Information

information $\propto$ uncertainty $\propto \frac{1}{p}$

Layman:If event bound to happen, then event happening does not give any information

$I(X) = \log_2 \frac{1}{p_i}$

$I(X)_{N \to M} = \log_2 \frac{N}{M}$ bits , where N number of equally probably choice with M possible choices

2's complement

-Inverse, Add 1 to binary

Number Range

Given X bits

-Signed bit : -2^{x-1} to $2^{x-1} - 1$

-Unsigned bit : 0 to $2^x - 1$

-We can encode 2X choices, or random variables

CMOS Recipe

−Only use NFETs in pulldown circuits and PFETs in pullup circuits − PO-NI - PFETs conducts when 0, NFETs when 1 − AB is parallel in PD, but series in PU. − Ensure no direct connection from VDD to GND

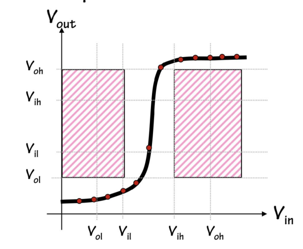
Combinational Device

1. Inputs
2. Outputs
3. Functional specification that details the output for each input
4. The propagation time to get a valid output given a valid input

Static Discipline

If a system is given a valid input, then it guarantees that it will give a valid output.

VTC



1. Gain > 1 (Because of static discipline)
2. Non-Linear Gain

Noise Margin

V_{ol} or V_{oh} is the voltage that your system outputs

V_{ih} or V_{ih} is the voltage that your system receive as input from another system.

The MOSFET

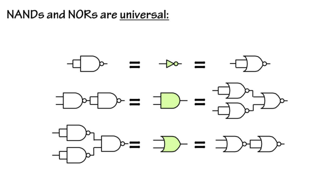
1. The side with the higher potential is drain, the one with the lower potential is source.
2. The p-type : majority care holes (impurities), bulk is connected to VDD
3. The n-type : majority are electrons, bulk is connected to GND
4. P0N1 (ON)

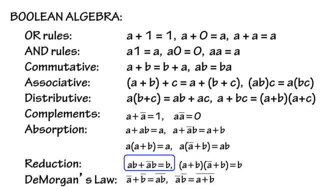
Timings

1. The Propagation Delay t_{pd} - the time delay from valid input to valid output. The effective tpd of an entire circuit is the maximum cumulative propagation delay over all paths from inputs to outputs.
2. The Contamination Delay t_{cd} - the time delay from invalid input to invalid output. The effective tcd of an entire circuit is the minimum cumulative contamination delay over all paths from inputs to outputs.

Logic Synthesis

Universal:can implement any boolean function.





BOOLEAN ALGEBRA:

OR rules:	$a + 1 = 1,\ a + 0 = a,\ a + a = a$
AND rules:	$a1 = a,\ a0 = 0,\ aa = a$
Commutative:	$a + b = b + a,\ ab = ba$
Associative:	$(a + b) + c = a + (b + c),\ (ab)c = a(bc)$
Distributive:	$a(b+c) = ab + ac,\ a + bc = (a+b)(a+c)$
Complements:	$a + \overline{a} = 1,\ a\overline{a} = 0$
Absorption:	$a + ab = a,\ a + \overline{a}b = a + b$
	$a(a+b) = a,\ a(\overline{a}+b) = ab$
Reduction:	$\boxed{ab + \overline{a}b = b,}\ (a+b)(\overline{a}+b) = b$
DeMorgan's Law:	$\overline{a} + \overline{b} = \overline{ab},\ \overline{a}\,\overline{b} = \overline{a+b}$

Karnaugh Map

1. "1" cells, No diagonal grouping, Cells can overlap.
2. The top/bottom and left/right edges of map are continuous

Multiplexer (MUX)

It is made up of logic gates (INV, AND, and OR, or NANDs).

- Muxes are universal,
- 2^k data inputs, k bits select inputs, and only can have 1 output
- Three components: the inputs, the selector signal(s), and the output.

Decoder

1. A Decoder is the opposite of Mux
2. It has k select inputs, and 2^k possible data outputs
3. The selected output i is HIGH (1), and the rest of the $2^k - 1$ data output is LOW (0).

