Q.7. loan (l-no, b_name, amount)
branch (b_name, b_city, assets)
customer (c_name, c_sheet, c_city)
account (a-no, b_name, balance)
depositor (c-name, a_no.)
borrower (c_name, l_no.)

* From clause

→ Eg: cartesion product borrower X loan then:
select *
From borrower, loan

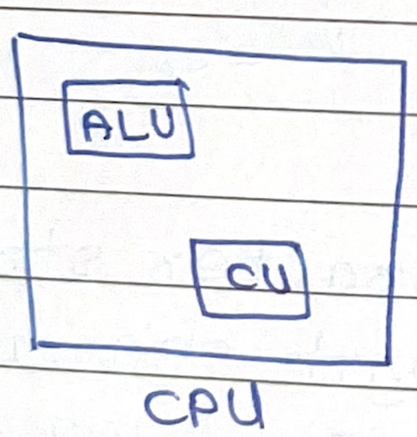
Imp

—

29/09/23
Tuesday

CO Lecture

*



Control ~~hardware~~ signal generations

Hardware signal generation
↓
In RISC V
↓
Here instruction length is Fixed.
↓
Faster approach

Microprogrammed approach
↓
In CISC
↓
Eg: x86.
↓
Here instruction length can be variable
↓
slower approach

* Integer

↳ 1] overflow
sign
added,
result

2] subtra
take tw

-128

Imp

* Range
→ Range
by incre
↳ Insign-
by shif
and fi

a] 1001
+ 0101
1110

c] 0011
+ 0100
0111

* Integer representation → signed / → unsigned.

↳ 1] overflow rule: If two numbers with same sign [both positive or both negative] are added, then overflow occurs if and only if the result has opposite sign.

2] Subtraction Rule: To subtract B from A, take twos complement of B and add it to A.

-128	64	32	16	8	4	2	1

Imp.

Imp.

* Range extension

→ Range of numbers can be expressed is extended by increasing the bit length.

↳ In sign-magnitude notation this is accomplished by shifting the sign bit more towards left, and filling remaining unfilled values with 0.

a] 1001 = -7
+ 0101 = 5
1110 = -2

b] 1100 = -4
+ 0100 = 4
$\boxed{1}$ 0000 = 0.

c] 0011 = 3
+ 0100 = 4
0111 = 7

d] 1100 = -4
+ 1111 = -1
$\boxed{1}$ 11011 = -5
-32 16 8 4 2 1

eg: 1001 = -7
+ 1010 = -6

1 0011 = -13 overflow
-16 8 4 2 1

* Block diagram of Hardware for addition and subtraction.

Ch.10

* RISCV has 32 64-bit general purpose register.

* Arithmethic for Multimedia
→ Advanced vectorised extension. 256
cpufetch 512

—

29/09/23 MSD Lecture
Tuesday

* shape in Numpy.
Eg: (2,3) → 2 rows 3 columns.

* Ndim, size.

* np.arrange() function → similar to the list slicing
└ create 1D array.

Eg: a=np.ar
print (a.
print (a)

numpy.arra

* Numpy Ar

* operatio

* a= np.c
b = np.a
a*b = [
↳ so

∴

Multiplicatio

* For matrix
m1 = np.
m2 = n

* b= num
b= num

For normal distribution mean, mode and median are the same.

↑

Normal Distribution

* → A continous random variable X is set to follow normal distribution if its PDF is give by:

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \left[\exp\left(\frac{-1}{2} \left(\frac{x-\mu}{\sigma} \right)^2 \right) \right]$$

$\sqrt{2\sigma\pi}$

where $-\infty < x < \infty$

$-\infty < \mu < \infty$

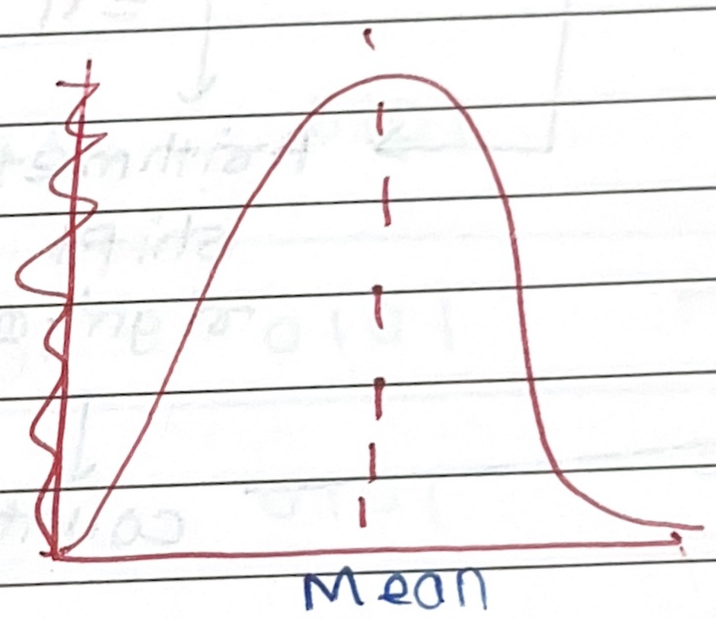
$\sigma > 0$

$\mu \rightarrow$ Mean

$\sigma \rightarrow$ standard Deviation.

