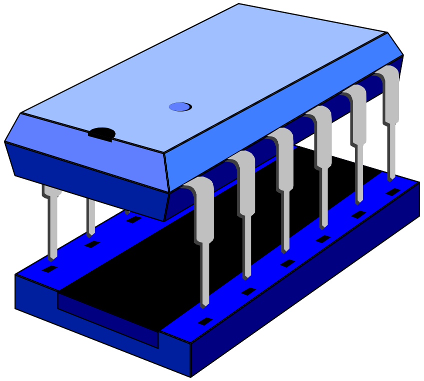


REVISION

Bernhard Kainz

b.kainz@imperial.ac.uk

Boolean Algebra – Truth Tables

- All possible outcomes of the operators can be written as truth tables

AND

•

A	B	R
0	0	0
0	1	0
1	0	0
1	1	1

OR

+

A	B	R
0	0	0
0	1	1
1	0	1
1	1	1

NOT

'

A	R
0	1
1	0

Boolean Algebra – Rules

- **Note: A and B can be any Boolean Expression**

Negation:

(A')' = A

A • A' = 0

A + A' = 1

Associative:

(A • B) • C = A • (B • C)

(A + B) + C = A + (B + C)

Commutative:

A • B = B • A

A + B = B + A

Distributive:

A • (B + C) = A • B + A • C

A + (B • C) = (A + B) • (A + C)

Note the precedence

Boolean Algebra – Rules

Single variables (Idempotent law):

$A \cdot A = A$

$A + A = A$

Simplification rules with 1 and 0:

$A \cdot 0 = 0$

$A \cdot 1 = A$

$A + 0 = A$

$A + 1 = 1$

Boolean Algebra – de Morgan's Rule

$(A + B)' = A' \cdot B'$

$(A \cdot B)' = A' + B'$

as before, A and B can be any Boolean expression

Can generalise to n Boolean variables:

$(A + B + C + D + \ldots)' = A' \cdot B' \cdot C' \cdot D' \cdot \ldots$

$(A \cdot B \cdot C \cdot D \cdot \ldots \cdot X)' = A' + B' + C' + D' + \ldots + X'$

