Chapters 1, 2, 4, 5, & 6
Textbook: Computer Organization and Design: The Hardware Software Interface: ARM Edition
, by D. A. Patterson and J. L. Hennessy, Morgan Kaufmann, ISBN-13: 9780128017333

8 GREAT IDEAS

-Moore's Law– Exponential growth
-Abstraction – Using only necessary details, subtracting low-level details
-**Common Case** – Make the usual program activities the most Important/Fastest.
-**Pipeline** – Series of Instructions
-**Parallelism** – Multiple tasks at once.
-**Prediction** – Operating on the Common Cases, before told to.
-**Hierarchy of Memory**- (Memory Triangle [More Speed higher on Triangle])
-**Dependability** via **Redundancy** –

-CISC vs RISC – (Complex vs Reduced) Instruction Set Computer
Application Software – High-level Language
System Software
-Compiler- translate HLL code to machine code
-Operating Systems: service code

Components of Computers:
1. Input 2. Output 3.Memory (main (DRAM) , cache (SRAM)
Processor = (4.Datapath 5. Control)

Response time vs Throughput
-**Response time** = How long it takes for 1 task (Execution Time)
-**Throughput** = Total work done per unit time (something per hour)

-Performance = Execution time B / Execution time A
-CPU Clock cycles (Hardware) = Instructions (of a Program) * Avg. Clock cycles per instruction
-Clock Cycles per Instruction (CPI) (Software)- Avg. # of Clock Cycles per instruction for a program or module. (Lower = Better)

EX) Computer A has clock cycle time of 250 ps and a CPI of 2.0 for some program, and
Computer B has clock cycle time of 500 ps and a CPI of 1.2 for some program
Which computer is faster for this program and by how much?
CPU Time(A) = Instruction Count x CPI x Cycle Time
= I * 2.0 * 250 ps = I * 500 ps
CPU Time(B) = Instruction Count x CPI x Cycle Time
= I * 1.2 * 500 ps = I * 600 ps
CPU TimeA / CPU TimeB = I * 600 ps / I * 500 ps = Com A 1.2x Faster

---

Instruction Count & CPI
Instruction Count – Instruction set of Architecture / Program / Compiler (1 Python Line = 4 Assembly Language Line)
Average cycles per instruction
-Determined by CPU hardware
Clock Cycles = E(CPI1 x Instruction Count1)

Clock Cycles = 2 x (1) + 1 x (2) + 2 x (3) = 10
For 5 Instructions -> 10/5 = 2.0

In Java = 15 seconds
A new Java Compiler = 0.6 * I (Instruction)
= Increases CPI by 1.1
How fast can we expect the application to run using this new compiler?
15 * 0.6 * 1.1 = 9.9 Seconds of Execution time
A / B = 15 / 9.9 = 1.51 Faster

Execution Time = Instruction Count * CPI * Clock Cycle Time

CLASSWORK Turn it in into Moodle

2 Different Processors = P1 & P2
P1 = 4 GHz Clock rate
= 1 CPI
P2 = 3 GHz Clock Rate
= 1.5 CPI
Which processor has the highest performance expressed in instructions per second?
P1 = 4GHz / 1 CPI = 4 $*10^9$ Instructions per second
P2 = 3 GHz / 1.5 CPI = 2 $*10^9$ Instructions per second

Consider 3 Different processors: P1, P2, P3
P1 = 3 GHz Clock rate
= 1.5 CPI
Instruction Per Second = 2 $*10^9$
P2 = 2.5 GHz Clock rate
= 1 CPI
Instruction Per Second = 2.5 $*10^9$
P3 = 4 GHz Clock rate
= 2.2 CPI
4 / 2.2 =
Instruction Per Second = 1.82 $*10^9$

HIGHEST PERFORMANCE: P2
If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.
P1 = (2 $*10^{9)}$ * 10 = 2 $*10^{10}$ Instructions 10 Seconds / 1.5 = 6.67 Cycles
P2 = (2.5 $*10^{9)}$ * 10 = 2.5 $*10^{10}$ Instructions 10 Seconds / 1 = 10 Cycles
P3 = (1.82 $*10^9$) * 10 = 1.82 $*10^{10}$ Instructions 10 Seconds / 2.2 = 4.55 Cycles

---

RISC vs CISC
RISC = More Instructions per program = Decreases CPI / Clock Cycle Time
CISC = Less Instructions per program = Increases CPI / Clock Cycle Time
RISC = Higher performance

Average CPI review

Instruction Class	CPI	Frequency of Instructions	(Frequency * Instruction) / CPI
A	2	60%	(0.6 * (1.0 x 10^6)) / 2 = 0.3 x 10^6
B	1	10%	(0.1 * (1.0 x 10^6)) / 2 = 0.05 x 10^6
C	2	30%	(0.3 * (1.0 x 10^6)) / 2 = 0.15 x 10^6

Average CPI if Instruction count of 1.0 x 10^6 Instructions

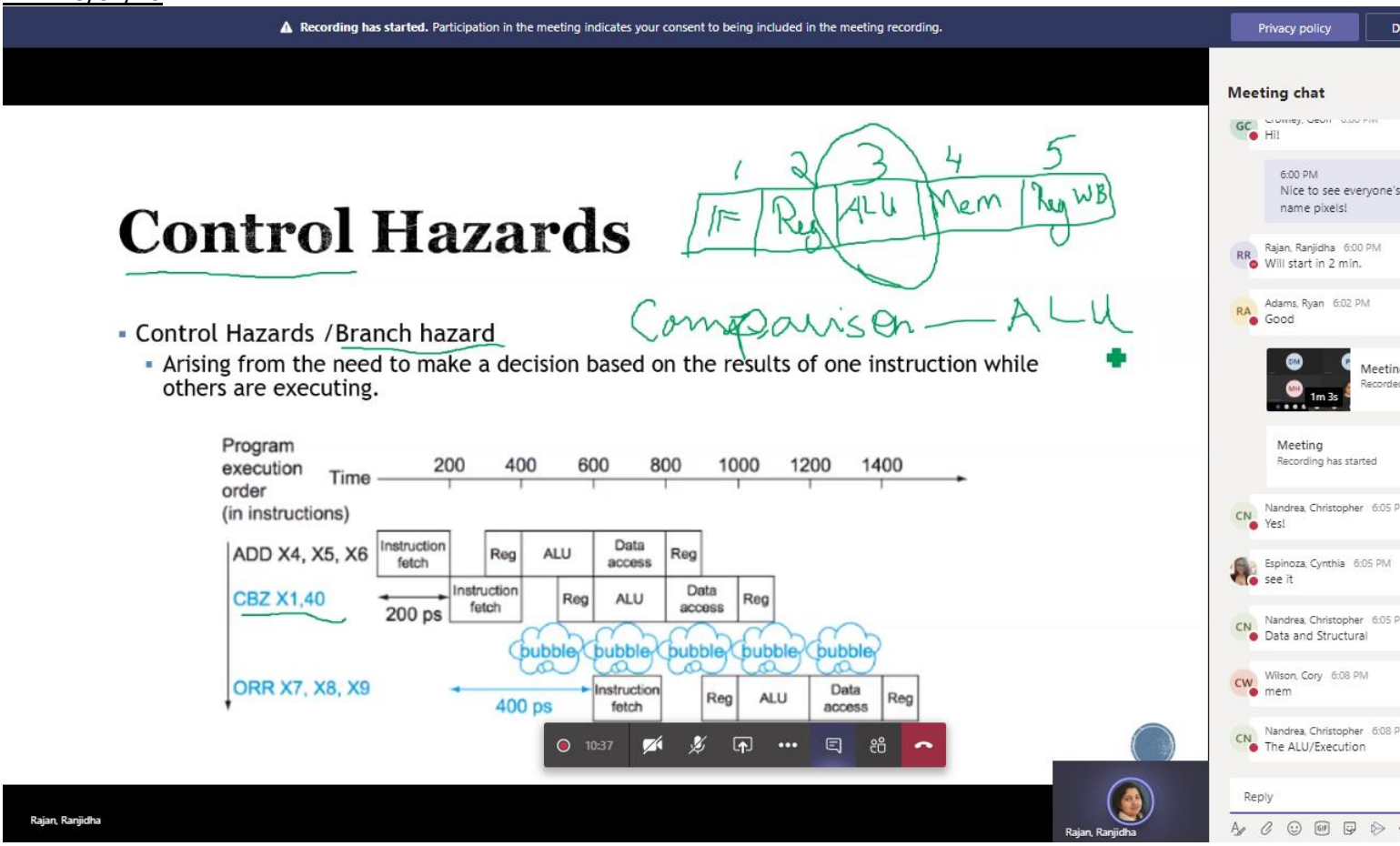
POWER WALL-Higher Clock rate = More Heat = More Electricity (Power)

Procedure- Store subroutines with parameters.
BL = Branch with Link
JUMPS to Specified Registers

Uses:
Link register: (Each Line of Code) Stores the Address of the Callee
Program Counter: Points to Instructions

MOV PC, R14 ; PC = R13/ Link Register=R14 Returns to a particular line of code.
Like a **function Call**. Calls the function , then returns to the last command before Branching.

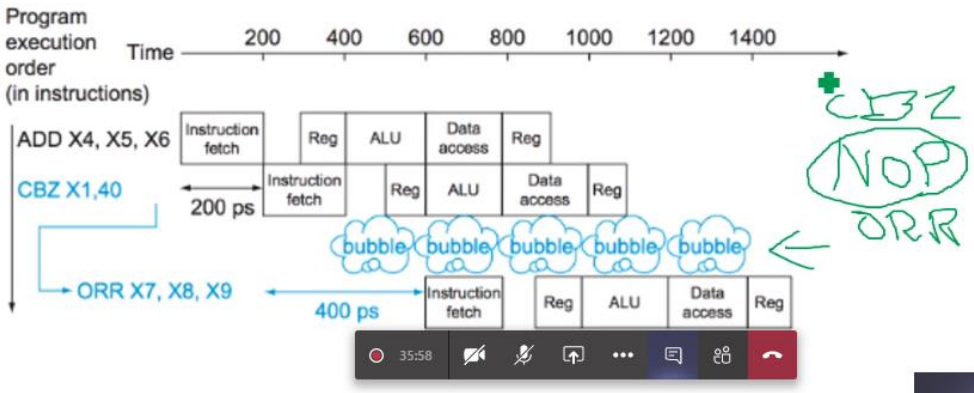
DATE: 3/31/20



Stalling = Inserting Bubbles

Control Hazards

- ***Branch prediction***: A method of resolving a branch hazard that assumes a given outcome for the conditional branch and proceeds from that assumption rather than waiting to ascertain the actual outcome.



PREDICTION = Assume it's not Branching
Implemented on the 3rd Command, once it reaches the ALU
NOP = No Operation, Bubbles

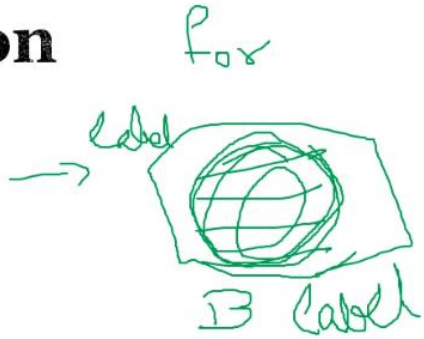
Control Hazard

- One improvement over branch stalling is to **predict** that the conditional branch will not be taken and thus continue execution down the sequential instruction stream. If the conditional branch is taken, the instructions that are being fetched and decoded must be discarded.

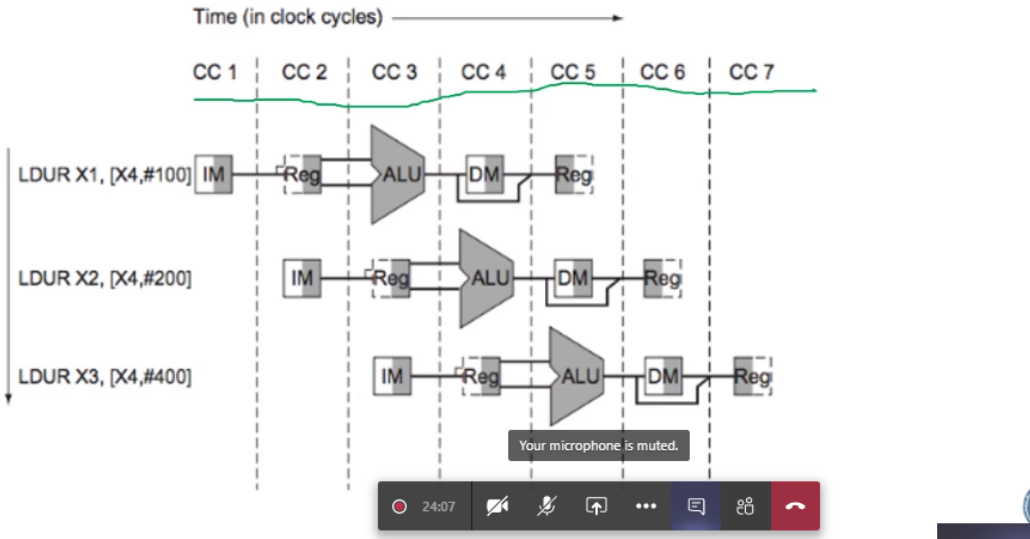
- ***Flush***: To discard instructions in a pipeline, usually due to an unexpected event.

Branch Prediction

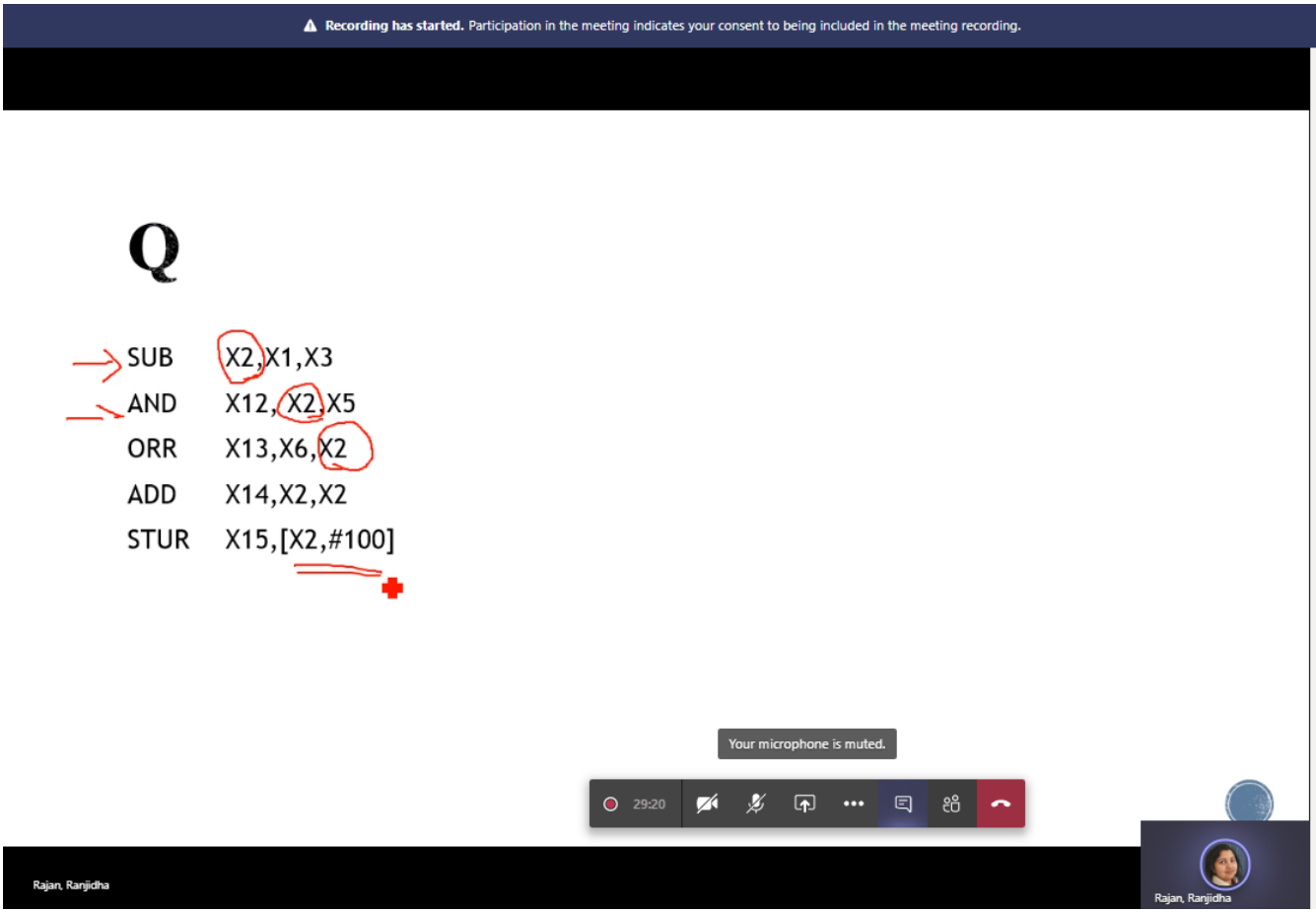
- Static branch prediction
  - Based on typical branch behavior
  - Example: loop and if-statement branches
    - Predict backward branches taken
    - Predict forward branches not taken
- Dynamic branch prediction
  - Hardware measures actual branch behavior
    - e.g., record recent history of each branch
  - Assume future behavior will continue the trend
    - When wrong, stall while re-fetching, and update history