Read-Only-Memories (ROM)

1. For an N-input boolean function, the size of ROM is roughly 2^N X number of outputs. FA: $2^3 * 2 = 16$

The Dynamic Discipline

The input to a synchronous sequential circuit must be stable during the aperture (setup and hold) time around the clock edge. The dynamic discipline states that

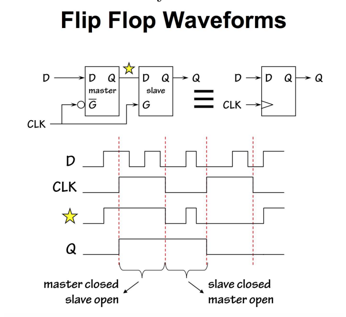
1. $T_{setup} = 2t_{pd}$
2. $T_{hold} = t_{pd}$

1. T_{setup} = the minimum amount of time that the voltage on wire D needs to be stable before the clock edge changes from 0 to 1.
2. T_{hold} = the minimum amount of time that the voltage on wire D needs to be stable after the clock edge changes from 0 to 1.
3. t_{pd} is the propagation delay of the D-latch

Flip-Flop

Created by putting 2 D-latches in series

1. There is an inverter on the G input on the master flip flop
2. When CLK signal is 0, the G wire of master latch receive a 1 while slave flip flop receive a 0 Master latch : Write mode Slave latch : Read mode
3. Vice-versa when CLk signal is 1
4. 1 D-Latch is on write mode why 1 D-latch is on memory mode at anytime.



Note that output wire Q only changes when CLK rises from 0 to 1

Timing Constraint

1. t_{CD} of a D-latch is the time taken for invalid CLK input to produce an invalid output on wire G
2. T_{PD} of a D-latch is the time taken for valid CLK input to produce a valid output on wire G

Sequential Logic Timing Constraint

$t_1 = t_{CD,R1} + t_{CD,1} > t_{HOLD,R2}$

$t_2 = t_{PD,R1} + t_{PD,1} <$
$t_{CLK} - t_{SETUP,R2}$

Metastable State

Properties

1. It corresponds to an invalid logic level
2. Unstable equilibrium which will settle to valid 0 or 1 eventually
3. Settling can be arbitraily long
4. All bistable system exhibits at least 1 metastable state
5. Cannot be avoided but can be minimize

Clock Skew

$t_1 = t_{CD,R1} + t_{CD,R1} >$
$t_{HOLD,R2} + t_{skew}$

$t_2 = t_{PD,R1} + t_{PD,CL1} <$
$t_{CLK} - t_{SETUP,R2} + t_{skew}$

Programmable Machines

Finate State Machine: Enumeration

FSM with i inputs, o outputs, s states

1. Truth table has 2^{i+s} rows with (o + s) columns each
2. $2^{(o+s)2^{i+s}}$ max state
3. Limitation : cannot solve problems with arbitrarily many states

MOORE MEALY

Moore: output drawn on states and depends only on state. Arcs leaving must be mutually exclusive and colletively exhaustive

Turing Machines

Turing Machine Specification

1. Doubly-infinite tape
2. Discrete symbol positions
3. Finite alphabet
4. Control FSM
   Inputs - Current Symbol
   Outputs - Write 0/1 , move Left/Right
5. Start State , Halt State

Properties

1. Can be used to compute integer functions of form $y = T_k[x]$
2. Where k: FSM index, x: input tape configuration, y: output tape configuration.
   *Not all integer functions can be computed with Turing Machines
3. Computable functions : f(x) computable $\Leftrightarrow \exists k : \forall x :$ $f(x) = T_k[x] = f_k(x)$
4. Church-Turing Hypothesis states that any computable function is computable by a TM

Universal Functions and Universality

Universal function: $U(k, j) = T_k(j)$

U is comptable by a Turing Machine

$\to$ k encodes a 'program'

$\to$ j encodes the input data to be used

$\to T_u$ interprets program

Von Neumann Model

4 components

1. CPU - contains several registers as well as logic
2. Memory = storage of N words with W bits, where W is a fixed archiertectural paramter, and N can be expanded to meet needs
3. Input/Output
4. Connection Bus

Week 5 Machine Language and Compilers

Compiler

1. Complier translate high-level language into low-level assembler machine language
2. Done before execution and slows program development
3. Decisions made during compile time,before execution

Interpreter

1. Computes exact instructinos
2. Done after execution and slows down program execution
3. Decisions made during run time, after execution