Half Adder

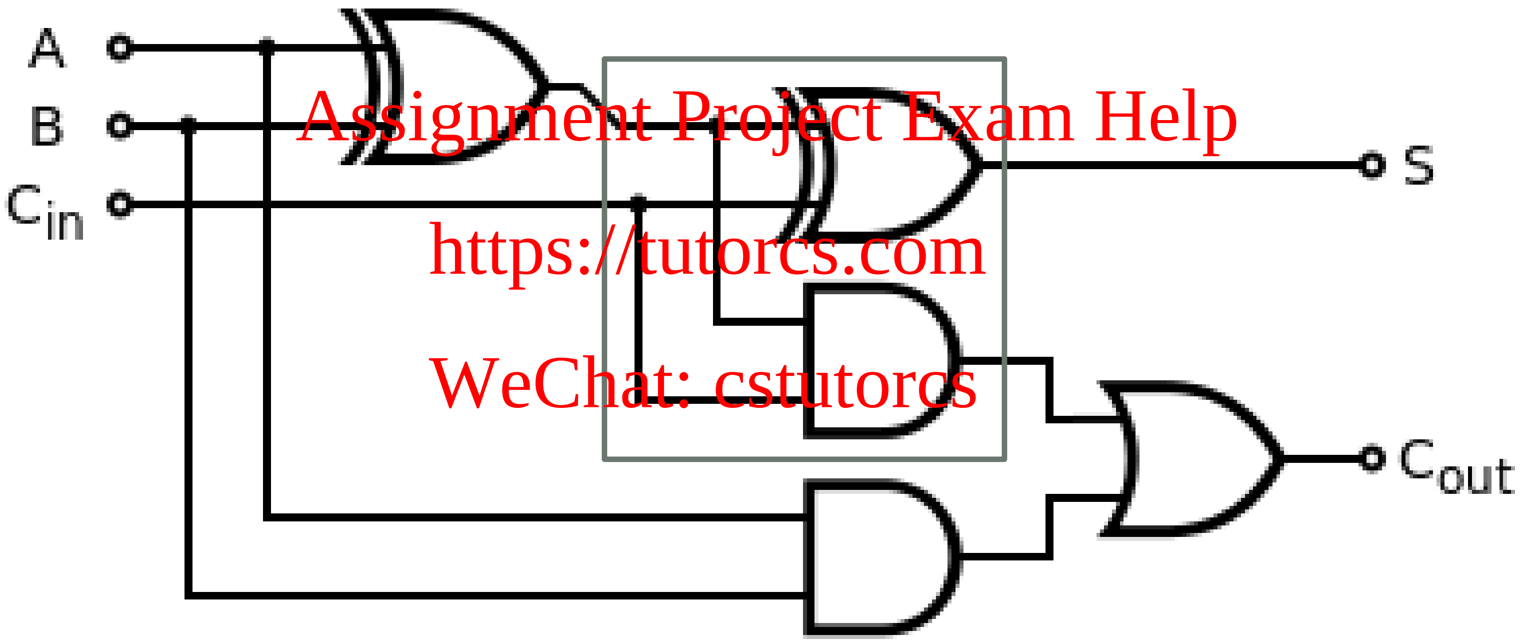
- Recall

	0	0	1	1
+	0	1	0	1
	00	01	01	10

- Truth Table

A	B	A + B	Sum	Carry
0	0	0	0	0
0	1	1	1	0
1	0	1	1	0
1	1	2	0	1

Full Adder

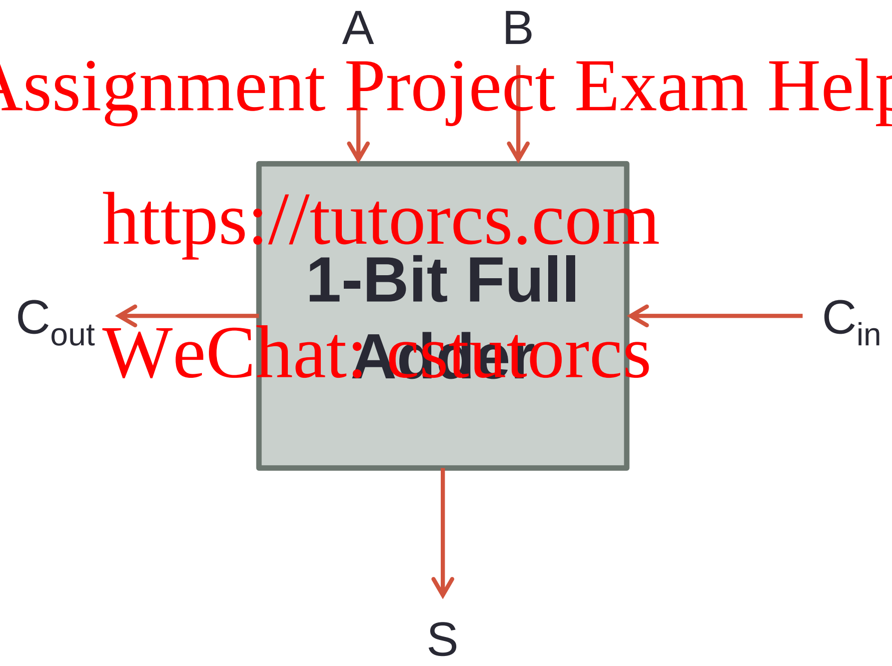


$S = A \oplus B \oplus C_{in}$

$C_{out} = (A \bullet B) + C_{in} \bullet (A \oplus B))$

Full Adder

- Conceptually



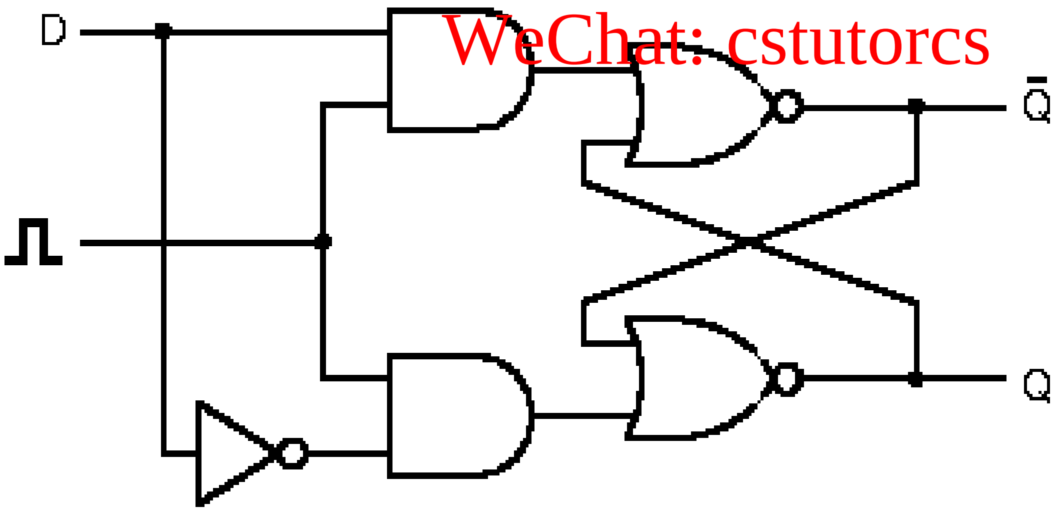
Latches

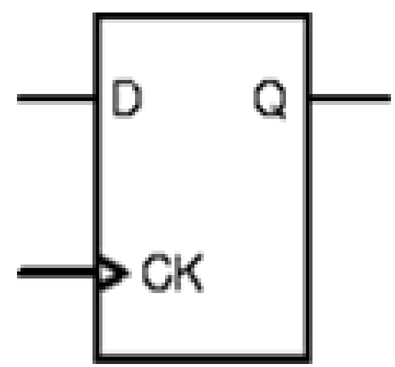
- SR-Latch: Truth table

S	R	Q	Q'
0	0	Latch	
0	1	0	1
1	0	1	0
1	1	Undefined	

Memory

- Useful variation on the SR latch circuit is the Data latch, or D latch
- Constructed by using the inverted S input as the R input signal
  - Allows for a single input → No race condition as input is inverted

