PRIME
First Step for Study

After MSE

Page No.
Date 30 10

Division Algorithm (Restoring Algorithm)

M - Divisor
Q - Dividend
count - n

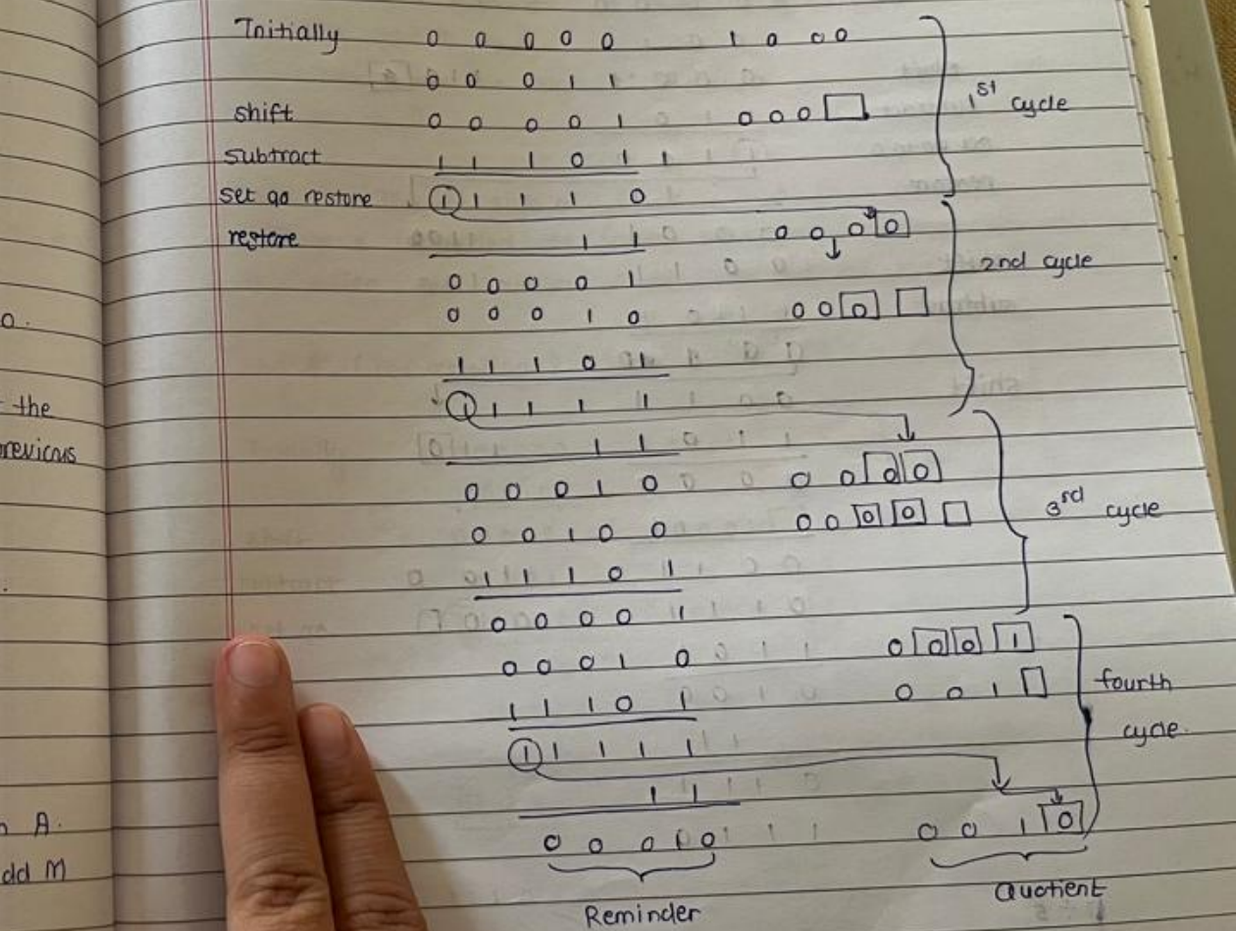
1 Step - Load 2's complement of divisor into m
Load dividend into A, Q register's.
dividend should be 2n bit positive number.

2 step - Shift A, Q left 1 bit position
3 step - A ← A-M Assign
4 step - a) 4th A if ~~reserved~~ result is non negative msb of A = 0.
then 1 is assign to Q_0
b) If result is negative i.e msb of A=1 at the time Q_0 must be assign 0. And restore previous values of A.
5 step - repeate 2-4 as many as bit a in Q.
6 step - reminder will be in A and Quotient in Q.

9.23 figure.
Do the following three steps n times :-
1. Shift A & Q left 1 bit position.
2. Subtract M from A and place the answer back in A.
3. If the sign of A is 1, set q_0 to 0 and add M back to A, otherwise, set q_0 to 1.

Page No.
Date 07 10 24

eg. 10 / 11) 1000 / 11 / 10



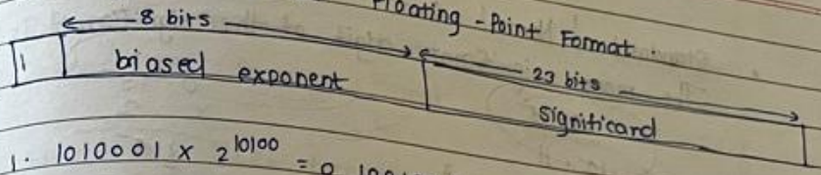
10001 000101 111010 + 1 = 111011

5) 17 (3 , -15 , 2

a) 17 ÷ 5

Initially	000000	10001	
	000101		
shift	000001	0001 □	1st cycle.
subtract	111011		
set qo	111100		
restore	000101		
	000001	0001 0	
shift	000010	001 0 □	2nd cycle
subtract	111011		
set qo	111101		
restore	000101		
	000010	001 0 0	
shift	000100	01 0 0 □	3rd cycle
subtract	111011		
set qo	111111		
restore	000101		
	000100	01 0 0 0	
shift	001000	1 0 0 0 □	4th
subtract	111011		
set qo	000011		
		1 0 0 0 1	
shift	000111		
subtract	111011		
set qo	000010	0 0 0 1 □	
	Reminder	0 0 0 1 ↑	Quotient.

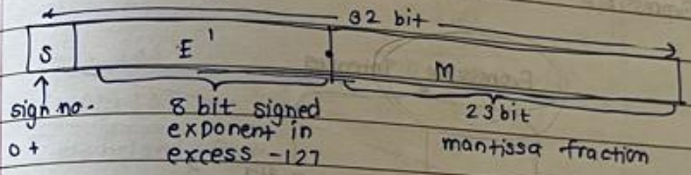
10.18 Typical 32-bit Floating-Point Format.



$1.1010001 \times 2^{10100} = 0\ 10010011\ 10100010000000000000000 = 1.6328125 \times 2^{20}$

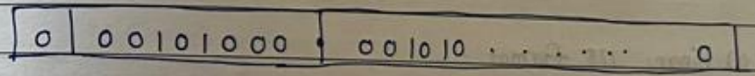
$-1.1010001 \times 2^{10100} = 1\ 10010011\ 10100010000000000000000 = -1.6328125 \times 2^{20}$

Homacher zaky.

