Video 1 : Introduction to cache memory

LRU takes care of Temporal locality.

Block/Paging takes care of spatial locality

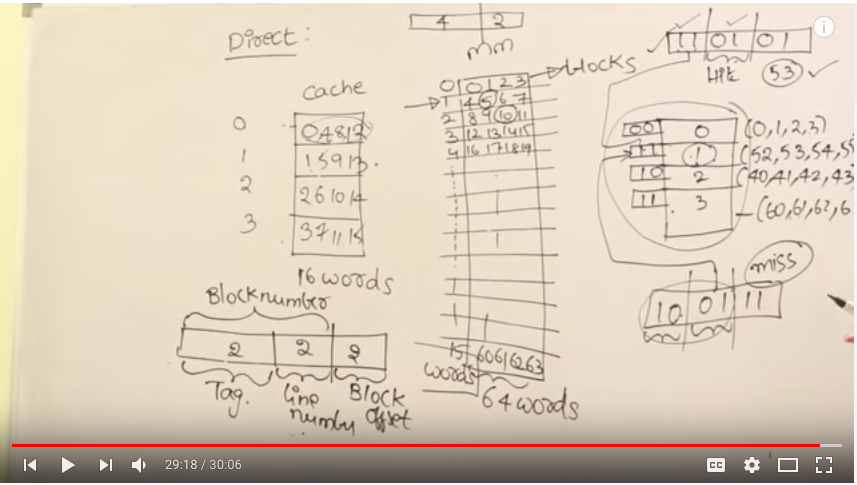
Video 2 : Direct Mapping

Process/Virtual memory – Pages

Main Memory – Frames/Blocks

Cache – Lines

1 Word is smallest addressable unit in memory. Can be anything say 1 word=1byte.

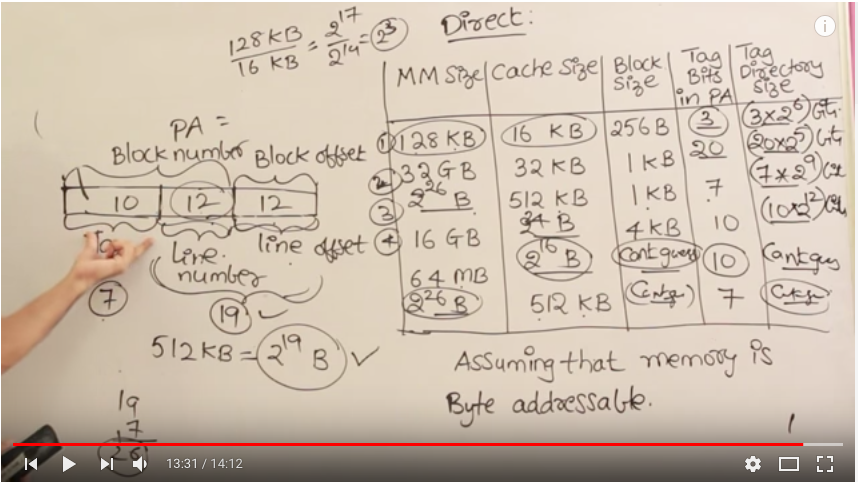


Video 4 : Direct Mapping Problems 2

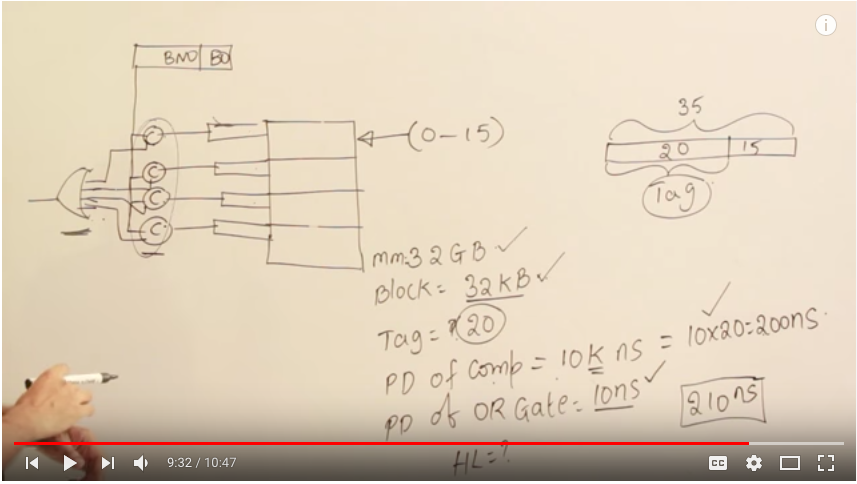
Address = Tag + LineNumber + BlockOffset/LineOffset