$\sigma^2 \rightarrow$ Variance

$$\text{Z-score} = z = \frac{x-\mu}{\sigma^2}$$



—

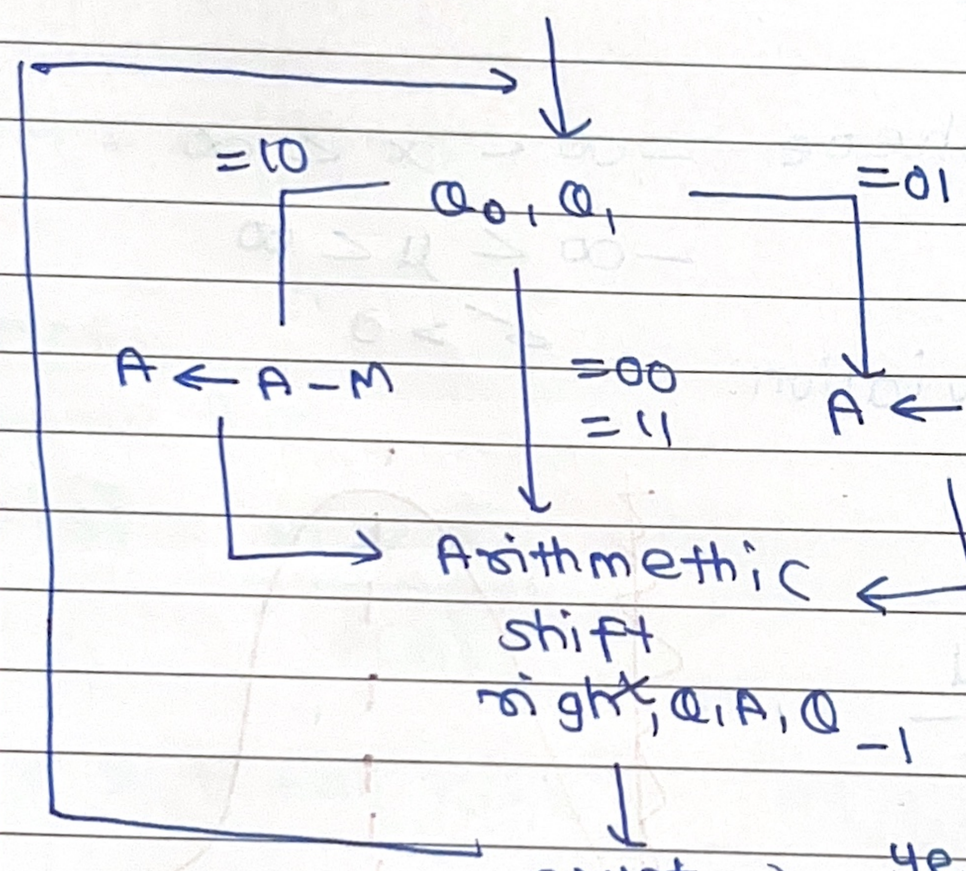
CO Lecture

09/10/23
Monday

* Unsigned and signed multiplication.

start
↓
set A to 0 and [M = Multiplicand
~~Q = 1~~ to 0 Q = Multiplier]
Q_{-1}
↓
Q_0, Q_{-1} = 01
↓
$A \leftarrow A + M$

Booth algo works for both signed and unsigned numbers



Q.1]. 7 × −3.

→

A	Q	Q−1	M	
0000	1101	0	0111	Intial
1001	0011	0	0111	A ← A−M
1100	1001	1	0111	Shift Q, A, Q−1
1100	0100	1	0111	Shift

↳ [this is wrong].

1100
0111
0011
1 0000
0000
1001001
1001
1101 3
0011
1100
16 − 3 = 13 (−3)
0011 1100
1101

~~algo works~~
For both
signed and
unsigned
numbers

1100
0111
0011
1 0000
1100 1001
1001
1101 3
0011
1100
16 − 3 = 13 (−3)
0011 1100
1101

~~Int~~
Intial

A ← A − M

Shift Q, A, Q−1

Shift

end

g].