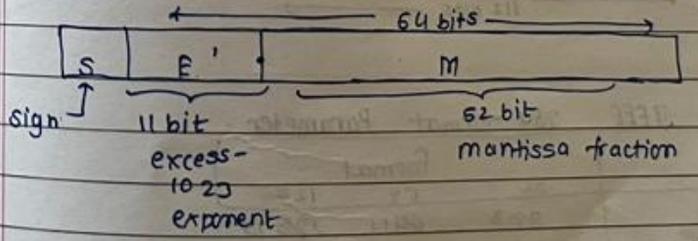


Value represented = $\pm 1.M \times 2^{E'-127}$

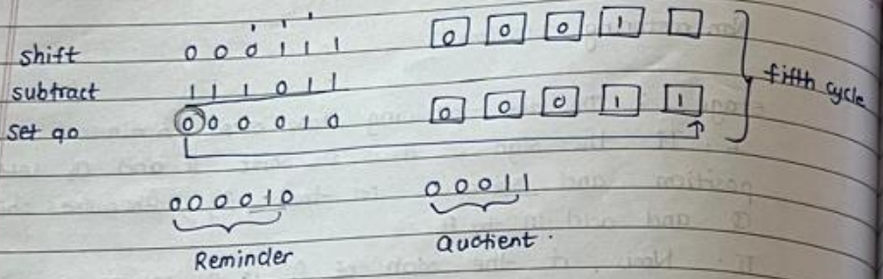
a) Single precision



Value represented = $1.00101000\cdots0 \times 2^{-87}$

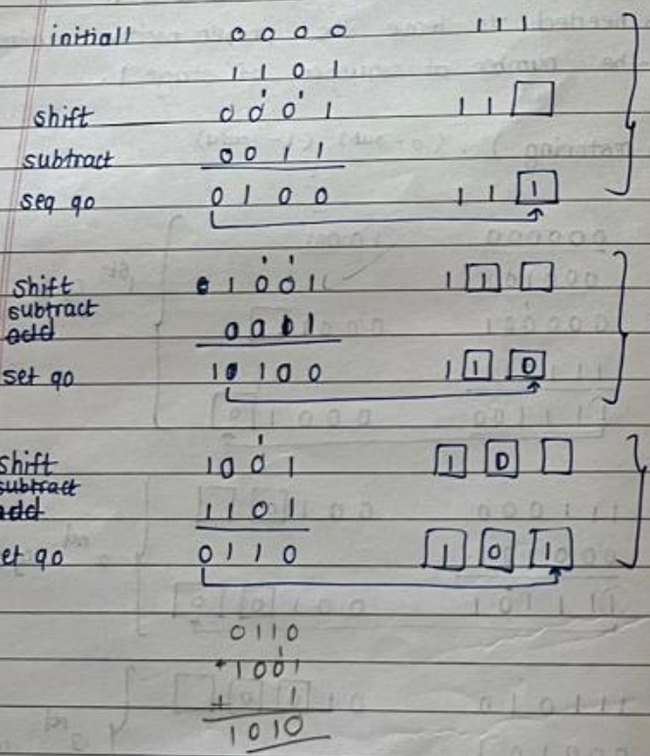


b) Value represented = $\pm 1.M \times 2^{E'-127}$



Reminder — Quotient.

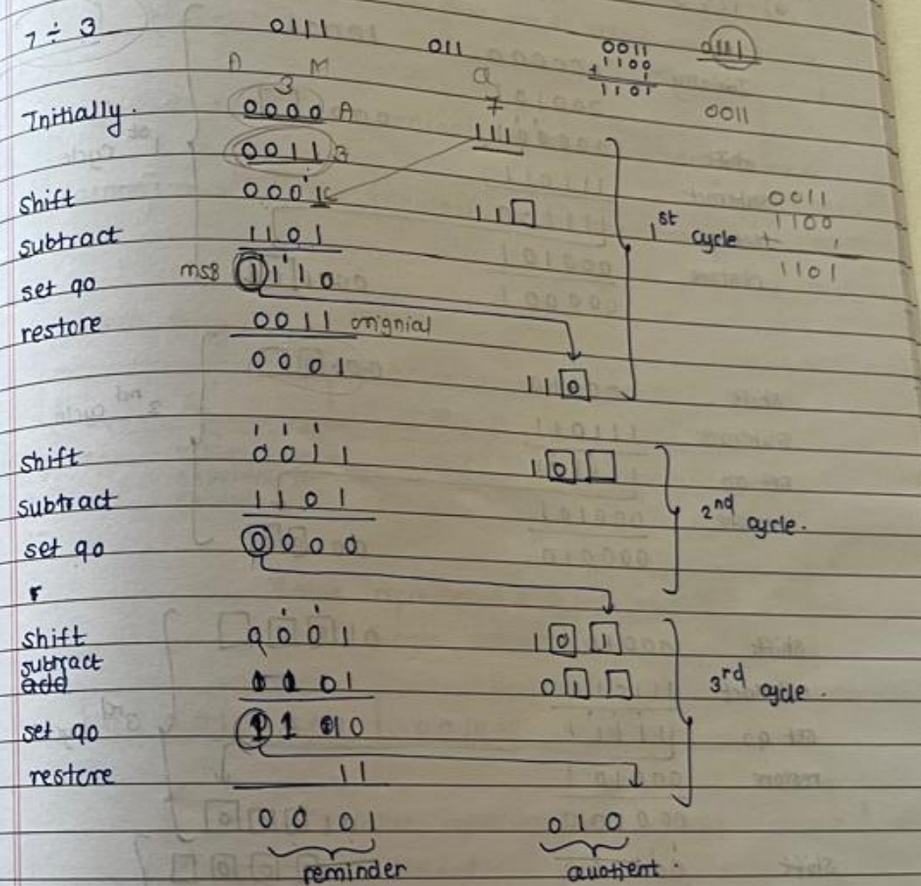
eg. $7 \div (-3)$ (-2) (1)



Note - There is no any fixed division for signed bit number.

Restoring Division. (1 - restore sub) (0 - shift, sub add)

2) $7 \div 3$



$A = S_A \cdot E_A \cdot M_A$

$B = S_B \cdot E_B \cdot M_B$

$(A + B)$

1. If $S_A = S_B = 0$ i.e both are positive numbers. Now we can add M_B and M_B when both biased exponents are same or equal.

$E_A = 128 + 10$

$E_B = 128 + 9$

$E_A - E_B = 1$ then Mantisa should be shifted by 1

Hence shift M_A

The M_A we can write as $010 \cdot 1010$

E_A you can write as 10001001.

The alternative of this can be : $E_A = 128 + 10$.

M_B can be written as : $00 \cdot 100010 \cdots \cdots 000$.

$E_B = 10001010$.

2. We can shift any one M_A or M_B but its difference between the exponents is large then so many bit shifting is required.

Key -: loosing LSB side is better than MSB side. Hence always made lower exponent convert to larger exponent.

M must always be 01.

eg. $A = 414C0000H$ $B = 40EB0000H$

$A = 0100\ 0001\ 0100\ 1100\ 0000$ $B = 0100\ 0000\ 1110\ 1011\ 0000$

	E_A		E_B	
0	100 00010	100 1100 0000 0000 0000	0	10000001	110 1011 0000 0000
sign	130	$M_A = 01 \cdot 10011$ 00000	sign	129	$M_B = 01 \cdot 1101000$

Shift M_B by 1 bit to wright. $M_B = 00 \cdot 11101$

$M_B + M_A$.

$00 \cdot 11101$ $M_R = 10 \cdot 10000$.

$+01 \cdot 10011$

$10 \cdot 10000$ $1 \cdot 0100000 \times 2^1$ normalized form.

normal we shift 1 that is add to get ER.

