1st Year Even Sem.

Assignment

ECE 1201

BTS

University Safa

Dreambird

ZAREEN TASNIM

Name : Zareen Tasnim Bipu

Institution : Safa

Class : ECE Roll : 1810016 1810016

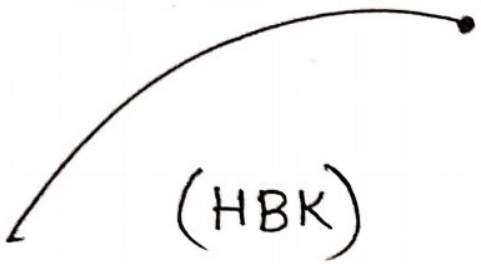
Reg. No : Board Roll :

Date Year :

ECE 3119

Computer Architechture and Design

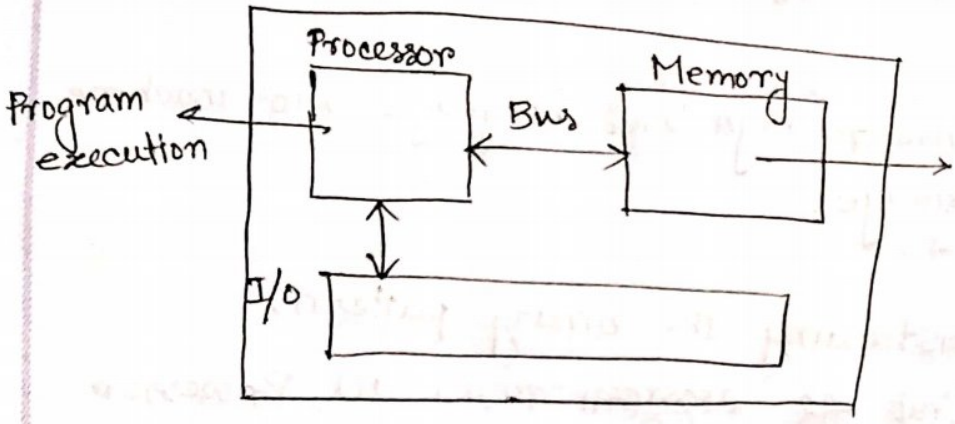
credit : 3.00



(HBK)

Subject
Date: 12.03.2022 Time: sat
L-①

Computer → compute or executing program
→ processor's job



* কোনটার কাজ কী?

Input & output Device
↙ ↘
keyboard, mouse
Monitor, Printer

Memory

* Program → a set of instruction
* Bus → connect করে রাখে

☐ Processor এর ভিতরে আছে :

→ Control Unit → Fetch decode
→ Execution Unit → Execution

Memory থেকে Bus এর মাধ্যমে Fetch করে decode করে program executed হয়।

* Processor এর কাজ কী? ↔ Fetch, Decode, Execution

Sweet Bird

Subject
Date Time

* What is Decode? → Convert এর পর আবার instruction টাকে understand করাটাই decode.

→ Machine Learning মানেই 0 and 1.

Compiler এর কাজ program কে machine Language এ convert করা।

Compiler : convert high level language into machine language.

Decode : Understanding the binary pattern.
Control Unit এই কাজগুলো করে। CU, Processor এর অংশ।

Assembler :
Converting to assembly Language into machine language.

Opt Code:

Program → (Compiler) Memory → fetch → Decode → Execution

fetch, Decode → CU

CU, Execution → Processor

ALU : Arithmatic Logic Unit :
execution এ help করে
Arithmatic operation

Sweet Bird

Subject
Date: Time

Memory : stores the program

- primary → RAM, ROM
- Secondary → Resister, FF

Cache → independent memory

How to Represent Number

→ Signed Number (can be + & −)

→ Unsigned Number (always +)

* How to represent negative number?

→ 2's compliment ও use করা হয়।

* কেন 2's complement use করি অন্য method use না করে?

✱ ৩টা method of number representation →

① Sign Magnitude :

3 bit = $2^3 = 8$ combinations

↙ Half positive ↘ Half negative numbers

+3 → 011
+2 → 010
+1 → 001
0 → 000

−1 → 101
−2 → 110
−3 → 111

(−) (−−)
sign magnitude

Subject .....
Date: ..... Time

① Sign Magnitude
② 1's compliment
③ 2's compliment

sign magnitude এ 100 represent করা যায় না কেন
it denotes negative zero which is impossible.
So, it's a drawback.

1's compliment :

0 1 1
0 1 0
0 0 1
0 0 0
1 1 0
1 0 1
1 0 0

এক্ষেত্রে,
111 missing.

এক্ষেত্রে, negative number এ sign position বাদে magnitude এর প্রত্যেক 0 থাকলে 1 হবে, 1 থাকলে 0.

So,

$111 \longrightarrow 100 = -0$

এটা negative zero denote করে which is impossible.

Subject
Date Time

2's compliment :

(1's compliment + add 1) → negative number হলে
positive number always unchanged.

+3	011
+2	010
+1	001
0	000
-1	111
-2	110
-3	101

-3 → 011
↳ 100
+1
101
✓

* short cut way for 2's compliment :

From the right hand side, copy the number as it is, till you get first 1, then invert everything.

↳ first 1 টা বসাবে, এর পর থেকে যা যা থাকবে তা invert হবে।

এখানেই, 100 missing.

After checking,

100
↙
negative, so এটার আবার 2's compliment করব

100 = 4 ⟶ এখন 100, 4 কে represent করে।

011
+1
100 = 4 (Binary)

2's compliment এ কেন negative zero নাই, আমরা এক্ষেত্রে unique combination পাই তাই আমরা 2's compliment use করি।

Binary Number :

16	8	4	2	1
2^4	2^3	2^2	2^1	2^0

3 bit → 2^3 (8) → (-4, -1) (0, 3)

4 bit → 2^4 (16) → (-8, -1) (0, 7)

8 bit → 2^8 (128) → (-128, -1) (0, 127)

5 → +5 → 0101

-5 → 1011

(3bit) 101 → এটা 2's compliment এ নিয়ে then বলব যে এটা কী।

↓

sign (negative)

$$101 \Rightarrow \begin{array}{r} 010 \\ +1 \\ \hline 011 \end{array} = +3$$

অর্থাৎ, 101 হল (-3).

000000001 → positive number এ finite leading zero থাকতে পারে।

in a same way,

111111101 → negative number এ finite leading 1 থাকতে পারে।

Subject: Mon
Date: 14.03.2022 Time: Lec-②

3 bit / 1 bit

$2^3 = 8$ $2^1 = 16$

(-4, -1) (0, 3) (-8, -1) (0, 7)

5 → +5 = 0101
→ -5 = 1011

* 7 → +7 = 0111
→ -7 = 1001

9 → +9 = 0009 001
→ -9 = 1111 0111

এখানে 4টা না দিয়ে একটা দিলেই হবে।

67 → +67 = 01000011
→ -67 = 10111101

85 → +85 = 01010101
→ -85 = 10101011

128	64	32	16	8	4	2	1
1	0	0	0	0	1	1	

Subject.....
Date:..... Time

Floating Point Number

56.7
78.2
42.5
fixed position number

0.0158
1.275
23.5
floating point number

Decimal point fixed না।

* Any group of number where the position of the number is not fixed is called floating point numer

Computer এ আমরা কিভাবে দশমিক represent করব এটা বুঝতে floating point লাগে।

73.2
649
.847
5.63

convert the floating point into fixed point
↳ Normalization Process

fixed point এ যোগ বিয়োগ করা easy.

$73.2 \rightarrow 7.32 \times 10^{1}$

$649 \rightarrow 6.49 \times 10^{2}$

$.847 \rightarrow 8.47 \times 10^{-1}$

$5.63 \rightarrow 5.63 \times 10^{0}$

একটা list দিয়ে কাকে এগুলোকে Normalization করব।

Subject 16.03.2022
Date Wed Time Lec-③

Floating Point Number :

73.2	7.32×10^1
649	6.49×10^2
.847	8.47×10^{-1}
5.63	5.63×10^0

point কখনো pc তে store করা যায় না। এটা register এ অন্যভাবে stored হয়।

Fixed point এর ক্ষেত্রে 1st এ point ignore করে then last এ বসায়ে ~~scale~~ করব।

But floating point এর ক্ষেত্রে point store ও করা যায় না। আর ignore করলে থাকবে না। তাই floating point কে fixed point এ convert করা হয়। একে Normalization বলে।

Normalization :

Rule হল just দশমিকের left side এ only one non-zero number থাকবে।

There should be only one non-zero digit to the left of the point.

Exponent ⟶ position of the point represent করি

Mantissa ⟶ number এর digit represent করি

5.63×10^0

E	M
0	563

$0101.001 = (-1)^0 \times 1.01001 \times 2^2$

$11111.01 = (-1)^0 \times 1.111101 \times 2^4$

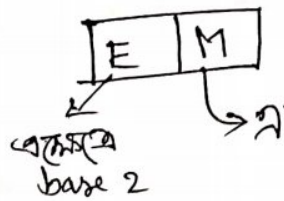
$0.0010 = (-1)^0 \times 1.01 \times 2^{-3}$

$-10.01 \Rightarrow = (-1)^1 \times 1.001 \times 2^1$

এক্ষেত্রে base 2.
দশমিক left এ যত ঘর যাবে, left এ (+) right এ (-)

Subject
Date: Time

Binary normalization এ always সর্বসামনের অংক 1 থাকে। তাই এটা extra করে store করি না।



এক্ষেত্রে base 2

এক্ষেত্রে Mantissa just সর্বসামনে দশমিক সহ digit গুলো হবে।

1.M format

Negative value গুলোর জন্য different problem:

ex pc কে hyphen set করা যাবে না।

এক্ষেত্রে,

$(-1)^S$ থাকবে S=0 positive
S=1 negative

S.E.M (digit of the number without 1 & .)

$$(-1)^S \times 1.M \times 2^E$$

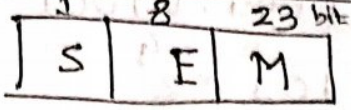
→ Normalize form for the binary pattern.

ex: M=1101
E=3

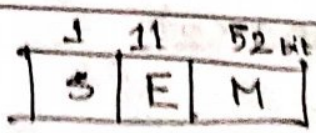
$1.1101 \times 2^3 = 1110.1$

Floating Point এর দুইটি structure.

*** IEEE-754-32 bit format (single precision)



*** IEEE-754-64 format (Double precision)



Subject:
Date: Time:

Prob যদি exponent এ minus আসে?

3×10^3
$(+)\ 2 \times 10^3$
5×10^3

3×10^3
2×10^4

extra বাদে কিভাবে লাগাবে exponent এর জন্য যদি neg হয়।
এই prb solve করবে।

Biased exponent :

এখানেও অসময় লাগে But কম।
অসময় কম লাগে কেন?
↳ ez এখানে একবারই change হয়
then ঐ change টাই সব জায়গায় use হয়।

> Bised Exponent = True Exponent + Bias
> BE = TE + Bias

Bias হিসেবে কত নিব।

Bias যদি 200 হয় negative side এ তাহলে আমরা positive number কম নিতে হবে।

For 32 bit case: 8 bit
single precision
$2^8 = 256$

200 + 56
↙ ↓
neg pos যা কম।

তাই 127 Bias নেই
Bias selection → support equal number of positive & negative

Subject
Date Time

(ii) 64-bit :

Doubb Precision

11 bit exponent

2^{11} =

1023 Bias

Q: Why we use bias?

Q: Bias 127 হয় কেন?

* Q: Convert $(14.125)_D$ → single precision

→

Step:
1. convert Binary
2. Normalization
3. Take the bias
4. Convert Bias exponent into binary
5. Substitute into required format

1110.001

1.110001×2^3

0.125 × 2 = 0.25 ↓ 0
0.25 × 2 = 0.5 ↓ 0
0.5 × 2 = 1 ↓ 1

BE = 3 + 127 = 130

$(130)_D = (10000010)_B$

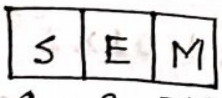
0	10000010	11000100000000000000000
S	E	M

23 bit → এখানে 23 পর্যন্ত 0 বসে এবং দশমিকের পরে যেসব সংখ্যা থাকে তা বসে যাবে

Neck done"

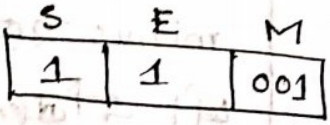
Subject: 21.03.2022
Date: Mon Time: Lec-(4)

Floating Point Number:



$\Rightarrow -5.63\ e^2 = -5.63 \times 10^2$

$\Rightarrow -563$



$\Rightarrow$ Normalize form $\Rightarrow (-1)^1 \times 1.001 \times 2^1$

$\Rightarrow -10.01.$

exponent এ neg sign থাকলে extra work করতে হয় প্রতি-calculation এ. এটা solve করতে Bias করা হয়।

Q. Bias কেন করা হয়?

→ To make the calculation faster. কেন?

exponent এ minus sign থাকলে addition করার সময় প্রতিবার compare করতে হয়। এক্ষেত্রে extra steps লাগে। আর addition এ extra step মানে multiplication, divide etc সব ক্ষেত্রেই 10 গুণ or extra steps লাগে।

Q: Biasing এর number কত করা হয়?