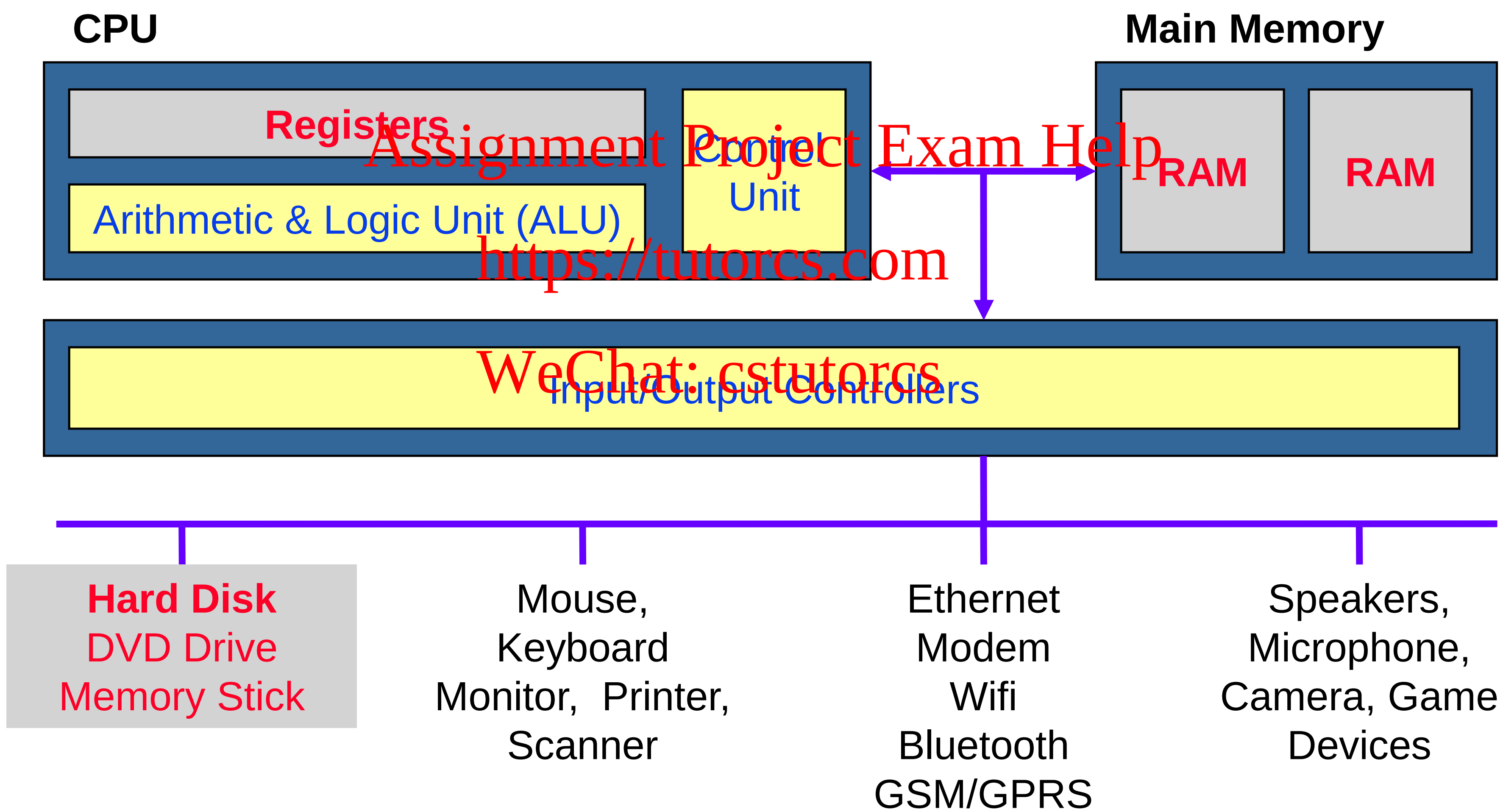




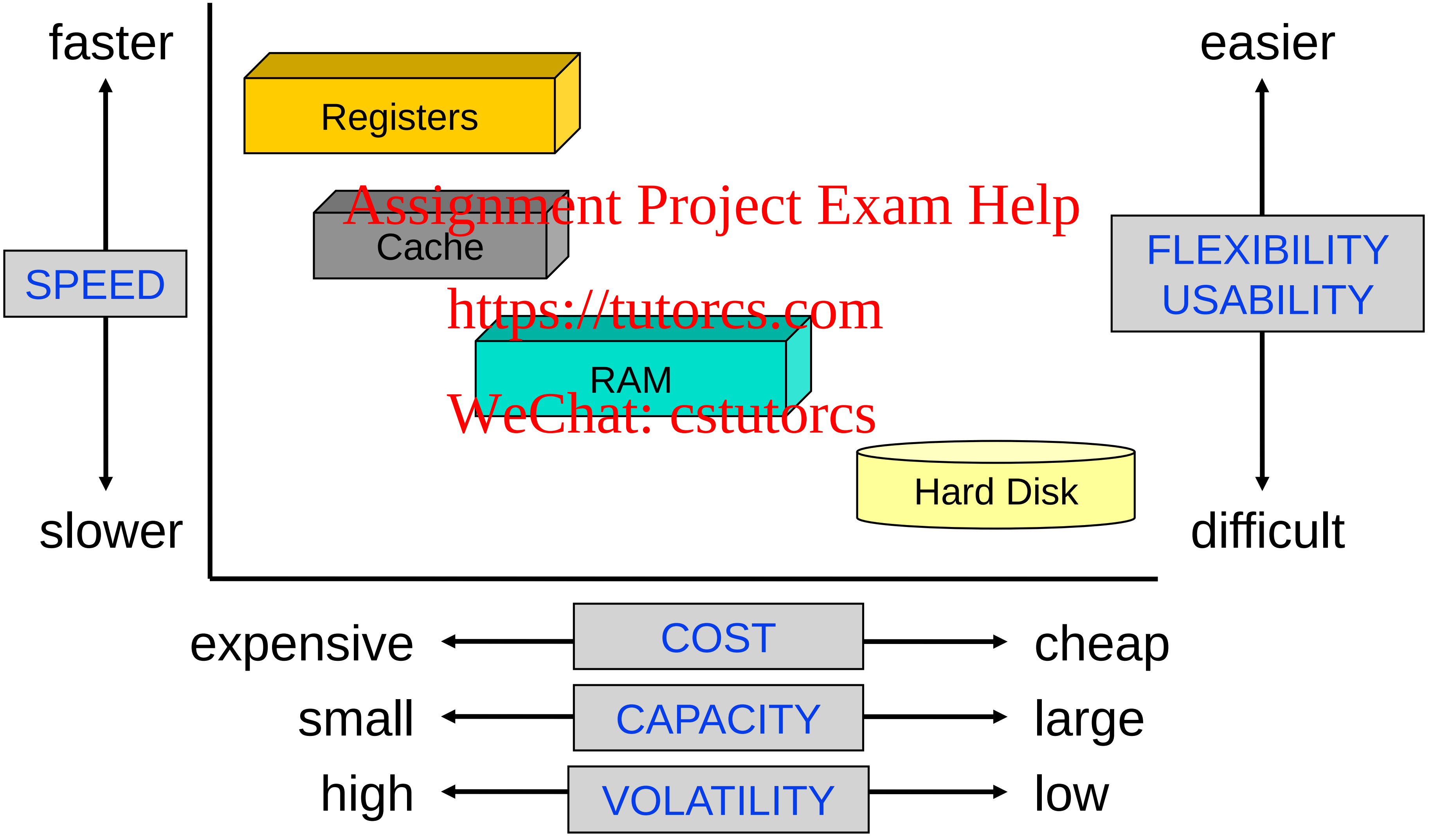
Memory

- Memories hold binary values
  - Data (e.g. Integers, Reals, Characters)
  - CPU Instructions (i.e. Computer Programs)
  - Memory Addresses (“Pointers” to data or instructions)
- Contents remain unchanged unless overwritten with a new binary value
  - Some of them *lose* the content when power is turned off (volatile memory)