$ER = EA + X$

$= 10000010 + 1$

$ER = 10000011$

$4 + 1 + 0 + 0$

$4 \times 16^3 + 1 \times 16^2 + 0 \times 16^1 + 0 \times 16^0$

0	10000011	0100 0000 ... 0

✓ 41A0 0000H → convert it into decimal = 16800

A = | 0 | 100 00010 | 100 1100 0000 0000 0000 ... |

decimal is ~~12.75~~ ~~12.05625~~ $1 \cdot 1001100 = 1.50703125$

B = | 0 | 10000001 | 110 1011 0000 00 | Decimal is ~~7.25~~

$= 1 \cdot 1101011 = 1.818359375$

~~decimal = 7.2733025432~~.

~~12.05625 + 7.2733025432 = 19.3295525432~~.

A Decimal value = 12.75 B = 7.34375

$A + B = 20.09375$ (binary)

10100.00000

Normalized form = $1 \cdot 0100000000 \times 2^9$

Exponent = 4 + 127 = 131. which is 10000011

Mantisa = 01000000

0	10000011	01000000 00000000

41A00000H ✓

same.

①111 ①0001

0011
1100
1101

eg. 7 ÷ 3.
m = 00011
Dividend = 00111

Initially	0 0 0 0 0	1 1 1
	0 0 0 1 1	
shift	0 0 0 1	1 1 0 □
subtract.	1 1 0 1	
set q0 to 0	①1 1 0 0	
restore	1 1	
	0 0 0 1	1 1 0 0
shift	0 0 1 1	1 0 0
subtract	1 1 0 1	
	0 0 0 0	
shift	0 0 1 1	
	1 1 0 1	1 1 0
	0 0 0 0	
	1 1	
	0 0 1 1	1 0 0
	0 1 1 1	0 0 0 □
	1 1 0 1	
	0 1 0 0	
	1 1	
	0 1 1 1	
	1 1 1 0	

17 ÷ 5

Page No.
Date 08 10 24

0101
1010
111011

Non restoring Division.

Stage 1: Do the following two steps n times.
I. If the sign of A is 0, shift A and Q left one bit position and subtract M from A; otherwise shift A and Q and add M to A.
II. Now, if the sign of A is 0 set q0 to 1; otherwise set q0 to 0.

Stage 2: If the sign of A is 1 add M to A.
Stage 2 is needed to leave the proper positive reminder in A after the number of sequence of stage 1.

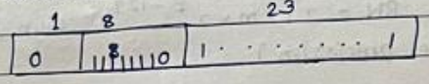
eg. 17 ÷ 5 (non restoring). (0 - sub) (1 - add)

	A	Q	
Initially	000000	10001	1st cycle
	000101		
shift	000001	0001□	
subtract	111011		
set q0	①11100	0001[0]	
shift	11000	001[0]□	2nd cycle
add	000101		
set q0	①11101	001[0][0]	
shift	111010	01[0][0]□	3rd cycle
add	000101		
set q0	①11111	01[0][0][0]	
Shift	111110	1[0][0][0]□	4th cycle.
add	000101		
set q0	⓪00011	1[0][0][0][1]	

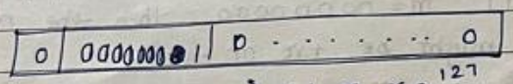
IV) E = all 0's and M = all 0's (it represents a value 0)

V) If E = all 0's but M ≠ all 0's (it gives a non zero value) so represented number is closest to zero and denormalized number.

Maximum number representation in single bit precession.

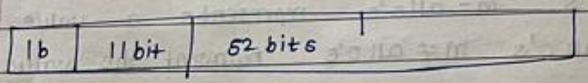


mininum number



$\downarrow -1^s \times 1.m \times 2^{-127}$

- The Biased values is 1023 (double precission)



If you consider 11 bit you can represent 2048 as maximum. 2 are reserved so 2048 − 2 = 2046 ÷ 2 = 1023 is your biased.

I) If E = all 1's (2047) all M = all 0's (∞ decimal) It represent's infinity value

II) If E = all 1's M ≠ all 0's (i.e m is non zero), it represents not a number.

III) If E = all 0's M = all 0's represented value is zero.

IV) If E = all 0's M ≠ all 0's, it represented as value closedest to zero which is a denormalized no.

V) If E lies between 1 and 2046 (i.e $1 \le E \le 2046$) the numbre represented is $\boxed{(-1)^s \times (1.m) \times 2^{E'-1023}}$

Maximum − $1.1\ldots \times 2^{1023} \Rightarrow \pm 1.8 \times 10^{308}$ ← (decimal representation)

min − $1.0000\ldots \times 2^{-1022} \Rightarrow \pm 2.2 \times 10^{-308}$

1.40625

1.41796875

0.6875 × 2 = 1.375 0

0.828125

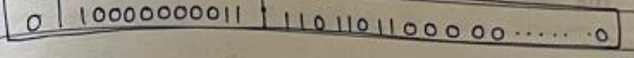
-45.83593 75

Exercise

1) Represent 29.6875 in double precission IEEE 754 format.

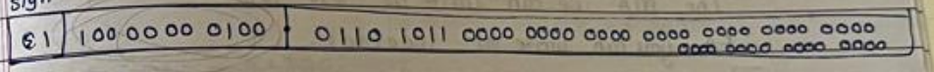
11101.1011

→ 1.11011011×2^4 → normalized form.



2) Convert C046B00000000000 H into decimal.

1100 0000 0100 0110 1011 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000



sign = 1

exponent = 1028 − 1023 = 5

Mantisa = $0\times\frac{1}{2} + 1\times\frac{1}{4} + 1\times\frac{1}{8} + 0\times\frac{1}{16} + 1\times\frac{1}{32} + 0\times\frac{1}{64} + 1\times\frac{1}{128}$

$+1\times\frac{1}{256} = 0.8359375$ 1.41796875

Decimal value = $(-1)^1 \times (1 \times 1.41796875) \times 2^5 = -45.375$

$2^{E'-127}$

$2^{E'-1023}$

UTF → search.

- Standard / Normal form.
  The most significant digit of the significand is nonzero.

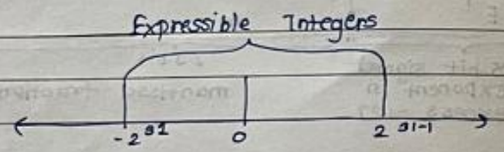
14.75
1110.11
11.1011 × 2^2
1.11011 × 2^3 → Normal form
0.111011 × 2^4
000 ↓ × 2^7

111.011 × 2^1
11.1011 × 2^2
1.11011 × 2^3

2)14
2)7 — 0
2)3 — 1
2)1 — 1
0 — 1

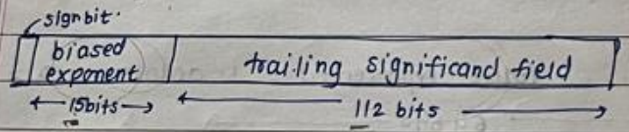
0.75 × 2 = 1.5 → 1
0.5 × 2 = 1.0 → 1

- 10.19 Expressible Numbers in Typical 32-bit formats.



- IEEE 754-2008

c) binary 128 format.



- Table 10.3 IEEE 754 format Parameter.

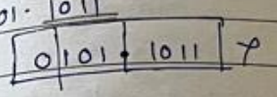
Parameter	Format		
	32	64	128
exponent bit	~~8~~ 8	~~8~~ 11	~~128~~ 15
Exponent bias	127	1023	16383

64 → 1 sig 11 biased

- Additional formats.
  1) Extended precision Format
  2) Extendable Precision Format

Table 10.4 IEEE formats.
Table 10.5 → 10.6

eg. 5.6875
101.1011



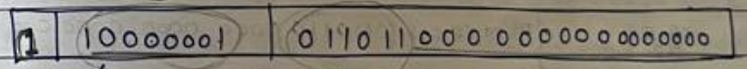
101.1011
1.011011 × 2^2
standard form.