−3 => 0011

~~110~~

0111

1100
+1
1101

1010
1011

1010
+1
1011
0100

1101
+0101
1010

0110
1001
1010

A	Q	Q −1	M	State
~~1100~~ 0000	1010	0	0101	Intial
0000	0101	0	0101	shift
1011	0101	0	0101	A ← A − M
1101	1010	1	0101	shift
1011	1010	1	0101	A ← A + M
1101	1101	0	0101	Shift

Q.2] 5 X −6.
→

→

5 4 3 2 1 0
1 1 1 0 1

5 4 3 2 1 0
0 1 1 1 1
0 0 1

010100
010111
011011

Q.1) Multiplicand = 23 Use by 6 bit representation.
Multiplier = 29

→

A	Q	Q-1	M	state
000000	011101	0	010111	Intial
101001	011101	0	010111	A←A-M
1st cycle 010100	101110	1	010111	Right shift
011011	101110	1	010111	A←A+M
2nd cycle 001101	110111	0	010111	Right shift
110110	110111	0	010111	A←A-M
3rd cycle 011011	011011	1	010111	Right shift
4th cycle 001101	101101	1	010111	Right shift
5th cycle 000110	110110	1	010111	Right shift
6th cycle 011101	110110	1	010111	A←A+M
001110	011011	0	010111	Right shift

↳ this is wrong.

A
000000
101001
1st cycle
110100
2nd cycle
001101
001
3rd cycle
101
1
4th
st
cycle
th

0
1

by 6 bit representation.

M	state
010111	Intial
010111	A←A−M
010111	Right shift
010111	A←A+M
010111	Right shift
010111	A←A−M
010111	Right shift
010111	Right shift
010111	Right shift
010111	A←A+M
010111	Right shift

010100
010111
011011

001101
101001
101110

① ①
000110
010111
011101

110100
+010111
1 001011

A	Q	Q−1	M	State
000000	011101	0	010111	Intial
101001	011101	0	010111	A←A−M
1st cycle 110100	101110	1	010111	Right shift
2nd cycle 001011	101110	1	010111	A←A+M
000101	110111	0	010111	Right shift
3rd cycle 101110	110111	0	010111	A←A−M
110111	011011	1	010111	Right shift
4th cycle 111011	101101	1	010111	Right shift
5th cycle 111011	110110	1	010111	Right shift
6th cycle 010100	110110	1	010111	A←A+M
001010	011011	0	010111	Right shift

Combine this = 667

* Division

→ Eg: 7 ÷ 3

→

A	Q	M	state
0000	0111	0011	Intial
0000	0011	0011	shift left A,Q
1st cycle			
1101	0011	0011	A ← A − M
1101	0011	0011	Q_0 ← 1

↳ this wrong.

```
  11101
+ 00101
  00010
```

Q.7. 17 ÷ 5.

→

A	Q	M	state
00000	10001	00101	Intial
00000	01000	00101	shift left A,Q
11011	01000	00101	A ← A − M
11011	01000	00101	Q_0 ← 0

```
  00100
  11011
  11111
  00000
```

1st cycle complete

```
  0000
  1111
  0101
  0100
```

Q.7. 17 ÷ 5

→

A
0000
0000
1110
1110
00
00
11

1st cycle

2nd cycle

3rd cycle

M	state
0011	Intial
0011	shift left A,Q
0011	A ← A − M
0011	Q_0 ← 1

wrong.

```
  11101
 +00101
 -------
1 00010
```

	state
101	Intial
01	shift left A,Q
01	A ← A − M
01	Q_0 ← 0


```
 00101
 11011
 -------
 11111
 00000
```

```
 +11011
 -------
  11101
 01000
```

```
 +00101
 -------
 00000
```

00000	01000	A ← A + M

1st cycle complete

```
 00101
 11111
 00101
 -------
1 00100
```

↳ this is wrong.

```
  11100
 +00101
 -------
 100001
  00001
  11011
 -------
  11100
```

Q.1. 17 ÷ 5

M = 00101

A — Q — M'2 complement = 11011

00000	10001	Intial
00001	00010	Left shift
11100	00010	A ← A − M
11100	00010	Q_0 ← 0
00001	00010	A ← A + M
00010	00100	Left shift
11101	00100	A ← A − M
11101	00100	Q_0 ← 0
00010	00100	A ← A + M
00100	01000	Left shift
11111	01000	A ← A − M
11101	01000	Q_0 ← 0
00100	01000	A ← A + M

1st cycle

2nd cycle

3rd cycle

4th cycle

00111	01000	
+11011	+11011	
00010	100011	
01000	10000	Left shift
00011	10000	A ← A − M
00011	10001	$Q_0 \leftarrow 1$
00111	00011	Left shift
00010	00011	A ← A − M
00010	00011	$Q_0 \leftarrow 1$

Remainder Quotient.