Computer Architecture



Summary


faster
slower
SPEED
Registers
Cache
RAM
Hard Disk
easier
difficult
FLEXIBILITY
USABILITY
expensive
COST
cheap
small
CAPACITY
large
high
VOLATILITY
low

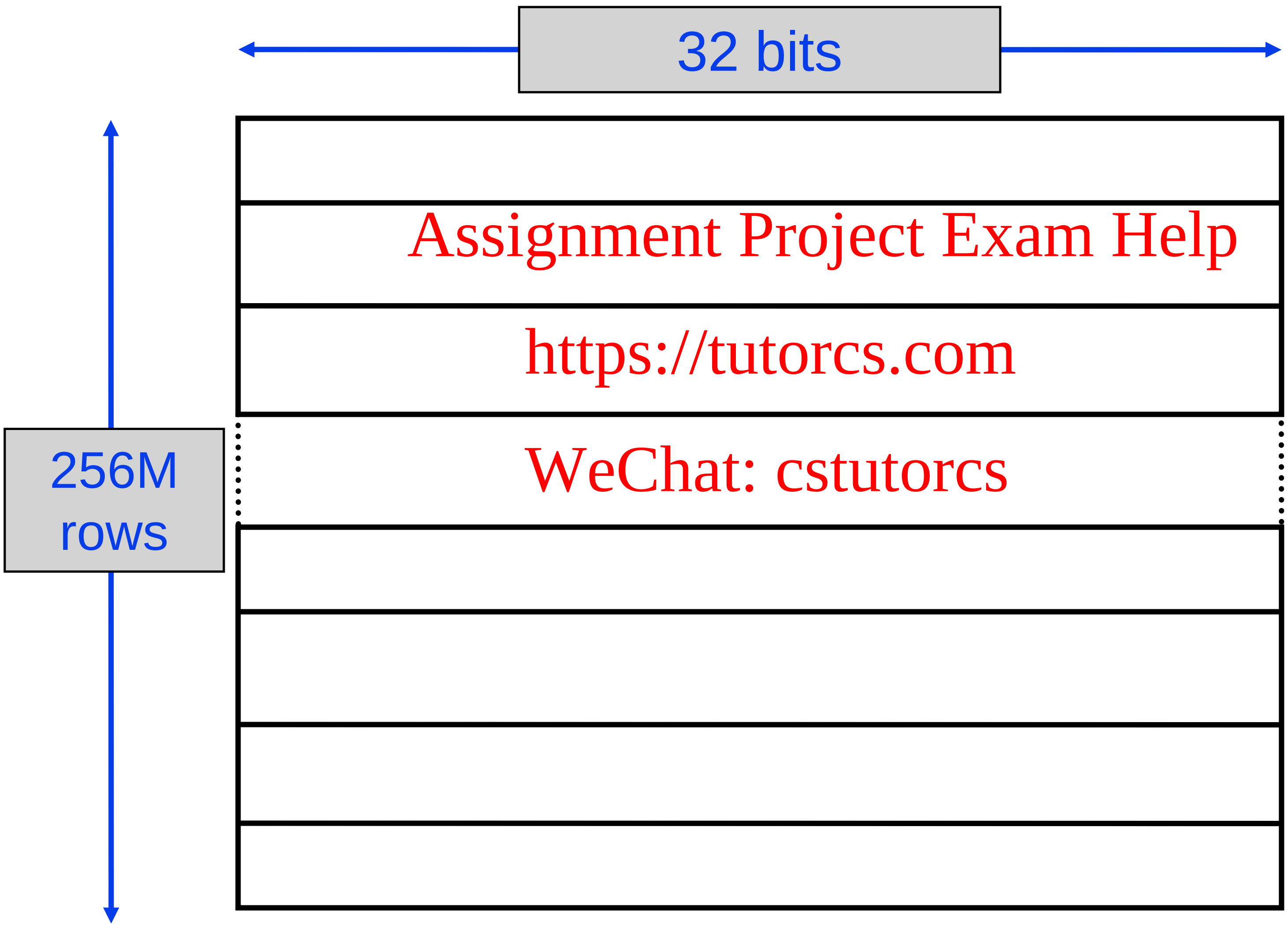
Byte Addressing (Big Endian)

Byte Address	Main Memory		Byte Address
0 →	0110 1101	1010 1101	← 1
2 →	0000 0000	0000 0011	← 3
4 →	0000 0000	0000 0000	← 5
6 →	1111 1111	1111 1111	← 7
8 →	0000 0000	0000 0000	← 9
10 →	1001 1010	1010 0010	← 11
12 →	0000 0000	0000 0000	← 13
14 →	1111 1111	1111 1110	← 15

Byte Addressing (Little Endian)