101.1011
1.011011 × 2^2

0.6875 × 2 = 1.3750 → 1
0.3750 × 2 = 0.75 → 0
0.75 × 2 = 1.50 → 1
0.50 × 2 = 1.0 → 1
× 2
0

1011

129



a) single precesion.

Assume 4 bit data D_0 D_3 : 1 1 0 1

P6 P5 P4 P8 P2 P1

D3 D2 D1 P4 D0 P2 P1

1 1 0 P4 1 P2 P1

$P_1 \rightarrow$ D3 D1 D0 P1	$P_2 \rightarrow$ D3 D2 D0 P2
1 0 1 0	1 1 1 1

00111001 = M Hamming code 1110

$C_1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$

$C_2 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$

$C_4 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$

$C_8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$

fig 5.9 William Stallling

Hamming code 0001 resulting block 00110101111

- CRC 110011

Steps:

1) $P(x) = x^4 + x^3 + 1 = 11001$
2) $P(x)$ is in order of 4 $\Rightarrow$ CRC 4
3) Modulo 2 (Ex-OR)
4) Message append CRC 0.

```
11001 ) 110011 0000C
        11001
        -----
        00001 0000
            11001
            -----
            01001 → CRC → 1100111001
```

1) let the message = 110011 generate a polynomial $P(x) = x^4 + x^3 + 1$ and we have string 11001
2) If the generated polynomial is of order 4 then CRC bits are also 4.
3) Modulo 2 division i.e Ex-oring is done.
4) While doing step 3 original msg is appended with CRC bits 0.

eg Original msg = 1011011

Polynomial = 1101 ($x^3 x^2 x^1 x^0$)

```
1101 ) 10110110000
       1101
       ----
       01100
       1101
       ----
       00011
       0000
       -----
         110 1
```

```
1101 ) 10110110000 ①
       1101
       ----
       01100
       1101
       ----
       01110
       1101
       ----
       00110 0
       1101
       ----
       0001 0
       1101
       ----
       0011 1
       1101
       ----
       0010 ✓ No error.
```

```
1101 ) 10110111001 ②
       1101
       ----
       01100
       1101
       ----
       0001110
          1101
          ----
          001101
             1101
             ----
             0000 ✓ No error.
```

Generated polynomial selection Criteria.

Generated polynomial should not divided by x but get divided by $x+1$

Page No.
Date 11 10 24

- Represent given number in IEEE format.

Q. Represent 14.75 in IEEE Single Precision format.

→ 1110 . 11
1.1011×2^3

1.1011×2^3

0.75
× 2
1.5
0.5
× 2
1.0
0.0
× 2
0

$\frac{1}{1}$ $\frac{1}{2}$ $\frac{1}{4}$ $\frac{1}{8}$ $\frac{1}{16}$

1 1

130

0	10000010	11011 . . . . .

1 2 4 8 16 32 64 128

Q. Find out decimal equivalent of this. 40860000 Hex

→ 40860000

0	100 0000	1011	0110 0000 0000 0000 0000

0	100 00001	011 0110 0000 0000 0000 0000

sign = 0 (+ve)
sign bit = 129 − 127 = 2 exp
Ex 2
E = 2
1.421875

Decimal value = $(-1)^{sign} \times 2^{exponent} \times M$

$(-1)^0 \times 2^2 \times 1.421875$

$= 5.6875$

Mantisa = 1.0110110 0000 0000 0000 0000

$M = 1 \times \frac{1}{1} + 0 \times \frac{1}{2} + 1 \times \frac{1}{4} + 1 \times \frac{1}{8} + 0 \times \frac{1}{16} + 1 \times \frac{1}{32} + 1 \times \frac{1}{64} = 1.421875$

$(-1)^S \times 1.M \times 2^{E'-127}$

$(-1)^0 \times 1.1.421875 \times 2^2 = 5.6875$

Q. Represent C0B60000 H find out equivalent decimal of the no.

→ 1100 0000 1011 0110 0000 0000 0000

1	10000001	011 0110 0000 0000 0000 0000

Sign exponent = 2 Mantisa = 1.421875

Decimal value $= (-1)^1 \times 2^2 \times 1.421875 = -5.6875$

Page No.
Date

Q. 41ED8000

0100 0001 1110 1101 1000 0000 0000 0000

0	10000011	11011011011000000000000

Decimal = 29.7109375 $\in (-1)^0 \times 2^4 \times 1.856993594$).

Q. Considering the eqn $RN = \pm 1.M \times 2^{E-127}$ sign bit is positive or negative. (Single precission)

M - fraction

E - excess 127

1) If E = 11111111 M = 00000000 then the represented no. if infinity might be +ve or -ve.

2) If E = all 1's and M ≠ all 0's then not a number

3) If 1 < E < 254 then the value represented is $(-1)^{S} \times (1.M) \times 2^{(E-127)}$

4) If E = all 0's M = all 0's represents a value zero.

5) If E = all 0's M ≠ all 0's represent the value closest to zero i.e some denormalized one.

1. The represented value equals to $\pm 1 - 1^S \times 1.M \times 2^{E'-127}$

2. If it is so sign represents wheather a no. is +ve or -ve significant M is usually a fraction.

3. Exponent E is in excess of 127 (biased value)

I case - if E = all 1's (254) and M = all 0's (0 represent it is represented as infinity +ve or -ve depend on sign.

II - If E = all 1's (255) but M ≠ 0's E in the case the number is represented as Not a Number.

III - If E lies between 1 and 254 (1 < E < 254) in the case the represented number is $(-1^S \times 1.M \times 2^{E'-127})$ refers to single preceission representation

- Meaning
3 bits represents 8 typical blocks getting from main memory, 6 bits represents point 1 block means (32 bytes) out of 64 blocks in one typical block. 5 bits represent out of 1 block that is from 32 bytes which one byte I need to take.

~~Processor try to match~~
- if 3 bits matching – Hit
- if 3 bits not matching – miss

When miss happens cache ~~need~~ must reload the required block by removing the existing block by applying certain policy

- If present cache memory block is modify then it must be kept to main memory first at proper position then fetch required new block.

If cache memory block is not modify then ~~=~~ overwritten 2 things
1) valid bit
2) Dirty bit.

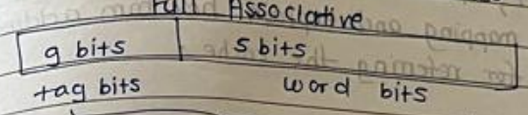
- valid bit is corresponding to if data in main memory loaded in cache memory the valid bit is 1
if nothing is loaded in cache memory then valid bit 0

- Dirty bit – if data from this 32 bytes (cache block) is modified then it is set to 1. otherwise set to 0.
eg. D=1 say need to put block in mm before it replaced.

for each block of 32 bytes we can have a valid bit, dirty bit and tag bits to make proper identification

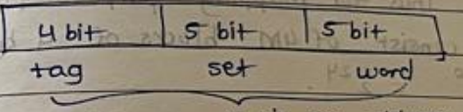
Ex. Consider main memory – 16 KB cache memory – 2K block size 32 bytes. How many bit address – 14 bits.

- Full Associative



9 bits of tag (msb) – represent the tag value. if it is matching the data from cache → then cache Hit. if not matching → then block of mm is fetched.

