Arithmetic Operations

S.Venkatesan
Network Security and Cryptography Lab
Department of Information Technology
Indian Institute of Information Technology, Allahabad
venkat@iiita.ac.in

Acknowledgement: The contents and figures are copied from various sources. Thanks to all authors and sources made those contents public and usable for educational purpose

Binary Addition & Subtraction

Direct Addition

Number 1	1	0	0	0	0	0	0	1	1	1
Number 2	0	1	1	1	1	0	0	0	0	0
Result	1	1	1	1	1	0	0	1	1	1

Direct Subtraction

Number 1	1	0	0	0	0	0	0	1	1	1
Number 2	0	1	1	1	1	0	0	0	0	0
Result	0	0	0	0	1	0	0	1	1	1

Subtraction Via addition using Two's complement [x-y] = x + (-y)

Number 1	1	0	0	0	0	0	0	1	1	1
Number 2	0	1	1	1	1	0	0	0	0	0
Two's complement	1	0	0	0	1	0	0	0	0	0
Result	0	0	0	0	1	0	0	1	1	1

Carries

		(0)		*(0)*		*(1)*		*(1)*		*(0)*		*(Carries)*
. . .		0		0		0		1		1		1
. . .		0		0		0		1		1		0
. . .	(0)	0	(0)	0	(0)	1	(1)	1	(1)	0	(0)	1

Overflow

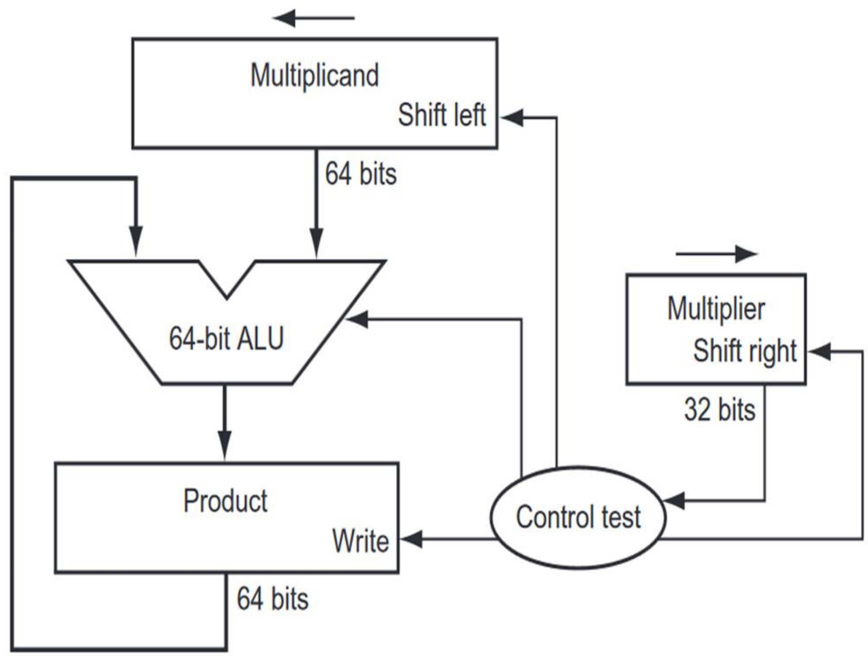
Operation	Operand A	Operand B	Result indicating overflow
$A + B$	≥ 0	≥ 0	< 0
$A + B$	< 0	< 0	≥ 0
$A - B$	≥ 0	< 0	< 0
$A - B$	< 0	≥ 0	≥ 0

Try
16 + 16
-1 + -1
(-16) – 10
16 – (-10)