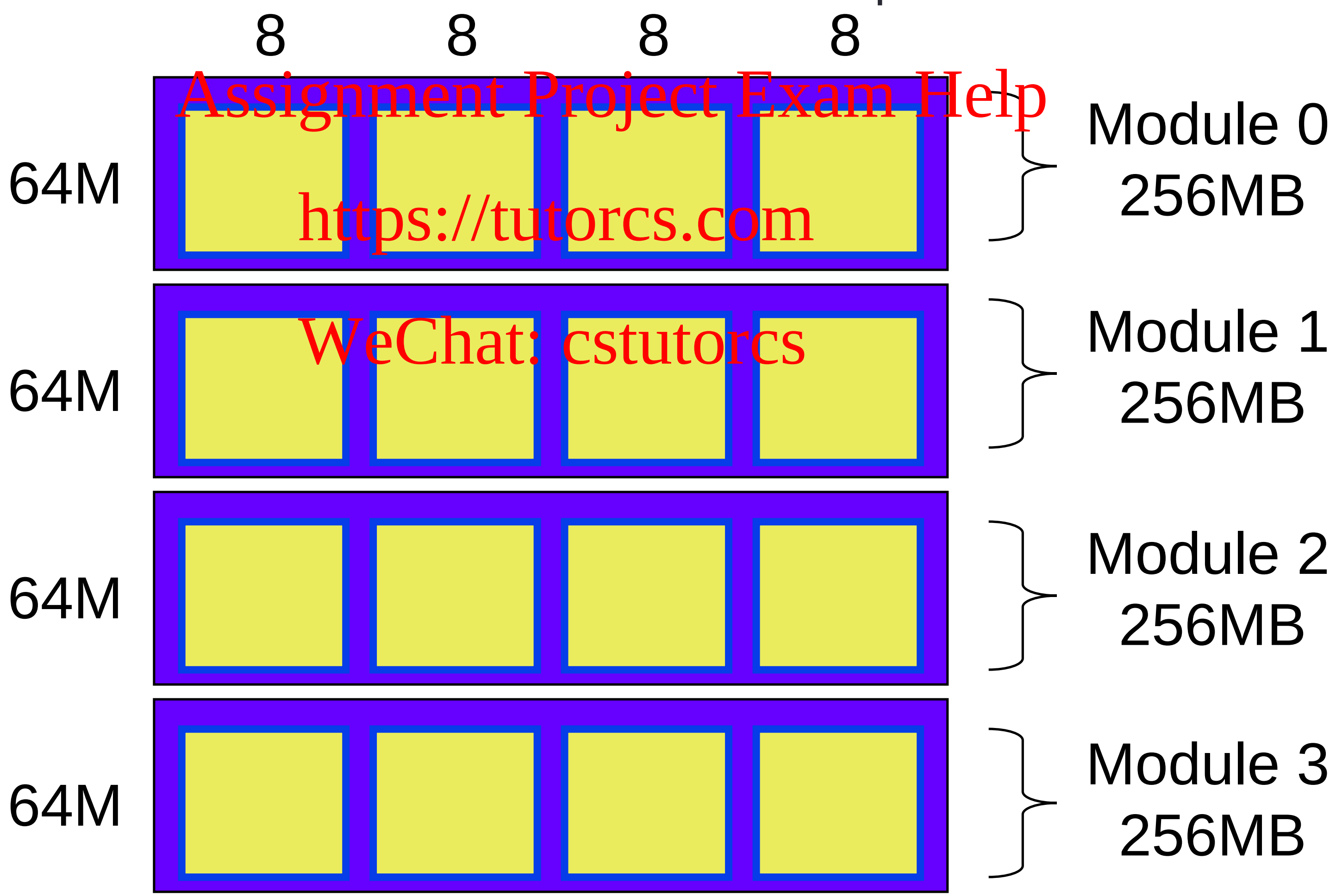
Byte Address	Main Memory		Byte Address
1	0110 1101	1010 1101	0
3	0000 0000	0000 0011	2
5	0000 0000	0000 0000	4
7	1111 1111	1111 1111	6
9	0000 0000	0000 0000	8
11	1001 1010	1010 0010	10
13	0000 0000	0000 0000	12
15	1111 1111	1111 1110	14

1GB (256M x 32-bit) Memory



1GB (256M x 32-bit) Memory

- Four 256MB memory modules
  - Each module has four 64M x 8-bit RAM Chips



Memory Interleaving

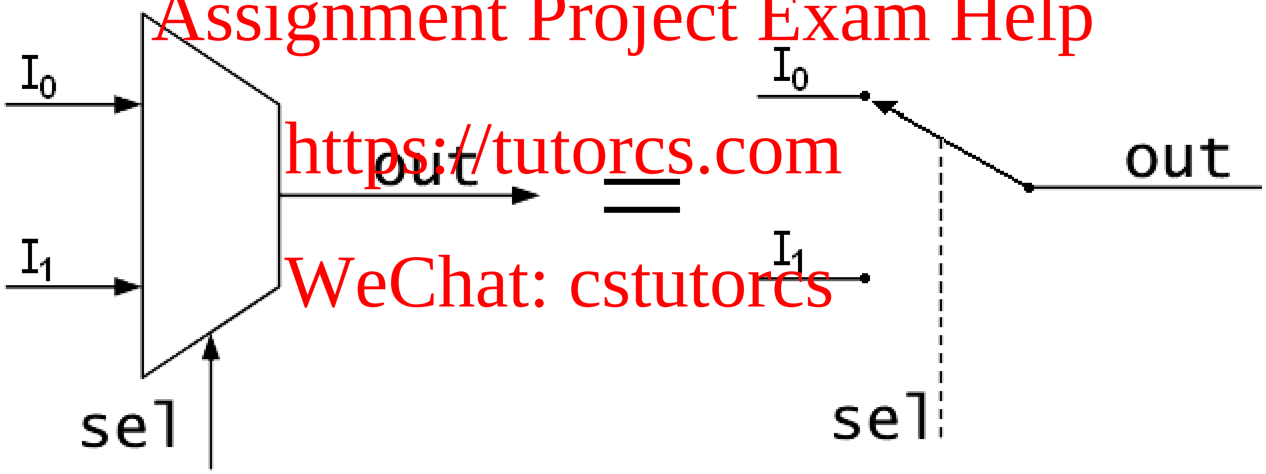
- Example:
  - Memory = 4M words, each word = 32-bits
  - Built with 4 x 1M x 32-bit memory modules
  - For 4M words we need 22 bits for an address
  - 22 bits = 2 bits (to select Modules) + 20 bits (to select row within Module)