* Booth multiplier recording table

Multiplier		version of multiplicand selected by bit i
Bit i	Bit i−1	
0	0	0 × M
0	1	1 × M
1	0	−1 × M
1	1	0 × M

(13) × (−6)

01101

→ 01101 ⟹ 01101 ⟹ 10010

× 11010 ⟹ × 0-1+1-10 ⟹ + 1

00000 ⟹ 00000 ⟹ 10011

00000 ⟹ 00000 ⟹ 00000

11101 ⟹ 11101 ⟹ 11101

00000 ⟹ 01101 ⟹ 01101

00100 ⟹ 10011 ⟹ 10011

00000 ⟹ 00000 ⟹ 00000

00000 ⟹ 00000 ⟹ 00000

00000 ⟹ 00000 ⟹ 00000

DBMS Lecture

10/10/23
Tuesday

* views, dependancies of views
* view Expansion.

↳ chapter 3 completed

* Modeling

* Types of Attribute

* Mapping cardinality of relations.

* ER diagrams.

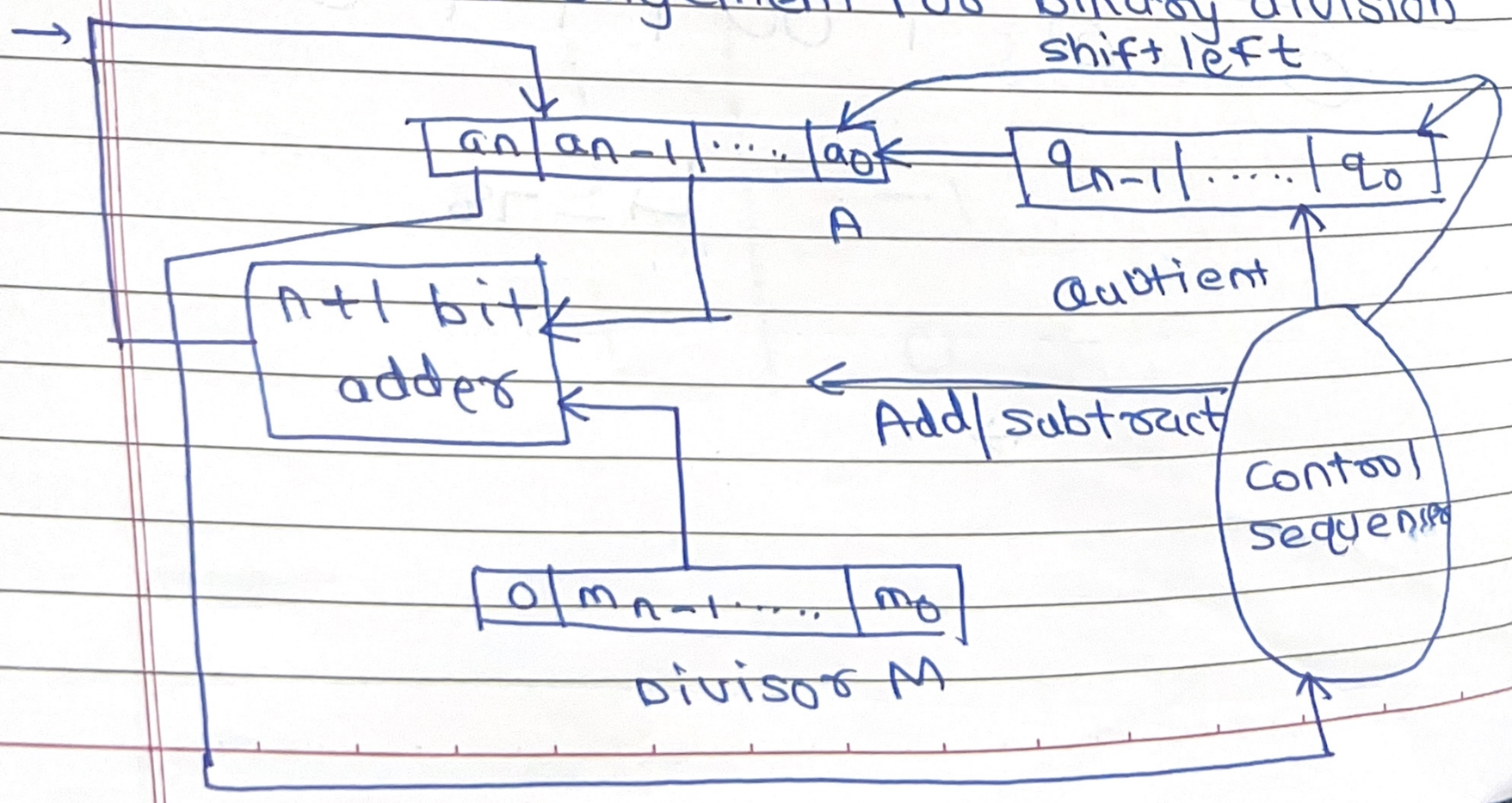
* cardinality limits

—

CO Lecture

10/10/23
Tuesday

* Circuit arrangement for binary division



Q.7] 8 -

→

	A
	0000
	0001
1st cycle	1110
	1110
	1011
	0110
	0011
→	A
	0000
	0001
1st cycle	1110
	1110
	0001
2nd cycle	0010
	1111
	1111
	0010