- Set Associative.
same example above → 2 block per set



set size is 2 blocks.
How mapping is done ?
0-31 of MM can load 0-31 set of cache respectively.
32-63 of MM can load 0-31 set of cache respectively.
So $\frac{2^9}{2^5} = 2^4 = 16$ blocks.
↓
means any set may load with any of 16 blocks. Hence tag bit comprisses of 4bit. 5 bit identify in which set block is address and 5 bit which bit to be taken from which block.

As these particular ~~partition~~ set consist of only 2 blocks this type of set assosicative is called 2 way set assosicative.
no. of cores increases → associativity increases

Q.1) Consider a computing system having 4 way set associative cache of 4 KB (capacity), mm = 64 KB & block size of 16 bytes. describe cache mapping and explain how mm address interpreted in for refering the cache.

22-10-24

fig 4.9 fig 4.10 Direct Mapping Example.

eg.2) The main memory is of capacity 16 MB, then the cache can hold 64 KB of data. Data transferd between main memory and the cache in block of 4 bytes each. This means that the cache is organized as $16K = 2^{14}$ lines of 4 bytes each. The mm consist of 16 MB, with each byte directly addressable by 24-bit address. Thus for mapping purposes, we can consider mm to consist of 4M blocks of 4 bytes each.

$16\ MB = 2^4 \times 2^{20} = 2^{24}$.

PM

cache block.

8 bit of tag	14 bit index	2 bits word

Direct Mapping Summary:

1) Address length = (s+w) bits.
2) No. of addressable units = 2^{s+w} words or bytes.
3) Block size = line size = 2^w words or bytes.
4) No. of blocks in main memory = $2^{s+w} / 2^w = 2^s$
5) No. of lines in cache = $m = 2^r$
6) Size of tags = (s-r) bits.

fig 4.11 fully Associative Cache Organization.

fig. 4.12, fig 4.13. fig 4.14 - K-way SA (Set Associative) Cache organization

c. Estimate time required to transfer 5 MB file?
D. What is brust transfer rate?

→ a-: $8 \times 512 \times 64 \times 1KB = 8 \times 512 \times 64 \times 1024 = 268423456$
$= 2^3 \times 2^9 \times 2^6 \times 2^{10} = 2^{28} = 268435456$

b-: $\frac{60}{3600} \times 2$

Rotation latency = $\frac{\text{Rotation time}}{2} = \frac{60/3600}{2} = 8.3$ ms. ($0.0167 / 2$)

average access time = seek time + Rotational latency = $8 + 8.3$
$= 16.3$ ms

c-: 8×64 s/t $\times 1KB$ /sect = 512 KB → each cylinder consist
such 10 cylinders are required. to transfer 5 MB file.
$\frac{8 \times 60}{3.6} = 133.3$ ms. to read 8 different tracks of 1 cylinder.

The access time = $8 + 9 \times (8.3 + 133.3 + 1.5) + 8.3 + 133.3$
(seek time; 9 = shift required; 8.3 + 133.3 + 1.5 = averaging across 8 different + EOT access time; + 8.3 + 133.3)

= 1437.5 ms. time required to transfer 5 MB file

d-: Brust rate = revolution /sec x sectors /revolution x bytes /sector
$= \frac{3600}{60} \times 64 \times 1KB = 3.84$ mbps.

Q. Consider a single platter disk with following parameter.
Rotational speed is 7200 rpm
no. of tracks on 1 side of platter = 30000
no. of sectors per track = 600
seek time = 1 ms. for every 100 tracks travels. let the disk receive a request to access random sequence sector on random track. Assume the disk head start at track 0.
A. What is avg seek time?
B. Rotational latency?
C. what is transfer time for sector?
D. what is total avg time to satisfy request?

→ a. Head start at track 0, then the moment seek time is 0 if requested track is 29999 track then seek time is the time to travel these tracks.
avg seek time = $\frac{29999}{2} = \frac{14999.5}{100} = 149.995$ ms

b. avg Rotational latency = $\frac{60/7200}{2} = \frac{8.333 \text{ ms}}{2}$ (normal RD) $= 4.1665$ ms (avg RD)
4.167 ms

c. $\frac{8.333}{600} = 0.01389$ ms transfer time.

d. $149.995 + 4.167 + 0.01389 = 154.17589$ ms

- Error Correcting code is Hamming code.
$2^k - 1 \geq m + k$.
$(m = 8)$ $k = 3 : 2^3 - 1 < 8 + 3$
$k = 4 : 2^4 - 1 > 8 + 4$

a) 1. F0010 (Hex)
Binary - 1111 0000 0000 0001 0000
tag - 1111 = 15
offset (4 bits) = 0000 = 0
index (12 bits) = 0000 0000 0001 = 1

2. 01234
Binary - 0000 0001 0010 0011 0100
offset (4 bits) - 0100 = 4
index (12 bits) = 0001 0010 0011 = (291) decimal Hex (123)
tag (4 bits) = 0000 = 0

3. CABBE
Binary - 1100 1010 1011 1011 1110
offset = 1110 = 14
index = 1010 1011 1011 = 2747 decimal ABB Hex
tag = 1100 = 12

b) Given any two main memory addresses with different tags that maps to the same cache slot for a direct mapped cache.

→

c) for the main memory addresses of F0010 and CABBE give the corresponding tag, cache set and offset value for a full associative cache.

1) F0010
Binary - 1111 0000 0000 0001 0000
offset (4 bits) = 0000 = 0 decimal
tag (16 bits) = 1111 0000 0000 0001 = F001 Hex
= 61441

2) CABBE
offset = 1110 = 14
tag = 1100 1010 1011 1011 = 51899

d) For the main memory addresses of F0010 and CABBE, gives the corresponding tag, cache sets and offset value for a two way associative cache.

→ no. of cache line = 2^{12} = 4096.
No. of sets = $\frac{4096}{2}$ = 2048

sets = log2 (2048) = 11 bits set index.

1) F0010
offset - 0000 = 0
set index (11 bits) = 000 0000 0001 = 1
tag = 11110 = 30

2) CABBE
offset = 1110 = 14
set = 010 1011 1011 = 699
tag = 11001 = 25

= Instructions / program × Misses / instruction × Miss penalty

eg. Average Access Time.

Hit time is also important for performace. Average memory access Time (AMAT)

AMAT = Hit time + Miss rate × Miss penalty.

Example - CPU with 1ns clock, hit time = 1 cycle, miss penalty = 20 cycles, I - cache miss rate = 5%

→ AMAT = 1 + 0.05 × 20 = 2ns.

2 cycles per instructions.

- Associative Cache < full associative
  n-way set associative.

Spectrum of Associativity. figure.

Set Associative Cache Organization figure.

Q. Assume that you have 4 Gbytes of mm at your disposal. 1 GB of the 4 GB has been reserved for process page table storage. Each page table entry consist of:

- A physical frame number.
- 1 valid bit
- 1 dirty bit
- 1 LRU status bit

Virtual addresses are 32 bits, Physical address are 26 bits, The page size is 8 KB. How many process page table can fit in the 1 GB space?

→ 8 KB ($2^3 \times 2^{10} = 2^{13}$) offset associated with a VA/PA.

- Thus remaining 19 bits used to represent VPNs & serves as PT
- Thus each PT has 2^{19} entries

Next, each PA is 26 bit. (Physical add).

- 13 bits of PA come from offset, other 13 from PT lookup.