Cache size = 2 pow (LineNumber +LineOffset)

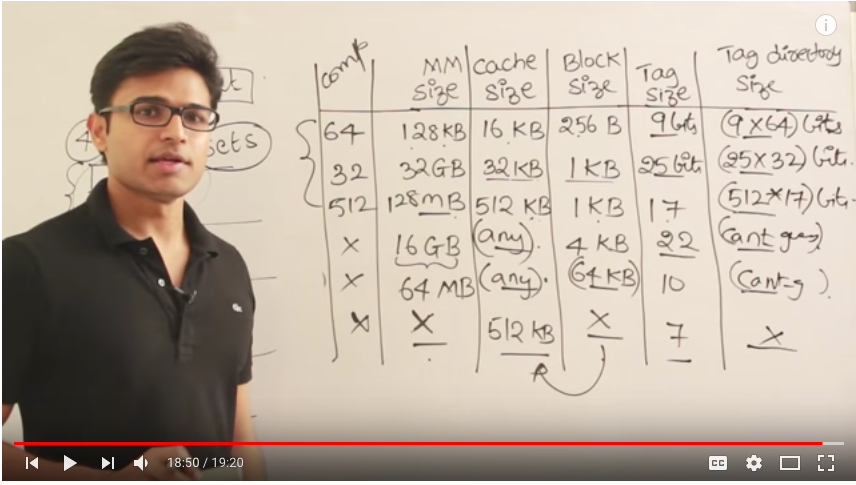
Tag Directory size = Tags \* 2 pow LineNumber



Video 5 : Introduction to associative mapping



Video 3 : Numericals on associative mapping



Video 6,7 : Problems on Set Associative Mapping

Similar to direct mapping

In direct mapping,

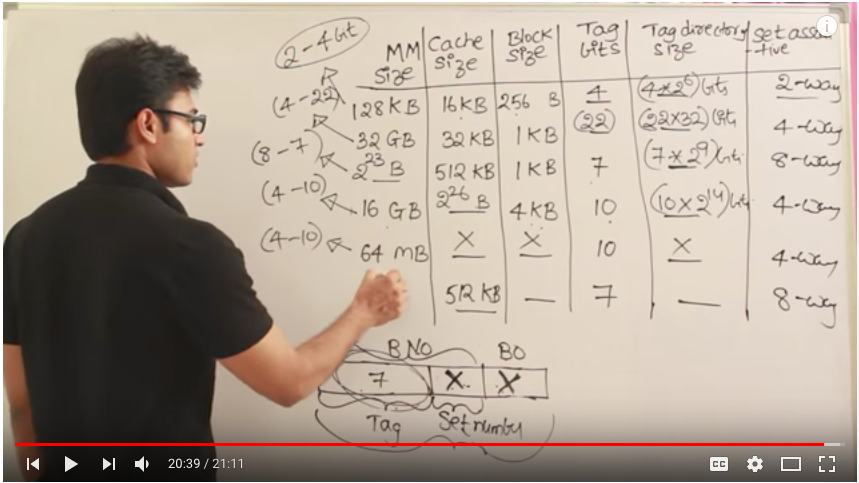
Address = Tag + LineNumber + BlockOffset/LineOffset