Q: single precision 127 কেন?

BE = 127 & BE = 1023 (Double precision)

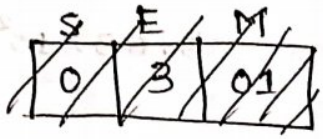
Exp: 8, $2^8 = 256 \longrightarrow \frac{255}{2} = 127.5$

So that it can support equal number of positive and negative.

Subject.....
Date:.....Time:

Q: 0.00101 represent the number in double precision.

$0.00101 = 1.01 \times 2^{-3} = (-1)^0 \times 1.01 \times 2^{-3}$

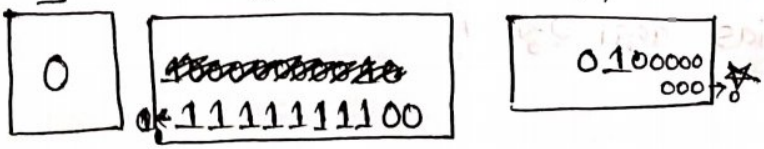


2 expo কে এভাবে রাখি না দেখেই Biasing করি।

Bias exponent = $1023 - 3 = 1020$

$(1020)_D =$ ~~1000000010~~ 1111111100

right side এ দশমিক নিলে minus হবে। left এ নিলে plus হবে।

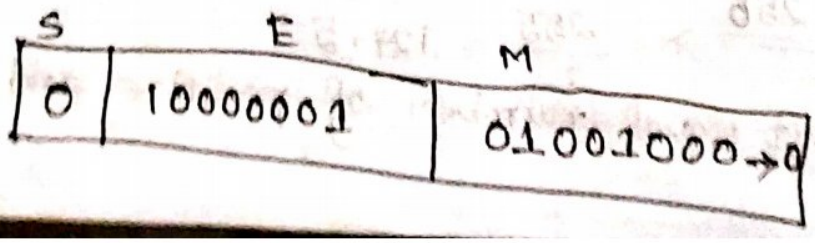


* Mantisa এর ডানে zero দিব কেন mantissa, decimal point এর পরের number represent করে। তাই এর right এ zero বসালে value change হবে না।
* Exponent এর ক্ষেত্রে left এ zero বসাতে হবে। bit fill up করার জন্য।

Q: 0101.001 → single precision.

⇒ $(-1)^0 \times 1.01001 \times 2^2$ ⇒ Normalized form

$BE = 127 + 2 = 129$



Mantissa always single precision এ 23 bit দিয়ে represented হয়।

Subject:
Date: Time:

সামনে always 0 হই দিব ex Bias টা কখনো neg দিয়া হয় না।
তাই 1 হবে না।

Book : William Stalling → W

Q: Convert (14.125) → Single precision

→ $(1110.001) = (-1)^0 \times 1.110001 \times 2^3$

BE = 130

0	10000010	11 0001 → 0

Q: (-14.125)

→ $(-1)^1 \times 1.11\,0001 \times 2^3$

S	E	M
1	10000010	110001 → 0

Q: (2A3B) H

2 → 0010, A → 1010, 3 → 0011, B → 1011

=> (0010101000111011) = (1.0101000111011)
$= (-1)^0 \times 0101000111011 \times 2^{13}$

BE = 13 + 127 = 140

1 bit	8 bit	23 bit
0	10001100	01010001110 11 → 0

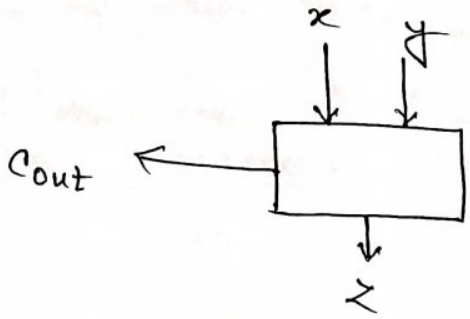
Wed
Subject 23.03.2022
Date: Lec-5 Time

Adder Circuit : দুই রকম : Half Adder, Full Adder

3	0 0 1 1	4 bit addition
+2	0 0 1 0	4 bit Adder
	0 1 0 1	

n bit adder for n bit addition

Half Adder : (1 bit Half Adder)



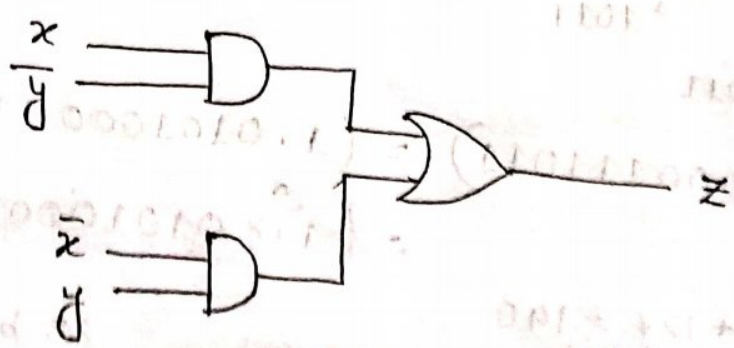
i/p → x, y

o/p → sum(z), carry (cout)

z এর মান,

$$z = x.\bar{y} + \bar{x}.y \quad \text{(sum 1 পাব)}$$

Gate Design :

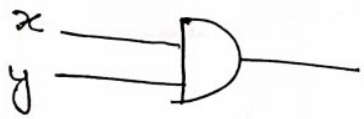


কখন আমরা sum এ 1 পাব?

Subject:
Date: Time:

carry = 1 পাওয়া যখন $x = 1$, $y = 1$

$$C_{out} = x \cdot y$$

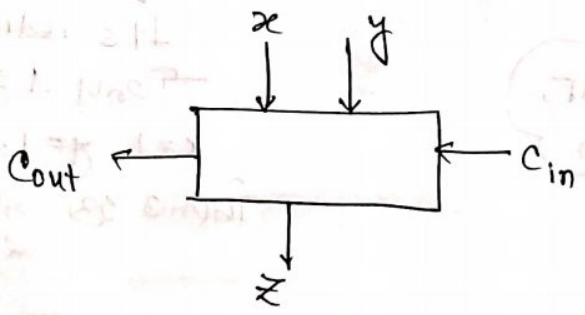


* কখন Half Adder কাজ করে?

It can add two 1 bit number independently, but it can't be used in a circuit with a series of number.

এজন্য Half adder থেকে full adder এ switch করি।

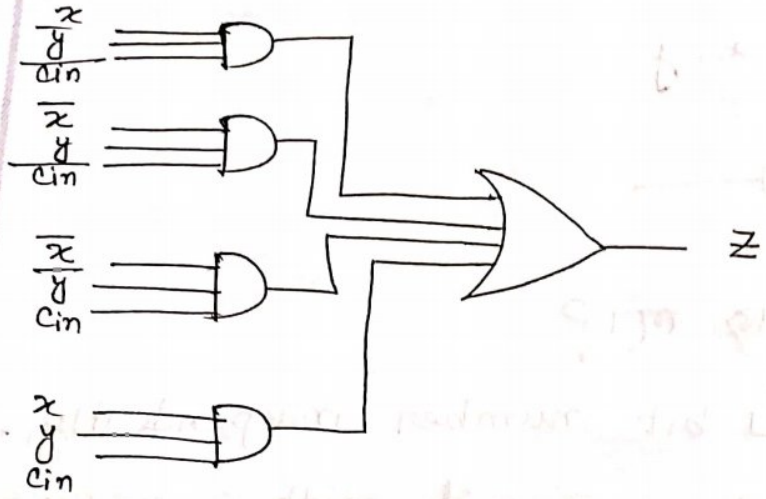
Full Adder : (1 bit Full Adder)



I/p : x, y, Cin
O/p : z, Cout

Sum 1 পাব যখন, যেকোনো একটা 1 হবে provided by the others zero, তিনটা input এর মধ্যে। আবার তিনটা 1 হলেও আমরা sum এ 1 পাব।

$$\therefore \boxed{z = x \cdot \bar{y}\,\bar{C_{in}} + \bar{x} \cdot y \cdot \bar{C_{in}} + \bar{x} \cdot \bar{y} \cdot C_{in} + x \cdot y \cdot C_{in}}$$

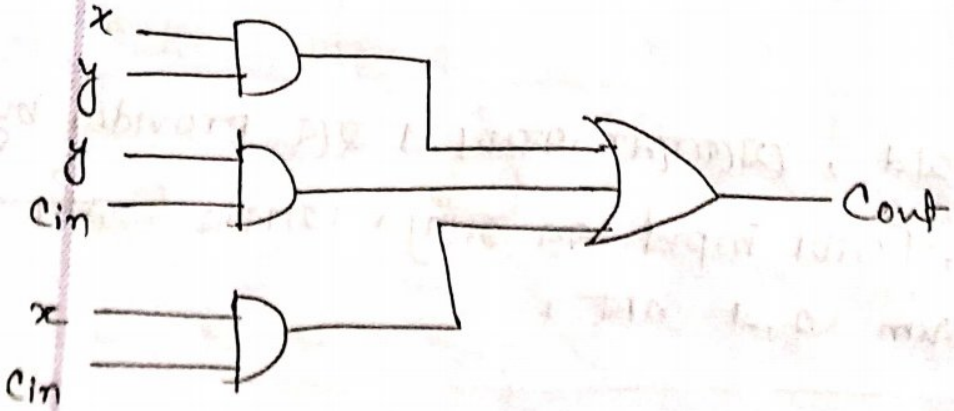


* $C_{out} = 1$ পাব তখনই— যেকোনো দুইটা 1 হলে। এক্ষেত্রে third টা দেখা লাগে না।

$$C_{out} = x.y + y.c_{in} + x.c_{in} + x.y.c_{in}$$

It's redundant
তিনটা 1 হলে:
$x=1, y=1, c_{in}=1$
দিলেও হয়, না দিলেও হয়।

generally নেই না,
$x.y.c_{in}$



—x—

Subject.....

Date:.....Time:.....

4 bit Full Adder :

$$\text{i/p} \begin{cases} x = x_0 \; x_1 \; x_2 \; x_3 \\ y = y_0 \; y_1 \; y_2 \; y_3 \end{cases} \; \& \; c_{in}$$

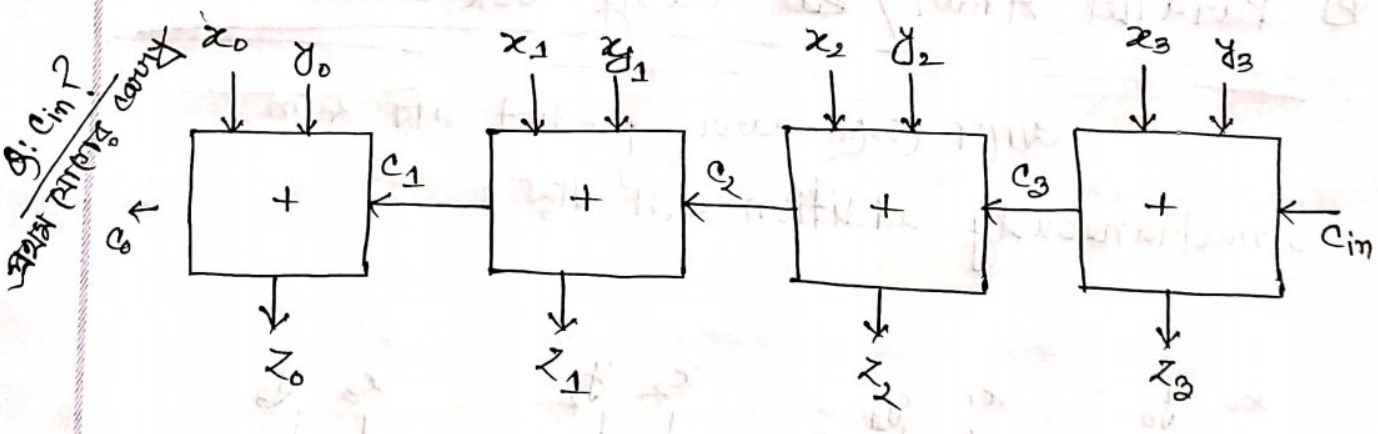
$$\text{o/p} \leftarrow z = z_0 \; z_1 \; z_2 \; z_3 \; \& \; c_0$$

$x = 0010$

$y = 10011$

Addition :

Serial /Slow/ripple ~~carry~~ Carry Adder :



প্রথম ক্ষেত্রে $c_{in} = 0$ ধরি। cz তখন কোনো carry থাকেনা।

এক্ষেত্রে ~~adding~~ addition serially হয়। serially হয় দেখে It's slow & এক্ষেত্রে carry টা ripple হয়।

অসুবিধা : একটা addition আরেকটার ওপর dependent so একটা না হলে অন্যটা হবে না।

এক্ষেত্রে, LSB টা জানা থাকব। এটা change করলে Figure change হবে।

4 bit এর জন্য 4 cycle, n bit এর জন্য n cycle হবে : time consuming.