Q.7) 8 ÷ 3

→

M = 0011
M's = 1101
two complement

A ← A+M

A	Q		
0000	1000	Intial	0001 + 1101 = 1110
0001	0000	shift left A,Q.	
1110	0000	A ← A−M	① 1110 + 1101 = ① 1011
1110	0000	Q_0 ← 0	
1011	0000	A ← A+M	① 0110 + 1101 = ① 0 0 1 1
0110	0000	shift left A,Q.	
0011		A ← A−M	

1st cycle

↳ wrong.

M = 0011
M's = 1101
complement

→

A	Q	state	
0000	1000	Intial	1110 + 0011 = ① 0001
0001	0000	shift left	
1110	0000	A ← A−M	
1110	0000	Q_0 ← 0	0010 + 1101 = 1111
0001	0000	A ← A+M	
0010	0000	shift left	
1111	0000	A ← A−M	① ① ① 1111 + 0011 = ① 0 010
1111	0000	Q_0 ← 0	
0010		A ← A+M	

1st cycle

2nd cycle

.... [HW].

* Non-restoring algorithm steps 'n'

→ 1]. Do the following step two times:

① If sign bit 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 ~~add~~ if the sign of A is 1, add M to A.

2]. If the sign of A is 1, add M to A.

Q.1] 8/3

↳ By non-restoring M = 0011

M's = 1101

2's complements

→

A	Q		
0000	1000	Intial	0001 + 1101 = 1110
0001	0000	shift A and Q left	1st time
1110	0000	A ← A − M	
1100	0000	Shift A and Q left.	1100 + 0011 = 1111
1111	0000	A ← A + M	2nd time
1110	0000	shift A and Q left	3rd time 1110 + 1101 = 1011
0001 ~~1011~~	0000	A ← A + M	
0010	0000	Shift A and Q left	0 1110 + 0011 = 0001

s 'n'
imes:

d Q left
+M Form A,
and add

dd M TO A.

M to A.

0001
+1101
1110

1st time

1100
+0011
1111

2nd time

1
1110
+1101
1011

3rd time

0
1110
+0011
0001

+1101
1111

1111 0000 A ← A − M } 4th time

sign of A is 1 so

~~ad~~ A ← A + M

1111
+0011
0010

A = 0010.

↳ wrong

* Non-Restoring division on signed no.

→

D	V	Q	R
7	3	2	1
7	−3	−2	1
−7	3	−2	−1
−7	−3	2	−1

Q.1. D = −145 and V = 13. M =

→ Don't put A = 0.

−145 = 1111 1001 0001 (3 2 1 0)

A Q.

→

1001

M = 001101

~~A =~~

2's complement =

110011

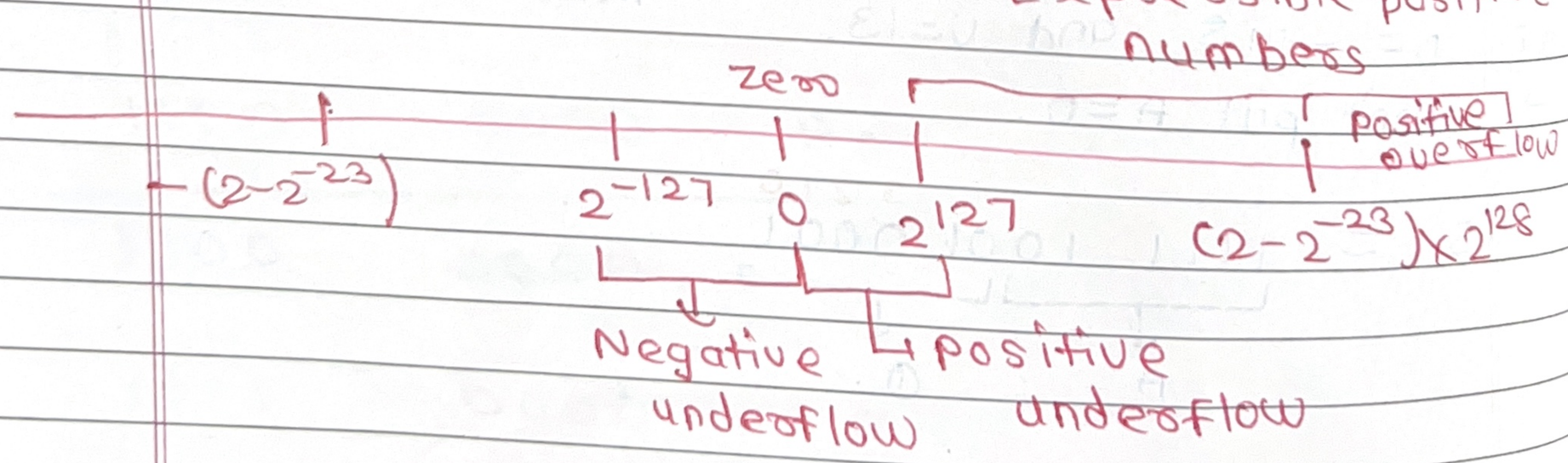
A	Q		
111110	010001	Intial	
111100	100011	shift left	
101111	100011	$A \leftarrow A - M$	
1st cycle 101111	10010	$Q_0 \leftarrow 0$	
111100	10010	$A \leftarrow A + M$	
		shift left A, Q.	

