Datapath Design

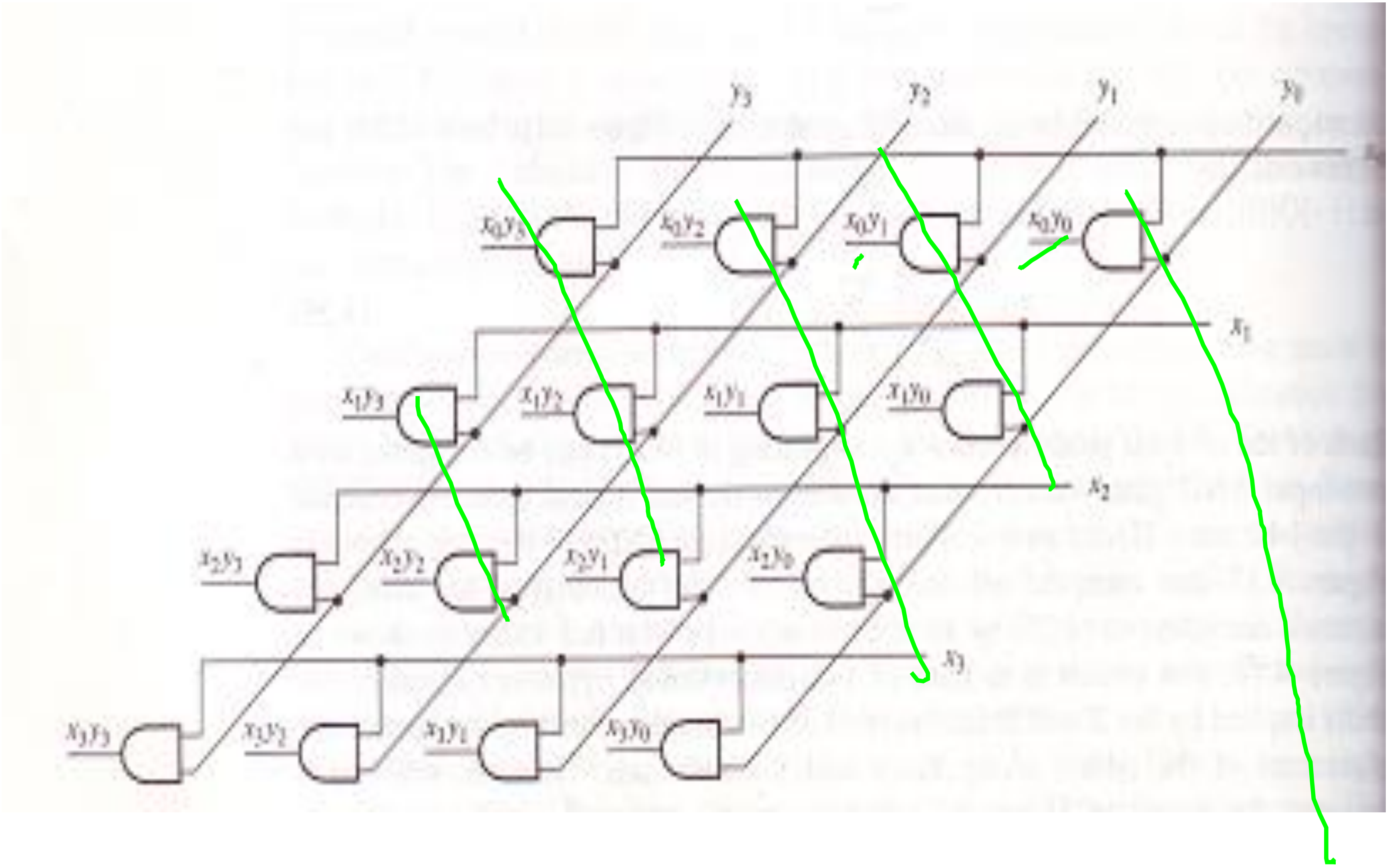
Book of John P. Hayes
Pg 240-244
Book of David A. Patterson
Pg 183-189