Ripple হয় অর্থাৎ ঢেউ এর মত কাজ pass হয়, এজন্য এই adder use করা হয় না। Computer fast calculation করে।

এজন্য parallal adder এ switch করি।

→

62 bit 62 cycle

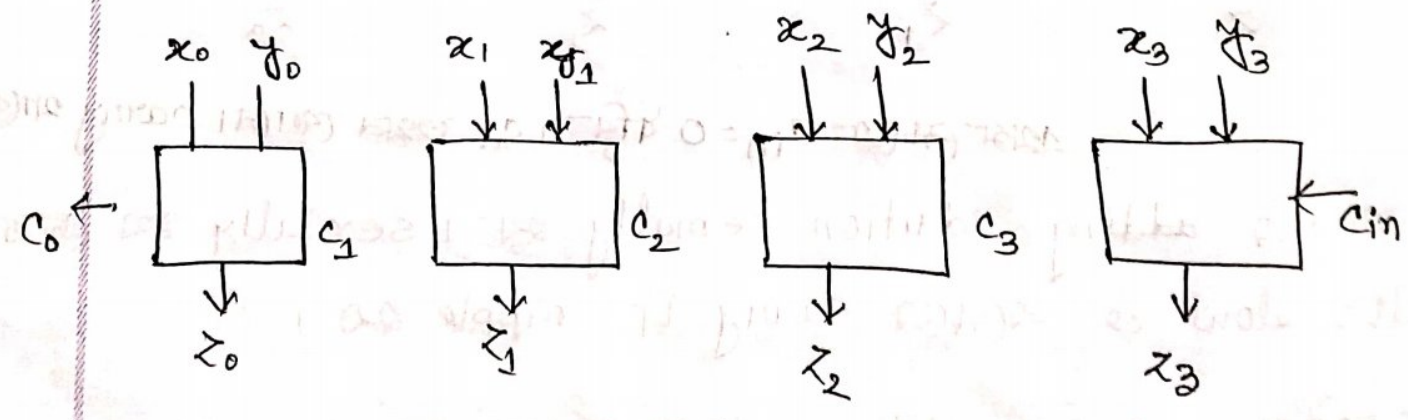
1 cycle এ 2 sec লাগলে

62 × 2 sec

for serial adder

Parallel Adder / ~~Add~~ Carry look ahead adder:

আগে থেকে carry predict করি ফলে simultaneously addition করা যায়।



generate, $g = x \cdot y$

propagate, $p = x + y$

$$C_3 = x_3 \cdot y_3 + y_3 \cdot C_{in} + x_3 \cdot C_{in}$$

$$= x_3 y_3 + C_{in}(y_3 + x_3)$$

$$= g_3 + C_{in} P_3$$

Subject.....
Date:..... Time:.....

input পেলেই C_3 generate করা যাবে।

$$C_2 = g_2 + P_2 C_3 = g_2 + P_2 (g_3 + C_{in} P_3)$$

$$C_1 = g_1 + P_1 C_2 = g_1 + P_1 (g_2 + P_2 (g_3 + C_{in} P_3))$$

C_3 বা C_2 এর জন্য অন্যটার ওপর dependent থাকা লাগে না। C_{in} দিয়ে আর input দিয়েই করা যায়।

$$\therefore C_0 = g_0 + P_0 C_1$$

$$= g_0 + P_0 \left[g_1 + P_1 \{ g_2 + P_2 (g_3 + C_{in} P_3) \} \right]$$

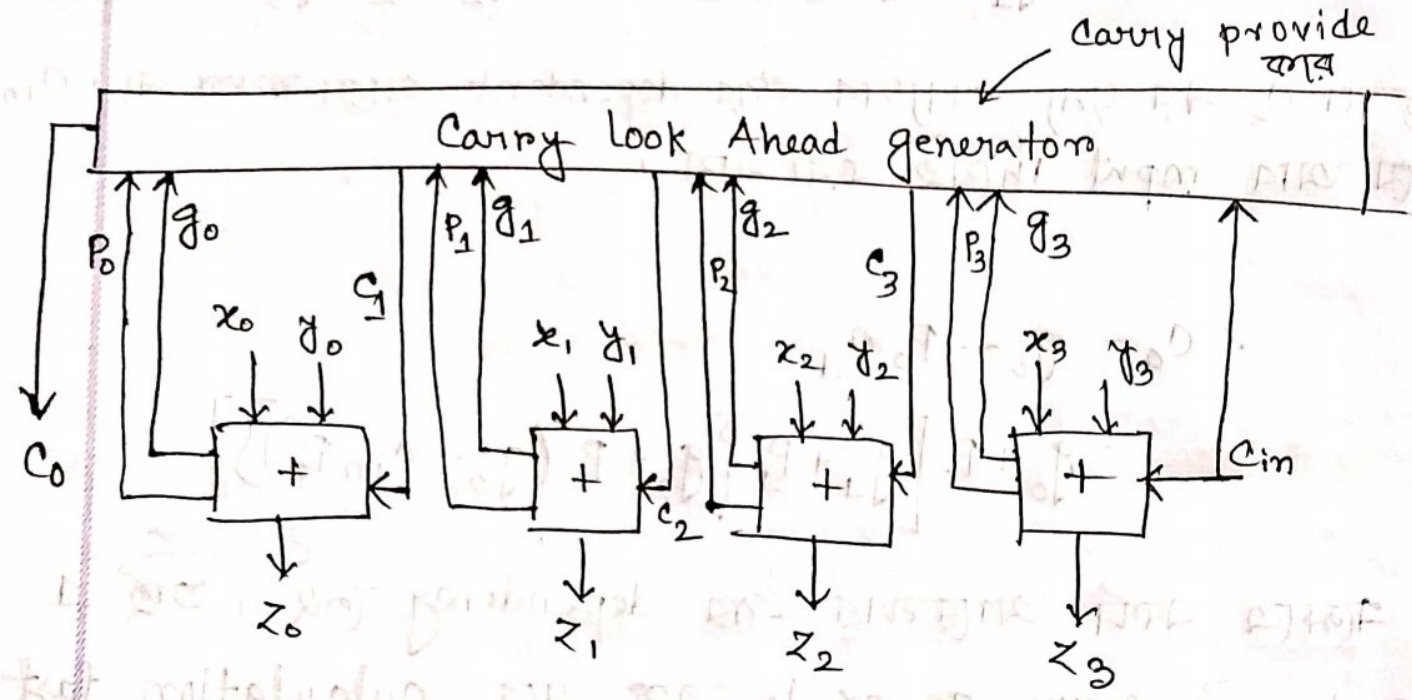
এক্ষেত্রে একটা আরেকটার ওপর dependency নেই। তাই 1 cycle এই carry generate হয়ে যাবে, calculation fast হবে।

* অনেক bit একসাথে calculation করা prb না, prb হল wait করে একটা আরেকটার ওপর depend করে calculate করা। So, parallal Adder effective.

* 1st cycle এ generate & propagate produce করবে।
2nd cycle এ carry produce করবে।
3rd cycle এ addition হবে।

Subject.....
Date:.....Time:

So, parallal adder এ 4 bit calculation এর জন্য 3টা cycle লাগবে, Serial এর জন্য 4 bit এ 4টা cycle লাগত। So time save হচ্ছে।



total টা parallal Adder

1st cycle : generate & propagate produce

2nd cycle : Carry

3rd cycle : Addition

i/p : $x = x_0\ x_1\ x_2\ x_3$

$y = y_0\ y_1\ y_2\ y_3$ & c_{in}

o/p : $z = z_0\ z_1\ z_2\ z_3$ & c_0

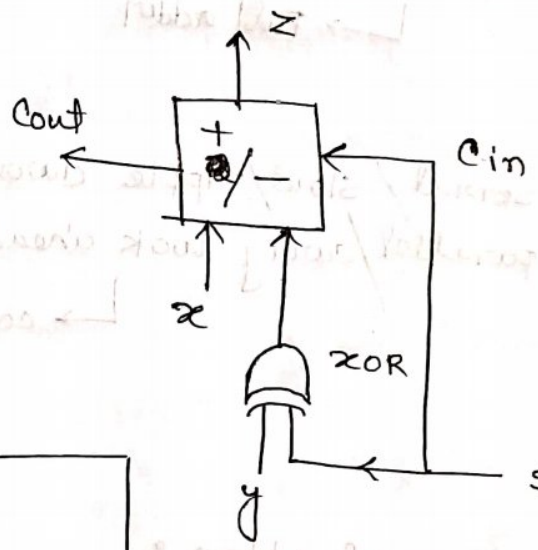
Subject:
Date: Time:

Subtractor :

Substractor এর ক্ষেত্রের carry mainly adder এর carry. carry লাগে না।

Adder দিয়েই subtract এর কাজ করা হয়,

4/16 bit এর জন্য আসলে এই চিত্র একেই just describe করে দিব ⇒



$x - y$

$= x + (2\text{'s complement of } y)$

$= x + (1\text{'s complement of } y + 1)$

XOR		
0	0	0
0	1	1
1	0	1
1	1	0

Substractor এর জন্য আলাদা কোনো circuit নেই। Adder দিয়েই করি।

* S = 1 হলে substraction হবে। কিভাবে?
* S = 0 হলে Addition হবে।

কোনো num কে 0 এর সাথে XOR করলে ঐ num ই পাব আর 1 এর সাথে XOR করলে inverse পাব।

" Week 2 Done "

Mon
Subject: 28.03.2022
Date: Lec - (6) Time

Q: Half Adder থেকে কেন Full adder এ switch করি।

Q: x number কে Half adder করতে পারবে? না পারলে— কী করণীয়।

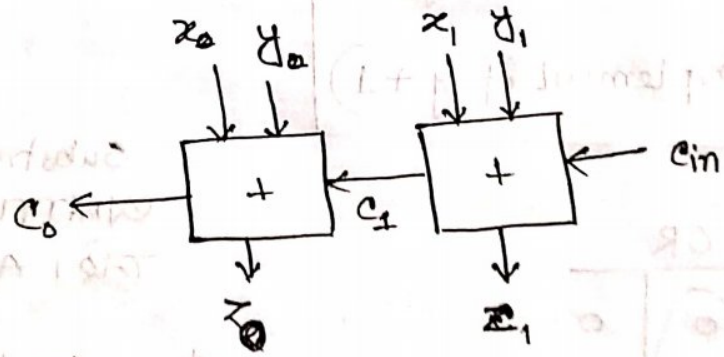
↳ Full adder

serial / slow / ripple carry

parallel / carry look ahead

↳ concurrently process করতে পারে।

2 bit serial adder :