① 111100
+110011
1 101111

1111
101111
+001101
111100

↳ Wrong

* Floating point numbers.



10/10/23
Tuesday

* Normal

→ $F(x) =$

$X \sim N(0$

$\phi(x)$

* Every
can be
variabl
as z-

z =

Some rules

① $P(X \leq b)$

② $P(X > a)$

* Portion for T2

→ 1]. TCP and UDP socket
2]. TCP, UDP.
3]. TCP 3-way handshake.
4]. Connection, Termination.
5]. Sliding window
6]. RDT version
7]. Congestion control
8]. IPV4 header
9]. Fragmentation
10]. Subnetting.

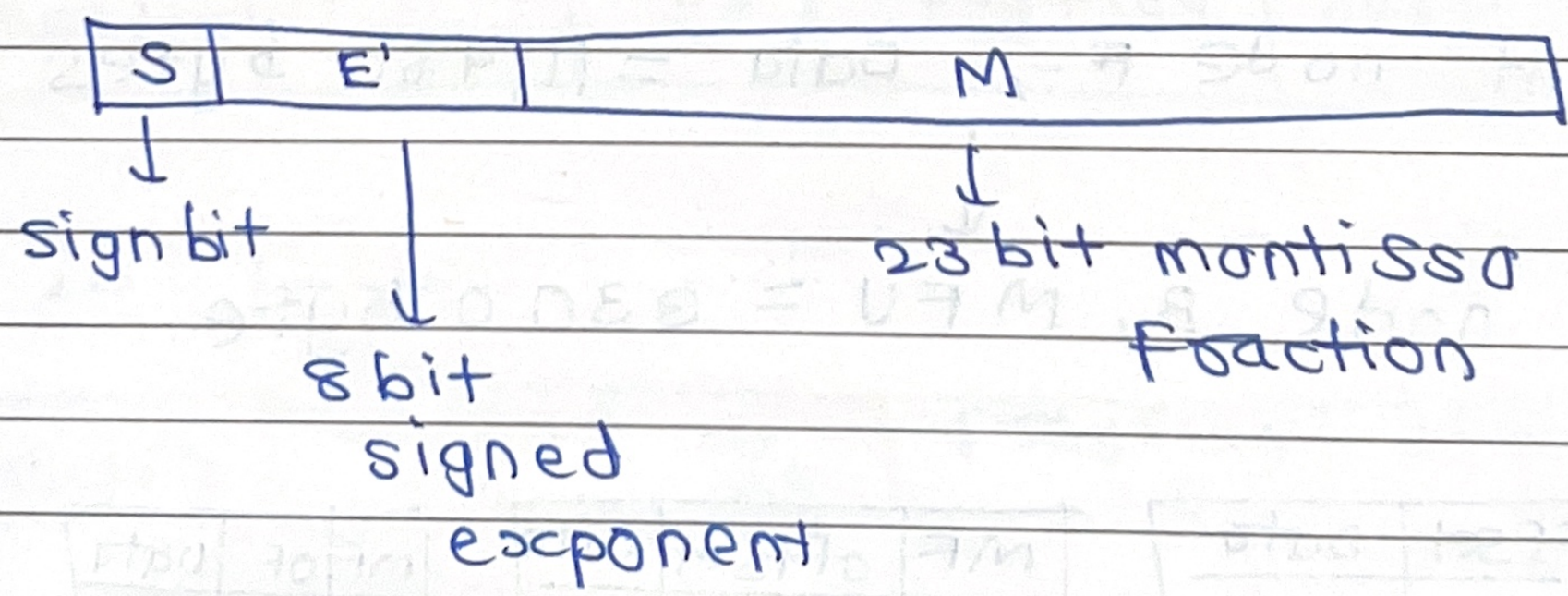
—

13/10/23 — CO Lecture

~~Mo~~ Friday

* Floating point numbers

→



a]. Single precision.

→

$0.110 * 2^5$
$110. * 2^2$
$0.0110 * 2^6$

S	E "	M
sign bit	11 bit excess	52 bit mantissa

b] Double precision

→ $B = 1.M = 1.b_{-1}\, b_{-2} \cdots b_{-23}$.