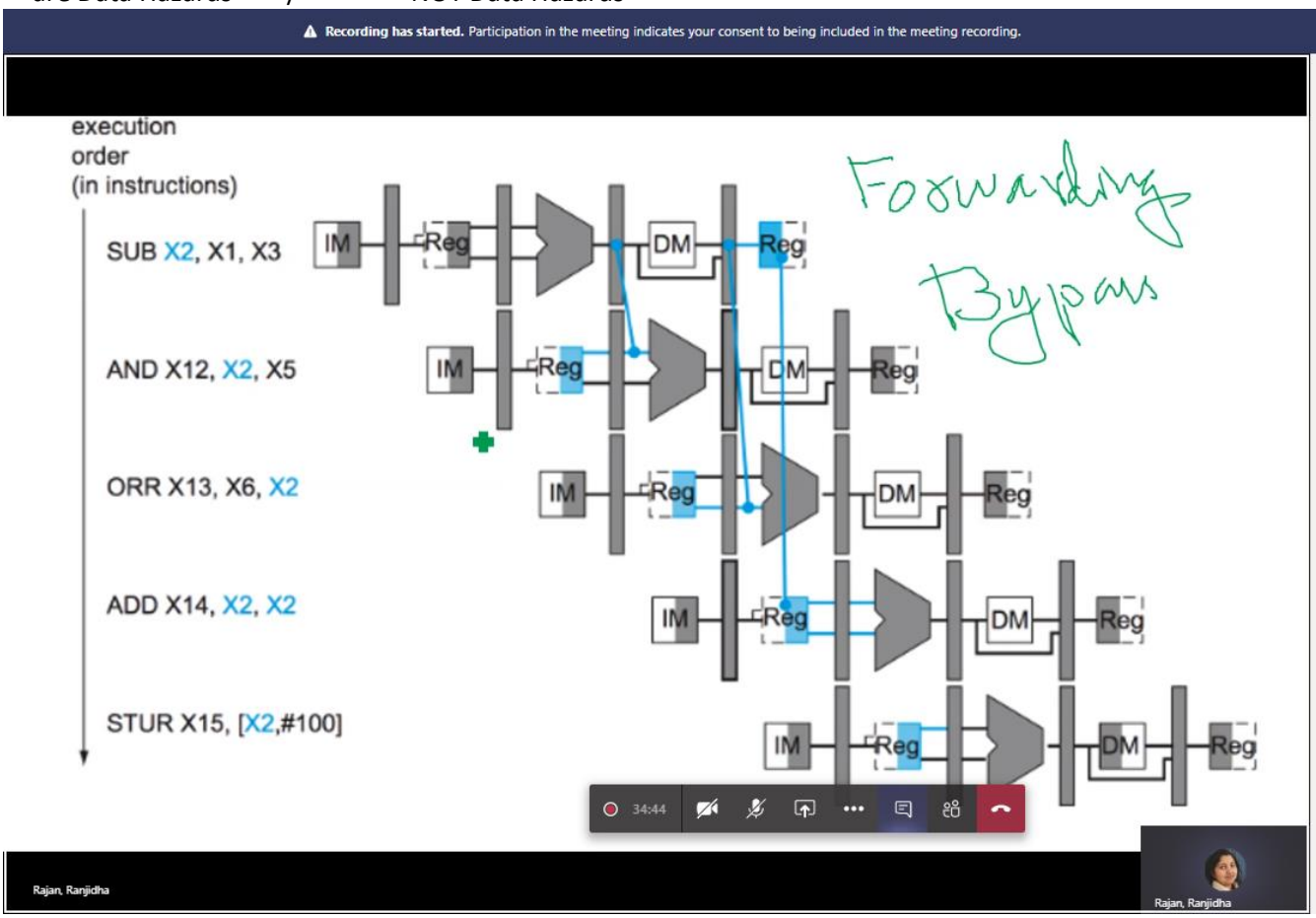
Pipeline Datapath



Instruction Fetch, Instruction Decode, Execution, Memory, Register WriteBack
Draw CC (Clock Cycles)



X2 is a Data Hazard
USES registers for 3 executions & then RELEASES
<- are Data Hazards / -> NOT Data Hazards

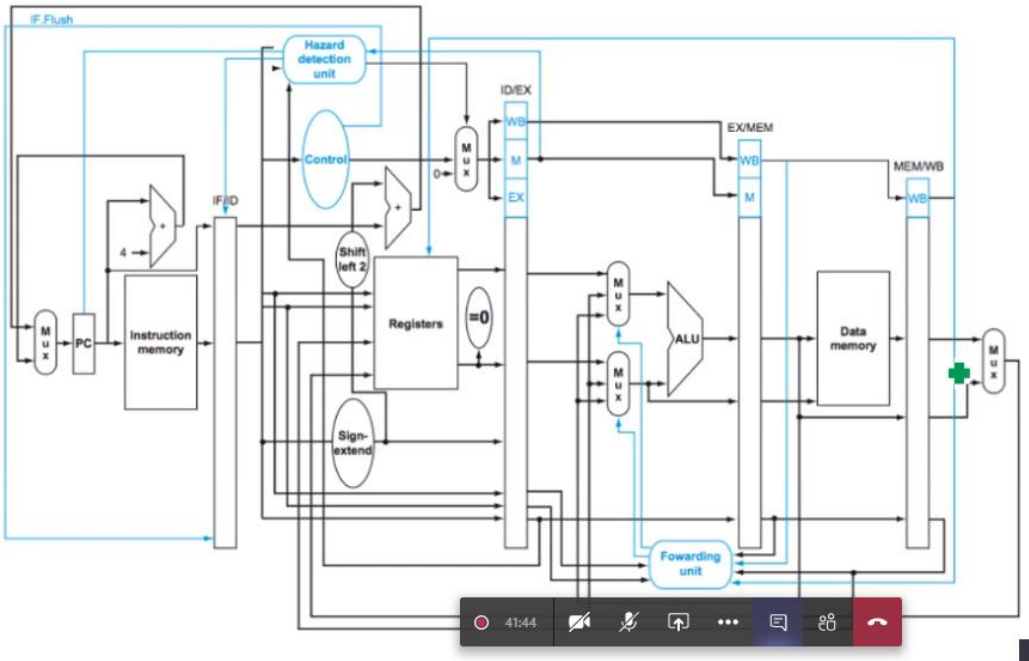


Forwarding / ByPassing / NOP (Bubbling)

Latency

- ***Latency*** (***pipeline***): The number of stages in a pipeline or the number of stages between two instructions during execution.

Latency = Time Taken



EXCEPTIONS

- ***Exception***: Also called ***interrupt***. An unscheduled event that disrupts program execution.
- ***Interrupt***: An exception that comes from outside of the processor.

Rajan, Ranjidha

Recording has started. Participation in the meeting indicates your consent to being included in the meeting recording.
Privacy policy

Exception Types

Type of event	From where?	ARMv8 terminology
System reset	External	Exception
I/O device request	External	Interrupt
Invoke the operating system from user program	Internal	Exception
Floating-point arithmetic overflow or underflow	Internal	Exception
Using an undefined instruction	Internal	Exception
Hardware malfunctions	Either	Exception or interrupt

Rajan, Ranjidha

Meeting chat

explain Static branch prdi

KH Hjelm, Kali 6:24 PM
Crowley, Geoff could you please mute, the eating s are being picked up

PS Safai-Rad, Parviz 6:27 PM
I dont think so

PP Panth, Prashant 6:27 PM
in line 2

CM Mechem, Christian 6:27 PM
X2 will have a data hazard

JB James Bierschwale 6:27 PM
Isn't there a data hazard the sub and and

KH Hjelm, Kali 6:27 PM
yes, there's a data hazard

PS Safai-Rad, Parviz 6:27 PM
yup line 2

CG Gamble, Cory 6:27 PM
and statement

CW Wilson, Cory 6:32 PM
so we should do this inste bubbles

CW Wilson, Cory 6:34 PM
gotcha, thanks!

Reply

Causes

Undefined instruction
Floating-point overflow and underflow
A hardware malfunction

Action

The processor must save the address of the unfortunate instruction in the *exception link register* (ELR) and then transfer control to the operating system at some specified address.

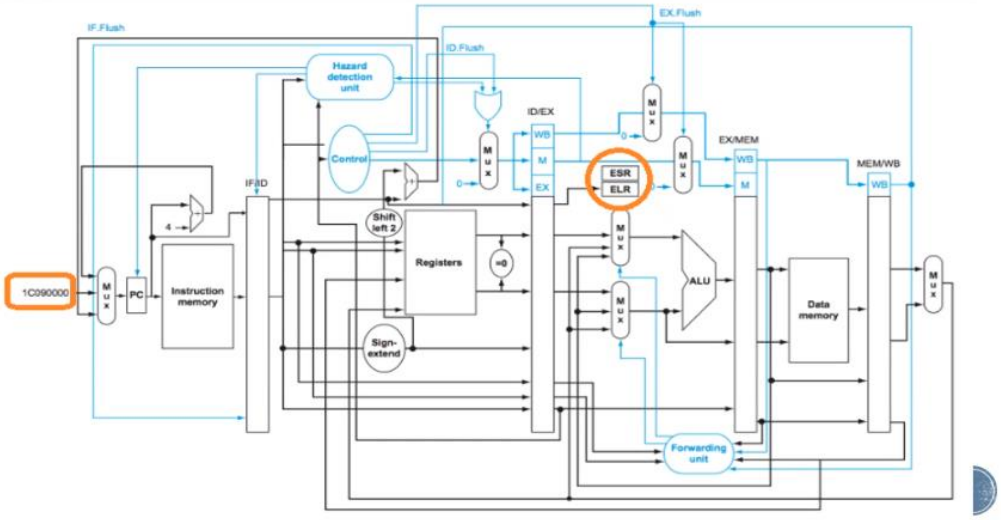
Exceptions Handling

- Operating system to handle the exception, it must know
  - The reason for the exception .
  - The instruction that caused it .
  - Register (called the *Exception Syndrome Register or ESR*), which holds a field that indicates the reason for the exception.
  - *Vectored interrupts*

Exception type	Exception vector address to be added to a Vector Table Base Register
Unknown Reason	$00\ 0000_{two}$
Floating-point arithmetic exception	$10\ 1100_{two}$
System Error (hardware malfunction)	$10\ 1111_{two}$

Exceptions Handling

- *ELR*: A 64-bit register used to hold the address of the affected instruction.
- *ESR*: A register used to record the cause of the exception. In the LEGv8 architecture, this register is 32 bits.



Recording has started. Participation in the meeting indicates your consent to being included in the meeting recording.
Privacy policy

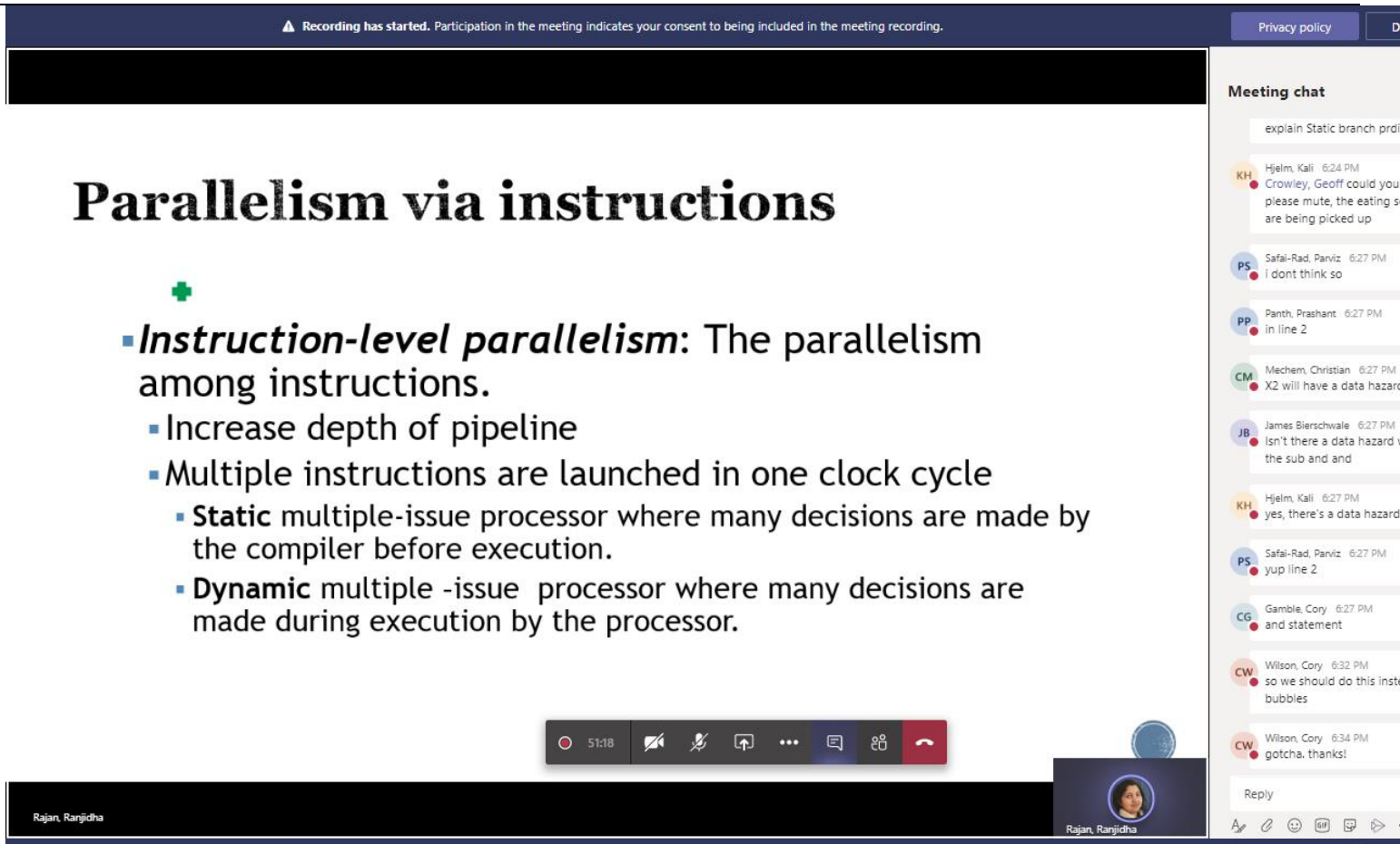
Q

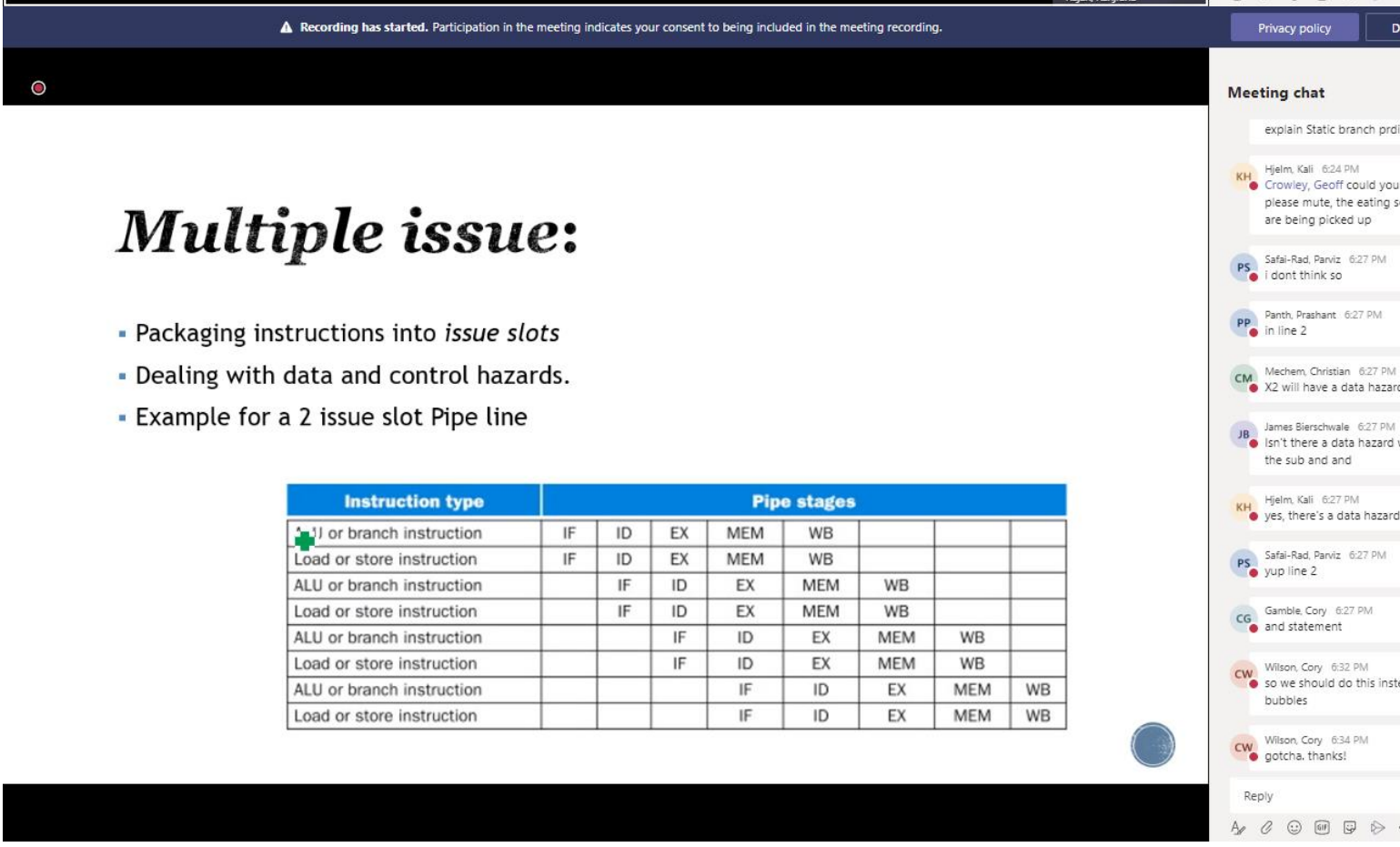
A five-stage pipeline (IF, ID, EX, MEM, WB) executes the following instruction sequence:
Which will be recognized first ?

```
XXX X1, X2, X1 // undefined instruction
SUB X1, X2, X1 // hardware error
```

Your microphone is muted.

Meeting chat
explain Static branch prdi
KH Hjelm, Kali 6:24 PM
Crowley, Geoff could you
please mute, the eating s
are being picked up
PS Safai-Rad, Parviz 6:27 PM
i dont think so
PP Panth, Prashant 6:27 PM
in line 2
CM Mechem, Christian 6:27 PM
X2 will have a data hazard
JB James Bierschwale 6:27 PM
Isn't there a data hazard
the sub and and
KH Hjelm, Kali 6:27 PM
yes, there's a data hazard
PS Safai-Rad, Parviz 6:27 PM
yup line 2
CG Gamble, Cory 6:27 PM
and statement
CW Wilson, Cory 6:32 PM
so we should do this inst
bubbles
CW Wilson, Cory 6:34 PM
gotcha, thanks!
Reply





Instruction type	Pipe stages							
ALU or branch instruction	IF	ID	EX	MEM	WB			
Load or store instruction	IF	ID	EX	MEM	WB			
ALU or branch instruction		IF	ID	EX	MEM	WB		
Load or store instruction		IF	ID	EX	MEM	WB		
ALU or branch instruction			IF	ID	EX	MEM	WB	
Load or store instruction			IF	ID	EX	MEM	WB	
ALU or branch instruction				IF	ID	EX	MEM	WB
Load or store instruction				IF	ID	EX	MEM	WB

Do question in MOODLE.
In Assignment Section? Submit as ASM file or whatever .s file??