~~i/p : x_1, y_1, c_{in}, x_0, y_0~~

~~o/p : z_0, z_1~~

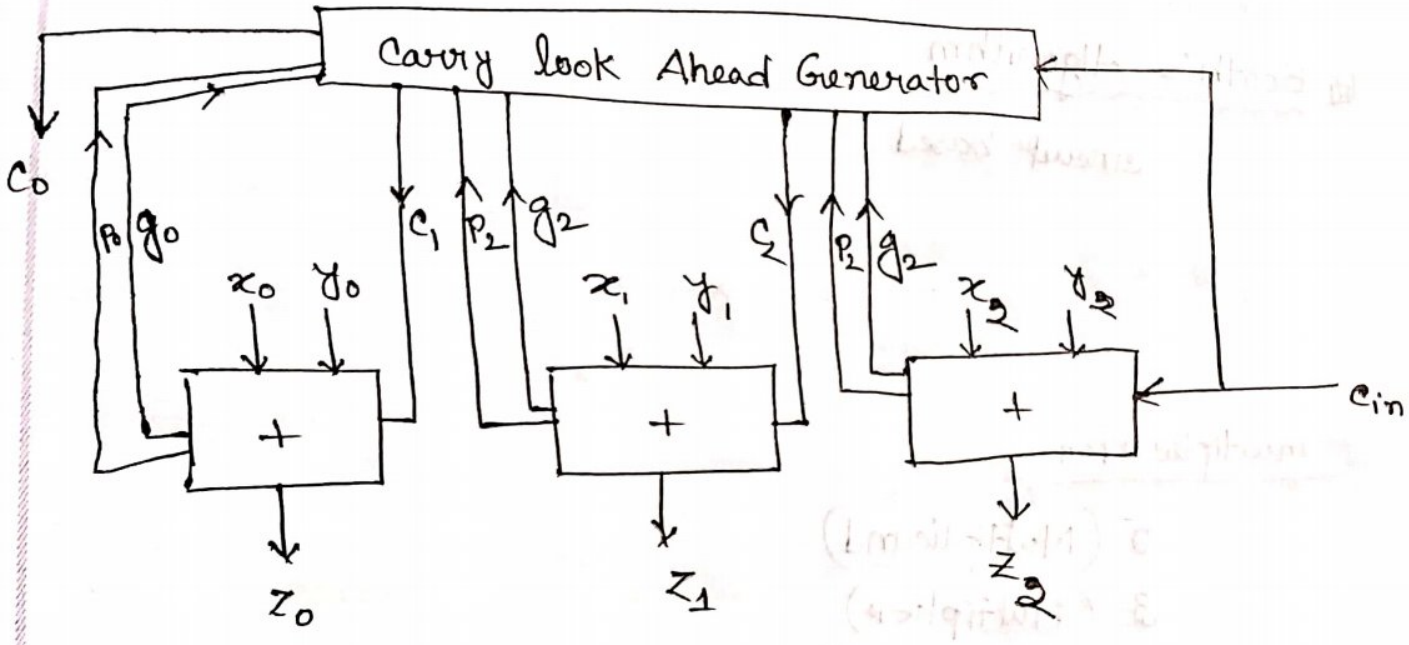
i/p : $x = x_0 \; x_1$ & c_{in}

$y = y_0 \; y_1$

o/p : $z = z_0 \; z_1$ & c_0

Subject:
Date: Time:

3 bit Parallal adder :



"খুব important"

i/p : $x = x_0 \quad x_1 \quad x_2$

$y = y_0 \quad y_1 \quad y_2$ & C_{in}

o/p : $z = z_0 \quad z_1 \quad z_2$ & C_0

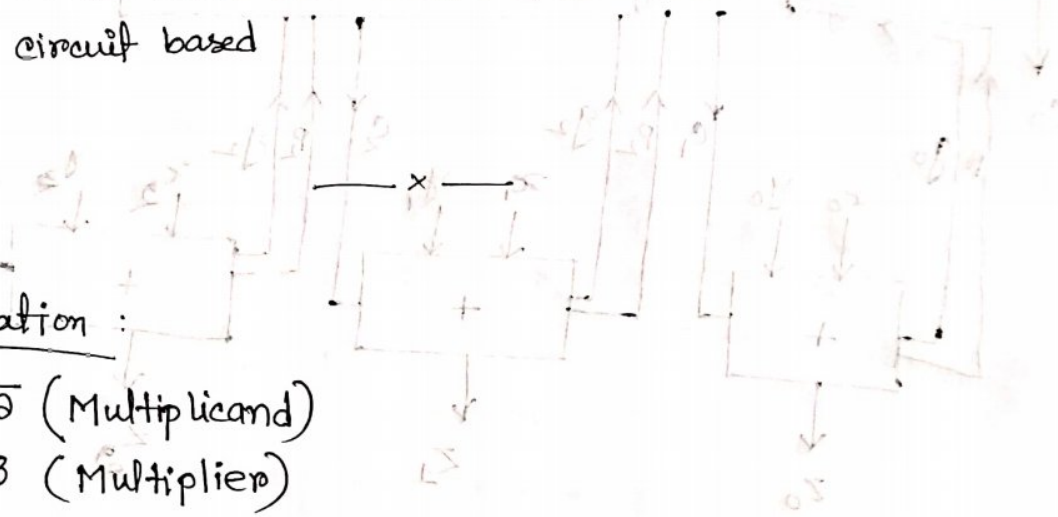
MSB থেকে operation start হবে

30.03.2022
Subject: Wed
Date: Lec-(7) Time:

Multiplication :

Booth's Algorithm

circuit based



multiplication :

5 (Multiplicand)
3 (Multiplier)

15

255 × 255 ← 255 কে 255 বার যোগ করা which is step consuming.
এই নিয়মকে বলে repeated addition.
এটা time consuming

To solve this, Basic Multiplication :

5	0101
3	0011
15	0101
	0101x
	0000xx
	0000xxx
	00001111 (15)

partial product

একেত্রে 4 step এই হয়ে যায়, অনেক বড় নম্বর এর ক্ষেত্রে এটা calculation easy.
*** এখানে mainly addition হচ্ছে ॥

Subject
Date Time

$2^8 \longrightarrow$ 8 steps

$2^4 \longrightarrow$ 4 steps

$2^n \longrightarrow$ n steps .

Addition + Shifting + Addition + . . . :

* steps of Basic Multiplication.

partial product যোগ করার মাধ্যমেই আমরা কাঙ্ক্ষিত result পাই।

Steps

1. Num of steps = bit of multiplier
2. Check the bit of multiplier from LSB
3. Multiplier = 1 $\longrightarrow$ Multiplicand Itself
   Multiplier = 0 $\longrightarrow$ 0
4. Add all partial products.

Register required :

We don't store partial product in register.

Q: $2^3 \times 2^5$ গুণের জন্য accumulator size কত হবে?

$\longrightarrow$ $2^3 \times 2^5 \Rightarrow 5+3 = 8$

store করা

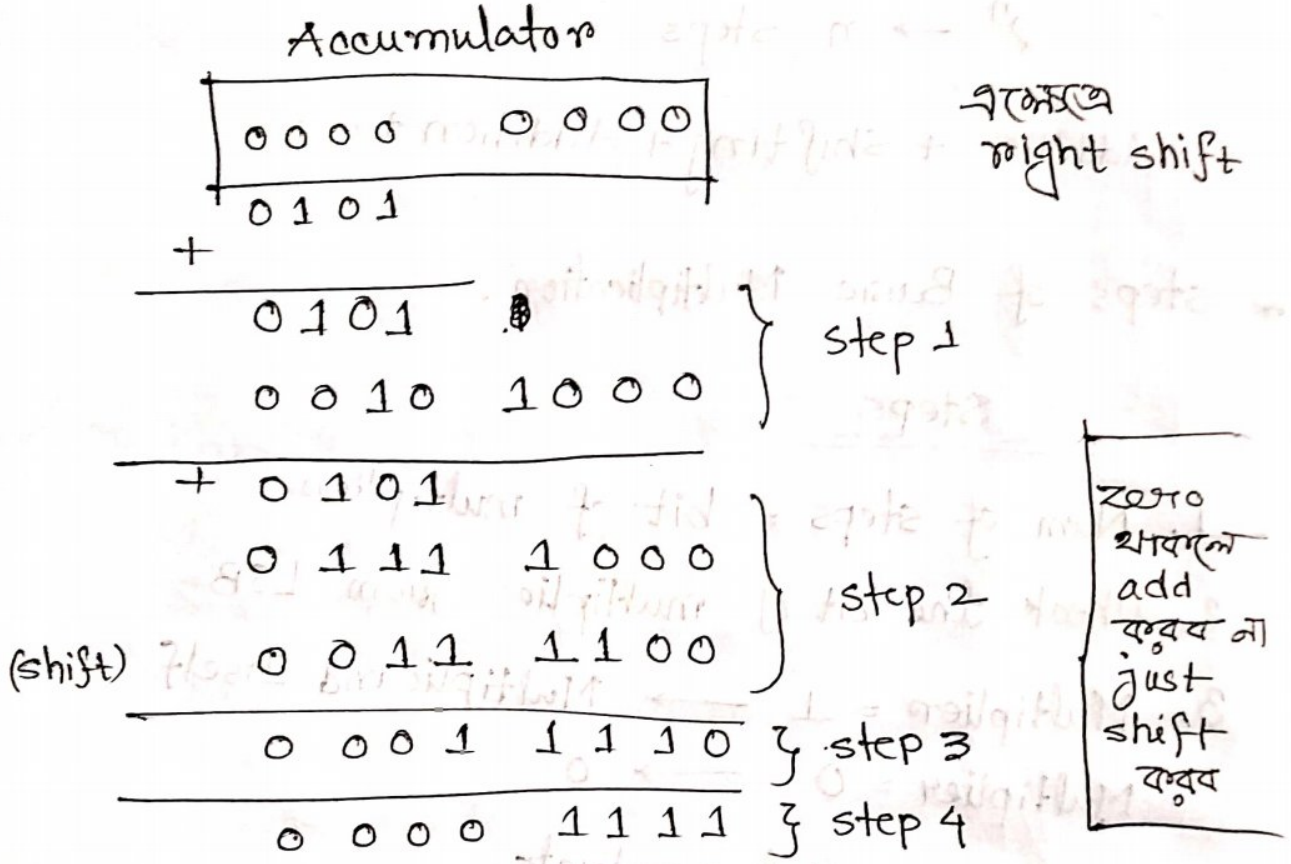
Accumulator

0000	0000

$\longrightarrow$ 8 bit

Sweet Bird

প্রতিটা step এ store করে add করি, এই storing procedure এর জন্য একে বলে accumulator.



Q:

Reg 1টি → 4 bit → Addition

Reg 1টি → 4 bit → Multiplication

" 1টি → 8 bit → Accumulator

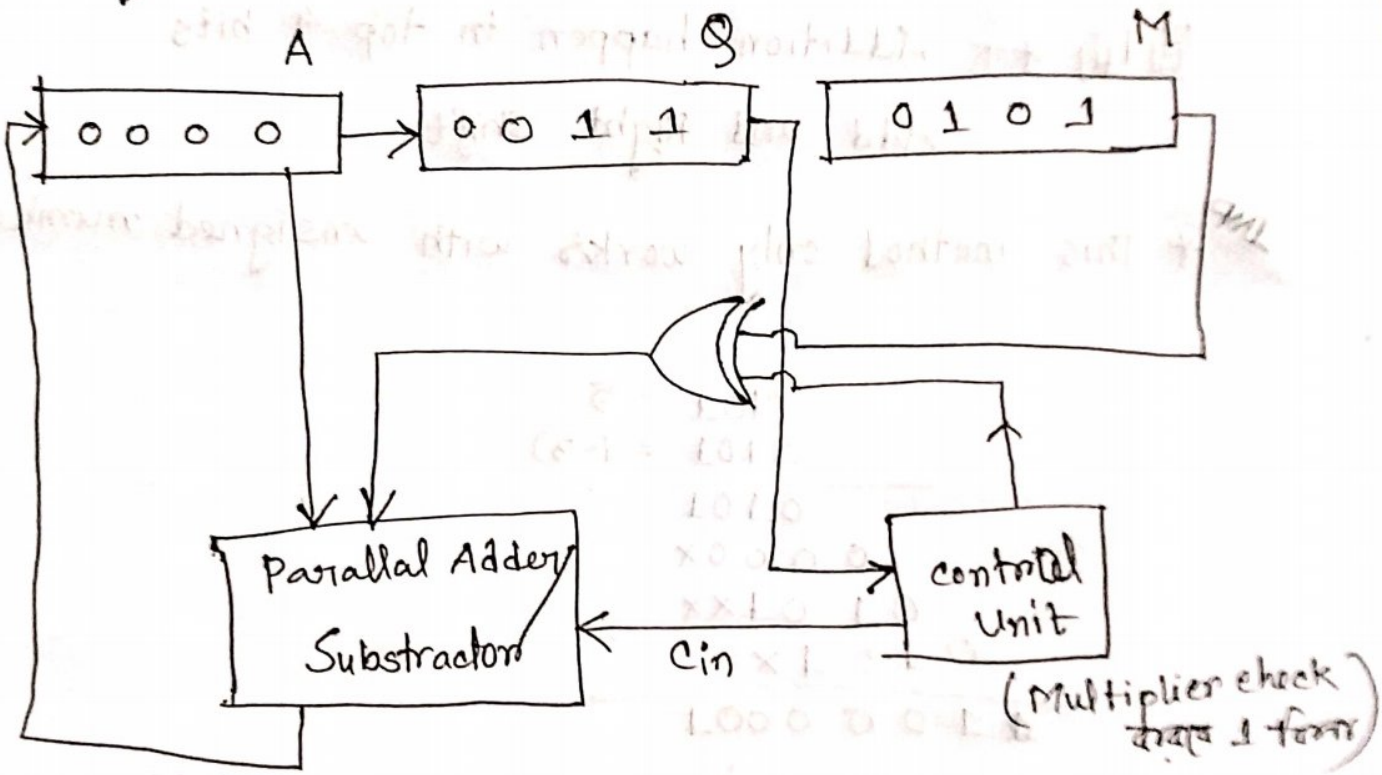
Subject
Date: Time:

Circuit

4bit + 4bit = 8 bit ← last step → A & Q মিলে result পাই

এখানে 8 bit না নিয়ে 4 bit কেন নেই?