$v(B) = 1 + b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} + \cdots + b_{-23} \times 2^{-23}$.

Q.1]

	2^8 64 32 16 8 4 2 1	
0	10001000	0010110....

excess-127 exponent

0.0010110×2^9

[their is no implicit 1 to left of binary point].

a] unnormalized value

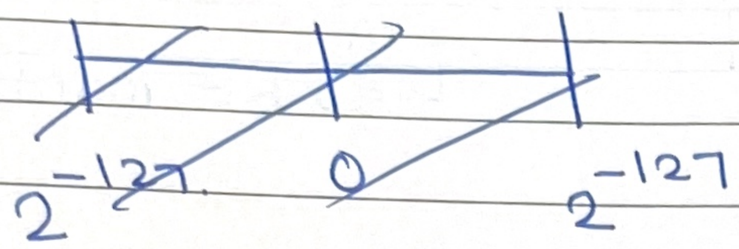
0	10000101	0110.....

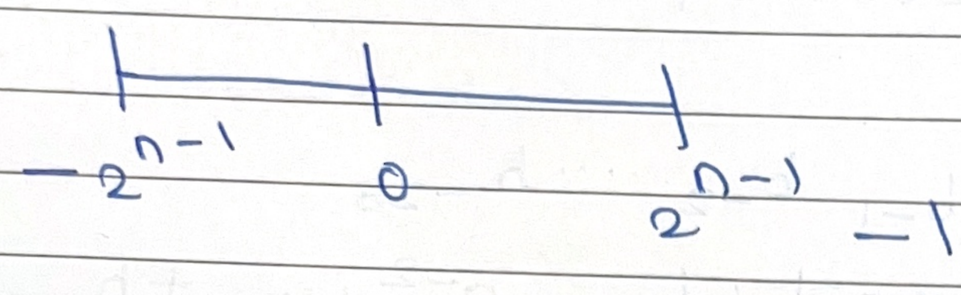
value $= 1.0110 \times 2^6$

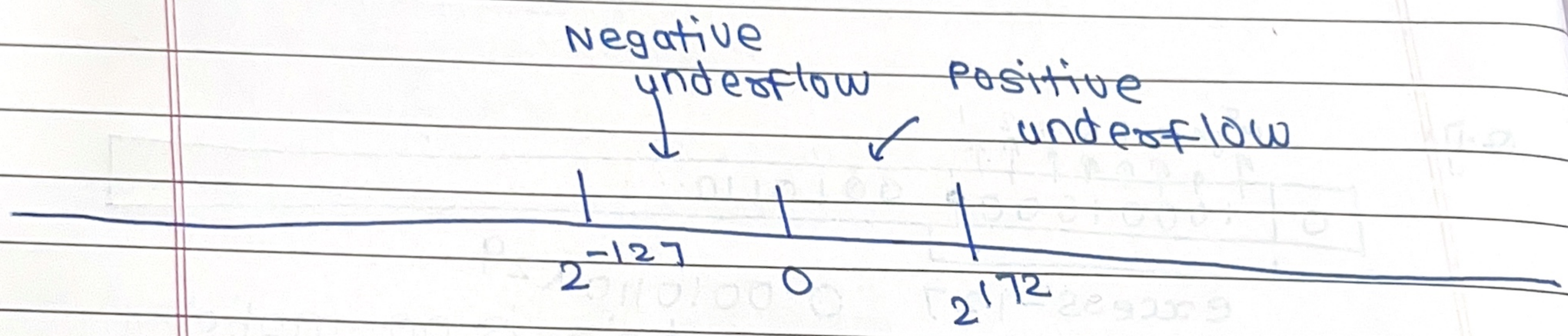
b] Normalized version.

* → Special Values
the end values 0 and 255 of excess-127 exponent

→







* IEEE standard 754

↳ ① Most Important Floating point representation is define.