2	20	
Module	Row within Module	High-Order Interleave

20	2	
Row within Module	Module	Low-Order Interleave

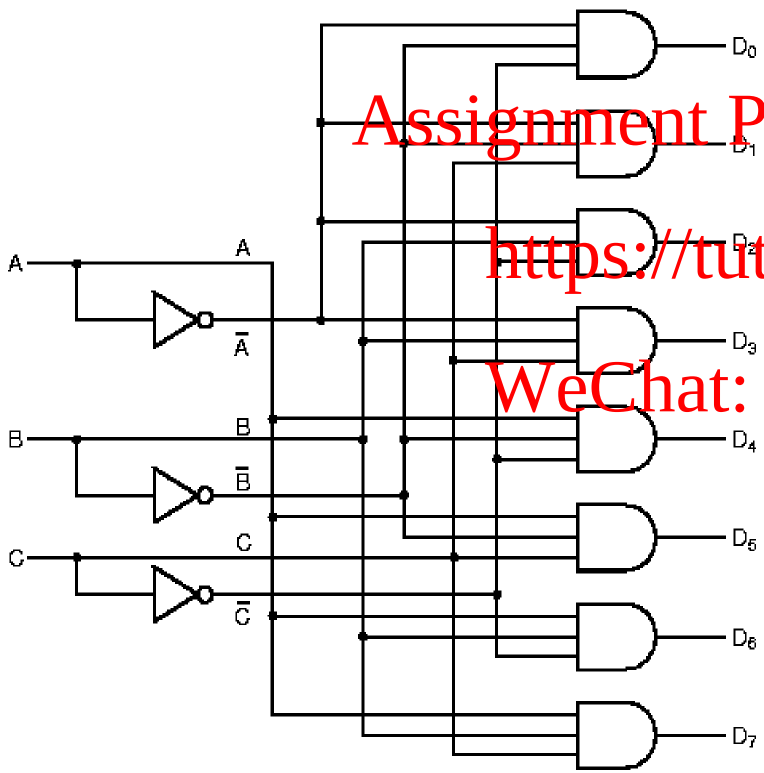
MSI Chips – Multiplexer

- A multiple-input, single-output switch
- Also called MUX for short ☺



- **sel** selects which of I_0 or I_1 is mapped to the output
- For example, **sel** = 0 selects I_0 and **sel** = 1 selects I_1
- Example is called a 2-to-1 MUX
- With n *selects*/control lines, we can have 2^n input lines

MSI Chips – Decoder



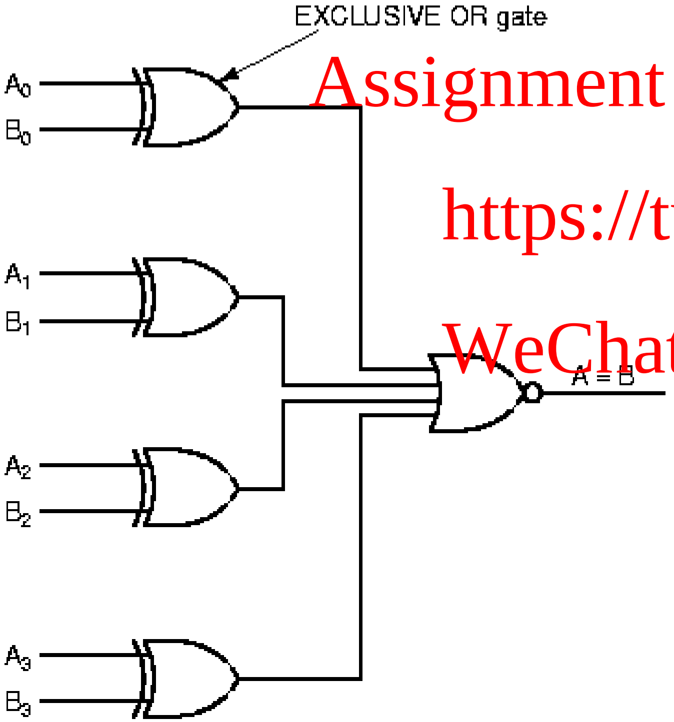
- Only one output is 1 – the one selected by the n-bit binary input number – the rest are zero
- Useful in transmitting line selection with fewer wires (e.g. selecting a memory chip)

MSI Chips – Decoder

- Truth Table

A	B	C	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

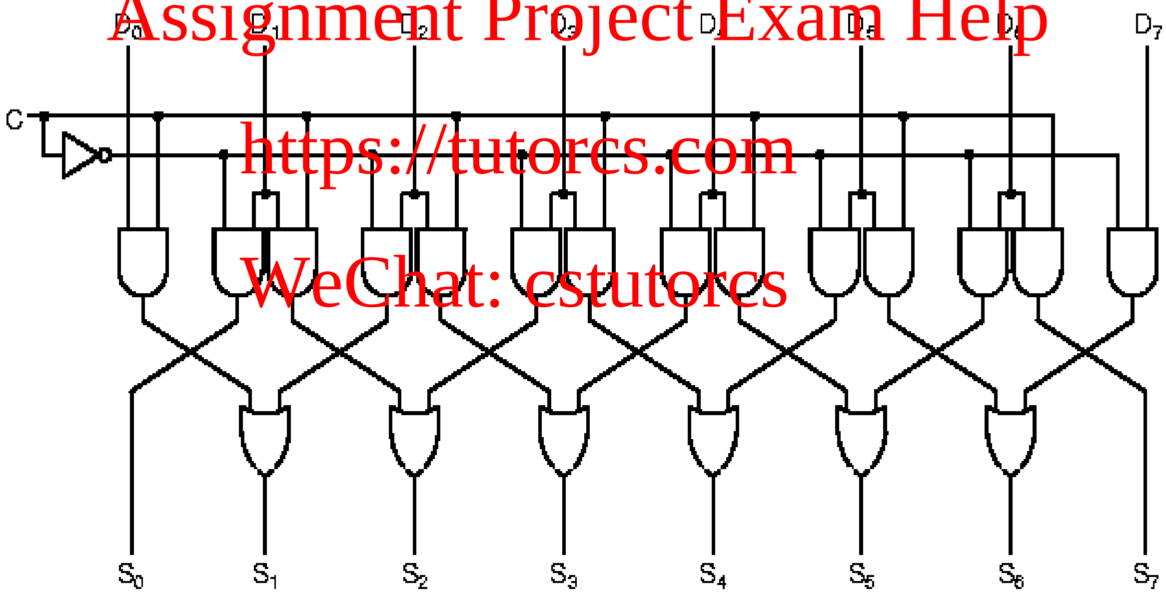
MSI Chips – Calculations – Comparator



- The comparator returns 1 if the two n-bit inputs A and B are equal, 0 otherwise