Multiplicant → M
Multiplier → Q
Accumulator → A



A	Q	M
0000	0011	0101
0101		
0010	1001	
0111		
0011	1100	
0001	1110	
0000	1111	

5 = 0101

3 = 0011

Accumulator Calculation

** Addition happen in top 4 bits

Add and Right Shift

IMP * this method only works with unsigned number.

```
  0101 = 5
  1101 = (-3)
 ------
  0101
  0000x
  0101xx
  0101x
 ---------
 1 0 0 0 0001
```

→ (-63) আসে যা ভুল

negative number এর partial product ও negative আসে। কিন্তু আমরা partial product এ যোগ করতাম। এখানে mainly one use করা হয় (এর আগে থেকে partial product এর MSB কত। যদি 1 হয় তখন shifting এর time এ 1 বাদ দেয়)

Q: Why fail?

Subject
Date: Time:

: Why fail ?

→ When you multiply neg number, your partial product will be negative.

negative numbers has infinite leading 1.

Subject: Lec - (8)

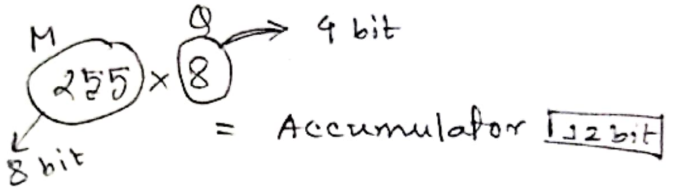
Date: 14.05.22 Sat Time:

After Eid-ul-Fitre Vacation.

5 → Multiplicant (M)

3 → Multiplier (Q)

```
    0101   = 5
    0011   = 3
   ------
    0101
   0101x
  0000xx
 0000x
--------
 00001111  = Accumulator
```



[1] Step's of Booth's Algorithm:

1. Number of steps = Number of bits in multiplier
2. At each step, examine two consecutive bits of multiplier from LSB.

*. If transition 0 → 1 (Substract M)
1 → 0 (Add M)

প্রতিটা step এ Right Shift হবে।

Sweet Bird

Subject.....

Date:.....Time:.....

Booth's Algorithm

(5×7)

+5 : 0 1 0 1 　　7 : 0 1 1 1

-5 : ~~1 1 0 1~~ 1011 　　-7 : 1 0 0 1

	A (0) Accumulator	Q(7) Multiplier	Q_{-1}	M(5) Multiplicand
	0 0 0 0	0 1 1 1		0 1 0 1
0 to 1 SubM RS	1 0 1 1 1 1 0 1	1 0 1 1	1	
1 to 1 RS	1 1 1 0	1 1 0 1	1	
1 to 1 RS	1 1 1 1	0 1 1 0	1	
1 to 0 AddM RS	+ 0 1 0 1 0 1 0 0 0 0 1 0	0 1 1 0 0 0 1 1		

right shift করব 1 হবে।

carry ignore করি। Why?

কয়টা step হবে তা depend করে Multiplier কত bit এর।

যেহেতু zero দিয়ে শুরু করে এটা positive Number

Subject: Lec -⑨

Date: 16.05.22 Mon Time:

5 × -7 = ?

+5 = 0101 +7 = 0111

-5 = 1011 -7 = 1001

steps	A(0)	Q(-7)	Q(-1	M(5)
	0000	1001	◌	0101
0 to 1	1011			
Sub M	1011	1001		
RS	1101	1100	1	
1 to 0	0101			
Add M	0010	1100	1	
RS	0001	0110	0	
0 to 0				
RS	0000	1011	0	
0 to 1	1011			
Sub M	1011	1011		
RS	1101	1101	1	

Answer

~~Accumulator~~ : 1101 1101 = -35 → (Ans)

Ans কে 2's compliment করে ans দেখাবে।

0010 0011 = 35

Ques: $14 \times 7 = ?$ by Booth's algo.

5 bit 4 bit

Steps = num. of bits in Multiplier

অর্থাৎ, Multiplier 7 কে বলে 5 bit এ represent করবি।

So, step 5 টা।

Example: $(-5 \times 7) = ?$

Booth's Devision → Restoring
→ Non-Storing

Dividend

Divisor ← 6) 25 (4 → Quotient
24
---
1 → Remainder

We don't use repeated subtraction, as ~~show~~ slow.

(Left shift)

Exam Important MUST ⟹ Booth's Algo

Sweet Bird

STORING

When step successful or unsuccessful.

$$4 \;\Big|\; 3\ 2 \;\Big|\; 0$$

$$4 \;\Big)\; -1$$

$$+\ 4$$

$$3$$

LS

Sub → +ve, S → Q = 1

" → -ve, US, Q = 0

Time loss হয়, ৫২ store করে & আবার কোন case এই unsuccessfull হয়।

Steps of Booth's Division ReStoring :

1) ~~Step = bit num of Dividend~~

Number of steps = Num. of bits in Dividend

ii) At each step, Left Shift the Dividend and then Subtract the Divisor.

$$4 \;\Big|\; 6\ 4 \;\Big|\; 1$$

$$4 \;\Big)\; 4$$

$$2$$

If, R = (+ve), Successful : Quotient = 1

R = (-ve), Unsuccessful : Quotient = 0

Restoration required, by adding back the divisor.

Subject..

Date:................................Time:..............

Microprocessor add back কেন করে?
→ ∵ overwrite করে তাই add back করে restore করা। এতে একটু time বেশি লাগে।

ex : 7 ÷ 3 = ?

+7 = 0111 +3 = 0011

−7 = 1001 −3 = 1101

	Accumulator A (0)	Dividend Q (7)	Divisor M (3)
	0000	0111	0011
LS	0000	111_	
Sub (Divisor) M	1101		
	1101	1110 (unsuccessful)	
∵ MSB = 1, −ve unsuccessful			
Restoration	0000	1110	
LS	0001	110_	
Sub M	1101		
	1110		
MSB = 1, −ve unsuccess → Restore	0001	1100	
LS	0011	100_	
Sub M	1101		
	0000		
MSB = 0, +ve Successful	0000	1001	
LS	0001	001_	
Sub M	1101	001_	
	1110		
MSB = 1, −ve unsu Restore	0001 → ভাগশেষ R	0010 → ভাগফল Q	

LS করার পর যেটা পাইছি তা দিবে।

Wed

Subject: Lec - 10

Date: 18.05.2022 Time:

Week-4

Re-storing division :

By the time procedure is over, the dividend will be empty, at every step, we start filling the quotient over here at last, the reg itself will be quotient.

By the time procedure is over, → Sum is completely out, if anything remains there it will be remainder.

$$\begin{array}{r|r|l} 5 & 6 & 1 \\ & 5 & \\ \hline & 1 & \end{array}$$

Q: $6 \div 2 = ?$

Ans:

Accumulator : 0000 → Remainder

Dividend : 0011 → Quotient

P.T.O →

Sweet Bird

Subject.....

Date:.....Time:.....

(*) $6 \div 2 = ?$

$+6 = 0110$ $\quad +2 : 0010$

$-6 = 1010$ $\quad -2 : 1110$

	Accumulator A(0)	Dividend Q(6)	

Sweet Bird

Non Restoring Devison:

1. Num of ~~bit~~ steps = num. of bits in dévidend
2. Left Shift & subtract M (divisor).
3. SubM: +ve, Success, Q = 1
   -ve, Unsuccessful, Q = 0, Next step, Add M
4. for last step,
   -ve, Unsuccessful, Q = 0, Restore

ex: $7 \div 3 = ?$

7 : 0111 3 : 0011

-7 : 1001 -3 : 1100

Accumulator A(0)	Dividend : 7 Q (~~0111~~)	Divison (3) (M	
0000	0111	0011	
0000 1101 1101	111_ 1110		LS Sub M MSB = -ve, Unsuccessful Next Add M
1011 0011 1110	~~1011~~ 111_ ~~0011~~ 1100		LS Add M MSB = -ve, Unsuccessful Next Add M
1110 0011 0000	100_ . 1001		LS Add M MSB = -ve, Successful

	A	Q
LS	0001	~~000~~
Sub M	1101	001_
	1110	
MSB = -ve, unsuccessful Restore	0001	0010
	1	2
	Remainder	Quotient

এগুলো সব unsigned. Signed হলে বা ঐ division এর ক্ষেত্রে sign number এর algorithm different.

// এখন না পড়ে পড়াবে //

এই process is only for unsigned number.

4] Non-Restoring :

$6 \div 2 = ?$

$+6 = 0110$ $\quad +2 = 0010$

$-6 = 1010$ $\quad -2 = 1110$

Mon

Subject..... Lec-(12)

Date:..23.05.22..Time:.....

Algorithm for adding floating Point Number

$010110 \longrightarrow 1.01 \times 2^{(2)E}$

M

S	M	E

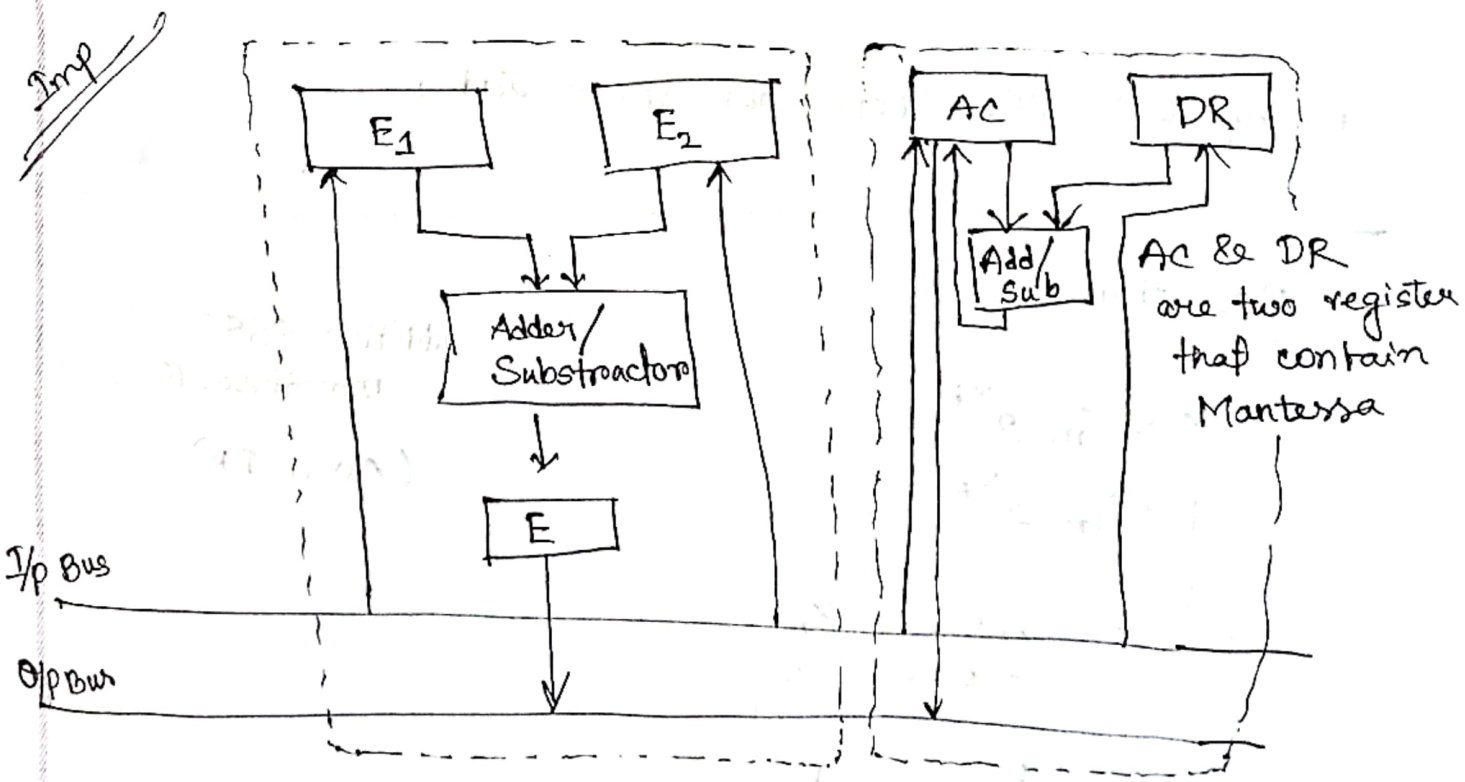
S	M	E

$$\begin{array}{r} 101 \\ 010 \\ \hline 111 \end{array}$$

$$(-1)^S \times 1.M \times 2^E$$

$$\begin{array}{r} 4 \times 10^2 \\ 42 \times 10^4 \\ \hline 8.2 \times 10 \end{array}$$

Imp



Subject.....