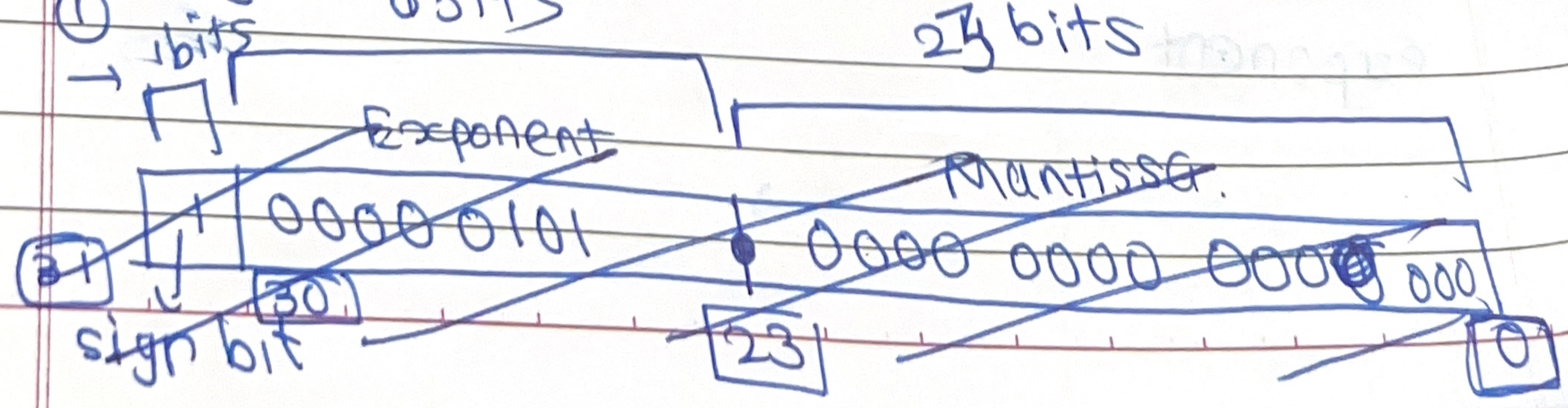
Floating point →
- Arithmethic format
- Basic format
- Interchange format.

Q.1. Express following numbers in IEEE 32bit floating point format.

→

① -5 8bits 23 bits

→ 1bits



→ 101×2^0

1.01×2^2 → excess 127
+ 2
129
↳ binary.

7654 3210
1000 0001

1 | 1000 0001 | ~~0100 0000 0000 0000 0000~~

0100 0000 0000 0000 0000 0000 000

② −6.

→ 110×2^0

~~1.10×2^2~~ → 127
~~+ 2~~
~~129~~

11.0×2^1

1.10×2^2 → 127
+ 2
129.

1 | 1000 0000 | ~~All 0's.~~

1 | 1000 0001 | 10 ..... All 0's

→

② $-1.5 \rightarrow$ sign bit $= 1$

$\rightarrow$ ~~1.5×2^0~~

~~$0.015 \times 2^2 \rightarrow 127 + 2 = 129$~~

③ -0.75

$\rightarrow = \left(\frac{-3}{2^2}\right)_{10}$

$= \frac{(-11)_2}{(2^2)_{10}}$

$= -0.11 \times 2^0$ two

$= -1.1_{two} \times 2^{-1} \rightarrow 127 - 1 = 126.$

$\boxed{1}$ $\boxed{0111\ 1110}$. $\boxed{1 \ldots \text{remaining all 0's}}$

④ $-1.5 \rightarrow$

$= \cancel{3 \times 2^{-1}}$

$= 1.$

$\boxed{1}$ $\boxed{0}$

⑤ 384

$\rightarrow$ 1 1

1.1

$\boxed{0}$ $\boxed{1000}$

⑥ 1/16 =

=

$\boxed{0}$ $\boxed{0000}$

$\boxed{0}$ $\boxed{0111}$

(4) $-1.5 \rightarrow$ sign bit $= 1$.

$= \cancel{3 \times 2^{-1}}$ $\quad \frac{3}{2}$

$= 1.10 \times 2^0 \longrightarrow \boxed{127}$

$\boxed{1}$ $\boxed{0111\ 0111}$ $\boxed{10 \ldots \text{all } 0's}$

(5) 384

$\rightarrow$ 1 1000 0000 $\times 2^0$

1.1×2^8 $\quad$ $127 + 8 = 135$

$\boxed{0}$ $\boxed{1000\ 0111}$ $\boxed{100 \ldots \text{all } 0's}$

(7) $1/16 = 0.0001 \times 2^0$

$= 1.\cancel{0} \times 2^{-4}$ $\quad$ $127 + \cancel{0}4 = 123$

$\boxed{0}$ $\boxed{0001\ 1111}$ $\boxed{0001 \text{ remaing all } 0's}$

$\boxed{0}$ $\boxed{0111\ 0011}$ $\boxed{000 \ldots \text{all } 0's}$

this last bit should come 1.

~~Many Many~~

T1	T2	Primary key
Many	Many	(P_{T1}, P_{T2})
Many	one	(P_{T1})
one	Many	(P_{T2})
one	one	(P_{T1}) or P_{T2}

$P_T \rightarrow$ primary key of table T

→ so this can be represented using two tables only.

6/10/23
Monday

Co lecture

Q.1) 0.75 ten

→ sign bit = 1.

$(0.11)_2 = 1.1 \times 2^{-1}$

$= 11.0 \times 2^{-2}$

here always 1 should come

so $127 - 1 = 126$

so $127 - 2 = 125$.

1 bits | 8 bits | 23 bits

1	0111 1101	all 0's

128 64 32 16 8 4 2 1

~~This always be negative~~