Test

Do description in Moodle / Open Book

EXAM 2: @ 6pm-8pm

Exam 2 (Total 70 points)

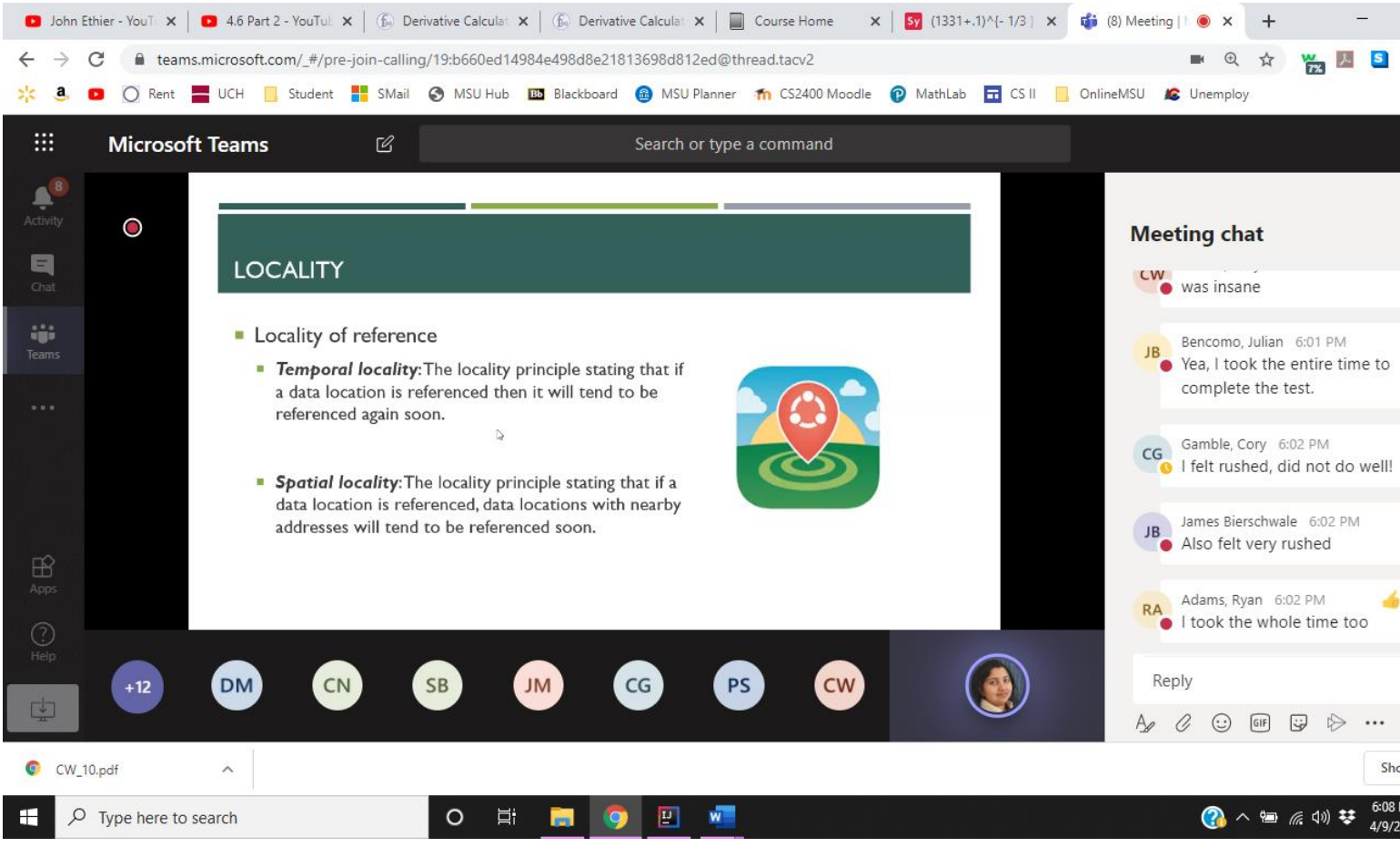
20 points take home program
Will be posting in Moodle by Friday (04/03/2020) 12 Noon
Submit the program as ARM assembly file in Moodle by Saturday(04/04/2020) Midnight.

50 points for online exam Tuesday (04/07/2020) through Moodle
Time duration 2 hours
Programming sections from Arrays and Chapter 4 (4.1 to 4.9).
Review all worksheets done in class.

THEN IN MOODLE CLASSWORK **AFTER 8PM** 4H gap
~Read Chpt 5.1-5.2 Intro/Memory Technologies

Programming Exam:
USES: Arrays & Stack

Online Exam:
Processors
Chapter 4.1-4.9



WORKSHEET Q2

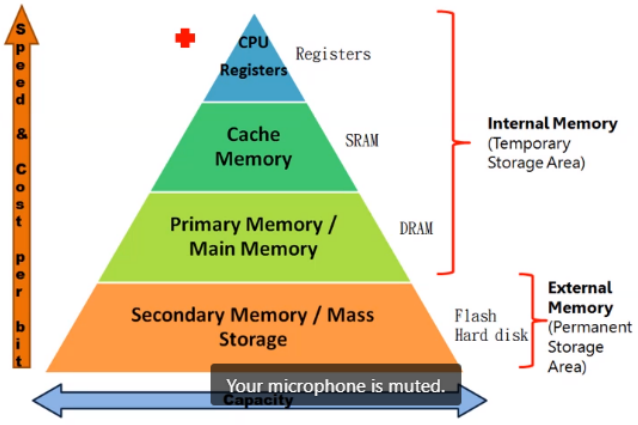
1. Given the following loop, the high likelihood of accessing multiple elements within array A is an example of _____ locality.

```
while (i < 10){
A[i] = A[i] + 2;
i = i + 1; }
```

2.Given the following loop, the high likelihood of accessing `i = i + 1` repeatedly is an example of _____ locality

MEMORY HIERARCHY

- ***Memory hierarchy***: A structure that uses multiple levels of memories; as the distance from the processor increases, the size of the memories and the access time both increase.



MEMORY

- ***Hit rate***: The fraction of memory accesses found in a level of the memory hierarchy.
- ***Miss rate***: The fraction of memory accesses not found in a level of the memory hierarchy.

- ***Hit time***: The time required to access a level of the memory hierarchy, including the time needed to determine whether the access is a hit or a miss.
- ***Miss penalty***: The time required to fetch a block into a level of the memory hierarchy from the lower level, including the time to access the block, transmit it from one level to the other, insert it in the level that experienced the miss, and then pass the block to the requestor.

13:32

Rajan, Ranjidha

Rajan, Ranjidha

Meeting chat

was insane

Bencomo, Julian 6:01 PM
Yea, I took the entire time to complete the test.

Gamble, Cory 6:02 PM
I felt rushed, did not do well!

James Bierschwale 6:02 PM
Also felt very rushed

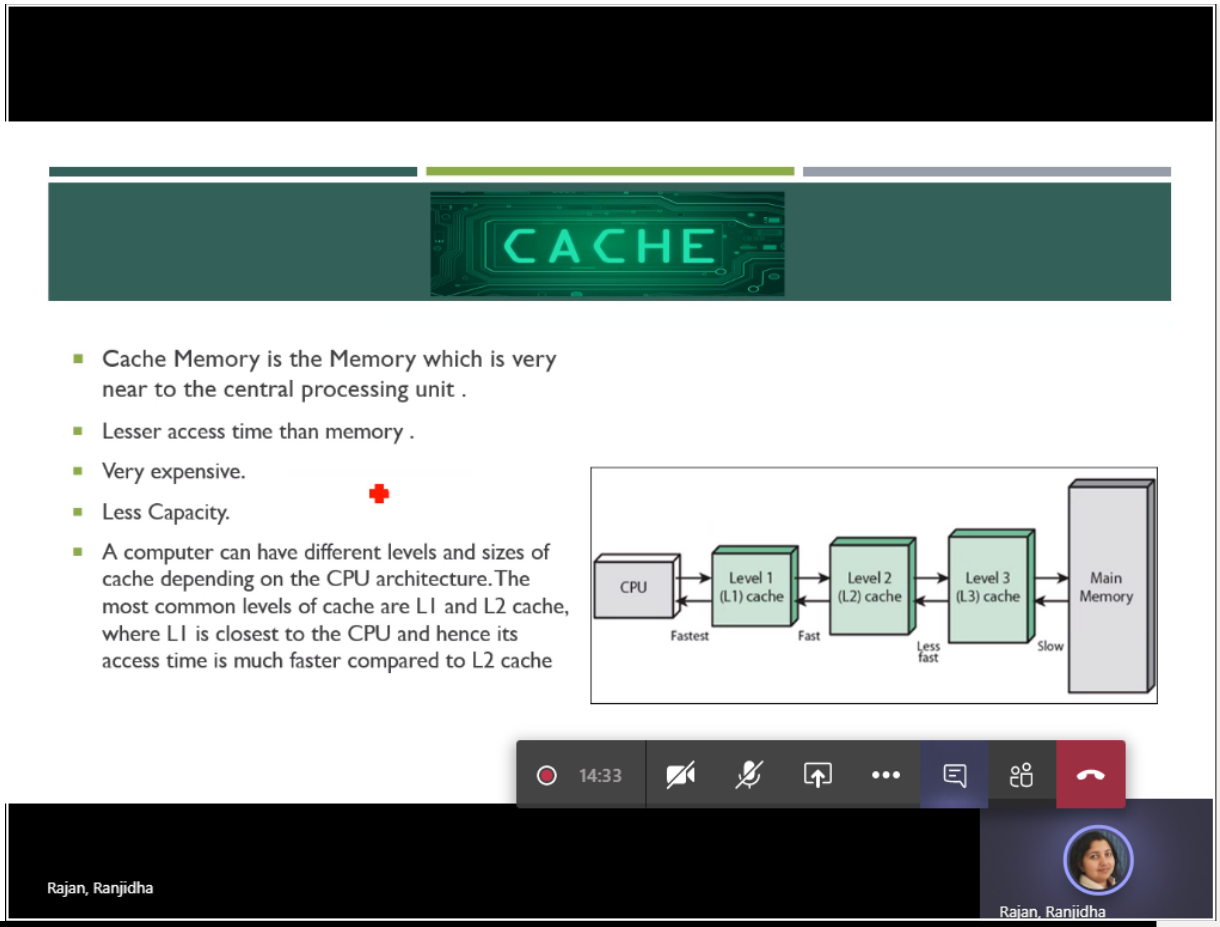
Adams, Ryan 6:02 PM
I took the whole time too

Panth, Prashant 6:04 PM
Yeah too many questions lol

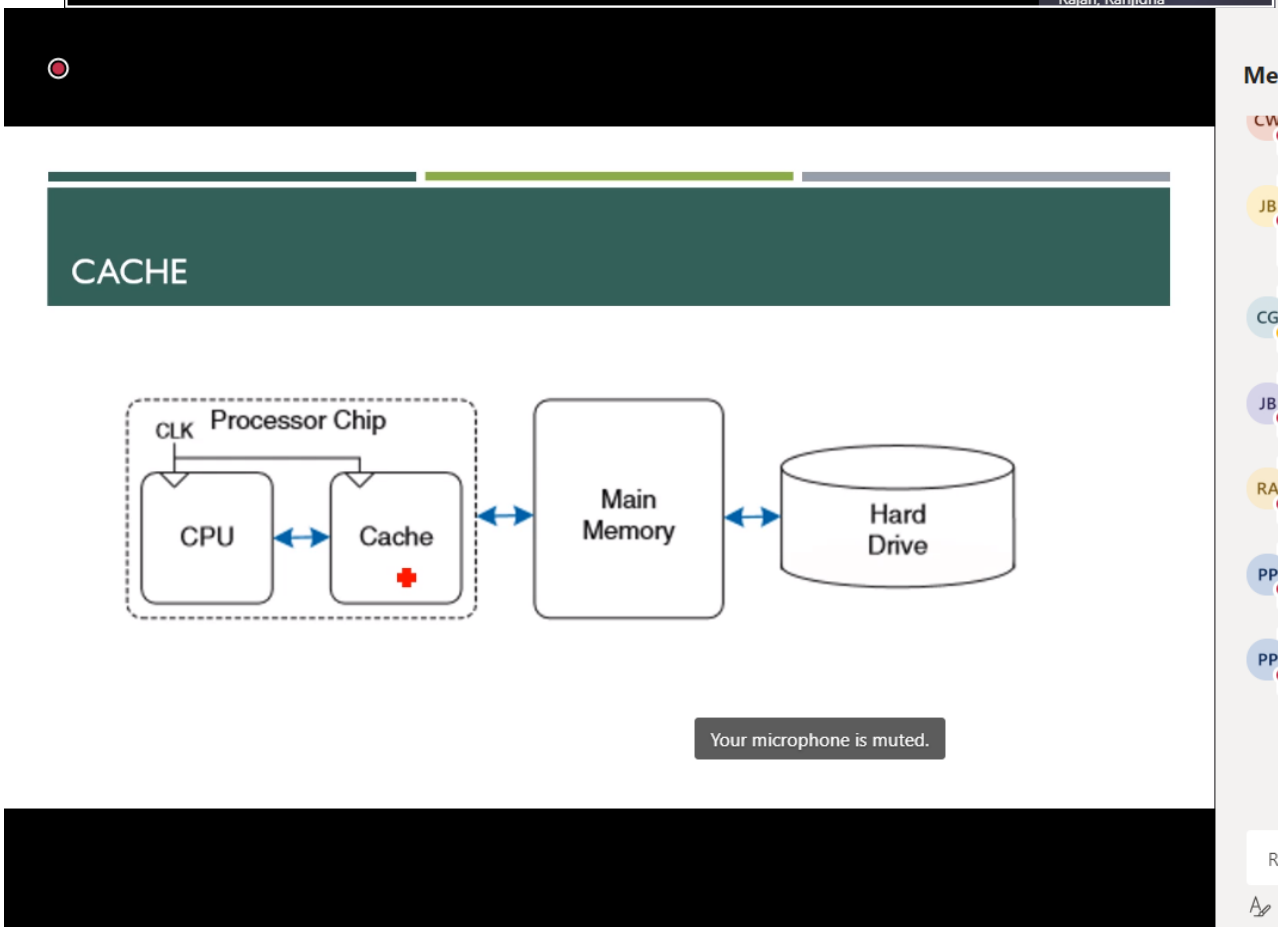
Panth, Prashant 6:04 PM
Yeah those took long

Meeting
Recording has started

Reply


CACHE
Cache Memory is the Memory which is very near to the central processing unit .
Lesser access time than memory .
Very expensive.
Less Capacity.
A computer can have different levels and sizes of cache depending on the CPU architecture. The most common levels of cache are L1 and L2 cache, where L1 is closest to the CPU and hence its access time is much faster compared to L2 cache
CPU
Level 1 (L1) cache
Level 2 (L2) cache
Level 3 (L3) cache
Main Memory
Fastest
Fast
Less fast
Slow
14:33
Rajan, Ranjidha
Rajan, Ranjidha

Meeting chat
was insane
Bencomo, Julian 6:01 PM
Yea, I took the entire time to complete the test.
Gamble, Cory 6:02 PM
I felt rushed, did not do well!
James Bierschwale 6:02 PM
Also felt very rushed
Adams, Ryan 6:02 PM
I took the whole time too
Panth, Prashant 6:04 PM
Yeah too many questions lol
Panth, Prashant 6:04 PM
Yeah those took long
Meeting
Recording has started
Reply


CACHE
CLK
Processor Chip
CPU
Cache
Main Memory
Hard Drive
Your microphone is muted.

Meeting chat
was insane
Bencomo, Julian 6:01 PM
Yea, I took the entire time to complete the test.
Gamble, Cory 6:02 PM
I felt rushed, did not do well!
James Bierschwale 6:02 PM
Also felt very rushed
Adams, Ryan 6:02 PM
2
I took the whole time too
Panth, Prashant 6:04 PM
Yeah too many questions lol
Panth, Prashant 6:04 PM
Yeah those took long
Meeting
Recording has started
Reply

CACHE

- ***Direct-mapped cache***: A cache structure in which each memory location is mapped to exactly one location in the cache.
- Cache index = Block address % Number of Blocks in Cache

- Example
- 8-block cache
- Cache index = Block address % 8

Your microphone is muted.

18:15

Rajan, Ranjidha

Rajan, Ranjidha

CACHE

- ***Index:*** *For 8 cache line =>2^3 ; last 3 bits represent the index.*
- ***Tag***: A field in a table used for a memory hierarchy that contains the address information required to identify whether the associated block in the hierarchy corresponds to a requested word. All bits other than the index is tag.
- ***Valid bit***: A field in the tables of a memory hierarchy that indicates that the associated block in the hierarchy contains valid data.

32:16

Rajan, Ranjidha

Rajan, Ranjidha

Meeting chat

was insane

Bencomo, Julian 6:01 PM
Yea, I took the entire time to complete the test.

Gamble, Cory 6:02 PM
I felt rushed, did not do well!

James Bierschwale 6:02 PM
Also felt very rushed

Adams, Ryan 6:02 PM
I took the whole time too

Panth, Prashant 6:04 PM
Yeah too many questions lol

Panth, Prashant 6:04 PM
Yeah those took long

Meeting
Recording has started

Reply

Meeting chat

Panth, Prashant 6:25 PM
3 and 5 I guess

6:26 PM
8 oneword block size mean lines?

6:26 PM
thank you

Mechem, Christian 6:29 PM
Yes

Isobe, Rin 6:29 PM
yes

Macias, Jasiel Rodriguez 6:29 PM
yes

Wilson, Cory 6:29 PM
thats a good trick haha. i like

6:30 PM
so you could AND the last

Reply

Q2 IN WORKSHEET

Address	Hit or Miss
10110	
11010	
10110	
11010	
10000	
00011	
10000	
10010	
10000	

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

35:15

Rajan, Ranjidha

Rajan, Ranjidha

Meeting chat

PP Panth, Prashant 6:25 PM
3 and 5 I guess

6:26 PM
8 oneword block size mean lines?