Date:.....Time:.....

Algorithm : Decimal

4×10^3 2×10^4	4×10^4 2×10^4	4×10^3 20×10^3
	2.4×10^4	24×10^3
	Normalized form (better)	

* Always get the lower one high. ??

ALU → Exponent unit
ALU → Mantissa unit

* exponent যেটাতে ছোট সেটা lower বার।

Algorithm :

addition করার আগে mantissa তে;
(AC & DR)

$X = X_m\ 2^{XE}$

$Y = Y_m\ 2^{YE}$

Input : $AC \leftarrow X_m$

$DR \leftarrow Y_m$

$E_1 \leftarrow XE$

$E_2 \leftarrow YE$

$AC.\text{overflow} \leftarrow 0$

$\text{Error} \leftarrow 0$

Sweet Bird

Subject.....

Date:.....Time:.....

Compare : $E \leftarrow E_1 - E_2$

Equalize : If $(E < 0)$ then

{ $AC \leftarrow RS(AC)$

$E \leftarrow E + 1$

go to equalize

}

if $(E > 0)$ then

{ $DR \leftarrow RS(DR)$

$E \leftarrow E - 1$

go to equalize

}

Add :

$AC \leftarrow AC + DR$

$E \leftarrow \max(E_1, E_2)$

Overflow : If (AC overflow = 1) then

{ if ($E = E_{max}$) then goto Error

$AC \leftarrow RS(AC)$

$E \leftarrow E + 1$

go to End

}

RS = Right shift

↳ We store the position of the point.

9×10^9

2×10^5

4×10^5 $E = 2$

$E = -1$

Defination of Ac overflow :

It is a flag that will automatically set by ALU as 1 if there are more than one digits to the left of the point.

Sweet Bird

Input
comp
equalize
Add

overflow
normalize
zero
underflow

E can only become E+1, if there is room for E to become E+1.

Normalize : (If AC normalized) then
go to End

Zero : If (AC=0), then E ← 1 → Error flag

Result = 0

Underflow : E > Emin
if ( ) then
{ AC ← L.S (AC)
E ← E-1
go to Normalise
}

Error : Error ← 1

End : End of process

Normalize form

1.01

E = Error flag.
We can't represent 0 in normalized form as in this form there should be at least one point to the left of the point.
↳ decimal

0.2×10^3
2.0×10^4
numbers left shift 2 ∴

(Imp Topic)

Subject.....

Date:.....Time:.....

Q: 0.00062×10

5.4×10^5

floating point number ৭ add =? & print in every steps.

$X = 6.2 \times 10^3$

$Y = 5.4 \times 10^5$

I/p : AC → 6.2 $E_1 \rightarrow 3$

DR → 5.4 $E_2 \rightarrow 5$

AC overflow ← 0

Error ← 0

Compare : $-2 \leftarrow 3-5$

Equalize : $(-2 < 0)$

$0.62 \leftarrow RS(6.2)$

$E = -1$

$0.062 \leftarrow R_s(0.62)$

$E = 0$

Add : ~~AC+DR~~ → AC ← AC+DR

~~00062~~ ~~5.4~~ 0.062

$+ 5.4$

5.462

$E \leftarrow 5$

Normalize :

End

Result : 5.462×10^5

Sweet Bird

Wed
Subject: Lec-(13)
Date: 25.05.22 Time:

CT-1

Microprocessor এর কাজ

2's C
floating point (Representation, ...) Normalization
floating point Algorithm

(Lec ⇒ 1-4, 12)

Q: What are the error conditions?

Q: Write the O/p of the algo fo the addition of two floating points _&_ Justify your answer. → [more marks]

Q: Write the only necessary parts of the algo for adding the given ~~flat~~ floating point number & explain your ans.

Q: Discuss the overflow/_/_ happen with example.
এমন দুইটা point বের করা যা overflow/zero etc দিবে।

Overflow এর জন্য
Emax লাগবে।

কিছু না দেওয়া
থাকলে ধরে নিব,

I/P: _ _
_ _
print ( )

compare:
_ _ _
_ _ _
print ( )

equalize:
_ _
_ _

Subject.....
Date:.....Time:.....

zero কে floating point এ represent করা যাচ্ছে না তাই, zero আসলে error flag on হবে। অর্থাৎ E=1 হবে।

#

Ans :

O/p :

AB
CD

4×10^2
2×10^3

where the print function is?

Now Justify
→ calculation দেখাতে হবে ॥

Example – 1 :

$4 \times 10^{(2)}$ E_1
$2 \times 10^{(3)}$ E_2

$E = E_1 - E_2 = 2 - 3 = -1$

print i/p
" compare
" equalize
" Add
End

E < 0 {
}

→

i/p
Compare
equalize
Add
End

Sweet Bird

i/p :

print ("i/p")

compare :

print ("compare")

Equalise :

if (E<0) {
print ("E1") ;
go to Equalise
}

(E>0)

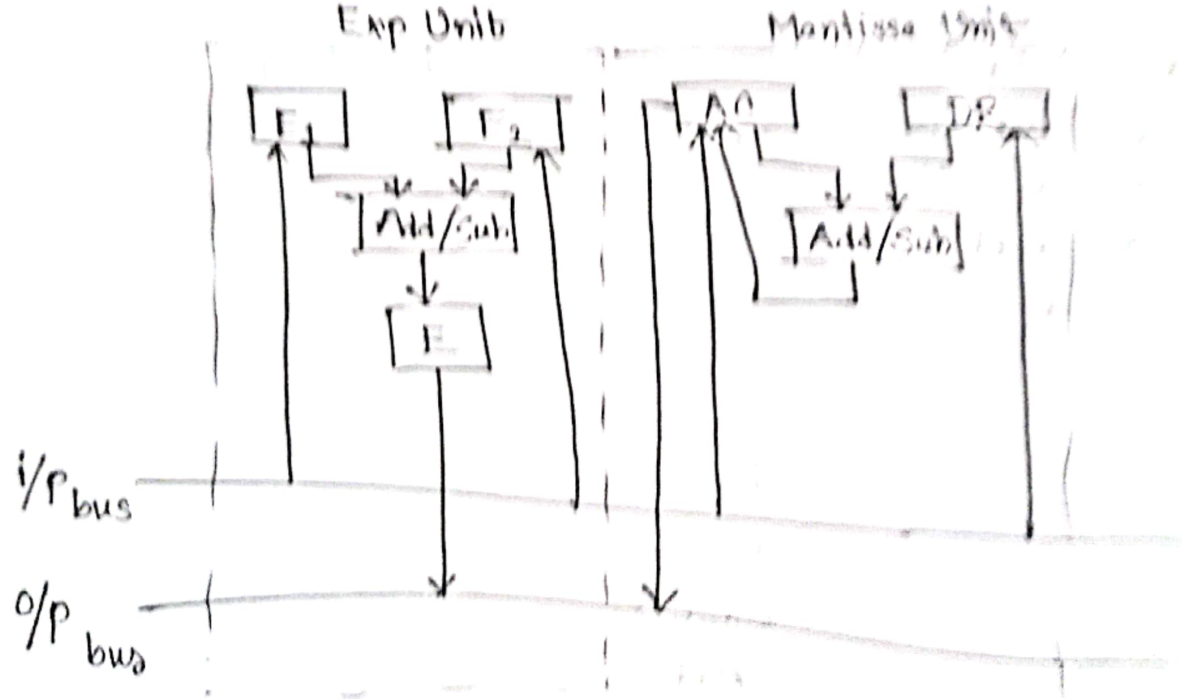
{ print ("E2")
go to equalise }

Arithmetic

i/p

compare

E



আগের be গুলো execution unit এর কাজ ছিল।

Subject:
Date: Time:

COA CU Instruction Pipelining

one of the reason why processor works fast.

COA → Computer Organization Architechture

CU → Control Unit (Processor এর একটা unit)

↳ Fetch & Decode

Q: Why processor becomes so fast?

→ Increase data bus size (Now 64 bit is used)

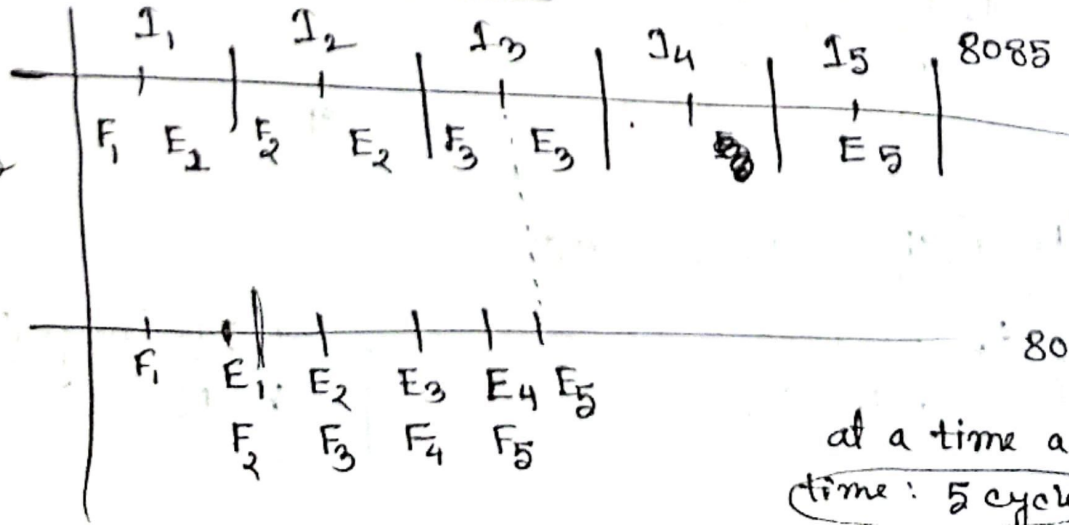
→ Increase clock frequency / Triggger

→ ~~Pipeling~~ Pipelining

8085 → Non pipelined

8086 → Pipelined

2 stage Pipelining:



figure

Decode খুবই কম time নেয় so লিখি নি।

time: 5 × 2 = 10 cycle

at a time a দুইটা কাজ হচ্ছে।

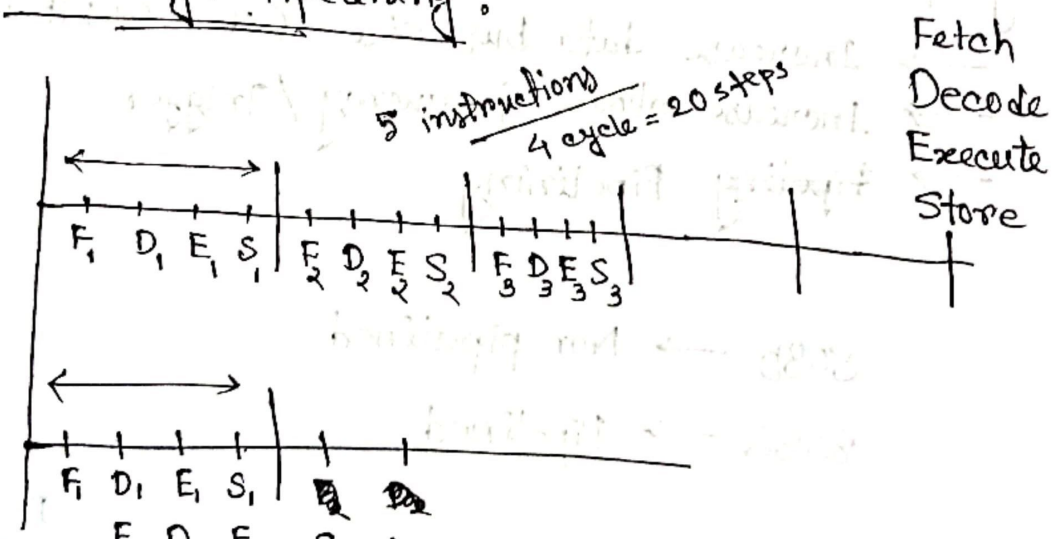
time: 5 cycle

Week-6
Subject Lec-(14)
Date 28.05.22 Time
Sat

As we are fatching two main process of an instructions & working on the two instructions at the same time, so it becomes fast, this process is called pipelining.

Q: figure of pipelining and non-pipelining. (exam imp)

4 stage Pipelining:



Pipelined

$K + (N-1) \times 1$ → Pipelining eqn

number of Stage

number of instructions

Non pipelining eqn

$K \times N$

Sweet Bird

Subject.....
Date:.....Time:.....