∴ of page table = 2^{30} bytes available × (1 PT entry / 2 bytes) × (1 process / 2^{19} PT entries)

$= 2^{10}$ or 1024

→ Given that each PT entry consists of a PFN, a valid bit, a dirty bit, and a LRU bit, each PT entry will be 2 bytes.

Q. Consider a magnetic disk drive with 8 surfaces then 512 tracks per surf, 64 sectors per track and each sector size is 1 KB. The average access-time sick time is 8ms, track to track is 1.5 ms drive rotates at speed is 3600 rpm.

Successive tracks in the cylinder can be read without head moment.

A. What is the disk capacity

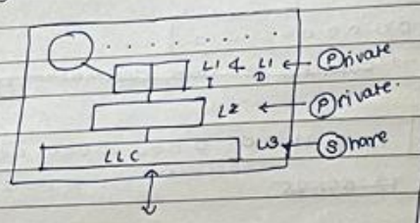
B. What is the average access time? Assume this file 8 is stored in successive sectors and tracks of successive cylinder, starting at sector 0 track 0 of certain cylinder i.

Page No.
Date 15 | 10 | 24

Unit 3 · Memory Hirerchy

Hamacher Chapter 8

figure 8.1 Connection of the memory to the processor.
figur 8.14 Memory Hierarchy.



Reading assignment
Differentiate among different levels of cache that are in intel is i7 and AMD Risen processor.
Diff - total capacity, block size, replacement policy

- Method of Accessing Units of Data.
- Characteristics of Memory System

Sequential Direct Random Associative.

- ~~Capacity performance~~ -
  Three perporformance parameter are used
  1) Access time 2) Memory cycle limit 3)

- Memory
  When block available - cache hit
  when block not available - cache miss.

Which containts are there in TLB

TLB is subset of page table.

Q. Who decide block size → architecture
Write down or mentioned the block size in AMD.

William Stealing

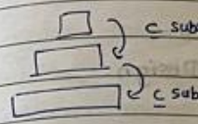
figure 4.4 Cache and main memory. 4.3 →
a) single cache b) Three level cache organization.

- figure 4.5 cache read operation.

Q. we are using what inclusive and exclusive.

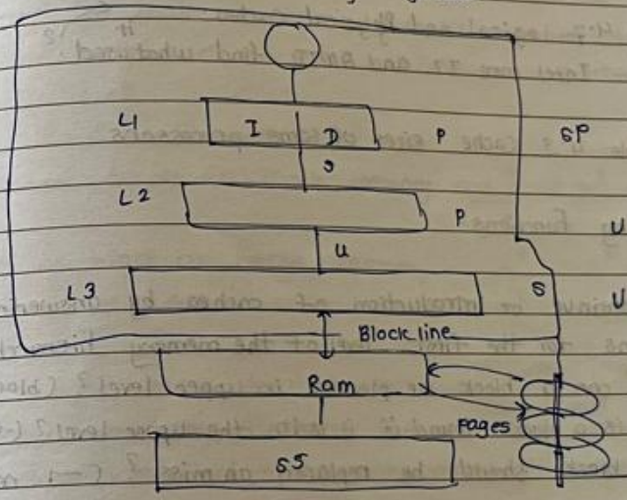
05- system software that manages performance of system.

Page No.
Date



11/10/24

Characteristics of Memory Systems.



A process of throwing out the block that is not required is called block eviction.

Block size is fixed designed by architecture.

- Method of Accessing Units of Data.
- Capacity and Performance -
- fig 4.1 The memory hierarchy.
- fig 4.3 cache and Main Memory.
- fig 4.4 cache | Main Memory structure.
- fig 4.5 cache Read operation.

Date 24 | 10 | 24

chapter 5 - Fast and Fast:

- Direct Mapped Cache
  - only one choice

- Tags and valids Bits

cache example - 8 blocks, 1 word/block, direct mapped

- initial state.

Ex.

Index	V	Tag	Data
000	~~N~~ Y	10	Mem [10000]
001	N		
010	~~N~~ Y Y	10 11	Mem [10010] [11010]
011	~~N~~ Y	00	Mem [00011]
100	N		
101	N		
110	Y	10	Mem [10110]
111	N		

1)

word addr	Binary addr	Hit/miss	cache block
22	10 110	Miss	110

2)

word addr	Binary addr	Hit/Miss	cache block.
26	11 010	Miss	010

Both are available together

3)

22	10 110	Hit	110
26	11 010	Hit	010

4)

word addr	Binary addr	Hit/miss	cache block
16	10 000	miss	000
3	00 011		011
16	10 000	Hit	000

5)

5)	18	10 010	miss	010

Page No.
Date

- Address Subdivision figure. → Hennesy Paterson book.

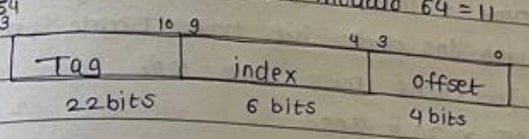
eg. Larger Block size.

64 blocks, 16 bytes/blocks.

To what block number does address 1200 map?

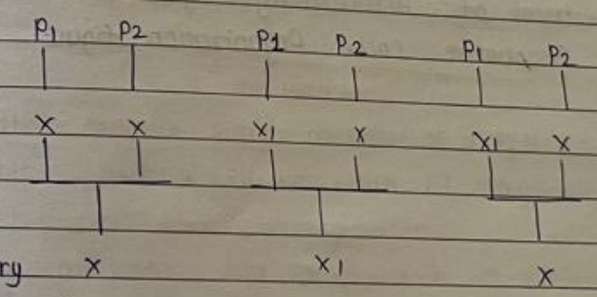
→ Block address = [1200/16] = 75

Block no. = 75 modulo 64 = 11



- Block Size Considerations.
- Cache Misses.
- Write - Through } — write Strategy
  Write - Back.

fig.



- Measuring cache Performance

* Components of CPU time
  - Program execution cycles :- includes cache hit time.
  - Memory stall cycles :- mainly from cache misses.

Memory stall cycles = $\frac{\text{Memory accesses}}{\text{program}} \times$ Miss rate x Miss penalty

- fig 4.6 Typical Cache Organization.
- Table 4.2 Elements of Cache Design.

- Cache Addresses -
  Virtual Memory

- fig 4.7 logical and Physical Caches.
- Reading Assign - Intel core I7 and AMID find what it used?

- Table 4.3 Cache sizes of some processors.

- Mapping functions.

- We continue or introduction of caches by answering the 4 questions for the first level of the memory hierarchy:

Q.1] Where can a block be placed in upper level? (block placement) dm, fam, SAM

Q.2] How is a block found if it is in the upper level? (identification) tag, valid bit

Q.3] which block should be replaced on miss? (replacement)

Q.4] What happens on a write? (write strategy).

→ 1) direct map — (Block address) Modulo (No. of blocks in cache)

full associative — ~~modulo (No of sets in cache)~~ anywhere

set associative — Modulo (No. of sets in cache)

→ 2) we have something that is known as address tag. - gives identities that the block is in certain set of cache. We with a address tag one bit is provided that is called valid bit. when bit is not set can't take it.

Block address		Block
Tag	Index	offset

- In order to pick contains from main memory we deploy something called mapping. Three types of mapping —
  1) Direct mapping 2) full associative 3) set associative.

Sir's map

Q. let main memory is 16 KB and cache memory is 2KB and cache memory block size is 32 bytes. find out total no. of blocks in main memory and cache memory.

→ MM 16 KB — 2^{14} — $2^9 \times 2^{10}$ — 3 bit for tag.

