ⇐ (1)1111 1001

↓ MSB is 1, So −ve no., So find 2's comp.

2's comp. of 1111 1001

0000 0110

+1

0000 0111 ⇒ 7

since, −ve ⇒ −7

∴ 6 − 13 = −7

(d) $-6-13$.

-6 : 11111010 (in 2's comp)

-13 : 11110011 (in 2's comp)

(1)11101101

↓ discard carry

(1)11101101

↘ MSB = 1, so -ve no.

So find 2's comp. of 11101101.

00010010
+ 1

00010011 ⇒ 19

since -ve, -19

$\therefore -6-13 = -19$

In each of 4 cases, addition is performed including sign bits, any carry out of sign bit is discarded and -ve results are automatically in 2's complement.

---

(5) Find the following differences using two's complement arithmetic.

Ans:- (a) 111000 − 110011

2's complement of 110011 :- 001100
+ 1

001101

Add this to 111000 :- 111000
001101

(1)000101

∴ Result = 000101

↙ Omit the carry.

(b) 11001100 − 101110

2's complement of 11001100 :- 00110011
+ 1

00110100

Add this to

2's complement of 00101110 :- 11010001
+ 1

11010010

Add this to 11001100 :- 11001100
11010010

(1)10011110

∴ Result = 10011110

↙ Omit carry.

(c) 1111 00001111 − 110011110011.

2's complement of 110011110011 :- 001100001100
+ 1

001100001101

Add this to 111100001111 :- 111100001111
001100001101

(1)001000001100

∴ Result = 001000001100

↙ Omit carry

(b) 11000011 − 11101000

2's complement of 11101000 ⇒ 00010111 + 1 = 00011000

Add this to 11000011 ⇒ 11000011 + 00011000 = 11011011

No carry. ∴ Result = 11011011

(6) Given x = 0101 and y = 1010 and in two's complement notation (i.e., x = 5 and y = −6), compute the product with Booth's algorithm.

Ans:- x = 0101, y = 1010.

Q = 5 = 0101, M = −6 = 1010 (2's comp. of 0110)

(Multiplier) (Multiplicand)

A ← 0000, C ← 4, Q−1 ← 0.

M	A	Q	Q−1	Operation	C
1010	0000	0101	0		
1010	0110	0101	0	A ← A−M	4
1010	0011	0010	1	Shift	3
1010	1101	0010	1	A ← A+M	3
1010	1110	1001	0	Shift	2
1010	1100	1001	0	A ← A−M	2
1010	0110	0100	1	Shift	1
1010	0100	1001	0	A ← A−M	2
1010	0010	0100	1	Shift	1
1010	1100	0100	1	A ← A+M	1
1010	1110	0010	0	Shift	0

Result = AQ = 11100010 = −ve

Find 2's comp. 00011101 + 1 = 00011110

$2^4 2^3 2^2 2^1 2^0$

30.

Result = −30.

(4) Use the Booth algorithm to multiply 23 (multiplicand) by 29 (multiplier), where each number is represented using 6 bits.

Ans:- Given, Multiplicand (M) = 23 = 010111 and multiplier (Q) = 29 = 011101.

A ← 000000 , C ← 6 (count) , Q-1 ← 0.

M	A	Q	Q-1	Operation	C
010111	000000	011101	0		6
010111	101001	011101	0	A ← A-M	6
010111	110100	101110	1	shift	5
010111	001011	101110	1	A ← A+M	5
010111	000101	110111	0	shift	4.
010111	101110	110111	0	A ← A-M	4
010111	110111	011011	1	shift	3
010111	111011	101101	1	shift	2.
010111	111101	110110	1	shift	1
010111	010100	110110	1	A ← A+M	1.
010111	001010	011011	0	shift	0.

Result = AQ = 001010011011

= 667.

(10) Explain Division restoring algorithm with an example.