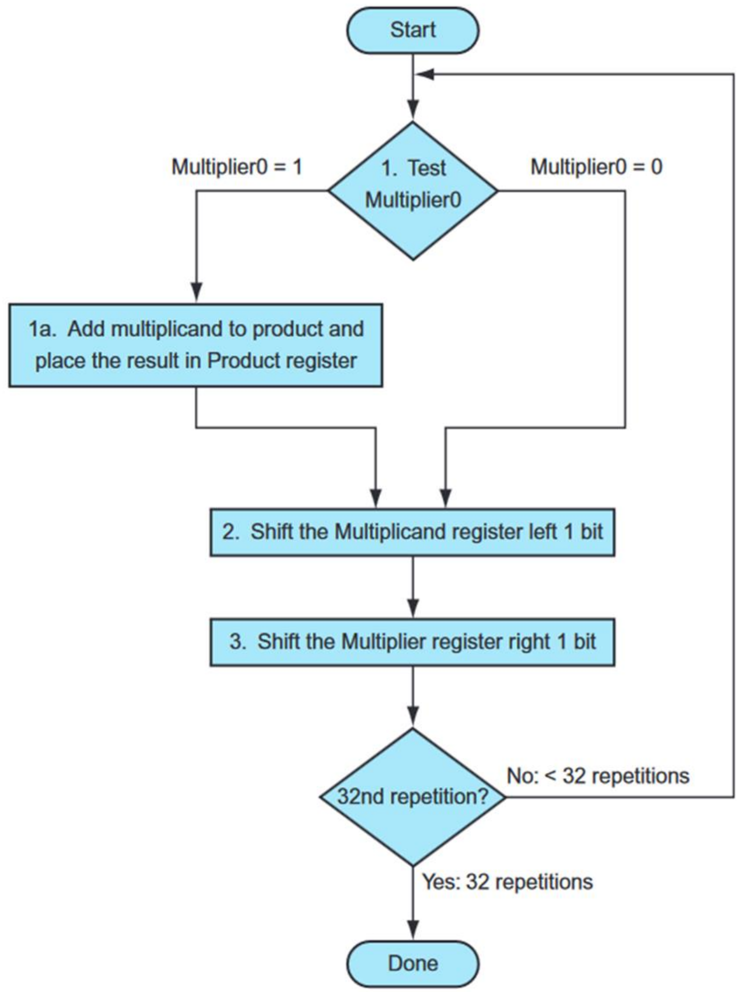
- Add (add), add immediate (addi), and subtract (sub) cause exceptions on overflow.
- Add unsigned (addu), add immediate unsigned (addiu), and subtract unsigned (subu) do not cause exceptions on overflow.

Multiplication

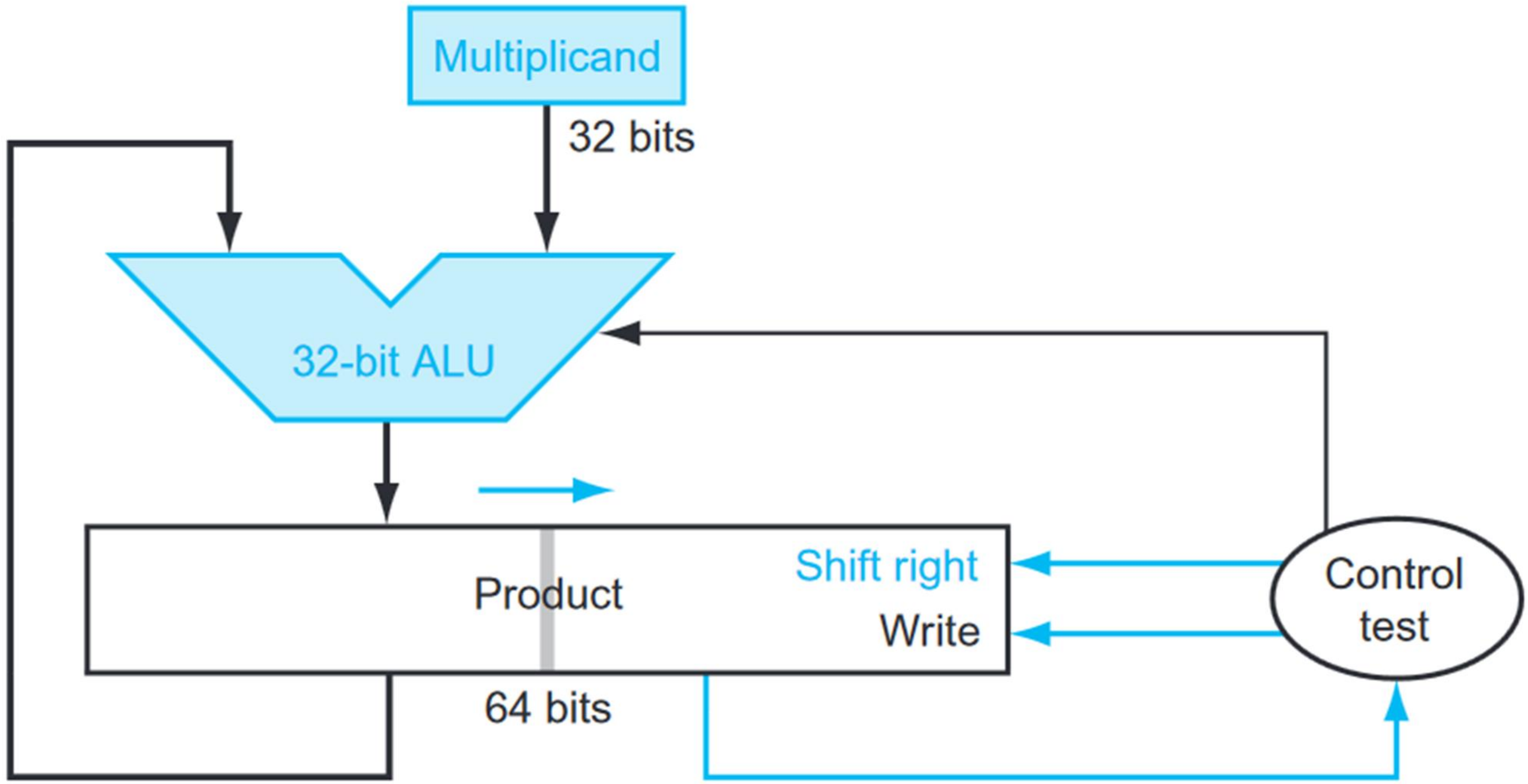
$$
\begin{array}{llr}
\text{Multiplicand} & & 1000_{ten} \\
\text{Multiplier} & \times & \underline{1001_{ten}} \\
 & & 1000 \\
 & & 0000 \\
 & & 0000 \\
 & & \underline{1000} \\
\text{Product} & & 1001000_{ten}
\end{array}
$$