Combinational Array Multiplier

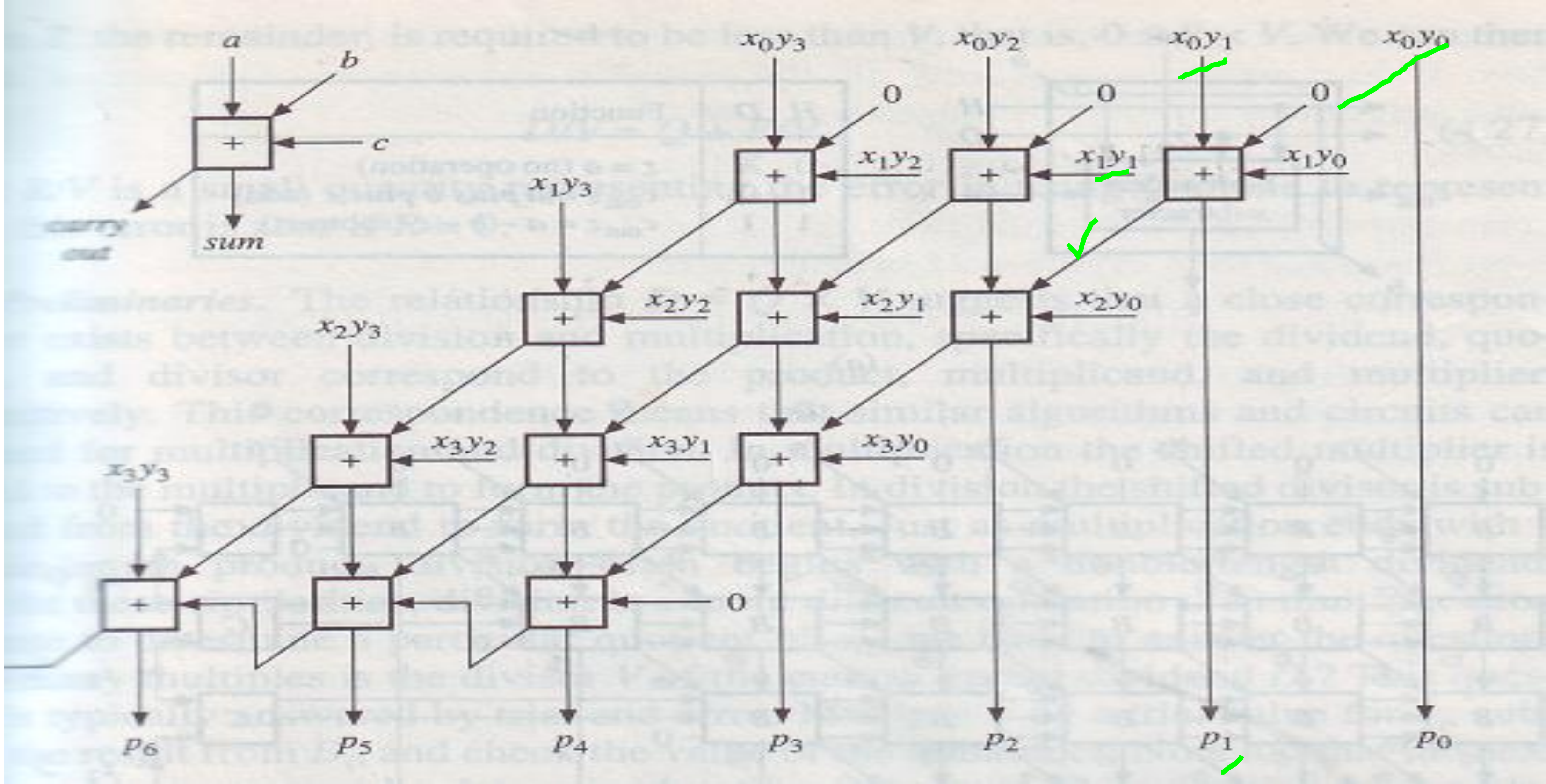
- Composed of arrays of simple combinational elements, each of which implements an add/sub and shift operation for small slices of the multiplicand operands.
- $X= x_{n-1}x_{n-2}\ldots\ldots x_1x_0$ and $Y=y_{n-1}y_{n-2}\ldots y_1y_o$ where both X and Y are unsigned integers. Now $P = X \times Y$ can be expressed as $P = \sum_{i=0}^{n-1} 2^i x_i Y$

  which can be rewritten as $P = \sum_{i=0}^{n-1} 2^i \left(\sum_{j=0}^{n-1} x_i y_j 2^j \right)$
- It requires $n \times n$ array of 2-input AND gate.
- The product terms are summed by an array of n(n-1) 1-bit full adders.