6:26 PM
thank you

CM Mechem, Christian 6:29 PM
Yes

RI Isobe, Rin 6:29 PM
yes

JM Macias, Jasiel Rodriguez 6:29 PM
yes

CW Wilson, Cory 6:29 PM
thats a good trick haha. i like i

6:30 PM
so you could AND the last

Reply

(8) Meeting | Microsoft Team

teams.microsoft.com/_#/pre-join-calling/19:b660ed14984e498d8e21813698d812ed@thread.tacv2

Rent UCH Student SMail MSU Hub Blackboard MSU Planner CS2400 Moodle MathLab CS II OnlineMSU Unemploy

Microsoft Teams

Search or type a command

Activity Chat Teams Assignments Apps Help

Q2 IN WORKSHEET

Address	Hit or Miss
10110	
11010	
10110	
11010	
10000	
00011	
10000	
10010	
10000	

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N	10	10110
111	N		

10110 Index

+10 GC IP KH DM CN SB JM CG PS CW

Type here to search

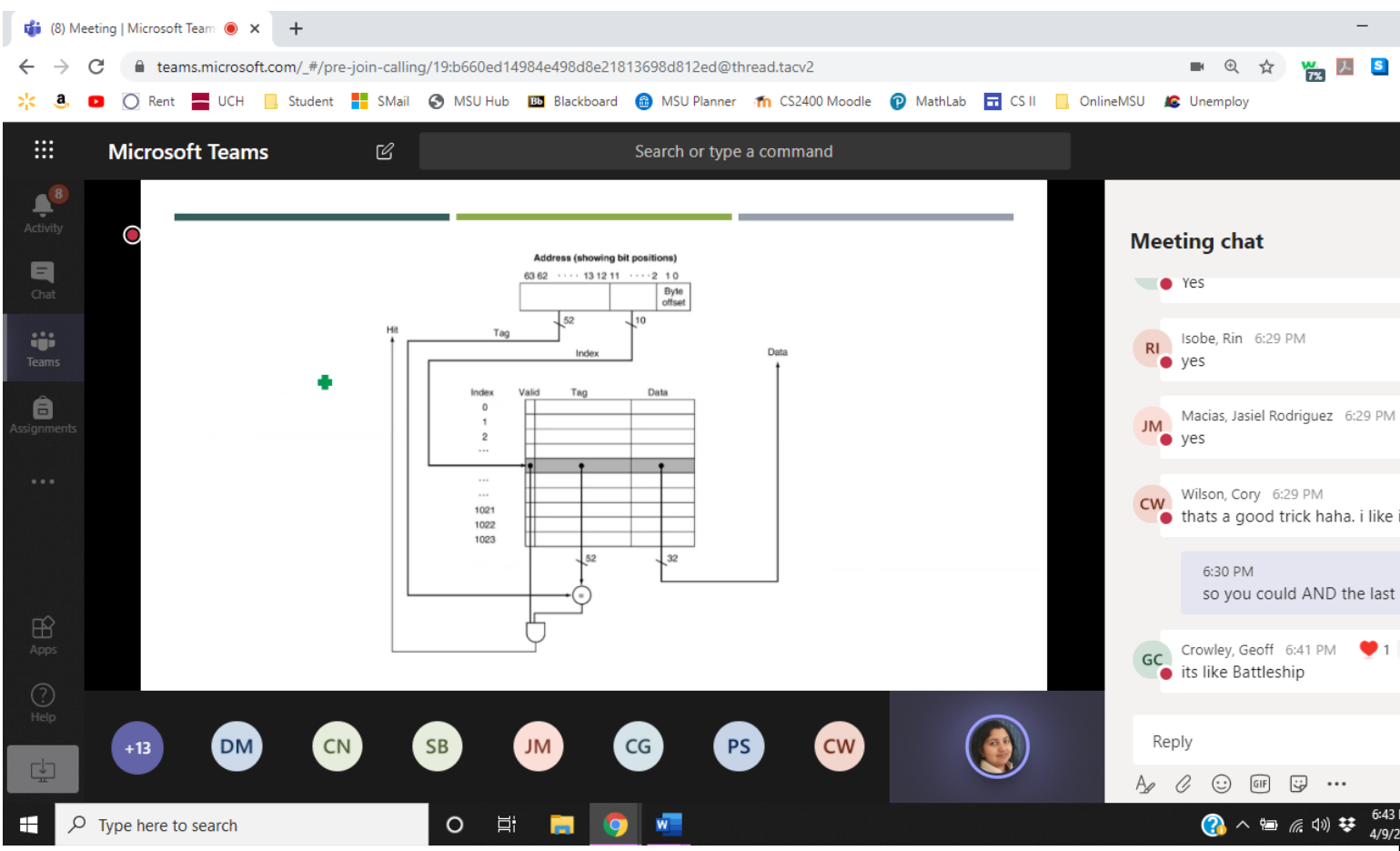
Address	Hit or Miss
10110	M
11010	M
10110	H
11010	H
10000	M
00011	M
10000	
10010	
10000	

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

(8) Meeting | Microsoft Team
teams.microsoft.com/_#/pre-join-calling/19:b660ed14984e498d8e21813698d812ed@thread.tacv2
Rent UCH Student SMail MSU Hub Blackboard MSU Planner CS2400 Moodle MathLab CS II OnlineMSU Unemploy
Microsoft Teams
Search or type a command
Activity Chat Teams Assignments Apps Help
Q2 IN WORKSHEET
Meeting chat
Yes
Isobe, Rin 6:29 PM
yes
Macias, Jasiel Rodriguez 6:29 PM
yes
Wilson, Cory 6:29 PM
thats a good trick haha. i like i
6:30 PM
so you could AND the last
Crowley, Geoff 6:41 PM
its like Battleship
Reply
+13 DM CN SB JM CG PS CW
Type here to search
6:42
4/9/2

Address	Hit or Miss
10110	M
11010	M
10110	H
11010	H
10000	M
00011	M
10000	H
10010	M
10000	H

Index	V	Tag	Data
000	Y	10	10010
001	N		
010	Y	10	10010
011	Y	00	00011
100	N		
101	N		
110	Y	10	10110
111	N		



HANDLING CACHE MISS AND WRITES

- ***Cache miss***: A request for data from the cache that cannot be filled because the data are not present in the cache.
  - Stall
- ***Write-through***: A scheme in which writes always update both the cache and the next lower level of the memory hierarchy, ensuring that data are always consistent between the two.
- ***Write-back***: A scheme that handles writes by updating values only to the block in the cache, then writing the modified block to the lower level of the hierarchy when the block is replaced.
- ***Split cache***: A scheme in which a level of the memory hierarchy is composed of two independent caches that operate in parallel with each other, with one handling instructions and one handling data.

MEASURING AND IMPROVING CACHE PERFORMANCE

- CPU time=(CPU execution clock cycles + Memory-stall clock cycles)×Clock cycle time

- The memory-stall clock cycles come primarily from cache misses.
- Memory-stall clock cycles can be defined as the sum of the stall cycles coming from reads plus those coming from writes:

- Memory-stall clock cycles=(Read-stall cycles + Write-stall cycles)

READS / WRITES

- Reads:
- The read-stall cycles can be defined in terms of the number of read accesses per program, the miss penalty in clock cycles for a read, and the read miss rate:
- Read-stall cycles = Reads-Program × Read miss rate × Read miss penalty

- Writes
- Write-stall cycles = (Writes Program × Write miss rate × Write miss penalty) + Write buffer stalls

MEASURING AND IMPROVING CACHE PERFORMANCE

- Memory-stall clock cycles= $\frac{Memory\ Access}{Program}$ × Miss Rate × Miss penalty
- Memory-stall clock cycles= $\frac{Instructions}{Program} \times \frac{Misses}{Instruction}$ × Miss penalty
- **Q3 in worksheet**
- The miss rate of an instruction cache is 2% and the miss rate of the data cache is 4%. If a processor has a CPI of 2 without any memory stalls, and the miss penalty is 100 cycles for all misses, determine how much faster a processor would run with a perfect cache that never missed. Assume the frequency of all loads and stores is 36%.
- ? Instruction miss cycle , data miss cycle , CPI time (including stall)

MEASURING AND IMPROVING CACHE PERFORMANCE

- Memory-stall clock cycles= $\frac{Memory\ Access}{Program}$ × Miss Rate × Miss penalty
- Memory-stall clock cycles= $\frac{Instructions}{Program} \times \frac{Misses}{Instruction}$ × Miss penalty
- **Q3 in worksheet**
- The miss rate of an instruction cache is 2% and the miss rate of the data cache is 4%. If a processor has a CPI of 2 without any memory stalls, and the miss penalty is 100 cycles for all misses, determine how much faster a processor would run with a perfect cache that never missed. Assume the frequency of all loads and stores is 36%.
- ? Instruction miss cycle , data miss cycle , CPI time (including stall)

SOLUTION

- The number of memory miss cycles for instructions in terms of the Instruction count (I) is
- Instruction miss cycles= I×2%×100=2.00×I
- The frequency of all loads and stores is 36%, we can find the number of memory miss cycles for data references:
- Data miss cycles=I×36%×4%×100=1.44×I

- The total number of memory-stall cycles is 2.00 I + 1.44 I = 3.44 I.
- CPI including memory stalls is 2 + 3.44 = 5.44.
- For a perfect Cache (without stall) = 2

- Performance = CPI_{stall} / $CPI_{perfect}$ = 5.44 / 2 = 2.72

SOLUTION

- The number of memory miss cycles for instructions in terms of the Instruction count (I) is
- Instruction miss cycles= I×2%×100=2.00×I
- The frequency of all loads and stores is 36%, we can find the number of memory miss cycles for data references:
- Data miss cycles=I×36%×4%×100=1.44×I

- The total number of memory-stall cycles is 2.00 I + 1.44 I = 3.44 I
- CPI including memory stalls is 2 + 3.44 = 5.44.
- For a perfect Cache (without stall) = 2

- Performance = CPI_{stall} / $CPI_{perfect}$ = 5.44 / 2 = 2.72

SOLUTION

- The number of memory miss cycles for instructions in terms of the Instruction count (I) is
- Instruction miss cycles= I×2%×100=2.00×I
- The frequency of all loads and stores is 36%, we can find the number of memory miss cycles for data references:
- Data miss cycles=I×36%×4%×100=1.44×I

3.44 I / I

- The total number of memory-stall cycles is 2.00 I + 1.44 I = 3.44 I.
- CPI including memory stalls is 2 + 3.44 = 5.44.
- For a perfect Cache (without stall) = 2
- Performance = CPI_{stall} / $CPI_{perfect}$ = 5.44 / 2 = 2.72

AMAT

- *Average memory access time* (AMAT) as a way to examine alternative cache designs. Average memory access time is the average time to access memory considering both hits and misses and the frequency of different accesses.
- AMAT = Time for a hit + Miss rate × Miss penalty
- Q4 in worksheet ?

ASSOCIATIVE CACHES

- ***Fully associative cache***:
  - A cache structure in which a block can be placed in any location in the cache.
- ***Set-associative cache***:
  - A cache that has a fixed number of locations (at least two) where each block can be placed.

Meeting chat

PP ok. Thank you

CW Wilson, Cory 7:03 PM 👍 1
When we get back from break. could you explain what each variable is and why we plugged them into the equations?

CW Wilson, Cory 7:04 PM
For question #3 in the class work. I got the AMAT one. Thank you!

Meeting
Recording has started

CW Wilson, Cory 7:05 PM
yes. thank yoU!

CW Wilson, Cory 7:10 PM
Yes. That helped. Thank you!

Reply

AMAT

- *Average memory access time* (AMAT) as a way to examine alternative cache designs. Average memory access time is the average time to access memory considering both hits and misses and the frequency of different accesses.
- AMAT = Time for a hit + Miss rate × Miss penalty
- Q4 in worksheet ?
- The average memory access time per instruction is

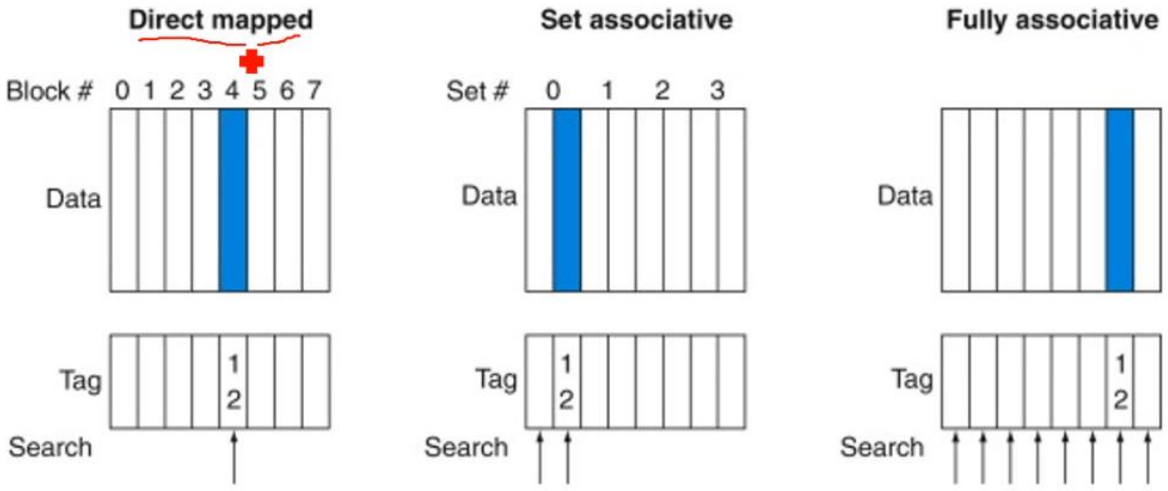
  AMAT = Time for a hit + Miss rate × Miss penalty

  = 1 + 0.05 × 20

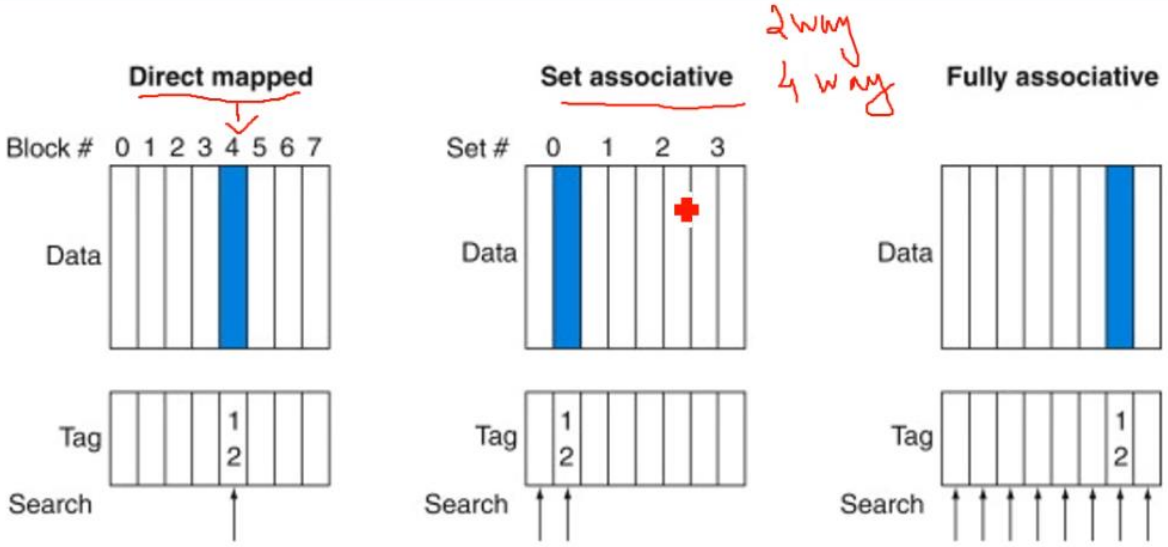
  = 2 clock cycles or 2 ns

Your microphone is muted.

CACHE MAPPING

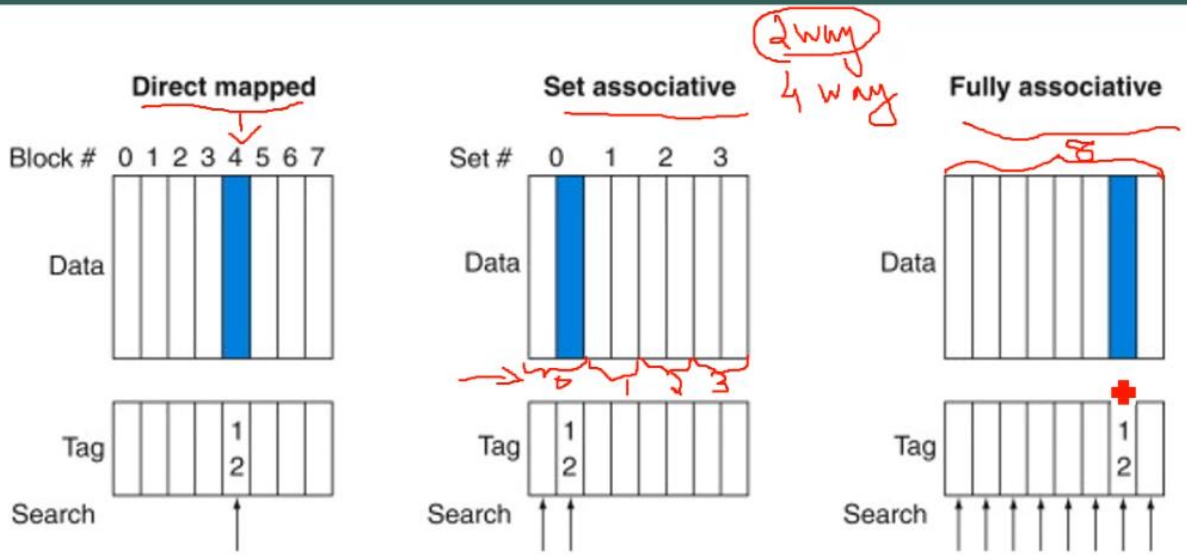


CACHE MAPPING



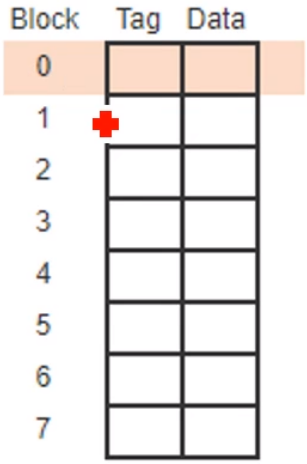
⚠ **Recording has started.** Participation in the meeting indicates your consent to being included in the meeting recording. Privacy policy Dismiss

CACHE MAPPING



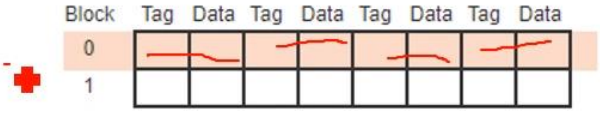
EXAMPLE

- Directed Mapped : Each cache entry holds one block and each set has one element.