Week 8

CPU Design Tradeoffs

1. Maximum Performance
2. Minimum Cost
3. Best Performance - MIPS = $\frac{ClockFrequency(MHz)}{clocksperinstruction}$

Instruction classes - OP,OPC,MEM,Transfer of Control

Multi-Port Register Files

1. 2 combinational Read ports
2. 1 clocked Write port - Write Address,Write Data,Write Enable

Exception

Bad Opcode

Reg[XP] ← PC + 4 , PC ← illOp

Other

Reg[XP] ← PC + 4 , PC ← Xadr

Extending Beta

LDX(R0,R1,R2)

ADD(R1,R0,R0) , LD(R0,0,R2)

Reg[Rc] ← Mem[Reg[Ra] + Reg[Rb]

STX(R0,R1,R2)

ADD(R1,R0,R0) , ST(R2,0,R0)

Mem[Reg[Ra] + Reg[Rb]] ← Reg[Rc]

Must amend data path and register file! Register file needs another RA/RD port. RA2SEL mux can be removed.

Week 9 Memory Hierarchy

From fastest and most expensive: Cache,SRAM,DRAM, Hard desk Both SRAM and DRAM are volatile storage

SRAM

Static RAM

1. 6 transitors and amp sense in each cell with 2 bit line
2. Value stays the same if word line is 0
3. Store a bit

DRAM

1. 1 transitors ,a capacitor and store a bit
2. A lot cheaper but significantly slower.

Disk

SSD/HDD

Non volatile storage (Can store information even after power source is cut)

The Cache Idea

1. Look for requested info in cache
2. If found, it's a hit. Else, go to physical memory and subsequently disk.

Locality of references

Reference to memory location X at time t implies that reference to X + change(X) at t +change(t) becomes more probable as change(X);change(t) approaches zero.

Type of Cache

Fully Associative Cache(FA)

Pros: Parallel Lookup and Flexible,address can be stored on any cache line

Cons: Needs many Bool operator(1 for each cache line). Also need a replacement strategy.

1. TAG contains the all bits of address A
2. DATA contains all bits of content at A: Mem[A]
3. Expensive, made up of SRAMS and other hardwares (comparator circuit at each row)
4. PARALLEL lookup, hence making it fast
5. Flexible because memory address + content can be stored on any TAG-DATA row.
6. Need replacement strategy

Direct mapping Cache(DM)

Pros: Cheaper as only 1 Bool operator , No need replacement strategy

Cons: Contention -

1. TAG contains T-upper bits of address A
2. DATA contains all bits of content at A: Mem[A]. Lower K-bits of A decides row of DM Cache. Mapped to exactly one of the rows of DM cache.
3. Also made of SRAMS but cheaper than FA cache(only one comparator circuit per DM cache)
4. No parallelism, but fast mapping between address and cache line index
5. Suffers contention, two different addresses can be mapped to the same location when the K-lower bits are the same. K-lower bits selected due to locality of reference, but does not completely eliminate contention.

Cache Design Issues

- Associativity : how many different address can be stored in the cache
- Replacement strategy
- Block size
- Write strategy

N-way set associative cache

1. if N = 1, it is a DM cache.
2. if k = 1, it is a FA cache
3. Same Column $\implies$ same cache line
4. k lower bits determines the set in which it will be in.

Replacement Strategy

1. Least Recently Used: overhead is O(Nlog$_2$ N)
2. FIFO: O($log_2 N$) bits/set
3. Random - uses psedu random generator to get reproducible behavior

Block Size

Blocks of 2^B words per row,

Special Bits

Valid Bit

The valid bit indicates that the particular cache row (also called cache line, but it is a different graphical representation from 'cache line' in the N-way set associative cache) contains data from memory and not empty or redundant value. We only check cache lines with valid bit = 1.

Dirty Bit

The dirty bit is set to 1 iff the CPU writes to cache and it hasn't been stored to the memory (memory is outdated).

LRU

The LRU bit is present in each cache line (for FA), and each cache set-cacheline cell (for NW), regardless of the block size because R/W with block size more than 1 is always done in parallel.

Cache Writes

Write-through

CPU writes are done in the cache first by setting TAG = Addr, and Data = new Mem[Addr] in an available cache line, but also written to the main memory immediately. This stalls the CPU until write to memory is complete, but memory always holds the "truth"

Write-behind

Write to the main memory is buffered or pipelined. CPU keeps executing next instructions while writes are completed (in order) in the background.