AND Array for 4 × 4 bit Unsigned Multiplication

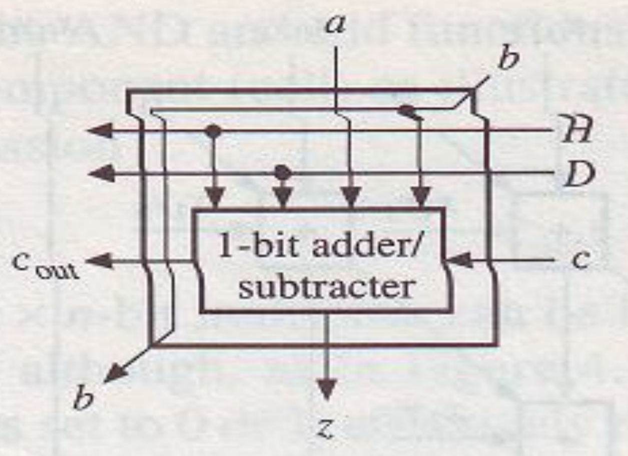


Full Adder Array for 4 × 4 bit Unsigned Multiplication



Array Implementation for Booth Multiplication

- It requires a multifunction cell capable of addition, subtraction and no operation (skip).



H	D	Function
0	X	$z = a$ (no operation)
1	0	$c_{out}z = a$ **plus** b **plus** c (add)
1	1	$c_{out}z = a - b - c$ (subtract)

- The functions of B are defined by z = a **xor** bH **xor** cH and C_{out} = (a xor D)(b+c) + bc
- An n-bit multiplier is constructed from $n^2 + n(n-1)/2$ copies of the B Cell.

Array Implementation for Booth Multiplication

- ✔ When HD = 10 the equations reduce to full adder equations.

  $z = a \text{ xor } b \text{ xor } c$ and $c_{out} = ab + ac + bc$

- ✔ When HD = 11 the equations reduce to full subtracter equations:

  $z = a \text{ xor } b \text{ xor } c$ and $c_{out} = \bar{a}b + \bar{a}c + bc$

- ✔ When H = 0 then z = a and carry plays no role in the final result.
- ✔ A n × n bit multiplier is constructed from $n^2 + n(n-1)/2$ cells.