Multiplication Process Flow



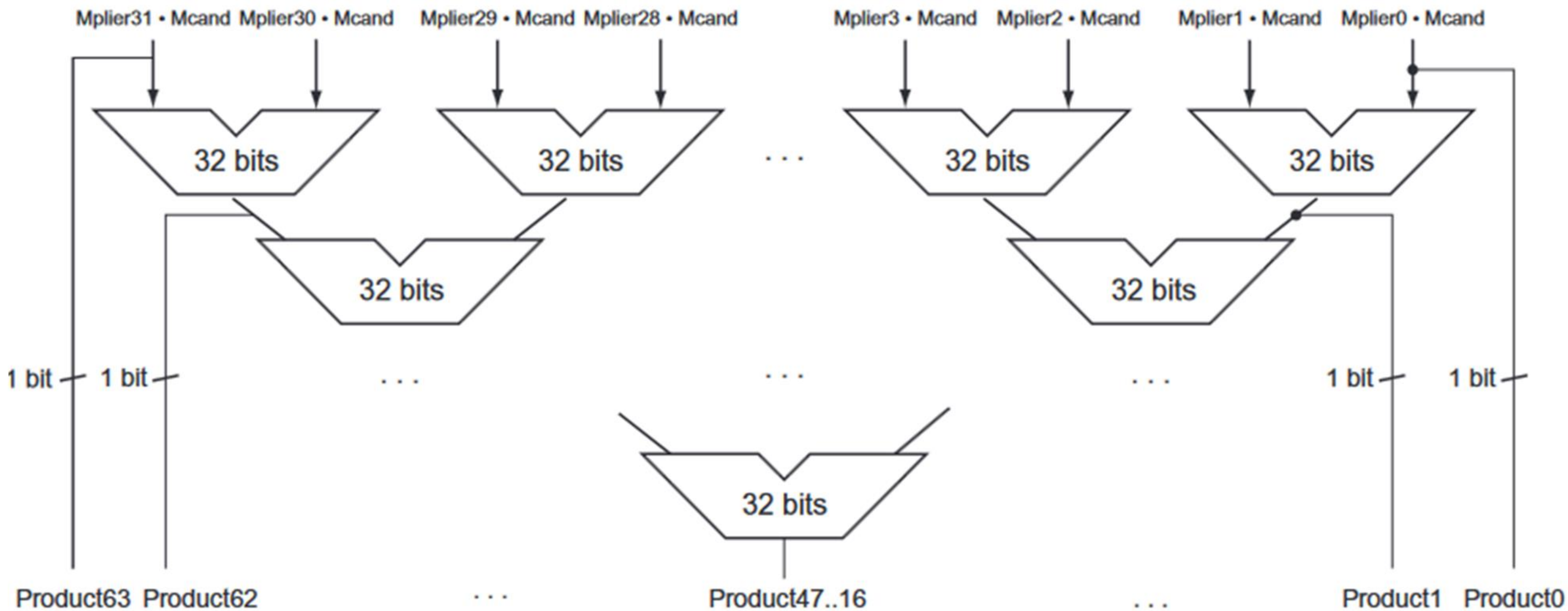
Refined version of the multiplication hardware