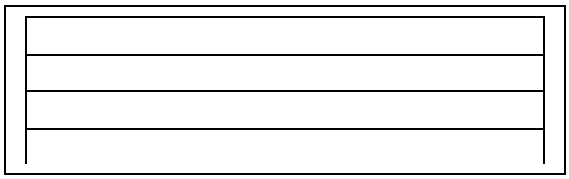
- Increasing the associativity decreases the number of sets while increasing the number of elements per set. A two-way set-associative cache contains four sets; each entry set has two elements.

- A four-way set-associative cache contains two sets; each entry set has four elements.



Q 5

- Assume there are three small caches, each consisting of four one-word blocks.
- One cache is direct-mapped, a second is two-way set-associative, and the third is fully associative .
- Find the number of misses for each cache organization given the following sequence of block addresses: **0, 8, 0, 6, and 8.**



SOLUTION - WORKSHEET Q5

- For Directed Cache Map - 0, 8, 0, 6, and 8.

Block address	Cache block
0	(0 modulo 4) = 0
6	(6 modulo 4) = 2
8	(8 modulo 4) = 0

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3

SOLUTION

- Directed Mapped Cache

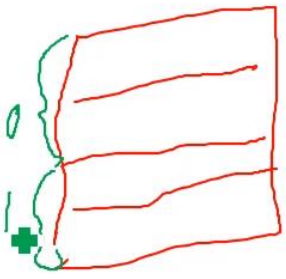
Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		0	1	2	3
0	miss	Memory[0]			
8	miss	Memory[8]			
0	miss	Memory[0]			
6	miss	Memory[0]		Memory[6]	
8	miss	Memory[8]		Memory[6]	

SOLUTION

- Set Associativity -2 way - 0, 8, 0, 6, and 8.

Block address	Cache set
0	(0 modulo 2) = 0
6	(6 modulo 2) = 0
8	(8 modulo 2) = 0

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Set 0	Set 0	Set 1	Set 1
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[6]		
8	miss				



Replacement Policy

Least recently used (***LRU***): A replacement scheme in which the block replaced is the one that has been u
the longest time.

SOLUTION

- Fully Associative

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[8]	Memory[6]	
8	hit	Memory[0]	Memory[8]	Memory[6]	

SOLUTION

- Fully Associative

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[8]	Memory[6]	
8	hit	Memory[0]	Memory[8]	Memory[6]	

SOLUTION - WORKSHEET Q5

- For Directed Cache Map - 0, 8, 0, 6, and 8.

Block address	Cache block
0	(0 modulo 4) = 0
6	(6 modulo 4) = 2
8	(8 modulo 4) = 0

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3

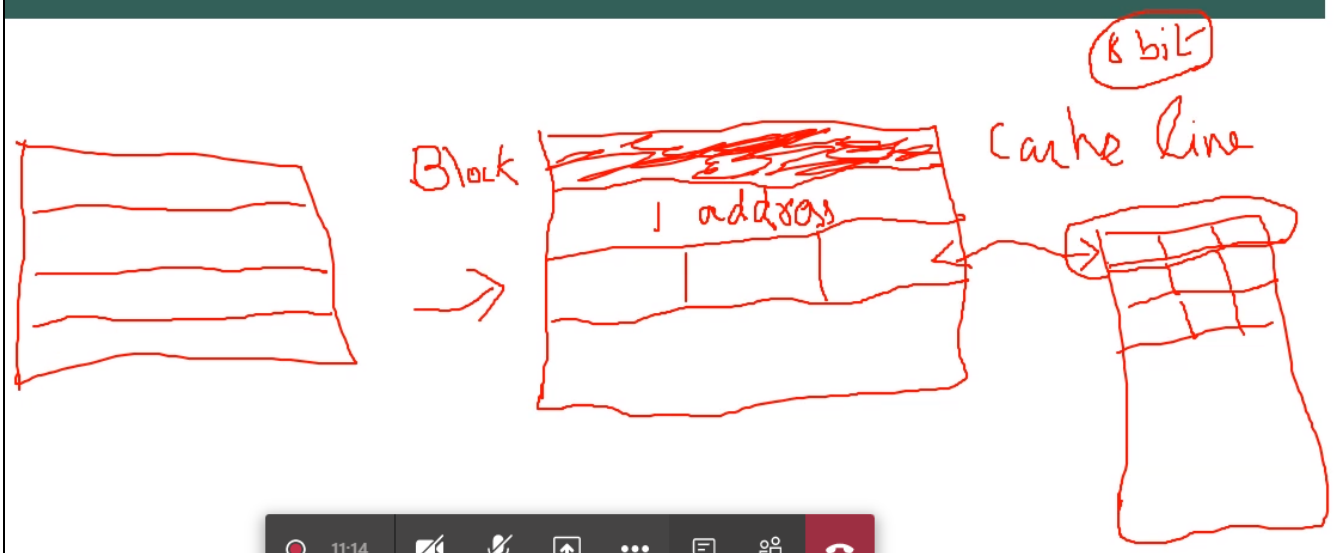
4/14/20

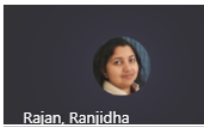
Index / Offset Bits

nx

8 Bit addressing Mode (Can use: Byte || 16,32,64 Bit Addressing Modes)

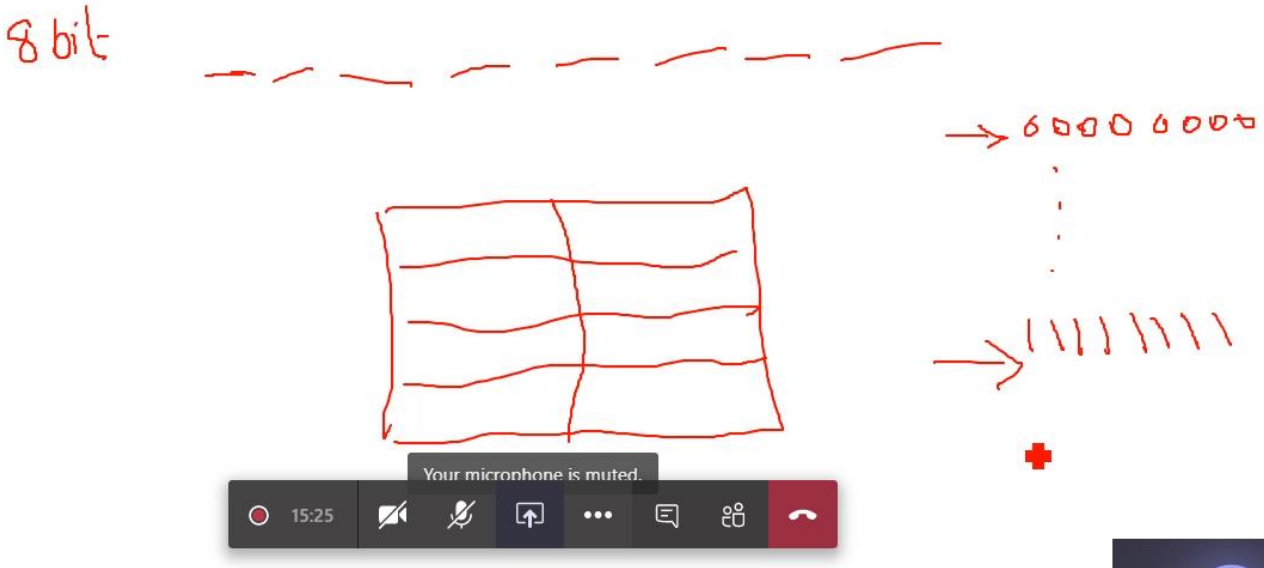
-Copies All Addresses in 1 line. Ex) 3 Addresses in 1 line



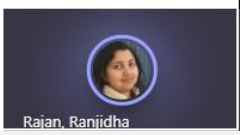


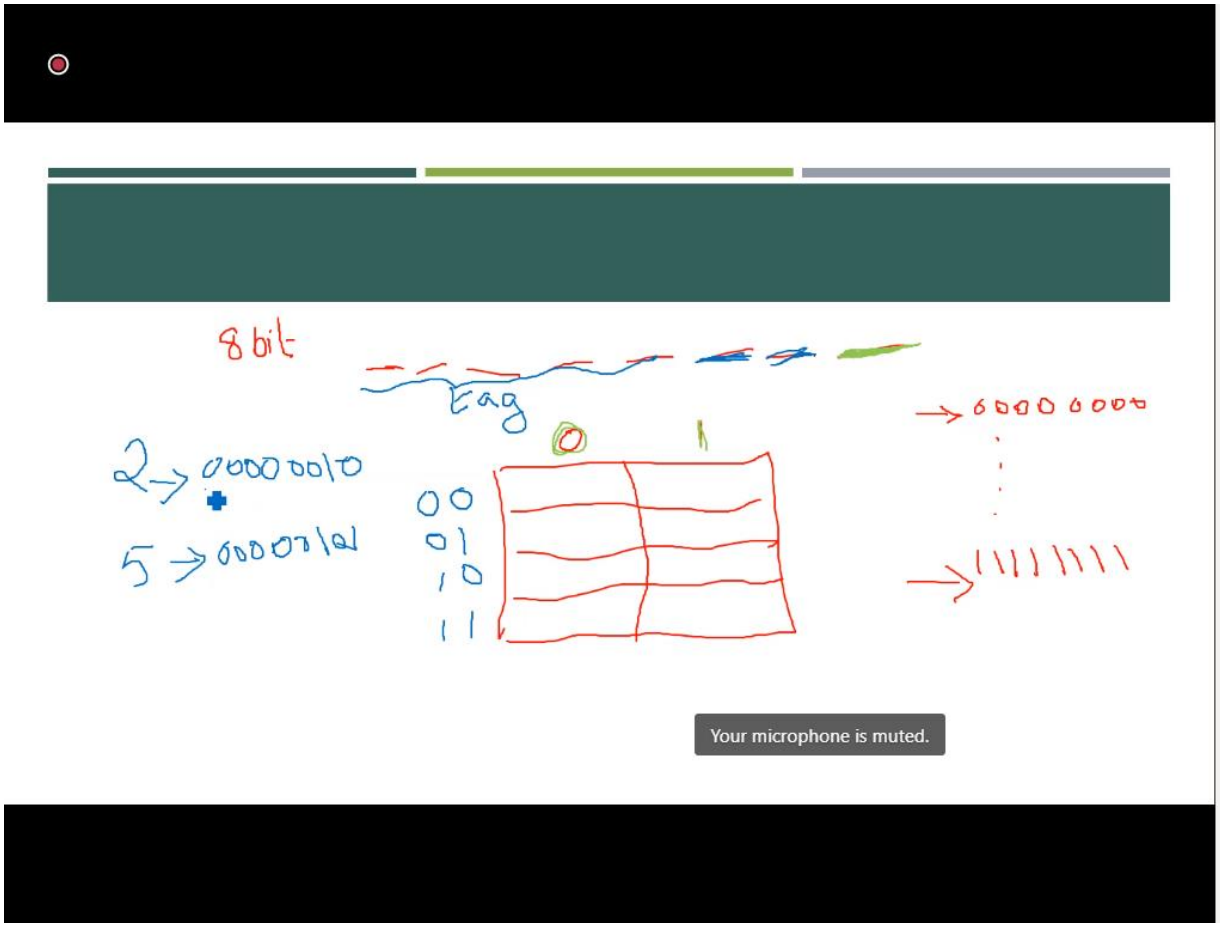
Rajan, Ranjidha

00000000 = Lowest Address
11111111 = Highest Address



Rajan, Ranjidha





Meeting chat

2 bits

6:17 PM
2

Tovar, Efren Paredes 6:17 PM
2

Wilson, Cory 6:18 PM
So the offset is always represented by that last bit in green? Just clarifying

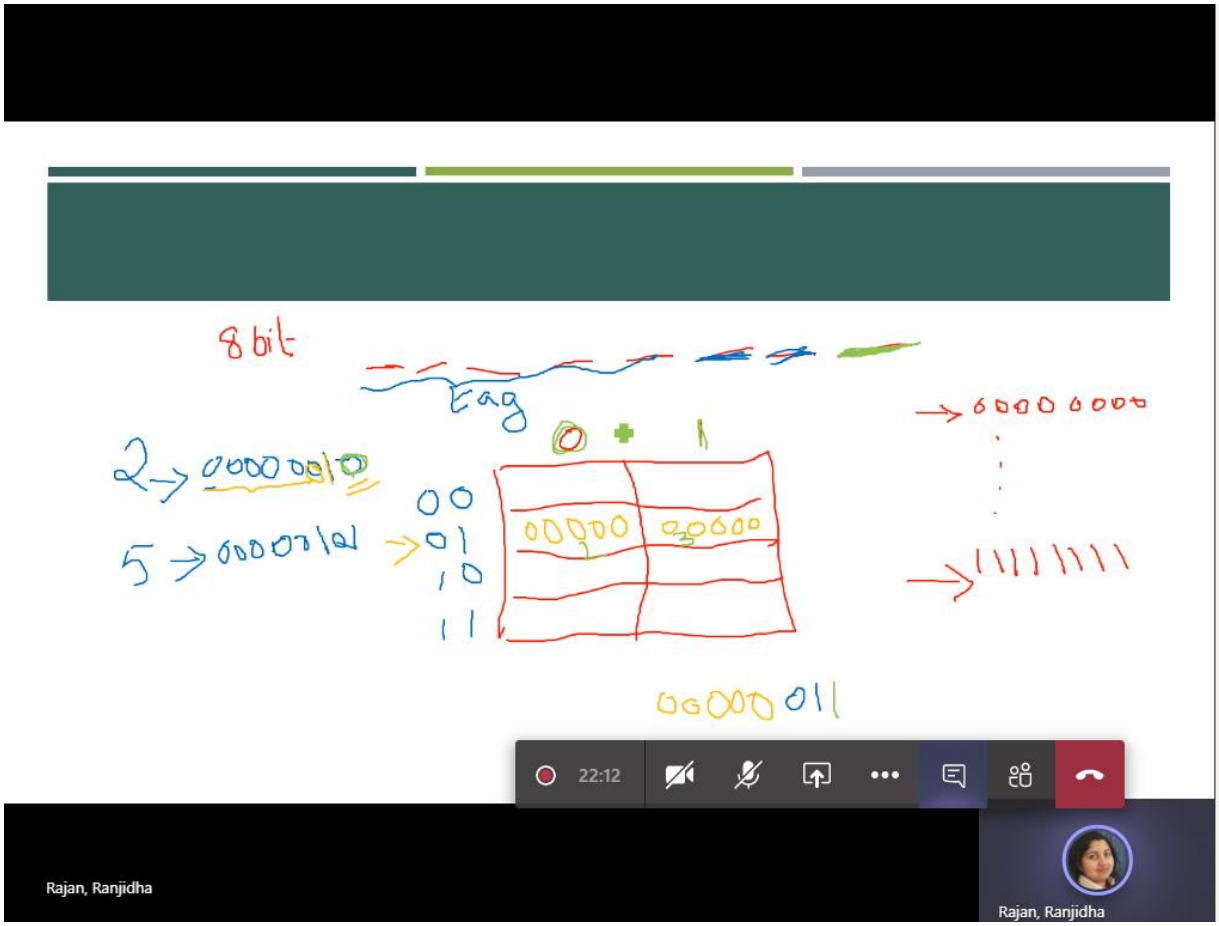
Wilson, Cory 6:18 PM
gotcha. Thank you!

6:19 PM
So Tag, then index, then offset?

6:19 PM
10

Hjelm, Kali 6:19 PM
0010

Reply



Meeting chat

2

ET Tovar, Efren Paredes 6:17 PM
2

CW Wilson, Cory 6:18 PM
So the offset is always represented by that last bit in green? Just clarifying

CW Wilson, Cory 6:18 PM
gotcha. Thank you!

6:19 PM
So Tag, then index, then offset?

6:19 PM
10

KH Hjelm, Kali 6:19 PM
0010

CW Wilson, Cory 6:22 PM
3

Reply

Q 2

- Additional specs for the 16 Kbyte cache include: - Each block will hold 32 bytes of data .The cache would be 2-way set associative. Data is addressed to the word are 32 bits.

- How many blocks would be in this cache? **512**
- How many sets **256**
- What is Tag, Index and offset ? **21, 8, 3**

49:30

Rajan, Ranjidha

Rajan, Ranjidha

Meeting chat

blocks 512
2 way associative 256
Index 8
tag 21

See less

CM Mechem, Christian 6:44 PM

ET Tovar, Efren Paredes 6:45 PM
yes

CG Gamble, Cory 6:46 PM
For #2, why is the tag 21? Thought it was the offset originally

CW Wilson, Cory 6:49 PM
can you go to the last slide really quick please?

CW Wilson, Cory 6:49 PM
ok thank you!@

Reply

TAG, INDEX, OFFSET BIT , Q 1

- If we have a standard 32 bit address machine with 4GB RAM .A 4 way set associative CPU data Cache that uses 32 Byte blocks. How many bits will be used for Tag, Index and offset.
- Total size of cache is 16 KiB.

- 32 Byte Blocks –> 3 bits for offset
- 16 KiB = 16 X 1024 Bytes
- Number of Blocks = 16 X 1024 / 32 =512 Blocks
- 4 Way = 512 /4 = 128 Sets -> 7 Bits for index
- Rest is Tag -> 22 bits

Meeting chat

CW Wilson, Cory 6:30 PM
4?