Write-back

Not immediately written to the main memory. Memory contents can be "stale". Typically CPU will write to the main memory only if the data in cache line needs to be replaced and that this data has been changed or is new. This requires the dirty bit in the cache.

MMU Details

1. 2^{v+p} bytes of VM
2. 2^{m+p} bytes of actual memory
3. (m + 2) 2^v bits in MMU Pagetable
4. 2^p bytes per page

Page-Map Design

Residence Bit - if R is 1, data is in RAM

Dirty Bit: if D is 1, data has to be written to Disk before removed from RAM

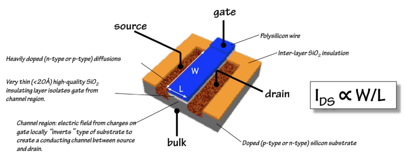
TLB

Cache to pagetable for mapping certain VPN to PPN. Locality of reference in address memory reference patterns.

Cache with VM

Each cache line stores a single data (not pages) and the address of each data.

Hex	Binary	Octal	Decimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
A	1010	12	10
B	1011	13	11
C	1100	14	12
D	1101	15	13
E	1110	16	14
F	1111	17	15
10	1 0000	20	16
11	1 0001	21	17
24	10 0100	44	36
5E	101 1110	136	94
100	1 0000 0000	400	256
3E8	11 1110 1000	1750	1000
1000	1 0000 0000 0000	10000	4096
FACE	1111 1010 1100 1110	175316	64206

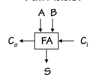

gate
source
Polysilicon wire
Inter-layer SiO_2 insulation
Heavily doped (n-type or p-type) diffusions
Very thin (<20Å) high-quality SiO_2 insulating layer isolates gate from channel region.
W
L
drain
$I_{DS} \propto W/L$
Channel region: electric field from charges on gate locally "inverts" type of substrate to create a conducting channel between source and drain.
bulk
Doped (p-type or n-type) silicon substrate

MOSFETs (metal-oxide-semiconductor field-effect transistors) are four-terminal voltage-controlled switches. Current flows between the diffusion terminals if the voltage on the gate terminal is large enough to create a conducting "channel", otherwise the mosfet is off and the diffusion terminals are not connected.

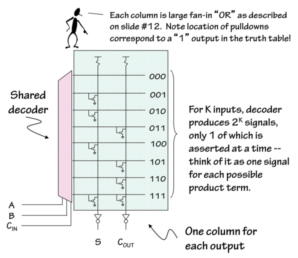
AB \ CD	00	01	11	10
00	0 A' B' C' D'	1 A' B' C' D	3 A' B' C D	2 A' B' C D'
01	4 A' B C' D'	5 A' B C' D	7 A' B C D	6 A' B C D'
11	12 A B C' D'	13 A B C' D	15 A B C D	14 A B C D'
10	8 A B' C' D'	9 A B' C' D	11 A B' C D	10 A B' C D'

Read-only memories (ROMs)

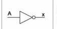
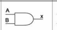
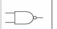
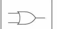
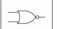
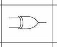
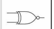
Full Adder

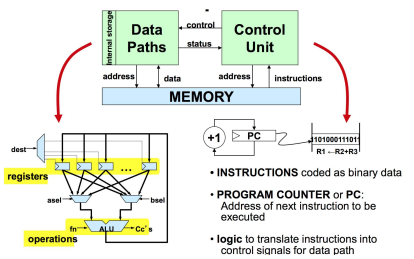


A	B	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1


Each column is large fan-in "OR" as described on slide #12. Note location of pulldowns correspond to a "1" output in the truth table!
Shared decoder
000
001
010
011
100
101
110
111
For K inputs, decoder produces 2^K signals, only 1 of which is asserted at a time -- think of it as one signal for each possible product term.
A
B
C_{IN}
S
C_{OUT}
One column for each output

Logic Gates

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol							
Truth Table	A X: 0 1; 1 0	B A X: 0 0 0; 0 1 0; 1 0 0; 1 1 1	B A X: 0 0 1; 0 1 1; 1 0 1; 1 1 0	B A X: 0 0 0; 0 1 1; 1 0 1; 1 1 1	B A X: 0 0 1; 0 1 0; 1 0 0; 1 1 0	B A X: 0 0 0; 0 1 1; 1 0 1; 1 1 0	B A X: 0 0 1; 0 1 0; 1 0 0; 1 1 1