Advantage & Disadvantage of 4-stage

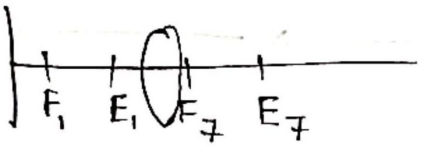
↓ fast

↓
1) Control Hazard (Broach Prediction)
2) Data Dependency
3) Structural hazard

* Control Hazard :

branching

A spam or empty spot was created which is called Pipelin Bubble.



এখানে pipeline failed

By Branch prediction algorithm it was overcome. It 95% accuracy

If stages increased তে discard increases.

* Data Dependency :

~~Destin~~ Destination of first instruction should not be source of second instruction.

```
INC  B
MOV  A, B
```

Before, B = 20
now, B = 21

↳ Increment করার আগেই নিয়ে নিলে prb হবে।
এখানে, B source হয়ে গেছে so prb হয়ে গেছে।

Sweet Bird

Subject.....

Date:.....Time:.....

Solution :

```
INC  B
NOP
MOV  A, B
```

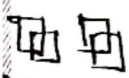
NOP → No operation

একটা,
fetch, execution
operation এ prob
হয় না।

* Structural Hazard :

```
ADD  A, B
ADD  A, [2000]
```

to solve this structural hazard



---

1. RISC Processor
2. PIC "

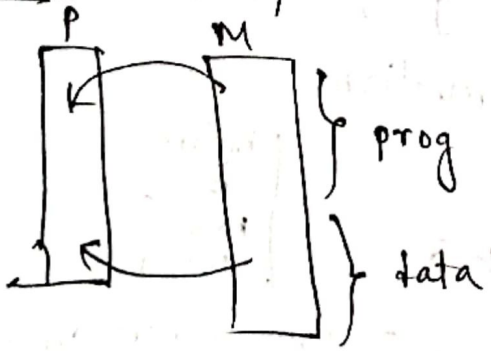
] → full form

Characteristics :

RISC has more register. It lessen the dependency on memory.

PIC → Peripheral Inter Controller

Before

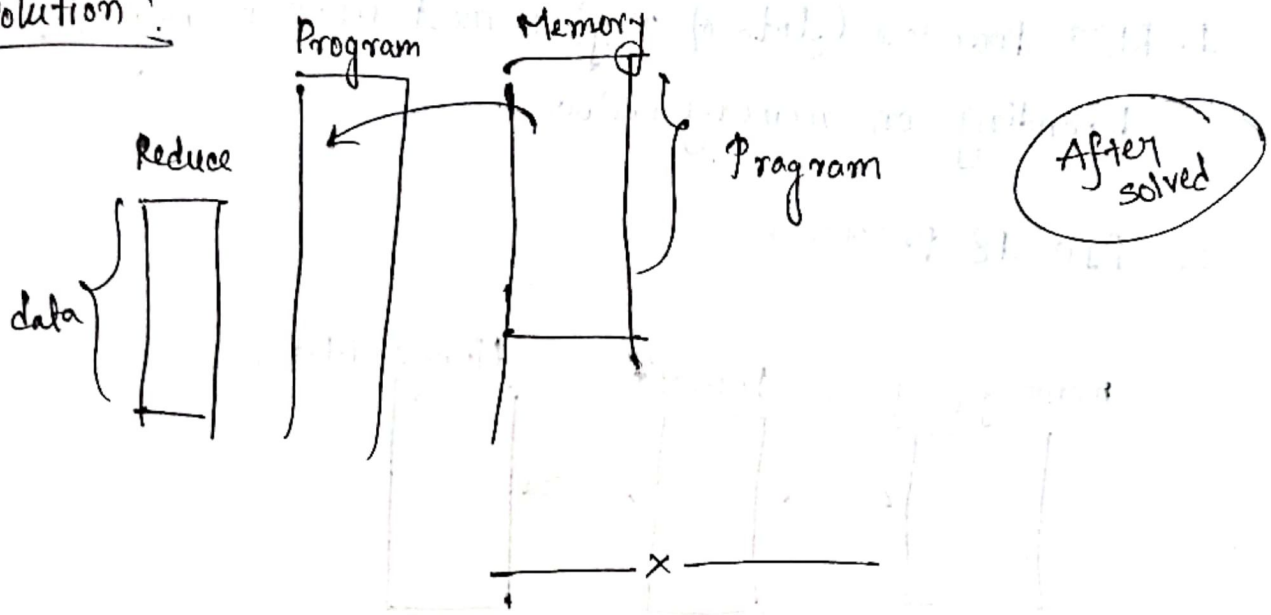


Sweet Bird

Subject.....

Date:.....Time:.....

Solution :



Overlapping of fetching & execution can only happen, if execution doesn't require memory.

Pipelining is imp as it helped to increased the speed of Microprocessor.

Sweet Bird

Mon

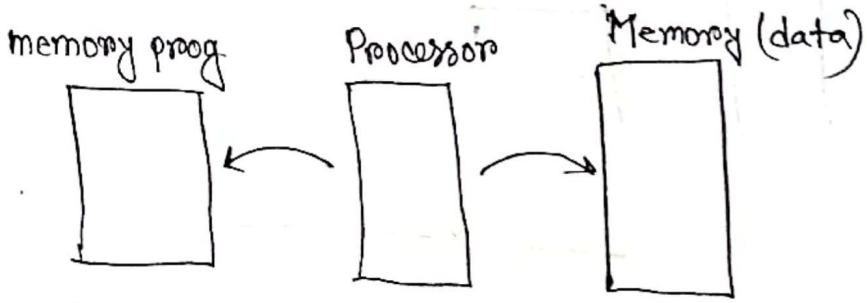
Subject: Lec-15

Date: W-6 Time:

30.05.22

1. RISC Processor (lots of reg) so most work in register, depending on memory reduce.

2. PIC 18 processor



two separated memory for program & data while executing if we need data use other memory, so fetching and executing never clash with each other.

Addressing Mode

Addressing mode is the manner by which an operand is given instruction is called--

operand: the number on which the operation is done is called operand.

Add, AL, 25H

operands

Operation → SUM, MOV, ADD

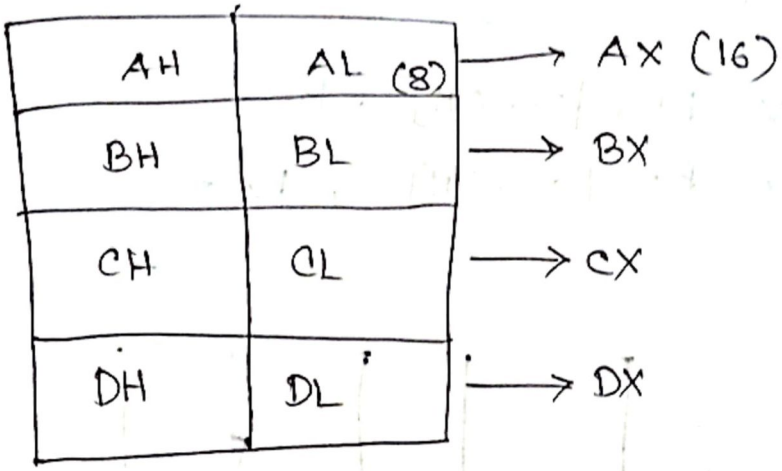
- Data from inidiate
  Add AL, 25 H
- Register
  Add AL, BL
- Indirect/memory location
  Add AL,

Sweet Bird

Subject.....

Date:.....Time:.....

General purpose register :



25 H → 0010 0101

1235 H → 0001 0011 0011 0101

① Immediate Addressing Mode :

Data is given in instruction

[To initialize]

```
MOV  BL, 25H ; BL ← 25H
MOV  BL, 2000H ; BX ← 2000H
```

যেখানে instruction এর সাথেই data লেখা থাকে so it can go to immediate operation.

② Registers Addressing Mode :

Data is stored in Reg.

```
MOV BL, CL ; BL ← CL
MOV BX, CX ; BX ← CX
```

processor এর ভিতর কাজ হচ্ছে ॥ It's smaller & faster.

Q: কোন addressing mode কখন use করবে?

Data নিয়ে কাজ করছে

Sweet Bird

Subject.....

Date:.....Time:.....

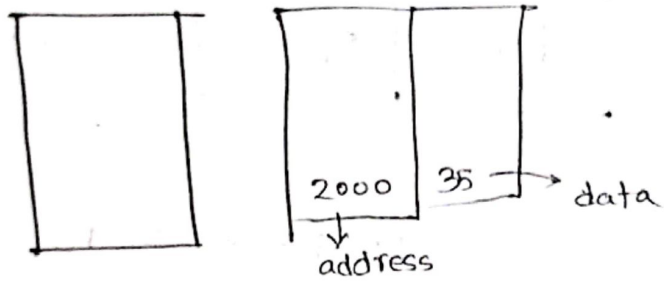
(III) Direct Addressing Memory:

Address is given in instruction

```
MOV  BL, [2000H] ; BL ←
MOV  BX, [2000H] ; BX ←
```

It's an address so 16 bit. though the Reg is 8 bit.

(BL gets the content of 2000H)



don't

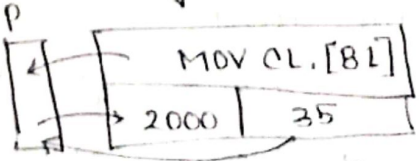
* When we know the data, we use this mode.

(IV) Indirect Addressing MODE: (BEST BEST)

Address is given in the register.

1) Register Indirect: addr in register

```
MOV CL, [BL]
```



BL এর ভিতর add.

23

best way because it's easier to increament & decrement.

Sweet Bird

Subject.....

Date:.....Time:.....

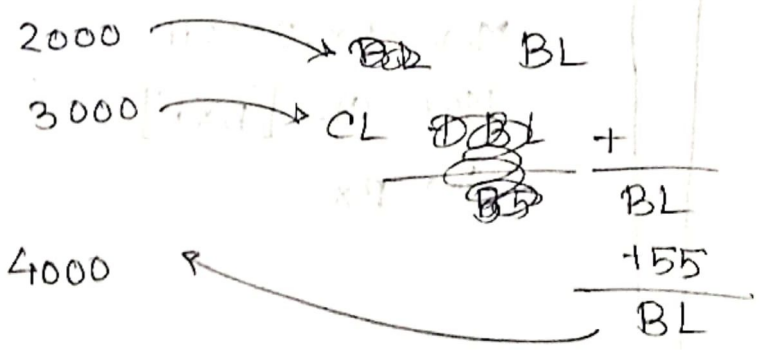
* ADD 25 + 35

```
MOV AL, 25H
MOV BL, 35H
ADD AL, BL   ← Reg A.M
```

* [2000H], [3000H]

```
MOV AL, [2000H]
MOV BL, [3000H]
Add AL, BL        → Direct Add. M
```

*



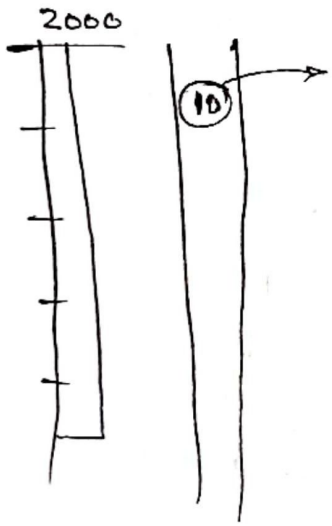

```
MOV BL, [2000H]
MOV CL, [3000H]
ADD BL, CL
ADD BL, 55H
MOV [4000H], BL
```

→ It's also a direct addressing mode as address is given in the ins.

* Operation Destination, Source
  ↓
  MOV AX, BX

Indirect Addressing Mode :

immediate : initialize
register : move between reg
direct : need to access few memory location




```
MOV CL, [2000H]
MOV BX, 2000H
MOV CL, [BXH]
INC BX
```

Give next 100 data from 2000 → Indirect
Give the data of 2000 memory location → Direct

Q: get data from location 2000 in CL.

Q: get data of hundred location starting from 2000.

Sweet Bird

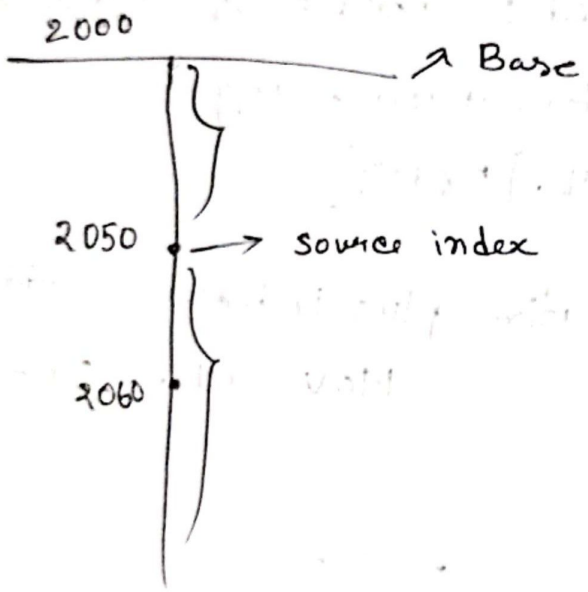
Subject.....