CM 2 KB — 2^{11}

↳ Block 32 bytes — 2^5

No. of blocks in main memory $= \frac{2^{14}}{2^5} = 2^9 = 512$ blocks.

No. of block in cache memory $= \frac{2^{11}}{2^5} = 2^6 = 64$ blocks

$\frac{2^9}{2^6} = 2^3$ tag bit.

- 32 byte block loaded into cache memory. Processor will output memory address w.r.t main memory block that - 32 byte block is loaded into cache memory. which part of main memory block will be loaded into cache memory is called mapping.

A block in cache memory will accomodate $\frac{2^9}{2^6} = 2^3 = 8$ typical block cache store one of the things 2^6 came from memory. These 8 typical blocks are 64 blocks apart. then cache memory ~~which one is in~~ can accomodate any of the 64 ~~can accomodate any of the 64~~ blocks ~~blocks~~. ~~is identify by 3 bits (2^3)~~?

These typical 8 blocks out of which ~~out~~ one is in cache memory is identify by 3 bits. The main memory is 16 KB processor will output 14 bit address out of that 3 bits are tag bits 6 bits are block (index remaining 5 bits are word or offset.

3 bit	6 bit	5 bit	⇒ 14 bit
Tag bit	block / index bit	word / offset bit.	

Page No.
Date 22 10 24

Full Associative Mapping Summary.

1) Address length = (s+w) bits.
2) No. of addressable units = 2^{s+w} words or bytes.
3) Block size = line size = 2^{w} words or bytes.
4) No. of blocks in mm = $2^{s+w} / 2^{w} = 2^{s}$
5) No. of lines in cache = undetermined.
6) size of tag = s bits.

Set Associative Mapping Summary.

1) Address length = (s+w) bits.
2) No. of addressable units = 2^{s+w} words or bytes.
3) Block size = line size = 2^{w} words or bytes.
4) No. of blocks in mm = $2^{s+w} / 2^{w} = 2^{s}$
5) No. of lines in set = k
6) No. of sets = $v = 2^{d}$
7) No. of lines in cache = $m = kv = k * 2^{d}$
8) size of cache = $k * 2^{d+w}$ words or bytes.
9) size of tag (s-d) bits.

Q.2) two way SA.

9 bits	13 bits	2 bits

blocks in mm = $\frac{2^{24}}{2^{22}} = 2^{2}$

tag bit = (s-d) = (22-13) = 9

Q.1) C = 4 MM = 64KB block size = 16

$= 64 \times 2^{10} = 2^{16}$

Block size. 2^{4}

~~block in mm = $2^{16+4} / 2^{4} = 2^{s+w}$~~ = 16. addressable units.

~~s=12~~ 16 - 12 = 4 bits offset.

k = 4 ways associative.

d = 6

(12 - 6) = 6 bit tag.

6 bit - tag
6 bit - set ✓
4 bit - offset

Q.3 Consider two-way set associative cache line offsets. Size 8 KB, 64 MB of MM & byte addressable format of MM address.

$2^{20} \times 2^6 = 2^{26}$

26 bit address.

22/10/24

Q.4 Consider a machine with a byte addressable main memory of 2^{16} bytes. & block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with the machine.

a) How is a 16 bit memory address divided into tag, line number and byte number?

→ MM = 2^{16} bytes = (64 KB)

Block size = 8 bytes

Cache = 32 lines (Directed mapped)

i) byte number (block offset) = $\log_2(8) = 3$ bits

ii) line number = $\log_2(32) = 5$ bits

iii) Tag = 16 − (3+5) = 8 bits

Total no. of bits = 16 bits

8 bits	5 bits	3 bits

b) Into what line would bytes with each of the following addresses be stored?

→ 1. 0001 0001 0001 1011

Tag = 0001 0001 (8 bits)

index (5 bits) = 00011

offset (3 bits) = 011

2. 1100 0011 0011 0100

Tag = 1100 0011

index = 00110

offset = 100

3. 1101 0000 0001 1101

→ Tag = 1101 0000

index = 00011

offset = 101

4. 1010 1010 1010 1010

Tag = 1010 1010

index = 10101

offset = 010

c) Suppose the bytes with address 0001 1010 0001 1010 is stored in the cache. What are the addresses of the other bytes stored along with it?

→ Since size of blocks is 8 bytes, when the bytes at address 0001 1010 0001 1010 is stored in cache, the following addresses in the same block will also be stored with it.

0001 1010 0001 1010 (given)

0001 1010 0001 1011

0001 1010 0001 1100

0001 1010 0001 1101

0001 1010 0001 1110

0001 1010 0001 1111

0001 1010 0010 0000

0001 1010 0010 0001

d) How many total bytes of memory can be stored in the cache?

→ The cache has 32 lines, and each line can store 1 block of 8 bytes. Therefore, the total memory stored in the cache is: 32×8 = 256 bytes.

e) Why is the tag also stored in the cache? (read from book)

→ The tag is stored in the cache to uniquely identify which blocks of main memory is currently stored in the corresponding cache line. Since multiple blocks from main memory can map to the same cache line the tag helps distinguishing which specific block is currently stored in that line.

Q.5) Consider a memory system that uses a 32 bit address to address at the byte level, plus a cache that uses a 64-byte line size.

a) Assume a direct mapped cache with a tag field in the address of 20 bits. Show the address format and determine the following parameter: no. of addresable units, no. of blocks in main memory, no. of lines in a cache, size of tag.

→ DM - address is divided into 3 parts

offset - $\log_2(64) = 6$ bits

index - $32 - 20 - 6 = 6$ bits

Tag - 20 bits.

20 bits	6 bits	6 bits

No. of addresable units - $2^{32} = 4,294,967,296 = 32$

No. of blocks in mm = $\frac{2^{32}}{2^6} = 2^{26} = 67,108,864$

Size of tag = 20 bits.

b) Assume an associative cache show the address format and determine the following parameters: number of address units, number of blocks in mm, no. of lines in cache, size of tag.

→ total add - 32 bit

offset - 6 bits

tag = 32 - 6 = 26 bits.

tag	offset
26 bits	6 bits

32 bits.

c) Assume a four way set associative cache with a tag field in the address of 9 bits. Show the address format and determine the following parameters. i) no. of address 2) no. of block in mm 3) no. of line in set 4) no. of set in cache 5) no. of line in cache 6) size of tag.

→ 1) addresable bit = 32

2) Tag bit = 9 bit

3) index bit = remaining bits after counting the tag & offset.

4) No. of line in cache = 2^n

5) Remaining bit = $32 - 9 - 6 = 17$ bits for index (6 = offset)

6) No. of set = $2^{17} = 131,072$

7) No. of line in cache = $2^{17} \times 4 = 2^{19} = 524,288$.

tag	index	offset
9	17	6

Q.6) Consider a computer with the following characteristics: total of 1 MB of main memory; word size of 1 byte; block size of 16 bytes; and cache size is 64KB.

a) For the main memory addresses of F0010, 01234 and CABBE, give the correct corresponding tag, cache line & word offsets for a direct-mapped cache.

→ mm = 1 MB (2^{20} bytes)

word size = 1 byte

block size = 16 bytes (2^4 bytes)

cache size = 64 KB ($2^6 \times 2^{10} = 2^{16}$ bytes).

block size = $\log_2(16) = 4$ bits → offset

No. of lines = $\frac{\text{Cache size}}{\text{Block size}} = \frac{2^{16}}{2^4} = 2^{12}$ ← index $= 4096$.