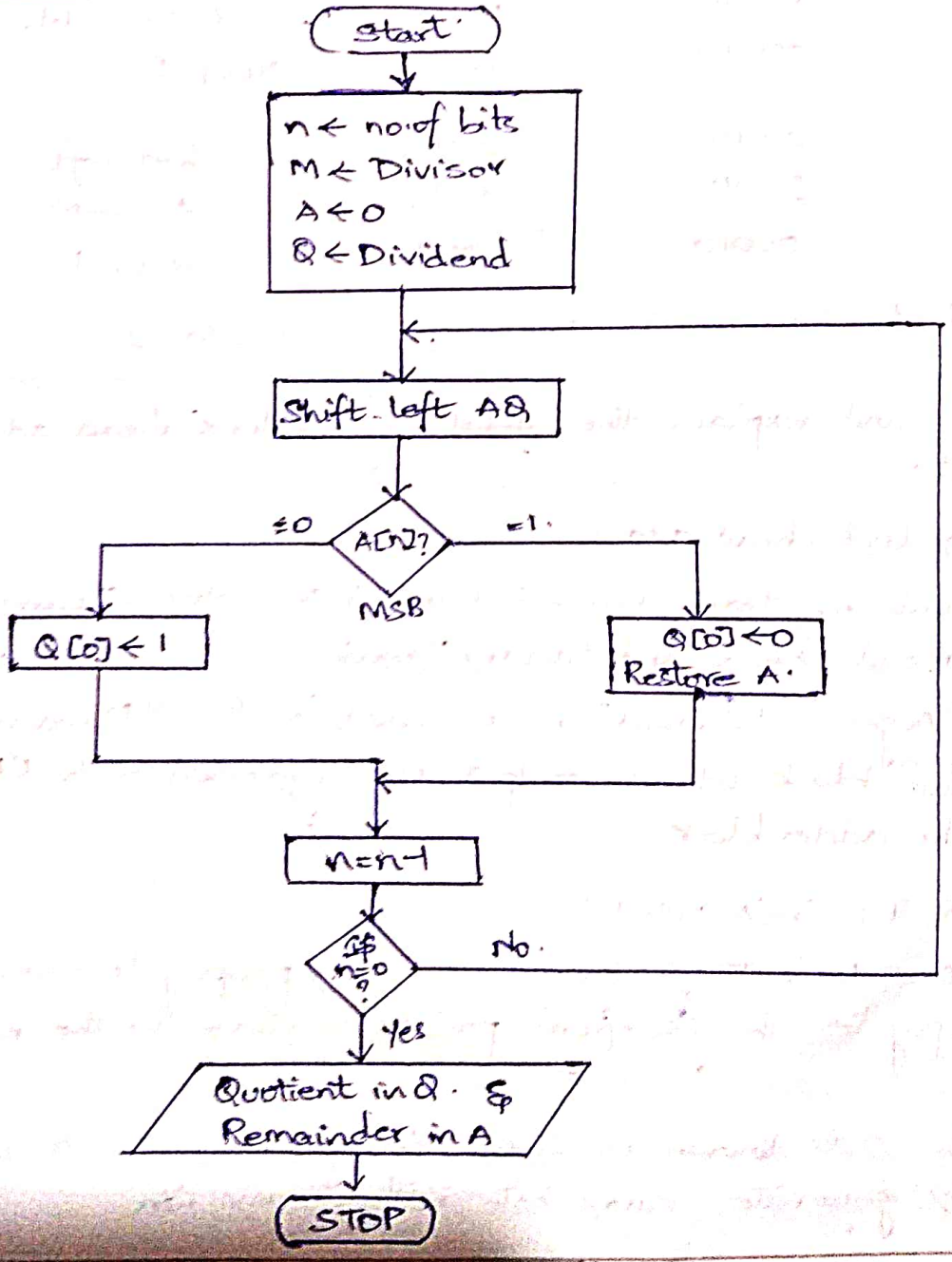
Ans:- Division Restoring Algorithm:-

A division restoring algorithm provides a quotient and a remainder when we divide two numbers

Restoring term is due to the fact that value of register A is restored after each iteration.

Here, register Q contain quotient and register A contain remainder. Here, n-bit divided is loaded in Q and divisor is loaded ~~with~~ in M. Value of Register is initially kept 0 and this is the register whose value is restored during iteration due to which it is named Restoring.

Flowchart:-



Ex:- 11/3 , Q ← 1011 , M ← 00011 n = 4.
(Dividend) (Divisor) A ← 00000

n	M	A	Q	Operation
4	00011	00000	1011	Initialization
		00001	011?	shift left AQ.
		11110	011?	A ← A-M
		00001	0110	Q[0] = 0, restore A.
3		00010	110?	shift left AQ
		11111	110?	A ← A-M.
		00010	1100	Q[0] = 0, restore A.
2		00101	100?	shift left AQ
		00010	100?	A ← A-M.
		00010	1001	Q[0] = 1.
1		00101	001?	shift left
		00010	001?	A ← A-M
		00010	0011	Q[0] = 1.