Date:..... Time.....

Base indexed :

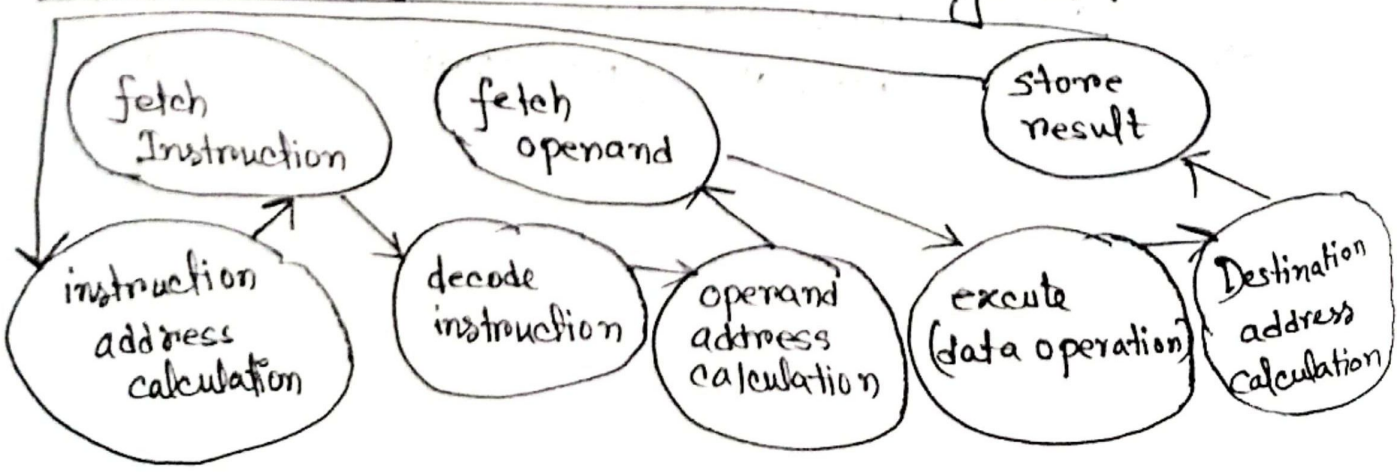
i) one register act as a base

ii) other reg act an an index



Indirect Addressing Mode :

Instruction cycle State: transition diagram :



Sweet Bird

```
int main ( )
{ cin >> a >> b;
  c = a+b
  cout << " done"; }
```

a = 2
b = 3

O/P :

done.

—x—

* program → set of instruction
* To execute these program, tiny operations are need that are called Micro operations.
(each instructions requires tiny tiny operations)
* Control unit control signal generate করে।

Q: এরে micro operation কেন বলে?

→ আগে clock speed micro sec এ ছিল, তখন থেকে এই নাম চলে আসছে। এখন technology improve হয়েছে। Now it's calculated in nanosec.

Sweet Bird

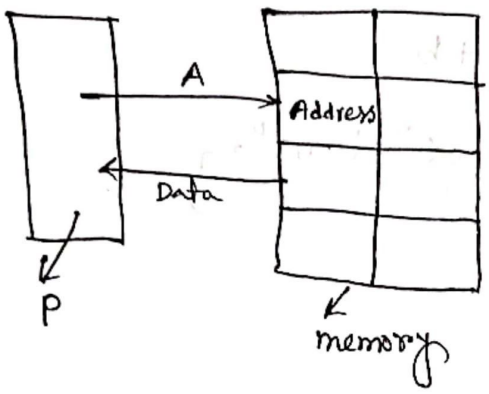
Subject.....

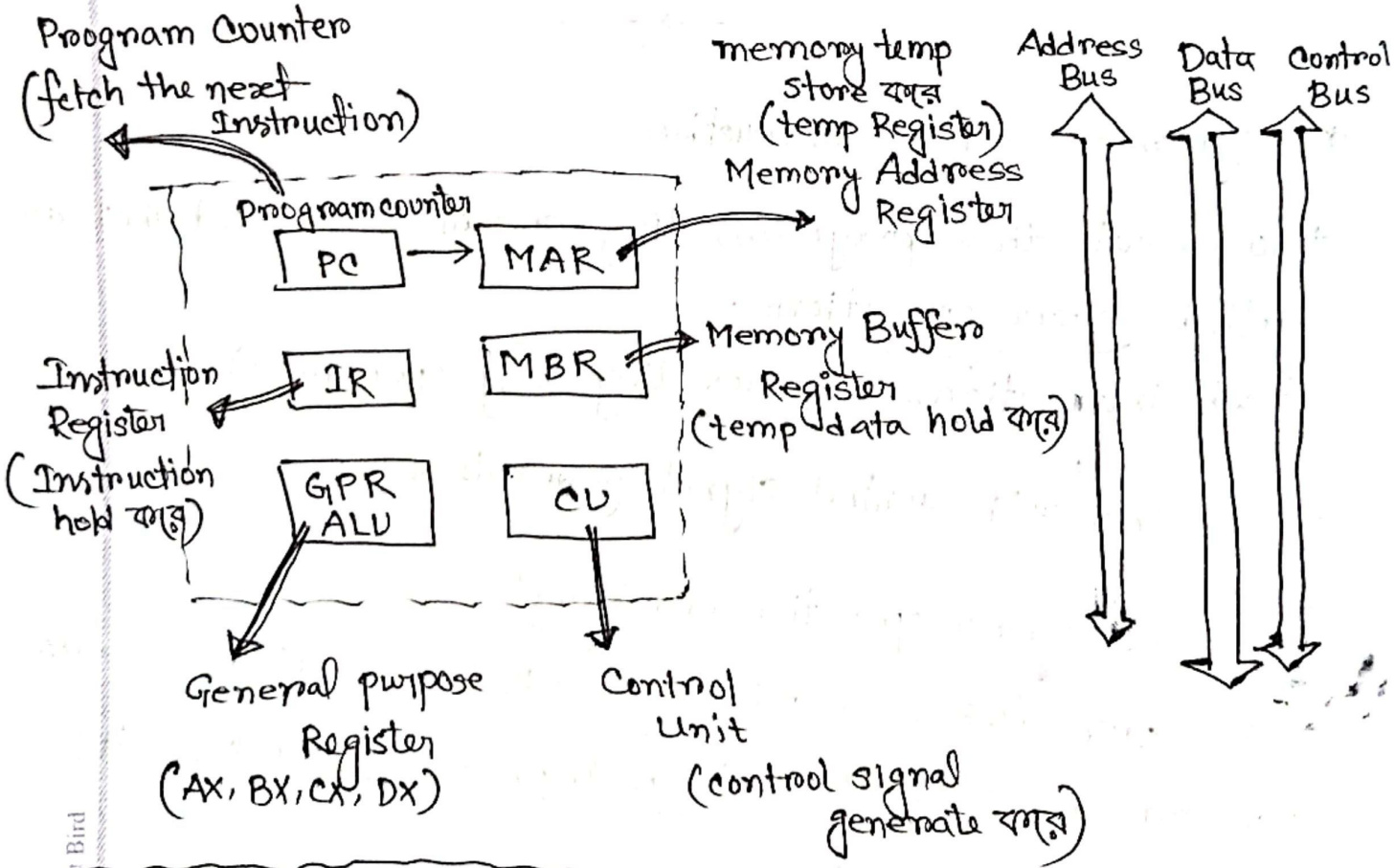
Date:.....Time:.....

Sytem Bus ⟶ Address Bus

⟶ Data Bus

⟶ Control Bus

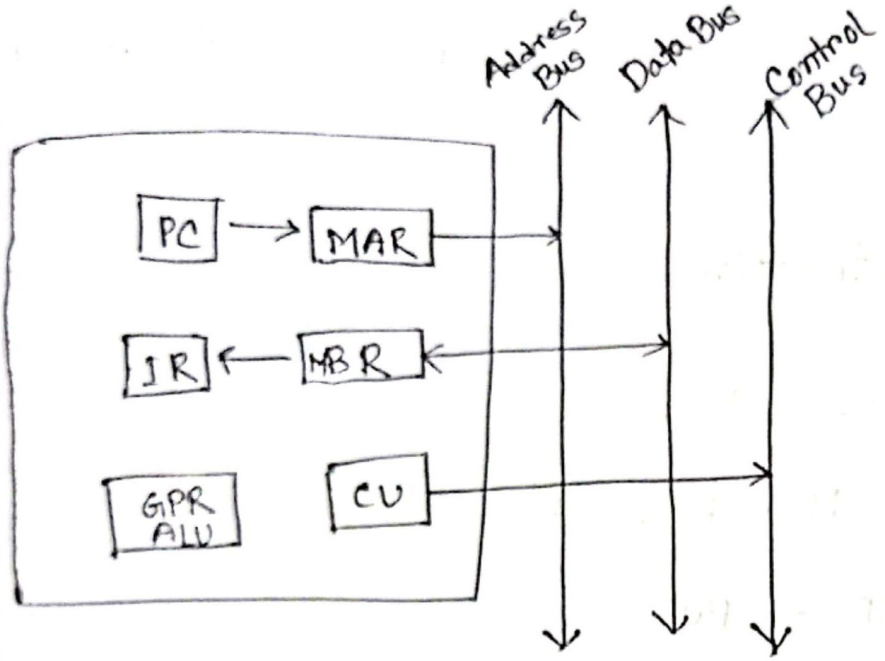




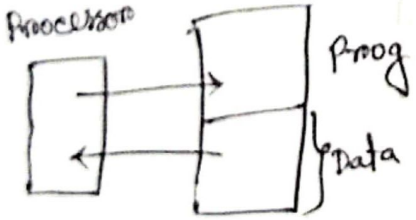
Imp Topic for Marks 4.

Subject Lec-(18) W-(7)
Date 08.06.22 Time Wed

Control Unit (micro operations)



Q: What is μ-operations?



Steps of Micro(μ) operations:

* μ-operation for fetching:

T_1 : MAR ← PC

T_2 : MBR ← mem (instruction)

T_3 : IR ← MBR
PC ← PC + 1

→ sequencially execute করানোর জন্য ।
→ Independent — তাই 4 টা না করে 3টা Tranzaction হবে ।

Subject.....

Date:.....Time:.....

clock pulse দ্বারা সকল Transaction controlled হয়।

* each clock pulse is a trigger to processor to do a new activity.

μ-operation instruction :

① Immediate Addressing Mode :

MOV BL, 25H

T_1 : MAR ← PC

T_2 : MBR ← mem (inst)

T_3 : IR ← MBR

PC ← PC+1

T_4 : BL ← 25H (IR)

② Reg Addressing Mode :

MOV BL, CL

T_1 : —

T_2 : —

T_3 : —

T_4 : BL ← CL

↑ data given in register

— same as before

Sweet Bird

Subject.....

Date:.....Time:.....

Q: Add BL, 25 H ?

⟹ T_1 :
T_2 :
T_3 :
T_4 : BL → BL + 25H (IR)

(3) Direct Addressing Mode :

MOV BL, [5000H]

address instruction এ দেয়া থাকে।

T_1 : —
T_2 : —
T_3 : —
T_4 : MAR ← IR (address)
T_5 : MBR ← mem
T_6 : BL ← MBR [5000]

Q: ADD reg, [add] ?

T_1 :
T_2 :
T_3 :
T_4 : MAR ← IR (address)
T_5 : MBR ← mem (data)
T_6 : Reg ← Reg + MBR

(4) Indirect Addressing Mode :

MOV BL, [CX]

address register এ দেয়া থাকে।

T_1 : —
T_2 : —
T_3 : —
T_4 : MAR ← CX
T_5 : MBR ← mem
T_6 : BL ← MBR

ADD, SUB etc.

Sweet Bird

Subject.....

Date:.....Time:.....

(5) Implied Addressing Mode :

carry flag 1 করতে হবে,

T_1 :

T_2 :

T_3 :

T_4 : CF ← 1

E_p : STC

↓

set the carry flag

~~data flow~~

—— × ——

CT-2

Booth's Algo

Booth's Multiplication, Division

~~Pipelinging~~

Adder, Substractor

Lec - 5 - 10 , Lec - 13, 14

Mid Break starts Here ....

Yayyy ☺

Sweet Bird