Example (2X3)

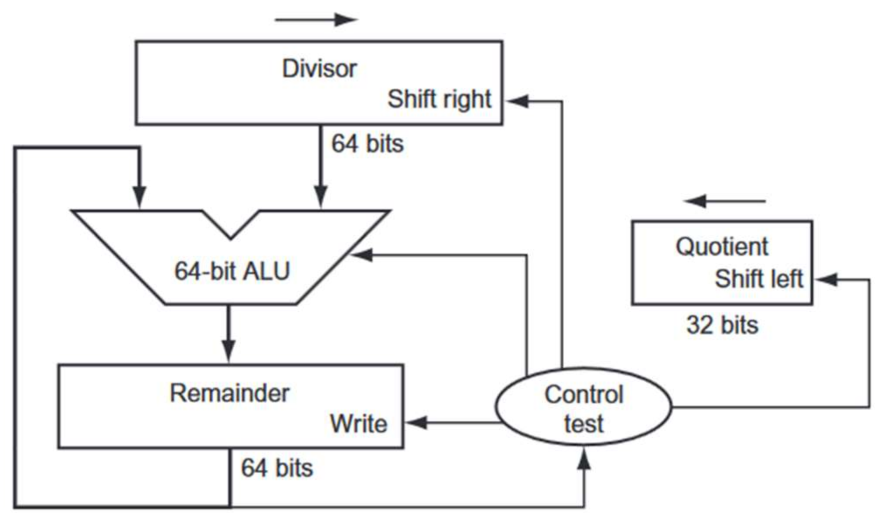
Iteration	Step	Multiplier	Multiplicand	Product
0	Initial values	0011	0000 0010	0000 0000
1	1a: 1 ⟹ Prod = Prod + Mcand	0011	0000 0010	0000 0010
	2: Shift left Multiplicand	0011	0000 0100	0000 0010
	3: Shift right Multiplier	0001	0000 0100	0000 0010
2	1a: 1 ⟹ Prod = Prod + Mcand	0001	0000 0100	0000 0110
	2: Shift left Multiplicand	0001	0000 1000	0000 0110
	3: Shift right Multiplier	0000	0000 1000	0000 0110
3	1: 0 ⟹ No operation	0000	0000 1000	0000 0110
	2: Shift left Multiplicand	0000	0001 0000	0000 0110
	3: Shift right Multiplier	0000	0001 0000	0000 0110
4	1: 0 ⟹ No operation	0000	0001 0000	0000 0110
	2: Shift left Multiplicand	0000	0010 0000	0000 0110
	3: Shift right Multiplier	0000	0010 0000	0000 0110