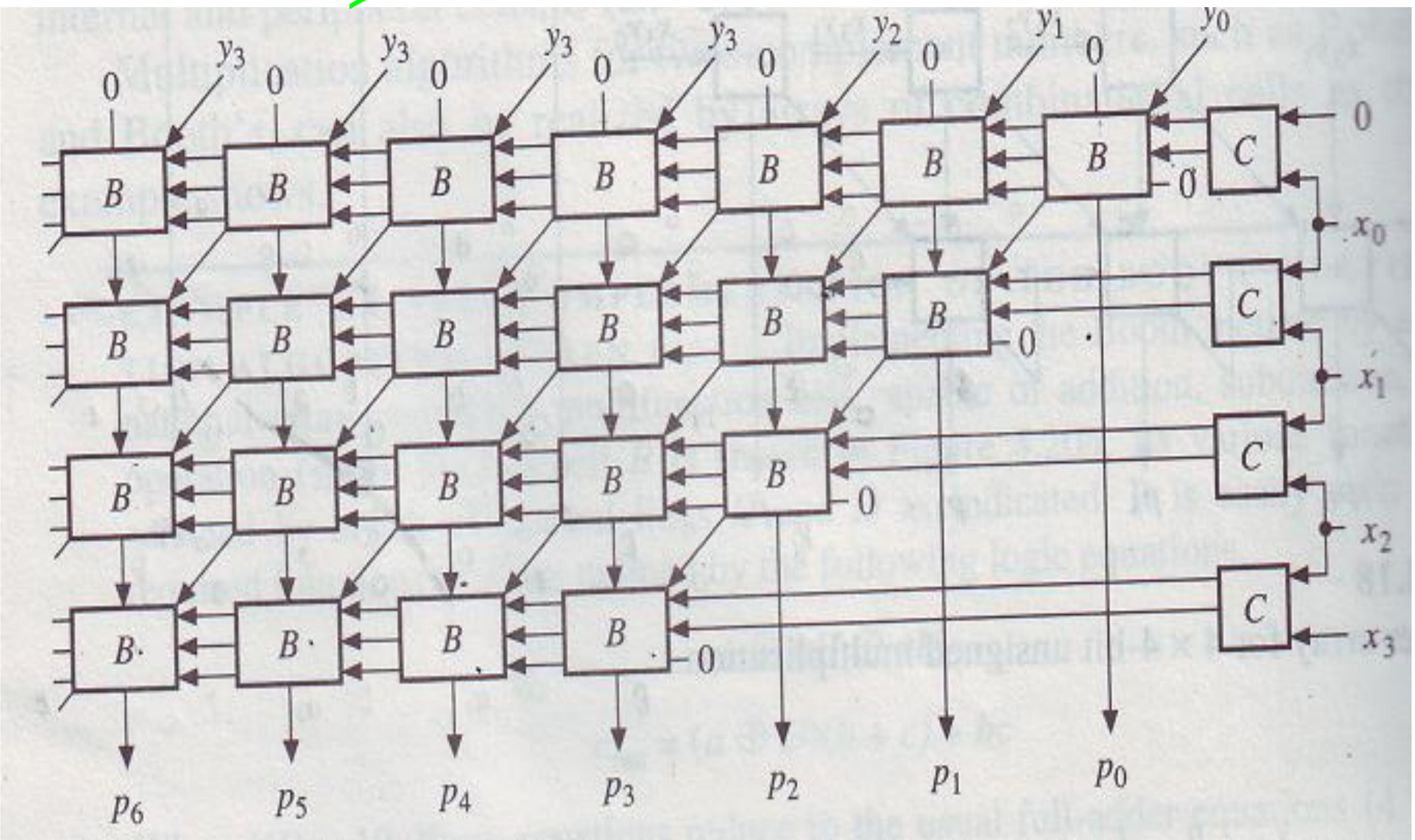
Array Implementation for Booth Multiplication

- C cell generates control input H and D required by the B cells depending on the combination of $x_i x_{i-1}$.