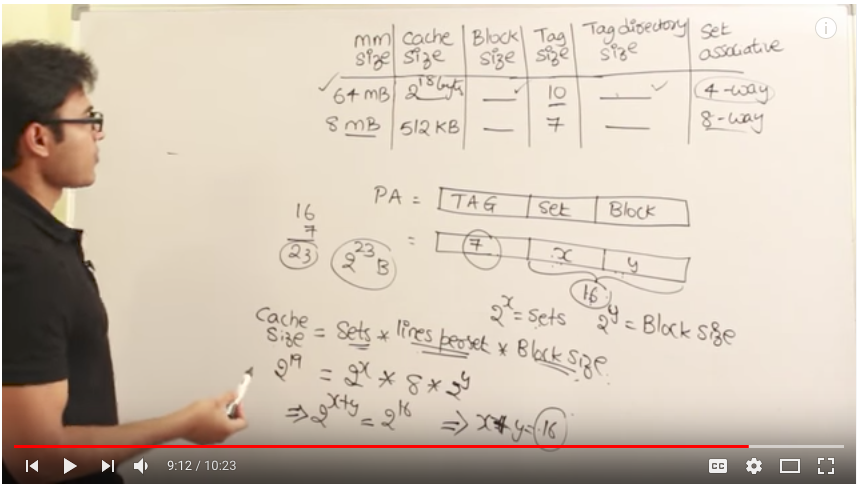
Here,

Address = Tag + SetNumber + BlockOffset/LineOffset

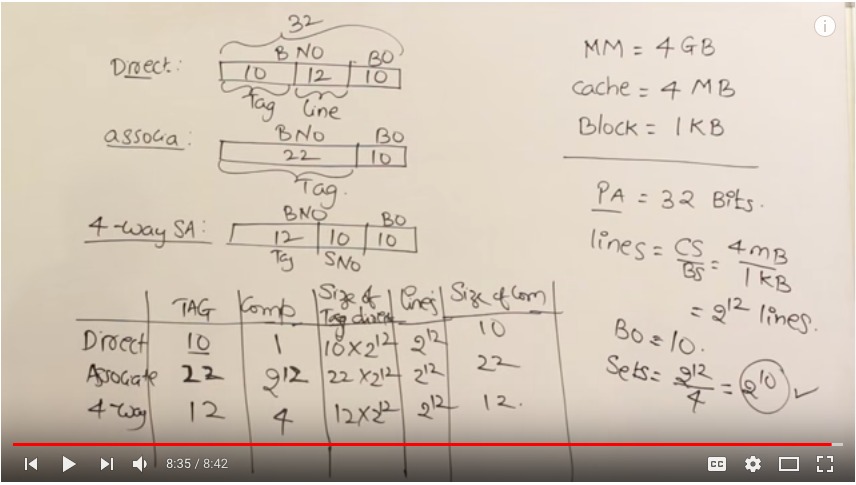
Only, Set Number = Line Number/ N-Set Associative

Less Comparaters reqd than Associative but more than Direct





Video 8 : Comparing all the mappings



Number System

Simple representation - 4bits – We can represent 2 pow 4 i.e. 16 numbers [0-15]

1’s Complement/Sign Rep- 4 bits – We can represent –(2pow3 -1) to +(2pow3-1) i.e. 14 numbers[-7 to +7]

2’s Complement - 4 bits - We can represent -2pow3 to +(2pow3-1) i.e.15 numbers[-8 to +7]

While addition of two signed two complement numbers if we have diff. carry in and carry out for MSB, overflow has occurred, if we are getting a positive no while adding two negatives or vice versa, an overflow has occurred.

Shift Left(Multiply by 2) And Shift right Operation(Divide by 2).

1.001\*2 = 10.01, 1.1\*2=11.0, 11.01/2=1.101

IEEE Represenation(32/64 bit)

32 bit = 1 Sign bit + 8 bit exponent + 23 mantissa bit, Biased no = 127

64 bit = 1 sign bit + 11 bit exponent + 52 mantissa bit, Biased no = 1023

Big-Endian Byte Ordering – Straight ordering 0,1,2,3 \n 4,5,6,7 for 4 byte add memory

Little-Endian Byte ordering – Reverse ordering 3,2,1,0 \n 7,6,5,4 for 4 byte add memory

Addressing Modes

$ - relative to program counter

# - constant

@ - indirect

Implied mode – definition of instruction gives operand eg.-complement accumulator

Immediate mode – operand specified in instruction only eg. MOV R,#20

Used for initializing a register with constant value.