Faster Multiplication

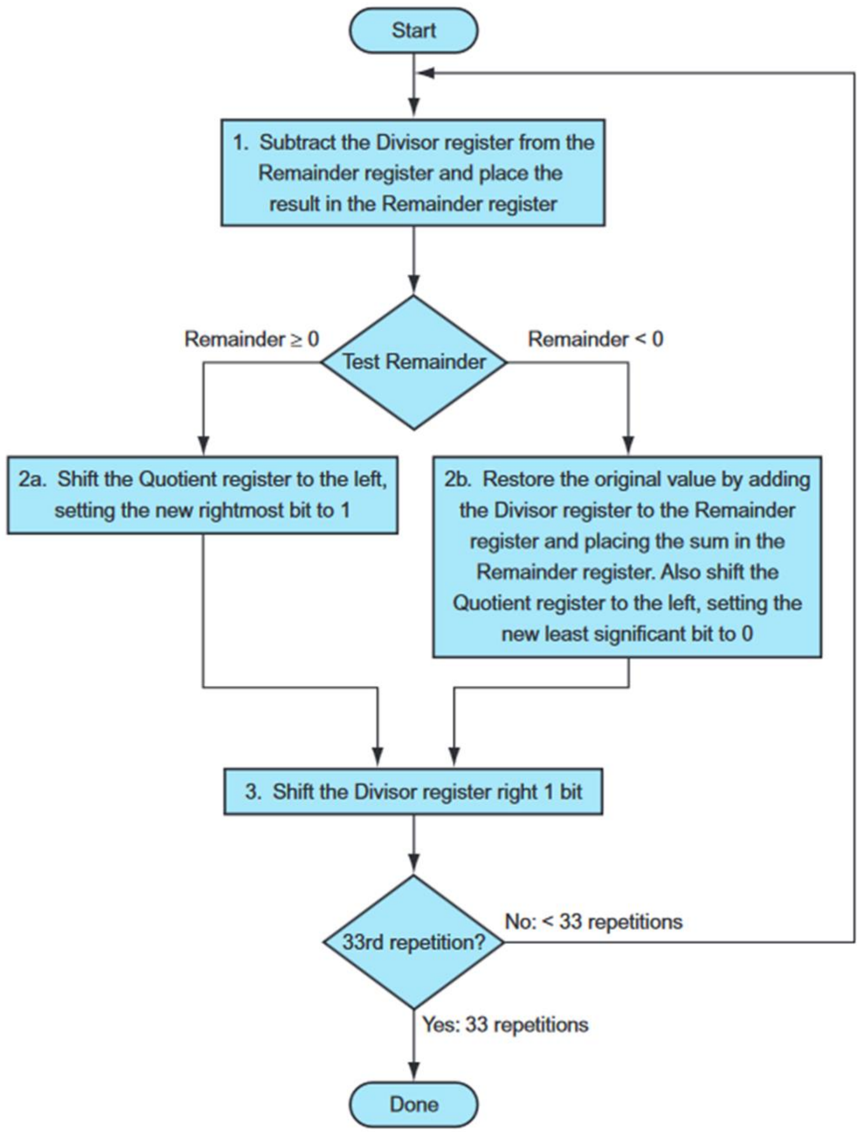


Division

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



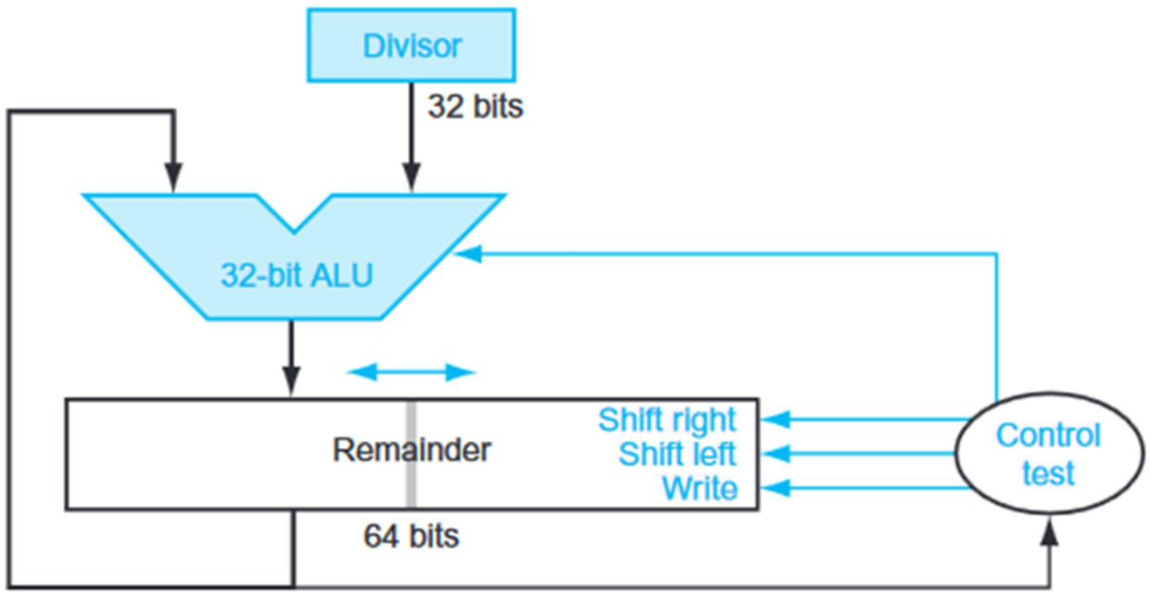
Division Flow



Example [Division]

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 $\Longrightarrow$ +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 $\Longrightarrow$ +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 $\Longrightarrow$ +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 $\geq$ 0 $\Longrightarrow$ 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 $\geq$ 0 $\Longrightarrow$ sll Q, Q0 = 1	0011	0000 0010	0000 0001
	3: Shift Div right	0011	0000 0001	0000 0001

Improved Version



Reference

- Computer Organization and Design (ARM edition) - The Hardware and Software Interface by David A. Patterson and John L. Hennessy