CN Nandrea, Christopher 6:30 PM
3

CW Wilson, Cory 6:31 PM
oh yeah. gotcha

6:32 PM
500?

PP Panth, Prashant 6:33 PM
512

KH Hjelm, Kali 6:33 PM
512

CG Gamble, Cory 6:34 PM
128

PP Panth, Prashant 6:35 PM
7

Reply

Q3

- Find the number of misses for each cache organization (Direct Map) given the following sequence of block addresses 8, 0, 5, 32, 0, 42, 9 and the address format is 8 bit ,if the cache has Capacity: 16B and Block size: 4B .

For 8 bit address

For 16 bit address

For 32 bit address

48:07

Rajan, Ranjidha

Rajan, Ranjidha

Recording has started. Participation in the meeting indicates your consent to being included in the meeting recording.

Privacy policy | Dismiss

Meeting chat

got it thank you

CG Gamble, Cory 6:43 PM
1) 512, 2) 256, 3) 3, 8, 21?

PP Panth, Prashant 6:43 PM
offset 3
blocks 512
2 way associative 256
Index 8
tag 21
See less

CM Mechem, Christian 6:44 PM
👍

ET Tovar, Efren Paredes 6:45 PM
yes

CG Gamble, Cory 6:46 PM
For #2, why is the tag 21? Thought it was the offset originally

Reply

Q3

- Find the number of misses for each cache organization (Direct Map) given the following sequence of block addresses 8, 0, 5, 32, 0, 42, 9 and the address format is 8 bit ,if the cache has Capacity: 16B and Block size: 4B .

For 8 bit address

For 16 bit address

For 32 bit address

- Address 8 = (binary)00001000 [for 8 bit address format]

Your microphone is muted.

56:10

Rajan, Ranjidha

Rajan, Ranjidha

Meeting chat

CG Gamble, Cory 6:46 PM
For #2, why is the tag 21? Thought it was the offset originally

CW Wilson, Cory 6:49 PM
can you go to the last slide really quick please?

CW Wilson, Cory 6:49 PM
ok thank you!@

CG Gamble, Cory 6:54 PM
no, I didn't catch why t was 21

CG Gamble, Cory 6:55 PM
ok thank you

Meeting
Recording has started

Reply

Q3

- Find the number of misses for each cache organization (Direct Map) given the following sequence of block addresses 8, 0, 5, 32, 0, 42, 9 and the address format is 8 bit ,if the cache has Capacity: 16B and Block size: 4B .

For 8 bit address

For 16 bit address

For 32 bit address

4 bytes = 32 bits

16 Bytes

4 Bytes

32/8 = 4 addr

- Address 8 = (binary)00001000 [for 8 bit address format]

Meeting chat

Gamble, Cory 6:46 PM
For #2, why is the tag 21? Thought it was the offset originally

Wilson, Cory 6:49 PM
can you go to the last slide really quick please?

Wilson, Cory 6:49 PM
ok thank you!@

Gamble, Cory 6:54 PM
no, I didn't catch why t was 21

Gamble, Cory 6:55 PM
ok thank you

Meeting
Recording has started

Reply

Recording has started. Participation in the meeting indicates your consent to being included in the meeting recording.
Privacy policy
Dismiss

Q3 A

00001001

	00	01	10	11
→ 00	0000 (0)	1	2	3
→ 01				
→ 10	0000 (8)	0000 (9)	0000 (10)	0000 (11)
11				

Your microphone is muted.

Q3 B

8 0 5 32 0 42 9

	0	1
00		
01		
10		
11		

Q4 SOLUTION

- Memory addresses are 16 bits, The cache has 8 rows (i.e., 8 cache lines) , Each cache row (line) holds 16 bytes of data . **Tag = 10 Index=3 Offset=3**

$\frac{16 \times 8}{16} = 8 = 2^3$

Memory Address	Tag	Index	Offset Copied	Hit or Miss
0010 1101 1011 0011		110		
0000 0110 1111 1100		111		
0010 1101 1011 1000		111		
1010 1010 1010 1011		101		

⚠ **Recording has started.** Participation in the meeting indicates your consent to being included in the meeting recording. Privacy policy Dismiss

Q4 SOLUTION

- Memory addresses are 16 bits, The cache has 8 rows (i.e., 8 cache lines) , Each cache row (line) holds 16 bytes of data . **Tag = 10 Index=3 Offset=3**

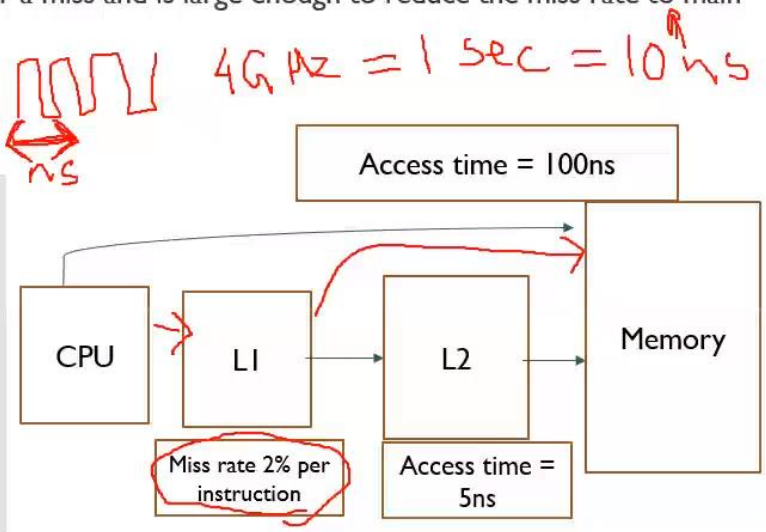
Memory Address	Tag	Index	Offset Copied	Hit or Miss
0010 1101 1011 0011		110	0010110110110 [000,001,010,.....,111]	M
0000 0110 1111 1100		111	0010110110 111[000,001,.....,111]	M
0010 1101 1011 1000		111		M
1010 1010 1010 1011		101	1010101010 101 [000,.....,111]	M

PERFORMANCE OF MULTILEVEL CACHE

- Suppose we have a processor with a base CPI of 1.0, assuming all references hit in the primary cache, and a clock rate of 4 GHz. Assume a main memory access time of 100 ns, including all the miss handling. Suppose the miss rate per instruction at the primary cache is 2%. How much faster will the processor be if we add a secondary cache that has a 5-ns access time for either a hit or a miss and is large enough to reduce the miss rate to main memory to 0.5%?
- Clock cycle per instruction (CPI) = 1
- Clock rate = 4 GHz

4 Ghz in 1 sec = pow(10,9) ns ; 1 period = pow(10,9) / 4GHz
1 period= pow(10,9) / 4 X pow(10,9) = 1/4 =0.25 ns
Miss penalty to Main memory = 100 / .25 =400 clock cycles
CPI = Base CPI + Memory Stall cycles / instruction = 1 + 2% X 400 =1+8 =9

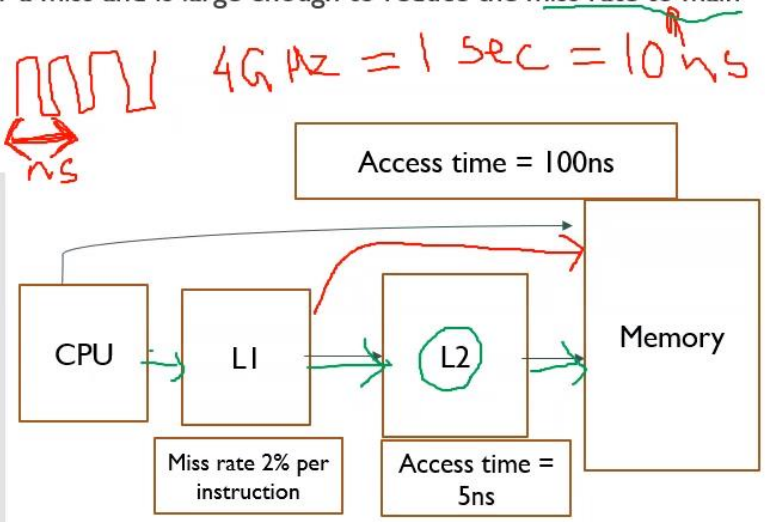
To Access With L1 ONLY



PERFORMANCE OF MULTILEVEL CACHE

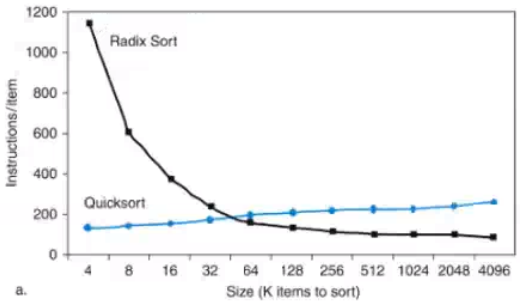
- Suppose we have a processor with a base CPI of 1.0, assuming all references hit in the primary cache, and a clock rate of 4 GHz. Assume a main memory access time of 100 ns, including all the miss handling. Suppose the miss rate per instruction at the primary cache is 2%. How much faster will the processor be if we add a secondary cache that has a 5-ns access time for either a hit or a miss and is large enough to reduce the miss rate to main memory to 0.5%?
- Clock cycle per instruction (CPI) = 1
- Clock rate = 4 GHz

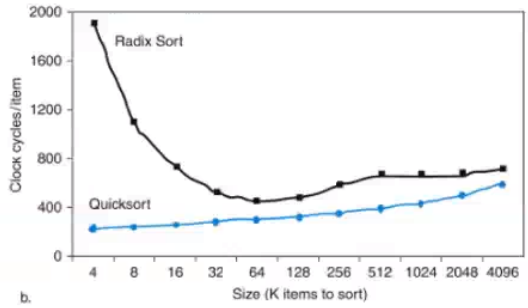
4 Ghz in 1 sec = pow(10,9) ns ; 1 period = pow(10,9) / 4GHz
1 period= pow(10,9) / 4 X pow(10,9) = 1/4 =0.25 ns
Miss penalty to Main memory = 100 / .25 =400 clock cycles
CPI = Base CPI + Memory Stall cycles / instruction = 1 + 2% X 400 =1+8 =9
Miss penalty of L2 = 5 / .25 = 20 clock cycles
CPI with L2 = Base CPi+ Stalls in L1+Stalls in L2
= 1 + 2% X 20 + 0.5% X 400 = 1+ .4 + 2 = 3.4
CPI =9 , with L2 CPI=3.4 , 9/3.4 times faster

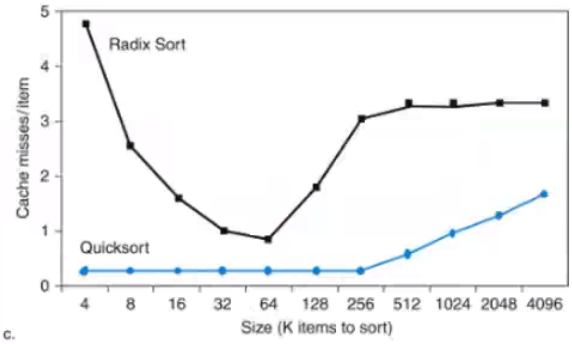


9 with L1 9/3.4 faster with L2

COMPARING QUICKSORT AND RADIX SORT







ERROR CODES

- **Error detection codes** – are used to detect the error(s) present in the received data (bit stream). These codes contain some bit(s), which are included (appended) to the original bit stream. These codes detect the error, if it is occurred during transmission of the original data (bit stream).
- **Example** – Parity code, Hamming code.

- **Error correction codes** – are used to correct the error(s) present in the received data (bit stream) so that, we will get the original data. Error correction codes also use the similar strategy of error detection codes. **Example** – Hamming code.

Parity Code- Uses 1 Parity Bit to detect Errors
Hamming Code – Checks multiple bits.

If Odd Number: Even Parity Bit = True

PARITY CODE

- Even Parity Code
  - *The value of even parity bit should be zero, if even number of ones present in the binary code.*
  - *Otherwise, it should be one.*

Binary Code	Even Parity bit	Even Parity Code
000	0	0000
001	1	0011
010	1	0101
011	0	0110
100	1	1001
101	0	1010
110	0	1100
111	1	1111

Your microphone is muted.

ODD PARITY

- Odd Parity Code
  - The value of odd parity bit should be zero, if odd number of ones present in the binary code.
  - Otherwise, it should be one.

Binary Code	Odd Parity bit	Odd Parity Code
000	1	0001
001	0	0010
010	0	0100
011	1	0111
100	0	1000
101	1	1011
110	1	1101
111	0	1110

Your microphone is muted.

Odd Parity = If Odd # of bits, Parity bit = 0;

PARITY BIT

7 bits of data	(count of 1-bits)	8 bits including parity	
		even	odd
0000000	0	00000000	00000001
1010001	3	10100011	10100010
1101001	4	11010010	11010011
1111111	7	11111111	11111110

PARITY BIT

7 bits of data	(count of 1-bits)	8 bits including parity	
		even	odd
0000000	0	00000000	00000001
1010001	3	10100011	10100010
1101001	4	11010010	11010011
1111111	7	11111111	11111110

PARITY BIT

- Example: even parity
  - 1000000
  - 1111101
  - 1001001
- Example: odd parity
  - 1000000
  - 1111101
  - 1001001

# Bits = Parity
1.1=1

2.6=0
3.3=1

4.1=0
5.6=1
6.3=0

PARITY BIT

- Assume we are using parity bit with 7-bit ASCII.
- The letter V in 7-bit ASCII is encoded as 0110101.
- How will the letter V be transmitted?

0 1 1 0 1 0 1 0

- For even parity ?
- For odd parity ?

Your microphone is muted.

- How will we do this using logic ?

FINDING EVEN PARITY

- XOR Truth Table

even

	0	0	1	1
^	0	1	0	1
	0	**1**	**1**	**0**

- 1011 check number of 1s => 1 ^ 0 ^ 1 ^ 1 = 1
- 1100
- 100100
- 111000

Your microphone is muted.

XOR

FINDING EVEN PARITY

- XOR Truth Table

even

	0	0	1	1
^	0	1	0	1
	0	**1**	**1**	**0**

- 1011 check number of 1s => 1 ^ 0 ^ 1 ^ 1 = 1
- 1100
- 100100
- 111000

Your microphone is muted.

XOR First 2 bits, use answer to XOR 3rd bit, and so on...

EXAMPLE –EVEN PARITY

- To transmit: **1001**
- Compute parity bit value:
  - 1+0+0+1 (mod 2) = **0** 0r 1^ 0 ^ 0^ 1 = **0**
- Adds parity bit and sends: **1001**0
- **Transmission error**

Your microphone is muted.

Sender & Receiver computes Parity bit & compares respectivey. If ParityBit != ParityBitSender, It's an ERROR.

EXAMPLE –EVEN PARITY

- To transmit: **1001**
- Compute parity bit value:
  - 1+0+0+1 (mod 2) = **0** 0r 1^ 0 ^ 0^ 1 = **0**
- Adds parity bit and sends: **1001**0
- **Transmission error**
- Receives: **1**1**01**0
- Computes parity: 1+1+0+1+0 (mod 2) = **1** 0r 1^ 1 ^ 0^ 1 = **1**

If the Parity Bit is corrupted, it may have a false positive||negative in the Error Checker.

HAMMING CODE – ERROR CORRECTION

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	✗		✗		✗		✗		✗		✗		✗		✗		✗		✗		...
	p2		✗	✗			✗	✗			✗	✗			✗	✗			✗	✗		
	p4				✗	✗	✗	✗					✗	✗	✗	✗					✗	
	p8								✗	✗	✗	✗	✗	✗	✗	✗						
	p16																✗	✗	✗	✗	✗	

More Thorough. Finds the position of the error. Goes by the Power of 2's.
Hamming Code = Even Parity Coverage

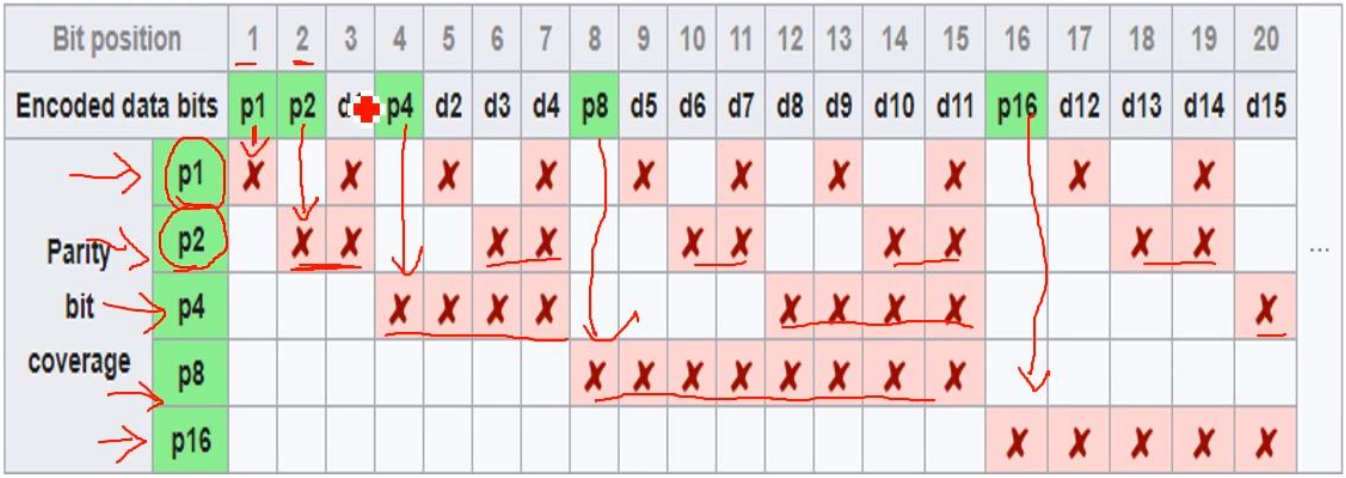
HAMMING CODE – ERROR CORRECTION

$2^0, 2^1, 2^2, 2^3, 2^4$ $\quad 2^{n-1}$ $\quad$ Hamm (15 (5))

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	✗		✗		✗		✗		✗		✗		✗		✗		✗		✗		...
	p2		✗	✗			✗	✗			✗	✗			✗	✗			✗	✗		
	p4				✗	✗	✗	✗					✗	✗	✗	✗					✗	
	p8								✗	✗	✗	✗	✗	✗	✗	✗						
	p16																✗	✗	✗	✗	✗	

Your microphone is muted.