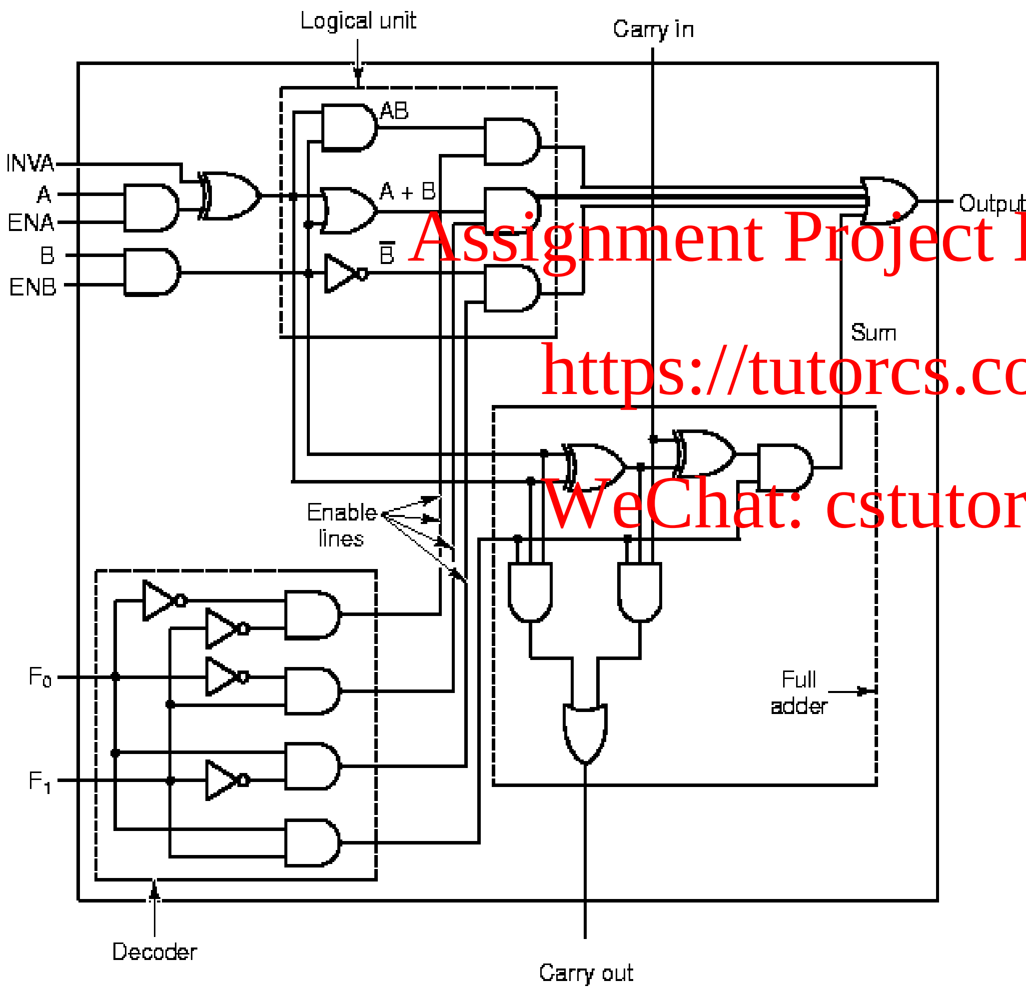
MSI Chips – Calculations – Bit-shifter

- Faster calculations for powers of 2
- Shift left and right (multiply and divide)



- c = 0 → shift left
- c = 1 → shift right

The Arithmetic Logic Unit (ALU)



- The ALU is able to perform multiple functions
- Depending on the input to the decoder (F_0,F_1) one of four functions is selected – A and B, A or B, not B, arithmetic A+B

Data representation

Bit Pattern	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Unsigned	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Sign & Magnitude	+0	+1	+2	+3	+4	+5	+6	+7	−0	−1	−2	−3	−4	−5	−6	−7
1s Complement	+0	+1	+2	+3	+4	+5	+6	+7	-7	-6	-5	-4	-3	-2	-1	-0
2s Complement	+0	+1	+2	+3	+4	+5	+6	+7	-8	-7	-6	-5	-4	-3	-2	-1
Excess-8	-8	-7	-6	-5	-4	-3	-2	1	0	1	2	3	4	5	6	7
BCD	0	1	2	3	4	5	6	7	8	9	-	-	-	-	-	-

ASCII Character Set

			Bit positions 654					Bit positions 3210
000	001	010	011	100	101	110	111	
NUL	DLE	SP	0	@	P	‘	p	0000
SOH	DC1	!	1	A	Q	a	q	0001
STX	DC2	“	2	B	R	b	r	0010
ETX	DC3	#	3	C	S	c	s	0011
EOT	DC4	$	4	D	T	d	t	0100
ENQ	NAK	%	5	E	U	e	u	0101
ACK	SYN	&	6	F	V	f	v	0110
BEL	ETB	‘	7	G	W	g	w	0111
BS	CAN	(	8	H	X	h	x	1000
HT	EM	)	9	I	Y	i	y	1001
LF	SUB	*	:	J	Z	j	z	1010
VT	ESC	+	;	K	[	k	{	1011
FF	FS	,	<	L	\	l	|	1100
CR	GS	-	=	M	]	m	}	1101
SO	RS	.	>	N	^	n	~	1110
SI	US	/	?	O	_	o	DEL	1111

Strings are represented as sequence of characters. E.g. **Fred** is encoded as follows:

English	F	r	e	d
ASCII (Binary)	0100 0110	0111 0010	0110 0101	0110 0100
ASCII (Hex)	46	72	65	64