Quotient = Q = 0011 = 3, Remainder = A = 00010 = 2.

---

(11) Draw and explain the 4-bit carry look ahead adder circuit.

Ans:- Carry Look Ahead adder:-

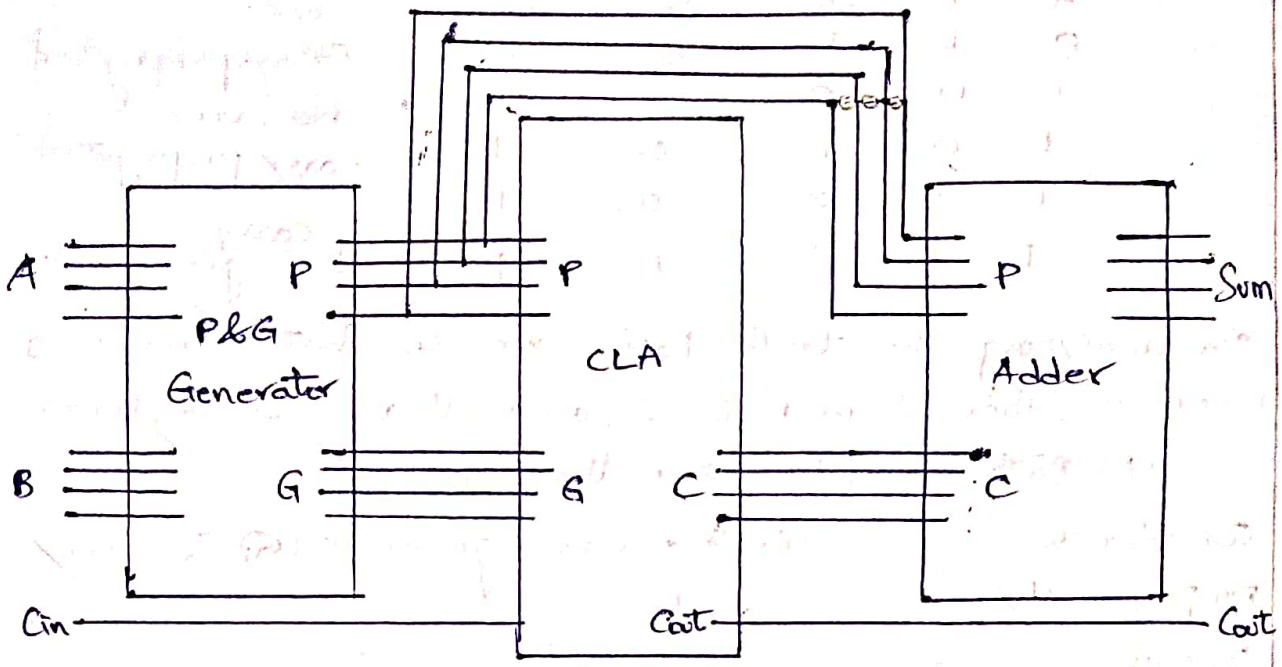
It contains three blocks :- "P and G generator", "Carry look ahead" block and "adder" block.

Input "Augend", "Addend" is provided to the "P and G generator" block whose output is connected with CLA and the adder block.

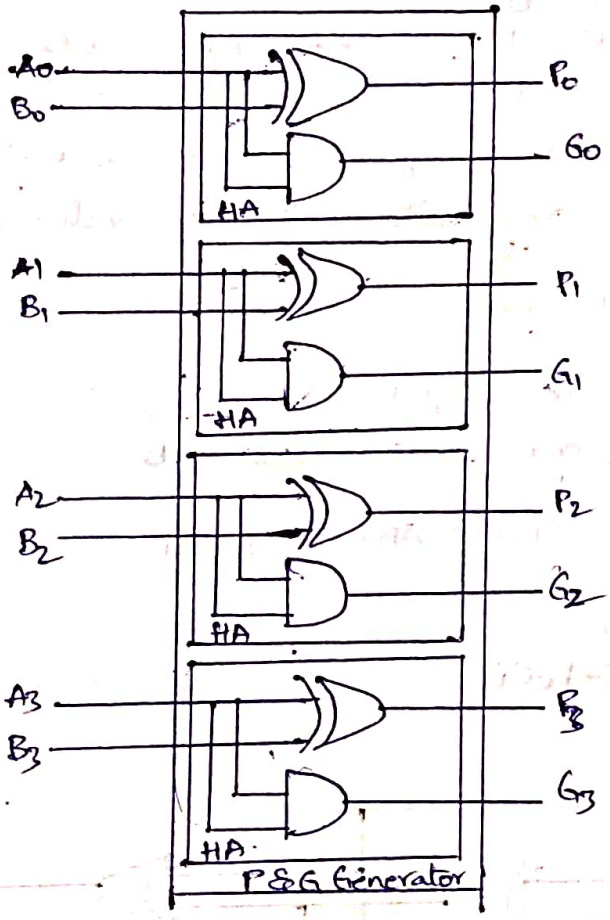
Carry = AB + Cin(A XOR B)