Thus we need 12 bits of index to identify cache lines.

finally remaning bits of address lines will be tag 4bits

a)

4 bits	12 bit	4 bit
tag	index	offset

20 bits. i.e (1MB)

New chapter.

william stalling

Page No.
Date 07 11 24

• Pipelining Strategy

- Pipelining Analogy
+ fig 14.9 Two stage instruction pipeline.
+ Additional Stages :- FI, DI, CO, FO, EI, WO

IF	ID	IE	M	WB

+ Page order

—
—
—
—

Inorder
out of order
(03)
(000)

- fig 14.10 Timing Diagram for instruction pipeline operation.
- fig 14.11, 14.12, 14.13, 14.14, 14.15, 14.16, 14.18, 14.19

- Pipeline Performance

$$T = \max_i [T_i] + d = T_m + d \quad | \quad 1 \le i \le k$$

- Speedup Time of the pipeline can written as-

$$T + (n-1) \times \frac{T}{k}$$

Time regular $= n \times T$

The speedup $= \dfrac{nT}{T + (n-1) \times \frac{T}{k}} = \dfrac{n}{1 + \frac{(n-1)}{k}} = \dfrac{nk}{n-1+k}$

- fig Pipeline Hazards.

types
- Data Hazard ← RAW, WAR, WAW.
- control Hazard
- Structure Hazard

Page No.
Date

- eg. Total answersheet 500 for 1 particular subject, we have 5 questions on different topics. Then one examiner takes 5 min per question so 25 min / answersheet. Total time 12500 min. If 5 examiners are working in pipeline how the show will look like.

	E1	E2	E3	E4	E5
t1	A1-1				
t2	A2-1	A1-2			
t3	A3-1	A2-2	A1-3		
t4					

1st examiner check only one question and give to another examine so 5 examiner check only question each. After 25 min A1 will be completely accessed. Therefore every 5 min other paper will complete.

Total time required $= 25 + 5 \times 499 = 2020$.

- Data dependencies of ~~three~~ four types.
1. Flow dependence denoted as $S_1 \rightarrow S_2$
2. Antidependence : output of S_2 overlaps the input of S_1 and S_2 follows S_1 in program order.
3. Output dependence : produce same output variable
4. I/O dependence :

- eg. Consider the following code fragment of four instruction.

S_1 :	Load R1, A	/ R1 ← memory(A) /
S_2 :	Add R2, R1	/ R2 ← (R1) + (R2) /
S_3 :	Move R1, R3	/ R1 ← (R3)
S_4 :	store B, R1	/ Memory (B) ← (R1) /

S_2 is flow dependent on S_1
S_3 is antidependent on S_2
S_3 is output - dependent on S_1

Note that dependence is a partial ordering relation.
S_2 and S_4 are totally independent.

b) Consider a code fragment involving I/O operation:

```
S1 : Read (4), A(1)    /Read array A from file 4/
S2 : Process           /process data/
S3 : Write (4), B(1)   /write array B into file 4/
S4 : Close (4)         /close file 4/.
```

Q. 2.4 Answer the following questions on program flow mechanisms and computer models:

a)
```
S1 : load R1, 1024        /R1 ← 1024/
S2 : Load R2, M(10)       /R2 ← Memory(10)/
S3 : Add R1, R2           /R1 ← (R1) + (R2)/
S4 : Store M(1024), R1    /Memory (1024) ← (R1)/
S5 : Store M((R2)), 1024  /Memory (64) ← 1024/
```

Where (Ri) means the content of register Ri and memory (10) contains 64 initially.

a) Draw a dependence graph to show all the dependences

b) Are there any resource dependences if only.

→ a) S_3 is flow dependent S_2 & S_1
S_4 is flow dependent on S_3
S_5 is flow dependent on S_2
S_3 is output dependent on S_1

b) Yes there is resource dependence if only one OP is available in cpu

- Control Hazards.
- Branch Prediction → next page.

Various techniques can be used to predict whether a branch will be taken:

1. Predict never taken
2. always taken
3. by opcode

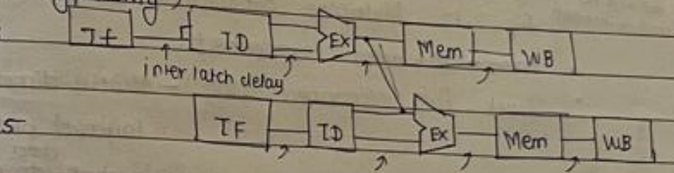
} These approach are static do not depend on execution history up to the time of the conditional branch.

1. Taken / not taken switch
2. Branch history table

} These approach are dynamic depend on execution history.

- fig 14.20
- Pipelining and ISA design
  - RISC-V designing for pipeling.
- Forwarding (aka Bypassing)

```
add x1, x2, x3
sub x4, x1, x5
```



- Load-Use Data Hazard

```
ld x1, 0(x2)
```

- Code Scheduling to Avoid Stalls.
- Control Hazard.
  - stall on Branch., to avoid control Hazard need proper Branch prediction
  - ~~Branch prediction~~ RISC-V Pipelined Datapath.
  - M Pipeline registers.

Pipeline operation < single line
multi line → not for exam

IF for load, store
EX for load
mem for load
WB for load

- RISC style characteristics.
  CISC style characteristics.

Unit 4 – I/O organization read own.

Chapter 4.

08/11/24

Data Bus.

- Address Bus Control Bus.

not mentioned in reference copy but have to read as introduct

fig. 3.16 Bus Interconnection Scheme.
fig 3.17. Multicore Configuration using QPI.

- Point-to-point interconnect.
- Quick Path interconnect (QPI) – Mutile direct layered protocol packet data transfer.

fig. 3.18, 3.21, 3.22, 3.23, 3.24, 3.25
The peripheral Component Interconnect (PCI).
Table 3.2.

Hamacher Zaky

7.1 Bus Structure. (fig. 7.2)
7.2.1 Synchronous Bus (fig. 7.3). (fig 7.4)
7.2.2 Asynchronous Bus (fig 7.6)
7.3 Arbitration (fig 7.9)
7.4) Parallel Interface (fig 7.10), 7.13)
7.4.2 Serial Interface
7.5 Interconnections – 7.5.1 Universal serial Bus (USB) (fig 7.17)

Don't read circuits.

7.5.2 Firewire
7.5.3 PCI (fig 7.18) (Table 7.1)
7.5.4 SCSI Bus
7.5.5 SATA 7.5.6 SAS 7.5.7 PCI Express. (fig 7.20)

William Stallings 17th ed.

Unit 6. 09/11/24

Multiple Processor Organization.
17.1 fig. A Taxonomy of Parallel Processor Architectures.
17.2 fig. Alternative Computer Organization.
Symmetric Multiprocessor (SMP).
fig 17.4 Generic Block Diagram of a Tightly Coupled Multiprocessor.
fig 17.5 Symmetic multiprocessor organization.
Disadvantages of the bus organization.
~~Multiprocessor~~ Directory protocol not there for exam. (os no part)
Snoopy protocol.
Write Update.
Multithreading & chip Multiprocessor
Definitions of Threads and Processes
Implicit and Explicit Multithreading
Approches to Explicit Multithreading.
fig 17.7, 17.8, Table 17.2, 17.9 fig., 17.10 fig., 17.11, 18.1
Clusters., Parallelizing Computation.
Clusters compared to SMP., Motivation., Numa pros & cons.
effective applications for multicore processor.

Hennessy Patterson 6.

SIMD.
Heterogeneous Multicore Organization
Heterogeneous system Architecture.

8.13 8.8 have a look 8.5 8.6 8.4 8.3 18.2