Two’s Complement – BNA Summary

- Addition
  - Add the values, discarding any carry-out bit

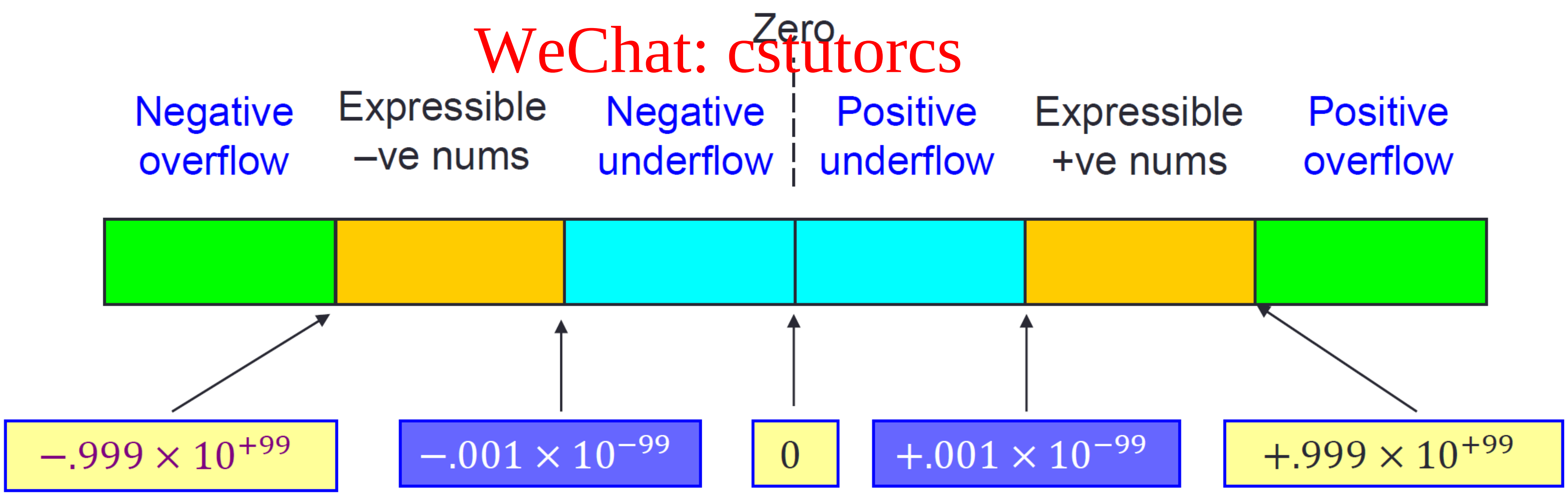
- Subtraction
  - Negate the subtrahend and add, discarding any carry-out bit

- Overflow
  - Adding two positive numbers produces a negative result
  - Adding two negative numbers produces a positive result
  - Adding operands of unlike signs never produces an overflow
  - **Note** - discarding the carry out of the most significant bit during Two's Complement addition is a normal occurrence, and does not by itself indicate overflow

Floating point zones of expressibility

- **Example**: assume numbers are formed with a **signed 3-digit coefficient** and a **signed 2-digit exponent**

- **Zones of expressibility:**



Normalised forms (base 10)

Number	Normalised form
23.2×10^4	2.32×10^5
-4.01×10^{-3}	-4.01×10^{-3}
$343\,000 \times 10^0$	3.43×10^5
$0.000\,000\,098\,9 \times 10^0$	9.89×10^{-8}

Binary fraction to decimal fraction

- What is the binary value **0.01101** in decimal?

	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
.	0	1	1	0	1

- $\frac{1}{4} + \frac{1}{8} + \frac{1}{32} = \frac{13}{32} = 0.40625$

32	16	8	4	2	1
.	0	1	1	0	1

- $\frac{8+4+1}{2^5} = \frac{13}{32}$
- What about 0.000 110 011?
- Answer: $\frac{32+16+2+1}{2^9} = \frac{51}{512} = 0.099\,609\,375$

Floating point multiplication

$$
\begin{aligned}
N_1 \times N_2 &= \left(M_1 \times 10^{E_1}\right) \times \left(M_2 \times 10^{E_2}\right) \\
&= (M_1 \times M_2) \times \left(10^{E_1} \times 10^{E_2}\right) \\
&= (M_1 \times M_2) \times \left(10^{E_1+E_2}\right)
\end{aligned}
$$

- That is, we **multiply the coefficients** and **add the exponents**
- Example:

$$
\left(2.6 \times 10^{6}\right) \times \left(5.4 \times 10^{-3}\right) = (2.6 \times 5.4) \times \left(10^{3}\right)
$$
$$
= 14.04 \times 10^{3}
$$

- We must also **normalise the result**, so final answer is 1.404×10^{4}

Floating point addition

- A floating point addition such as $4.5 \times 10^3 + 6.7 \times 10^2$ is not a simple coefficient addition, unless the exponents are the same. Otherwise, we need to align them first

$$N_1 + N_2 = \left(M_1 \times 10^{E_1}\right) + \left(M_2 \times 10^{E_2}\right)$$
$$= \left(M_1 + M_2 \times 10^{E_2 - E_1}\right) \times 10^{E_1}$$

- To align, **choose the number with the smaller exponent** and **shift its coefficient** the corresponding number of digits to the right

$$4.5 \times 10^3 + 6.7 \times 10^2 = 4.5 \times 10^3 + 0.67 \times 10^3$$
$$= 5.17 \times 10^3 = 5.2 \times 10^3$$

(rounded)

IEEE Single precision format (32-bit)

Sign S	Exponent E	Significand F
1 bit	8 bits	23 bits

- Coefficient is called the **significand** in the IEEE standard
- Value represented is $\pm\mathbf{1}.\boldsymbol{F} \times \mathbf{2}^{\boldsymbol{E}-\mathbf{127}}$

- The **normal bit** (the 1.) is omitted from the significand field ➔ a **hidden bit**
- Single precision yields **24 bits** (approx. **7 decimal digits** of precision)
- **Normalised ranges** in decimal are approximately:

$$-\mathbf{10^{38}}\ \textbf{to}\ -\mathbf{10^{-38}},\quad \mathbf{0},\quad \mathbf{10^{38}}\ \textbf{to}\ \mathbf{10^{-38}}$$

Special values

- IEEE formats can encode five kinds of values: **zero**, **normalised numbers**, **denormalised numbers**, **infinity** and **not-a-number (NaNs)**

- Single precision representations:

IEEE value	Sign field	Exponent	Significand	True exponent	Value
± 0	0 or 1	0	0 (all zeros)		$\pm 0.0 \text{ x } 2^{0}$
$\pm$ denormalised no.	0 or 1	0	Any non-zero bit pattern	-126	$\pm 0.\text{F x } 2^{-126}$
$\pm$normalised no.	0 or 1	1 ... 254	Any bit pattern	$-126 \ldots 127$	$\pm 1.\text{F x } 2^{\text{E}-127}$
$\pm\infty$	0 or 1	255	0 (all zeros)		$\pm 1.0 \text{ x } 2^{128}$
Not-a-number	0 or 1	255	Any non-zero bit pattern		$\pm 1.\text{F x } 2^{128}$


That's all Folks!