HAMMING CODE – ERROR CORRECTION



Bit position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Encoded data bits	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15
Parity bit coverage p1	X		X		X		X		X		X		X		X		X		X	
Parity bit coverage p2		X	X			X	X			X	X			X	X			X	X	
Parity bit coverage p4				X	X	X	X					X	X	X	X					X
Parity bit coverage p8								X	X	X	X	X	X	X	X					
p16																X	X	X	X	X

HAMMING (15,11)

P = 4

Bit position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	1	1	0	1	0	1	1	0	0	0	1	1	0	0	1	
	P1	P2		P4				P8								Parity
P1	?	X	0	X	0	X	1	X	0	X	1	X	0	X	1	1
P2		?	0	X	X	1	1	X	X	0	1	X	X	0	1	0
P4				?	0	1	1	X	X	X	X	1	0	0	1	0
P8								?	0	0	1	1	0	0	1	1

Parity Coverage

Your microphone is muted.

Compare Parity Bits with P1,P2,P4,P8

HAMMING (15,11)

P = 4

Bit position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	1	1	0	1	0	1	1	0	0	0	1	1	0	0	1	
	P1	P2		P4				P8								Parity
P1	?	X	0	X	0	X	1	X	0	X	1	X	0	X	1	1
P2		?	0	X	X	1	1	X	X	0	1	X	X	0	1	0
P4				?	0	1	1	X	X	X	X	1	0	0	1	0
P8								?	0	0	1	1	0	0	1	1

Parity Coverage

p1

Take the Bits of all non-matching Parity Bit positions and Add them.

HAMMING (15,11)

P = 4

Bit position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	1	1	0	1	0	1	1	0	0	0	1	1	0	0	1	
	P1	P2		P4				P8								Parity
P1	?	X	0	X	0	X	1	X	0	X	1	X	0	X	1	1
P2		?	0	X	X	1	1	X	X	0	1	X	X	0	1	0
P4				?	0	1	1	X	X	X	X	1	0	0	1	0
P8								?	0	0	1	1	0	0	1	1

Parity Coverage

p1
p2 in 2, p4 in 4,p8 in 8
2+4+8 = 14

Meaning the Error is in the 14th Position & Needs to be changed.

HAMMING (15,11)

P = 4

Bit position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	1	1	0	1	0	1	1	0	0	0	1	1	0	0	1	
	P1	P2		P4				P8								
P1	?	X	0	X	0	X	1	X	0	X	1	X	0	X	1	1
P2		?	0	X	X	1	1	X	X	0	1	X	X	0	1	0
P4				?	0	1	1	X	X	X	X	1	0	0	1	0
P8								?	0	0	1	1	0	0	1	1

Parity Coverage

Your microphone is muted.

p1
p2 in 2, p4 in 4,p8 in 8
2+4+8 = 14

HAMMING (15,11)

P = 4

Bit position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	1	1	0	1	0	1	1	0	0	0	1	1	0	0	1	
	P1	P2		P4				P8								
P1	?	X	0	X	0	X	1	X	0	X	1	X	0	X	1	1
P2		?	0	X	X	1	1	X	X	0	1	X	X	0	1	0
P4				?	0	1	1	X	X	X	X	1	0	0	1	0
P8								?	0	0	1	1	0	0	1	1

Parity Coverage

Your microphone is muted.

p1
p2 in 2, p4 in 4,p8 in 8
2+4+8 =

Change 14 from 0 to 1.

VIRTUAL MACHINES

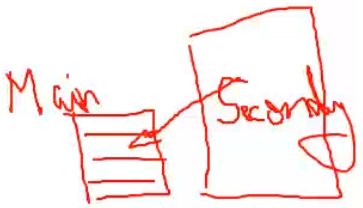
- *Virtual Machines are emulators.*
- *System Virtual Machines.*
  - Examples : IBM VM/370, VirtualBox, Vmware
- The software that supports VMs is called a *virtual machine monitor* (VMM) or *hypervisor.*

- *Why*
- The increasing importance of isolation and security in modern systems
- The failures in security and reliability of standard operating systems
- The sharing of a single computer among many unrelated users, in particular for Cloud computing
- The dramatic increases in raw speed of processors over the decades, which makes the overhead of VMs more acceptable

Virtual Machines
VM Monitor = VMM = Hypervisor

VIRTUAL MEMORY

- ***Virtual memory***: A technique that uses main memory as a "cache" for secondary storage.

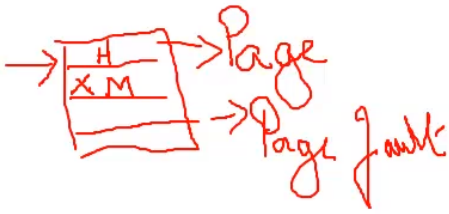


- Use main memory as a "cache" for secondary (disk) storage
  - Managed jointly by CPU hardware and the operating system (OS)
- Programs share main memory
  - Each gets a private virtual address space holding its frequently used code and data
  - Protected from other programs
- CPU and OS translate virtual addresses to physical addresses
  - VM "block" is called a page
  - VM translation "miss" is called a page fault

Virtual MEmory

VIRTUAL MEMORY

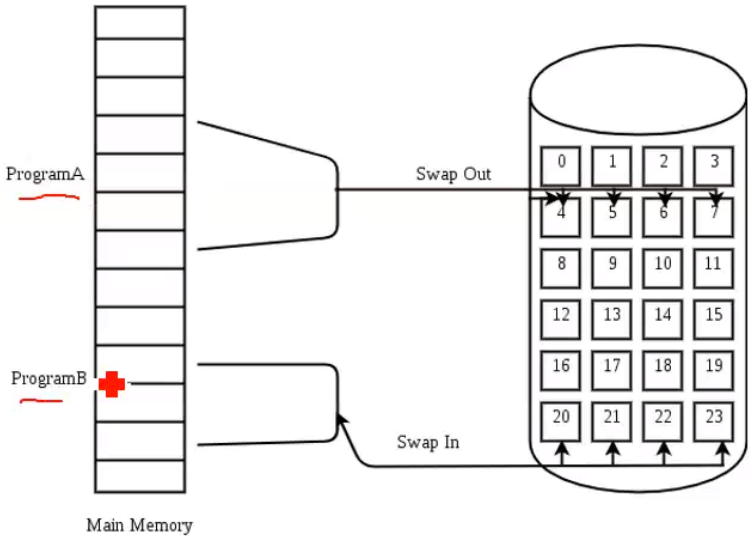
- ***Virtual memory***: A technique that uses main memory as a "cache" for secondary storage.



- Use main memory as a "cache" for secondary (disk) storage
  - Managed jointly by CPU hardware and the operating system (OS)
- Programs share main memory
  - Each gets a private virtual address space holding its frequently used code and data
  - Protected from other programs
- CPU and OS translate virtual addresses to physical addresses
  - VM "block" is called a page
  - VM translation "miss" is called a page fault

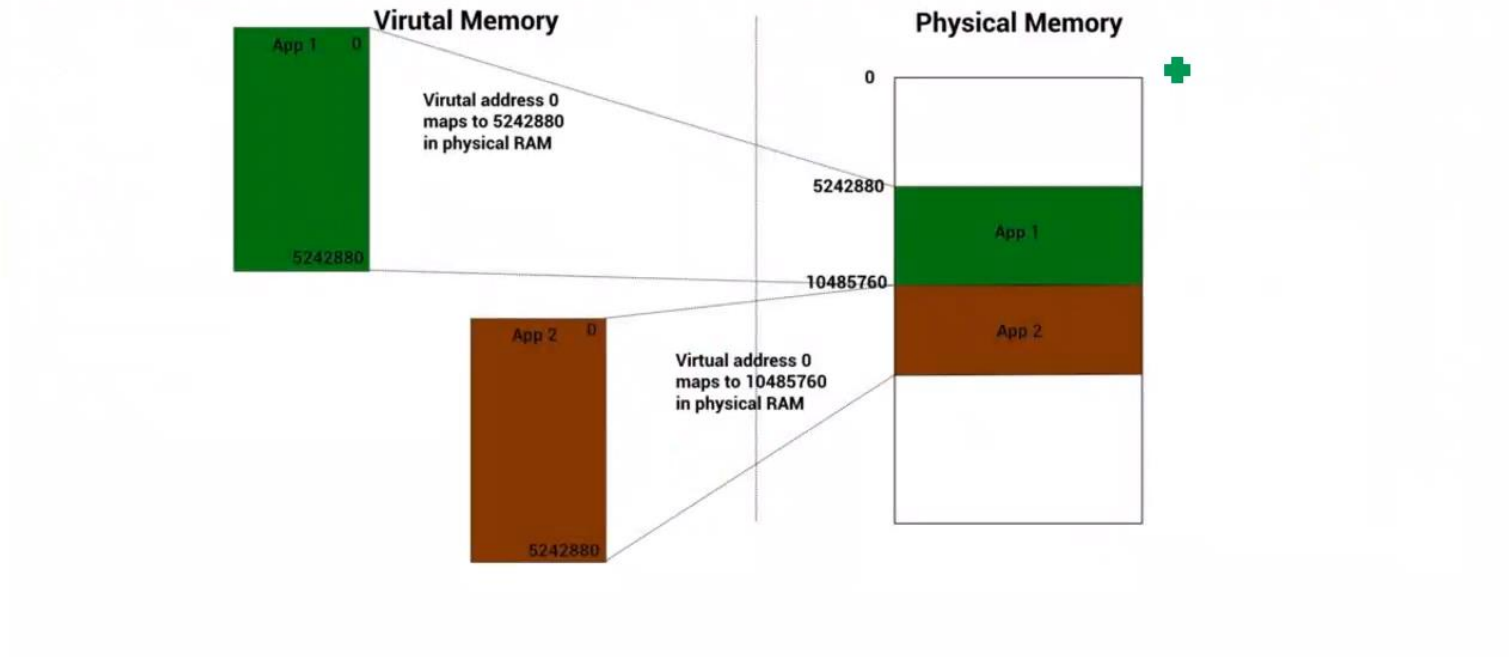
SWAP MEMORY

- Multiple program has to run at the same time.
- Swap memory helps multiprogramming .
- To execute program of any size.
- Main memory (with physical address) act as a "cache" for secondary (disk) storage.
- User process cannot access privileged information.



Swap Memory

VIRTUAL MEMORY



Virtual Mem vs Physical Mem

VIRTUAL MEMORY

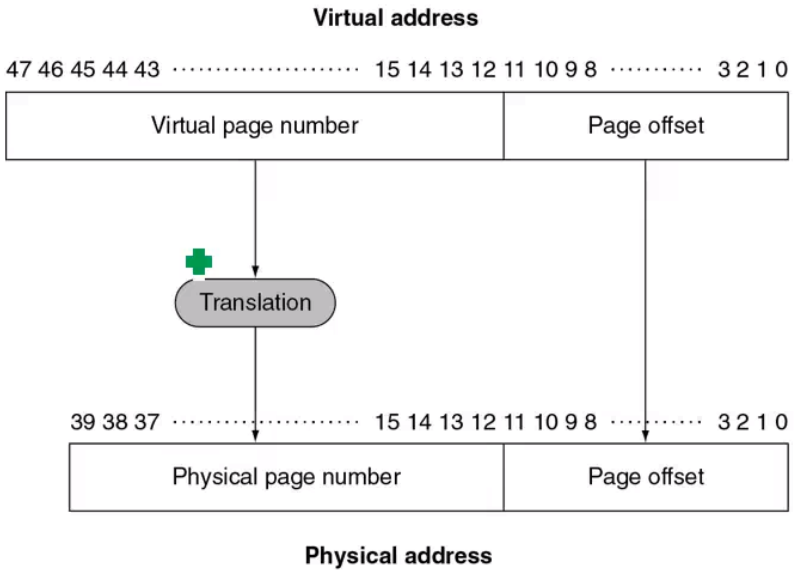
- VM "block" is called a page
- VM translation "miss" is called a page fault

Virtual addresses
Address translation
Physical addresses
Disk addresses

Main

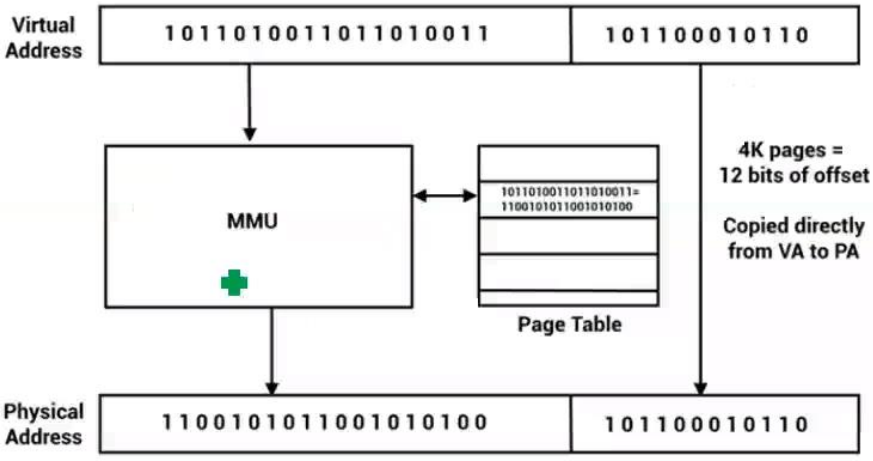
Page Fault – A miss

ADDRESS TRANSLATION



Address Translation

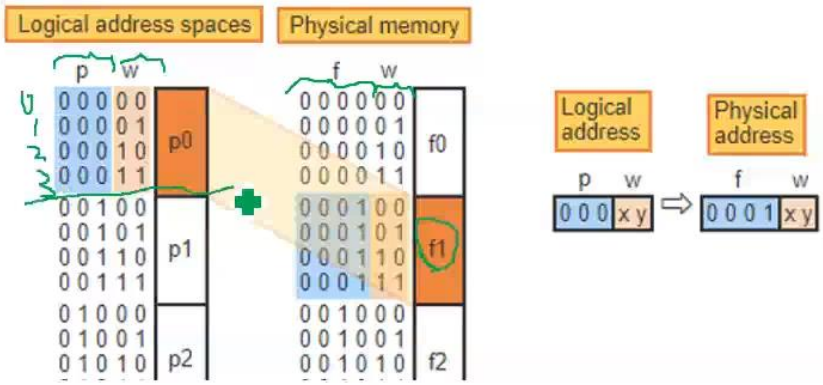
ADDRESS TRANSLATION



Page table: The table containing the virtual to physical address translations in a virtual memory system. The table, which is stored in memory, is typically indexed by the virtual page number; each entry in the table contains the physical page number for that virtual page if the page is currently in memory.

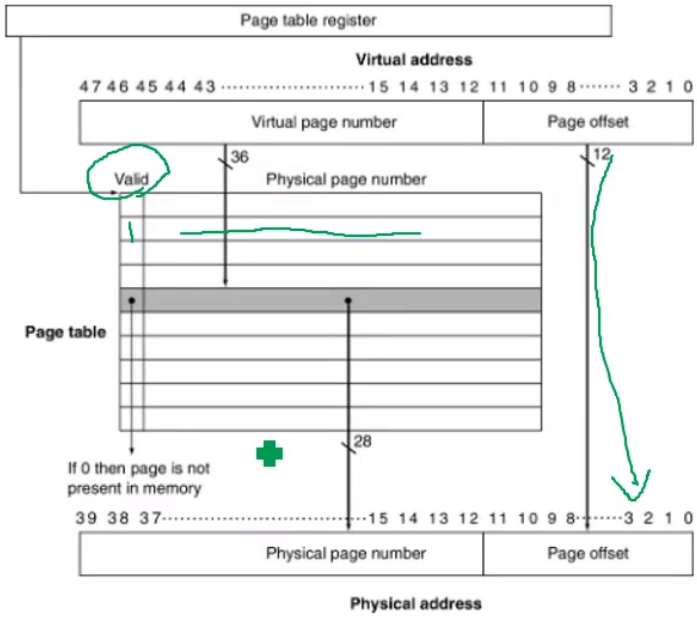
-Memory Management Unit.
-Uses a Page Table, like Hash Map.
-Uses a Boot Loader.

ADDRESS TRANSLATION



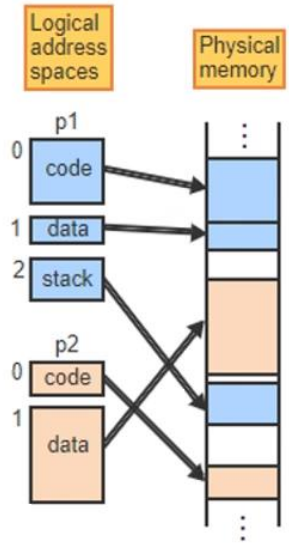
Page # (000's)& Offset (00,01,10,11)

PAGE TABLE



SEGMENTATION

- ***Segmentation***: A variable-size address mapping scheme in which an address consists of two parts: a segment number, which is mapped to a physical address, and a segment offset.



Segmentation

REFERENCE BIT

- ***Reference bit***: Also called ***use bit*** or ***access bit***. A field that is set whenever a page is accessed and that is used to implement LRU or other replacement schemes.
  - Reference bit in page table entry set to 1 on access to page
  - Periodically cleared to 0 by OS
  - A page with reference bit = 0 has not been used recently.

Your microphone is muted.

-Reference Bit, Use Bit, Access Bit. Bypassing/Forwarding(), Not Stalling(NOPs).