P = (A XOR B) : P is known as carry propogate, because it propogates the Cin from previous stage to the next stage.

G = AB : G is known as Carry Generate, because it can directly generate carry bit without any Cin



Block diagram:-



Let us consider a full adder. We have the input signals A, B and Cin, If we consider the addition of these three variables in every possible case, we get a truth table.

A	B	Cin	Sum	Carry	Condition
0	0	0	0	0	} No carry generated
0	0	1	1	0	
0	1	0	1	0	

A	B	Cin	S	C	Cond.
0	1	1	0	1	Carry propogated
1	0	0	1	0	No carry
1	0	1	0	1	Carry propogated
1	1	0	0	1	} Carry generated
1	1	1	1	1	

On analyzing the truth-table, we see that carry is 1 when (i) Either A or B is 1, as well as Cin is 1 (or)
(ii). Both A & B have the value 1.

Consider two new variable, Carry generate (Gi) & Carry propogate (Pi).

Case-(i):- Output carry is propogated, when we give an input carry. So

A	B
1	0
0	1

$P_i = A_i \oplus B_i$

$A\bar{B} + \bar{A}B = A \oplus B$

$\therefore P_i = A_i \oplus B_i$

Case-(ii):- Output carry is generated when both inputs A and B are high, regardless of the value of Cin.

A	B
1	1

So, $G_i = A_iB_i$

for a full adder, $Sum = A \oplus B \oplus C_i$

$Carry = C_{in}(A \oplus B) + AB$

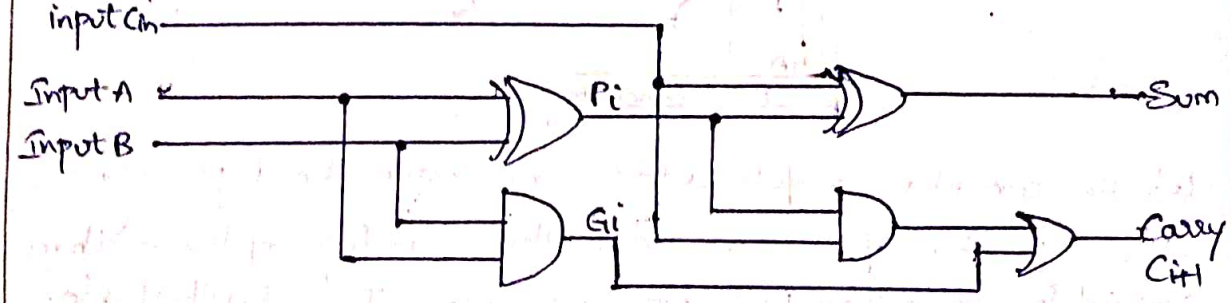
(or) $C_{i+1} = AB + BC_i + AC_i$

Thus,

$Sum = P_i \oplus C_i$

$Carry = G_i + P_iC_i$

Circuit:-



We can calculate the output carry C_1, C_2, C_3 and C_4 using the above derived equations as:

$C_1 = C_{in} \cdot P_0 + G_0$.

$C_2 = C_1 P_1 + G_1 = ((C_{in} P_0) + G_0)) P_1 + G_1 = C_{in} P_0 P_1 + G_0 P_1 + G_1$

$C_3 = C_2 P_2 + G_2 = (C_{in} P_0 P_1 + G_0 P_1$

$= (((C_1 P_1) + G_1) \cdot P_2) + G_2$.

$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}$.

$C_4 = C_3 P_3 + G_3$

$= C_{in} P_0 P_1 P_2 P_3 + P_3 P_2 P_1 G_0 + P_3 P_2 G_1 + G_2 P_3 + G_3$.