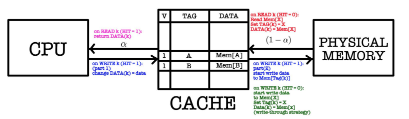
• **INSTRUCTIONS** coded as binary data

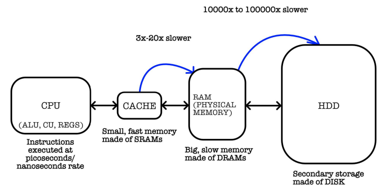
• **PROGRAM COUNTER** or **PC**: Address of next instruction to be executed

• **logic** to translate instructions into control signals for data path

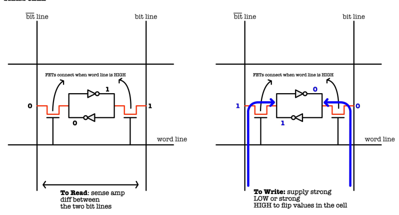
Macro	Definition
BEQ(Ra, label)	BEQ(Ra, label, R31)
BF(Ra, label)	BF(Ra, label, R31)
BNE(Ra, label)	BNE(Ra, label, R31)
BT(Ra, label)	BT(Ra, label, R31)
BR(label, Rc)	BEQ(R31, label, Rc)
BR(label)	BR(label, R31)
JMP(Ra)	JMP(Ra, R31)
LD(label, Rc)	LD(R31, label, Rc)
ST(Rc, label)	ST(Rc, label, R31)
MOVE(Ra, Rc)	ADD(Ra, R31, Rc)
CMOVE(c, Rc)	ADDC(R31, c, Rc)
PUSH(Ra)	ADDC(SP, 4, SP), then ST(Ra, -4, SP)
POP(Rc)	LD(SP, -4, Rc), then SUBC(SP, 4, SP)
ALLOCATE(k)	ADDC(SP, 4*k, SP)
DEALLOCATE(k)	SUBC(SP, 4*k, SP)

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

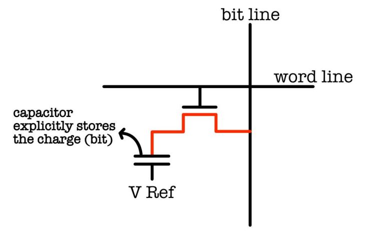

CPU
on READ k (HIT = 1): return DATA(k)
α
on WRITE k (HIT = 1): (part 1) change DATA(k) = data
V
TAG
DATA
1
A
Mem[A]
1
B
Mem[B]
CACHE
on READ k (HIT = 0): Read Mem[X] Set TAG(k) = X DATA(k) = Mem[X]
(1 − α)
on WRITE k (HIT = 1): part(2) start write data to Mem[Tag(k)]
on WRITE k (HIT = 0): start write data to Mem[X] Set Tag(k) = X Data(k) = Mem[x] (write-through strategy)
PHYSICAL MEMORY


10000x to 100000x slower
3x-20x slower
CPU
(ALU, CU, REGS)
Instructions executed at picoseconds/ nanoseconds rate
CACHE
Small, fast memory made of SRAMs
RAM (PHYSICAL MEMORY)
Big, slow memory made of DRAMs
HDD
Secondary storage made of DISK

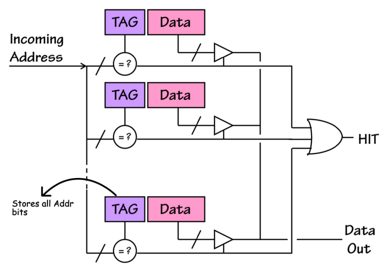
SRAM CELL


bit line
bit line
FETs connect when word line is HIGH
word line
To Read: sense amp diff between the two bit lines
To Write: supply strong LOW or strong HIGH to flip values in the cell