$$H = x_i \oplus x_{i-1}$$
$$D = x_i \bar{x}_{i-1}$$

x_i	x_{i-1}	H	D
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	0

Combinational Array implementing the Booth's Algorithm

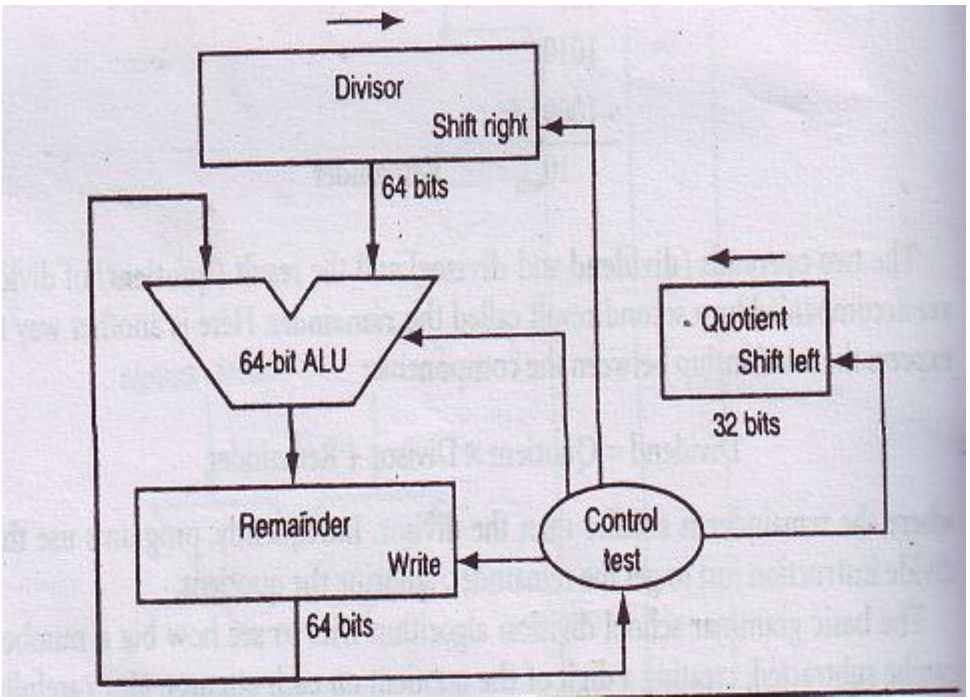


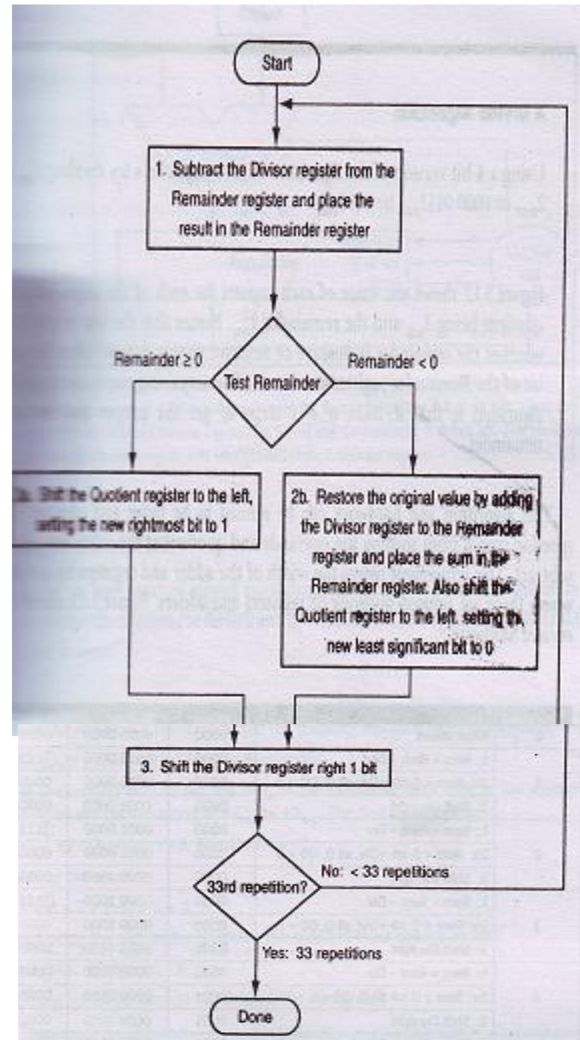
Division

$$
\begin{array}{rrl}
 & 1001_{ten} & \text{Quotient} \\
\text{Divisor } 1000_{ten} & \overline{)1001010_{ten}} & \text{Dividend} \\
 & \underline{-1000} & \\
 & 10 & \\
 & 101 & \\
 & 1010 & \\
 & \underline{-1000} & \\
 & 10_{ten} & \text{Remainder}
\end{array}
$$

✔ Dividend = Quotient × Divisor + Remainder

Division Algorithm and Hardware for Unsigned Number





Read the improved version of the circuit by yourself.

(n+1) it

2) 7 (3
6
1

Example

2 7

Iteration	Step	Quotient	Divisor	Remainder
0	Initial values	0000	0010 0000	0000 0111
1	1: Rem = Rem – Div	0000	0010 0000	1110 0111
	2b: Rem < 0 ⟹ +Div, sll Q, Q0 = 0	0000	0010 0000	0000 0111
	3: Shift Div right	0000	0001 0000	0000 0111
2	1: Rem = Rem – Div	0000	0001 0000	1111 0111
	2b: Rem < 0 ⟹ +Div, sll Q, Q0 = 0	0000	0001 0000	0000 0111
	3: Shift Div right	0000	0000 1000	0000 0111
3	1: Rem = Rem – Div	0000	0000 1000	1111 1111
	2b: Rem < 0 ⟹ +Div, sll Q, Q0 = 0	0000	0000 1000	0000 0111
	3: Shift Div right	0000	0000 0100	0000 0111
4	1: Rem = Rem – Div	0000	0000 0100	0000 0011
	2a: Rem ≥ 0 ⟹ sll Q, Q0 = 1	0001	0000 0100	0000 0011
	3: Shift Div right	0001	0000 0010	0000 0011
5	1: Rem = Rem – Div	0001	0000 0010	0000 0001
	2a: Rem ≥ 0 ⟹ sll Q, Q0 = 1	0011	0000 0010	0000 0001
	3: Shift Div right	0011	0000 0001	0000 0001

3 1