Register mode – operand in register eg. MOV r1,r2

Register Indirect mode – operand exists in memory address present in register given in instruction. Eg. MOV A,(r), used in pointers

Auto Increment Add Mode – used in loops, eg. ADD r1, (r2)+

Direct/Absolute mode – Instruction itself contains effective address, eg.- ADD r,2000

Used to implement variable

Index Add Mode – used in arrays, eg.- opcode memoryadd/base add, index

Relative Add mode – relative to program counter, eg. Opcode instruction, position-independent code, also used in relocation

Base Reg Add Mode – used in relocation of program in main memory, opcode base, partadd

ADDER

Half Adder – Sum – x ExOR y, and carry- x and y

Implement using an exor gate and one and gate.

Full Adder - Sum – x ExOR y ExOR z and carry- xy + yz + xz

Implement using 1 exor gate and 3 and and 1 or gates

Ripple carry Adder – For 2 4 bit numbers, use 4 Full Adders with two inputs and one carry input.

Delay in Ripple Carry Adder – Carry Delay = 2\*nbits(and/or).Sum Delay=(2n -1)Exor Delay

Overflow Delay = 2n +1

For Addition Carry0 is 0 and for subtraction it is 1

Carry Look Ahead Adder –

Generate fuction = Xi.Yi

Propagate function = Xi + Yi, We may use Xi ExOR Yi

Delay - 1 gate delay for Pi/Gi, 2 Gate Delay for CLA logic, 1 gate delay for Sum-> Total 4 Gate Delay.

FanOut –Maximum no of inputs a Gate can take input, Even 32 bit CLA adder can have only 5/10 inputs.

16 bit CLA Adder Using 4 4bitCLA Adders -> 3+3+3=9 delays before final Sum and 4 for Sum of last CLA -> 13 delays for Sum, 12 Delays for CaryyOut16.

For n bit carry look ahead adder, n\*(n+1)/2 AND gates required and n OR Gates

Multiplication

1.Array Multiplier, 2n-1 gates required, O(n) time complexity

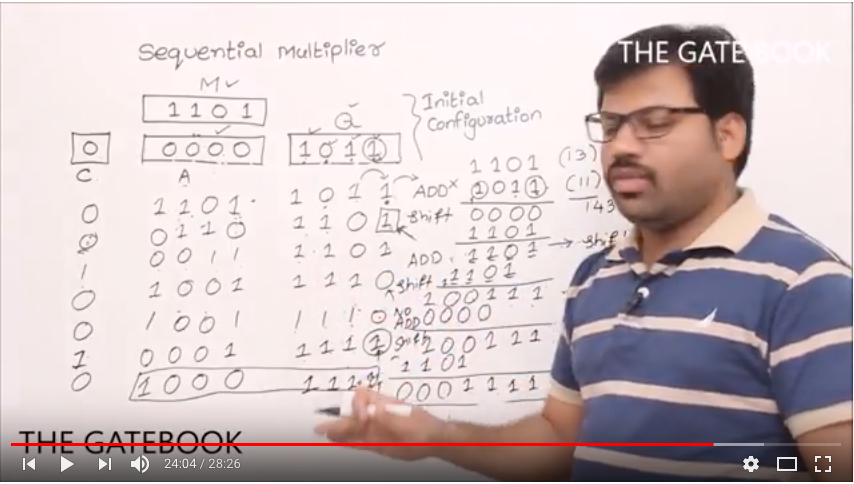
2.Sequential Multiplier

2.BoothMultiplier

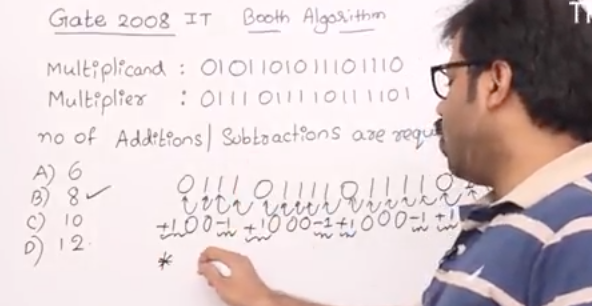
Realizing simple multiplication, and gates reqd at each level for multiplication and Full Adders required for adding current level with prev level result.

Delays in Array mult = (n-1) \* 2 + (n-1) = 3n-1 approx 3n delays, but very wasted spaces

To multiply 2 n bits no we req max 2n bits while adding we may req max n+1 bits.



Booth Recoding



modified booth algorithm reduces no of summands. [radix 4]