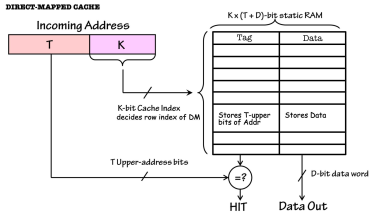
DRAM CELL



FULLY ASSOCIATIVE CACHE

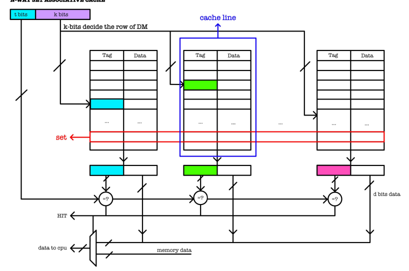


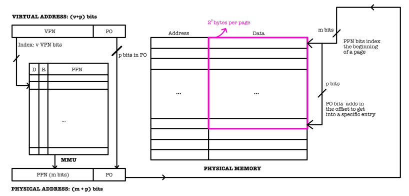
DIRECT-MAPPED CACHE

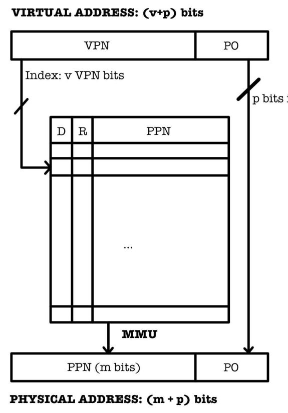

Incoming Address
T
K
K x (T + D)-bit static RAM
Tag
Data
K-bit Cache Index decides row index of DM
Stores T-upper bits of Addr
Stores Data
T Upper-address bits
=?
D-bit data word
HIT
Data Out

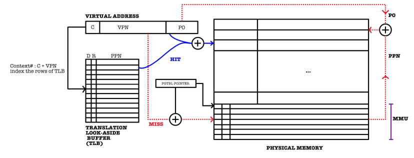
	FA Cache	DM Cache
Tag Content	All address bits	Lower t address bits
Tag Index	None	Higher k address bits
Data	Mem[A]	Mem[A]
Performance	The gold standard on how well a cache should perform	Performs slower than FA cache on average
Contention	Does not have address contention. Address-data can fit on any cache line.	Has address contention. **The probability of contention is inversely proportional to cache size.**
Cost	Expensive, many boolean circuits for comparators	Cheap, only one comparator in a cache
Replacement Strategy	Yes: LRU, FIFO, Random	No
Mapping	No available cache line mapping, must check all cache lines and compare the entire address	Fast mapping, lower k-bits for cache line index, and then compare upper t-bits for tag content
Application	Good when cache size is small, less important when cache size is large	Bad when cache size is small (more contention), good when cache size is large

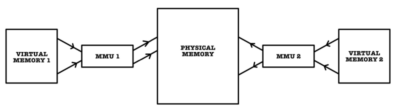
N-WAY SET ASSOCIATIVE CACHE

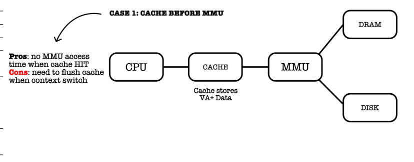


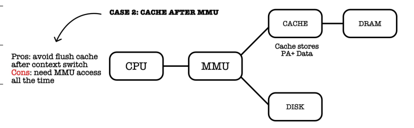

VIRTUAL ADDRESS: (v+p) bits
VPN
PO
Index: v VPN bits
p bits in PO
D R PPN
MMU
PPN (m bits)
PO
PHYSICAL ADDRESS: (m + p) bits
Address
Data
2^p bytes per page
m bits
PPN bits index the beginning of a page
p bits
PO bits adds in the offset to get into a specific entry
PHYSICAL MEMORY



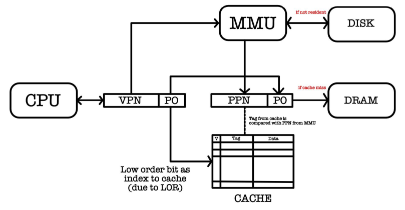

VIRTUAL ADDRESS
C
VPN
PO
Context# : C + VPN index the rows of TLB
D R
PPN
HIT
MISS
TRANSLATION LOOK-ASIDE BUFFER (TLB)
PPN
MMU
PHYSICAL MEMORY




CASE 1: CACHE BEFORE MMU
Pros: no MMU access time when cache HIT
Cons: need to flush cache when context switch
CPU
CACHE
Cache stores VA+ Data
MMU
DRAM
DISK


CASE 2: CACHE AFTER MMU
Pros: avoid flush cache after context switch
Cons: need MMU access all the time
CPU
MMU
CACHE
Cache stores PA+ Data
DRAM
DISK

CASE 3: THE BEST OF BOTH WORLDS


MMU
if not resident
DISK
CPU
VPN
PO
PPN
PO
if cache miss
DRAM
Tag from cache is compared with PPN from MMU
V
Tag
Data
Low order bit as index to cache (due to LOR)
CACHE