WRITE

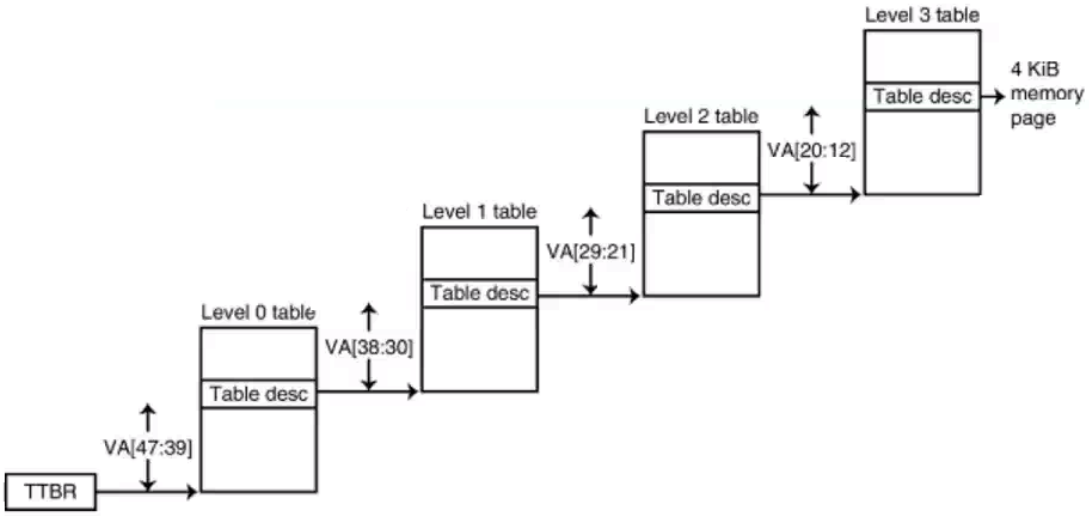
- Write-through will not work for virtual memory, since writes take too long. The Write buffer to allow the system to write-through to disk would be completely impractical due to high cost.
- Write –back is used .

- To track whether a page has been written since it was read into the memory, a *dirty bit* is added to the page table. The dirty bit is set when any word in a page is written.
- Modified page is called dirty page.

Write-Back uses a Dirty bit (True if Updated/Modified)

MULTIPLE PAGE TABLES

- *Translation Table Base Register* (TTBR) gives the starting address of the first page table.



Tranlation Table Base Register (TTBR)

?

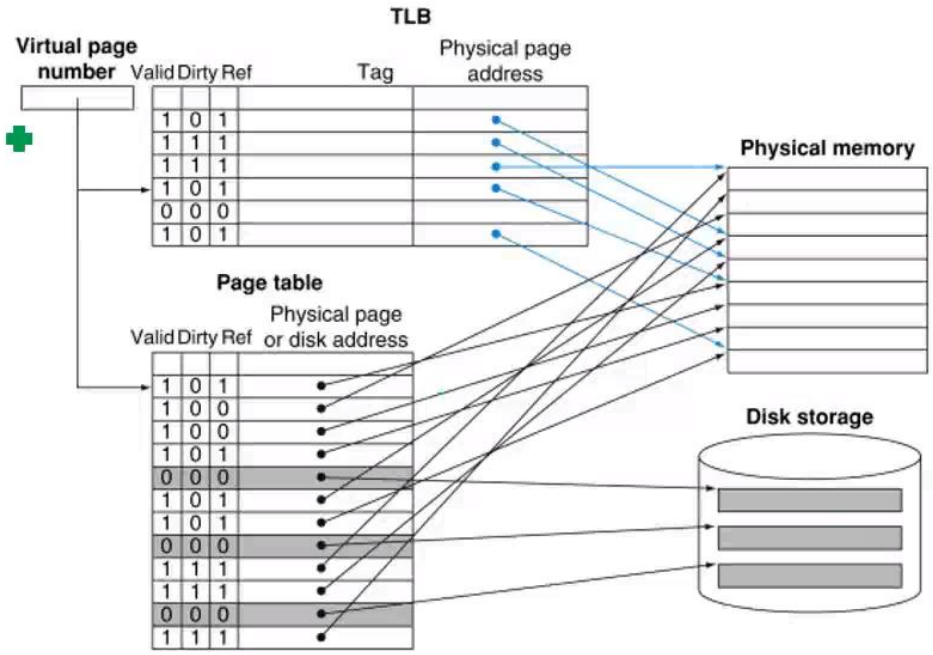
- What is this Page Table and where is the Page Table stored ?

Your microphone is muted.

Page Table is stored in main Memory.

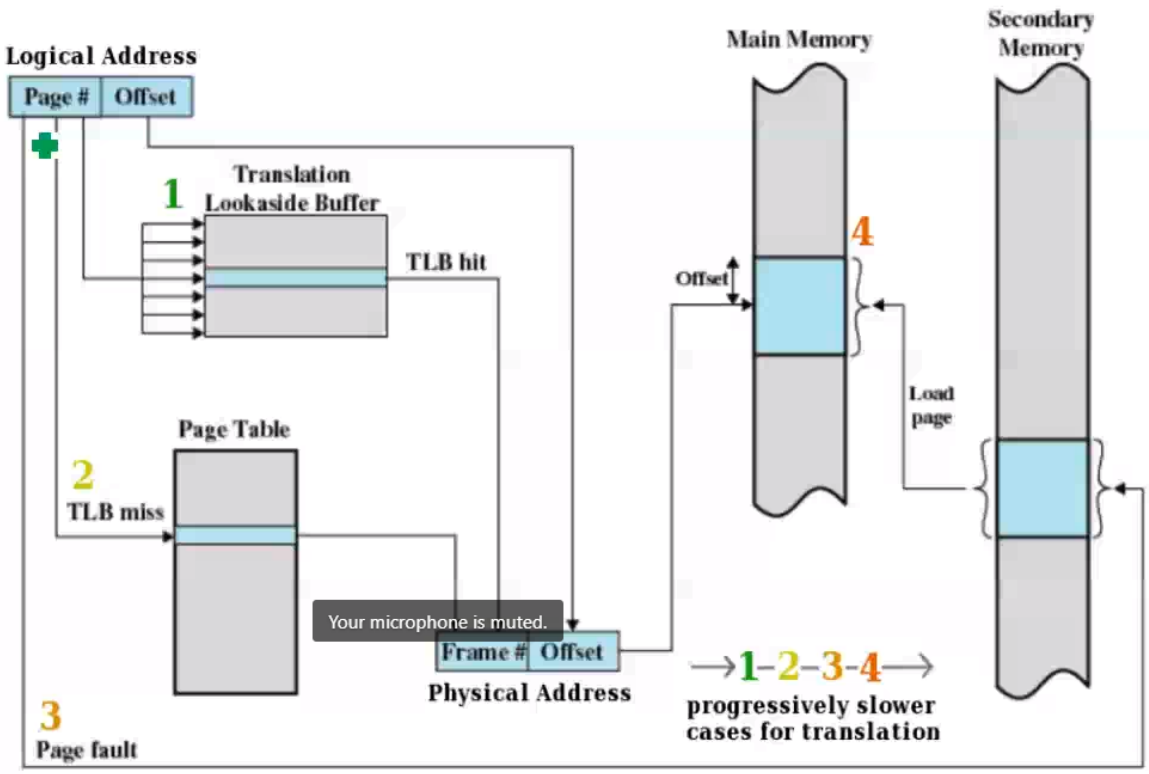
- ***Translation-lookaside buffer (TLB)***:

A cache that keeps track of recently used address mappings to try to avoid an access to the page table.

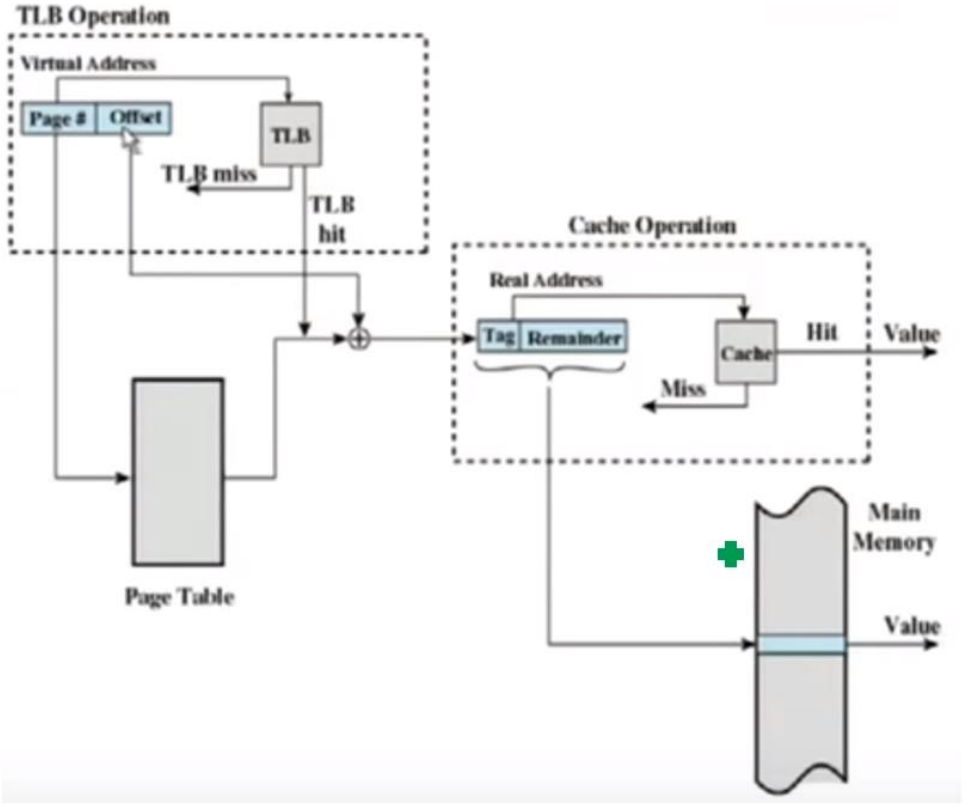


Translation-lookaside buffer (TLB) – Virtual Memory Cache.
Page Fault- Can't find value.

- TLB Miss is not Page fault



TLB AND CACHE



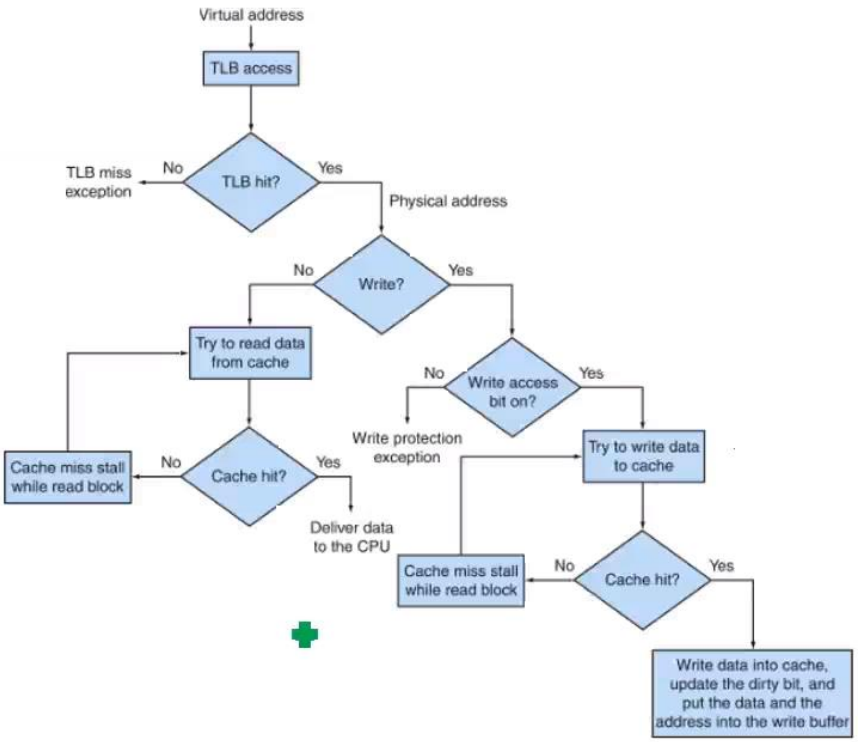
TLB & CACHE

PROTECT

- Access bits
  - Show permission –Read only, write ,execute only .

- ***Aliasing***: A situation in which two addresses access the same object; it can occur in virtual memory when there are two virtual addresses for the same physical page.

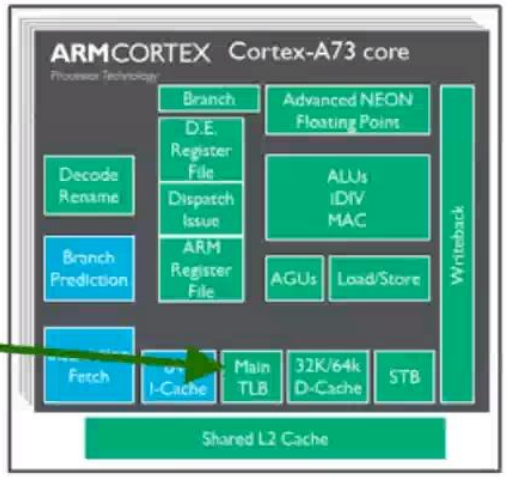
Access Bit = Permission Bit
Aliasing =



Waiting for substrate.office.com...

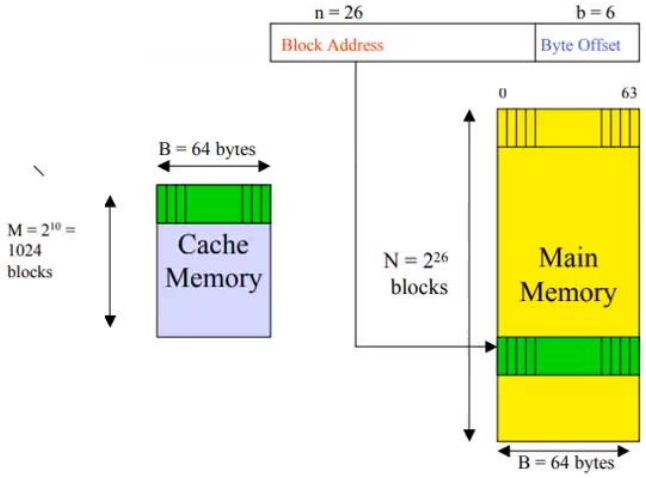
TLB

Translation Lookaside Buffer on ARM Cortex-A73



Arm Corex-A73 Archetecture

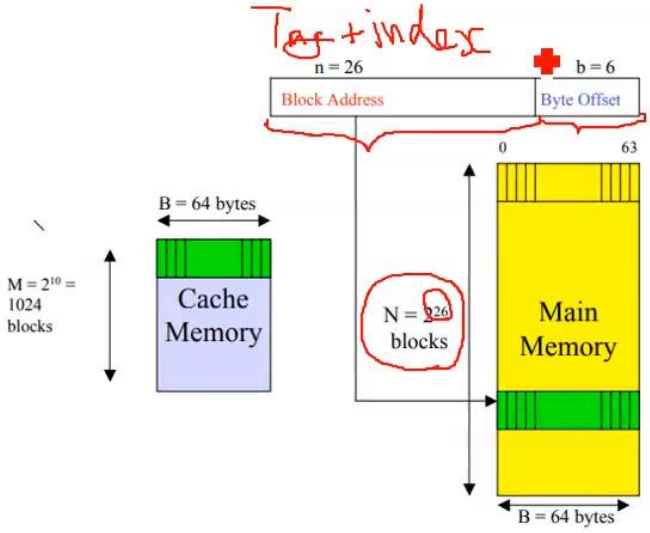
BYTE ADDRESSABLE



- Main memory: Byte addressable memory of size 4GB = 2^{32} bytes
- Cache size: 64KB = 2^{16} bytes
- Block (line) size : 64 bytes = 2^6 bytes
- Number of memory blocks = $2^{32} / 2^6 = 2^{26}$
- Number of cache blocks = $2^{16} / 2^6 = 2^{10}$

Byte Addressable Mode Programs
?'s are Bit || Byte Addressable
Find Tag/Index/Offset

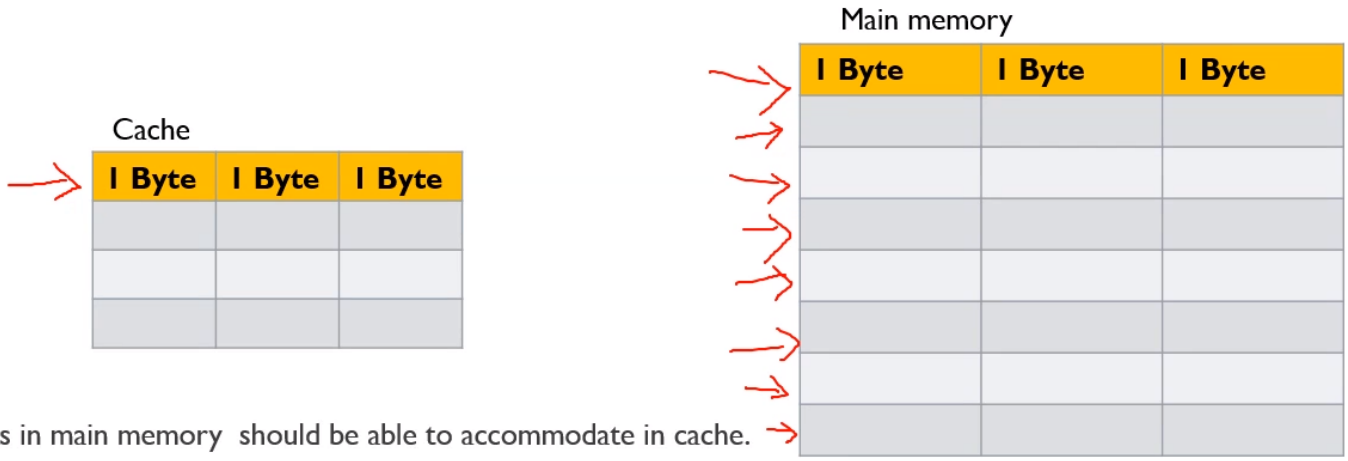
BYTE ADDRESSABLE



- Main memory: Byte addressable memory of size 4GB = 2^{32} bytes
- Cache size: 64KB = 2^{16} bytes
- Block (line) size : 64 bytes = 2^6 bytes
- Number of memory blocks = $2^{32} / 2^6 = 2^{26}$
- Number of cache blocks = $2^{16} / 2^6 = 2^{10}$

Byte Addressable gives Block address of Tag+Index

BYTE ADDRESSABLE EXAMPLE



- All blocks in main memory should be able to accommodate in cache.
- Tag + index = bits required to accommodate main memory blocks
- In the example Number of bits in Tag + Index = [There are 8 blocks in main memory $=2^3$] => 3 bits

BYTE ADDRESSABLE EXAMPLE

- A two-way set-associative cache has lines of 16 bytes and a total size of 8 KBytes. The 64-Mbyte main memory is **byte addressable**. Show the format of main memory addresses.

Your microphone is muted.

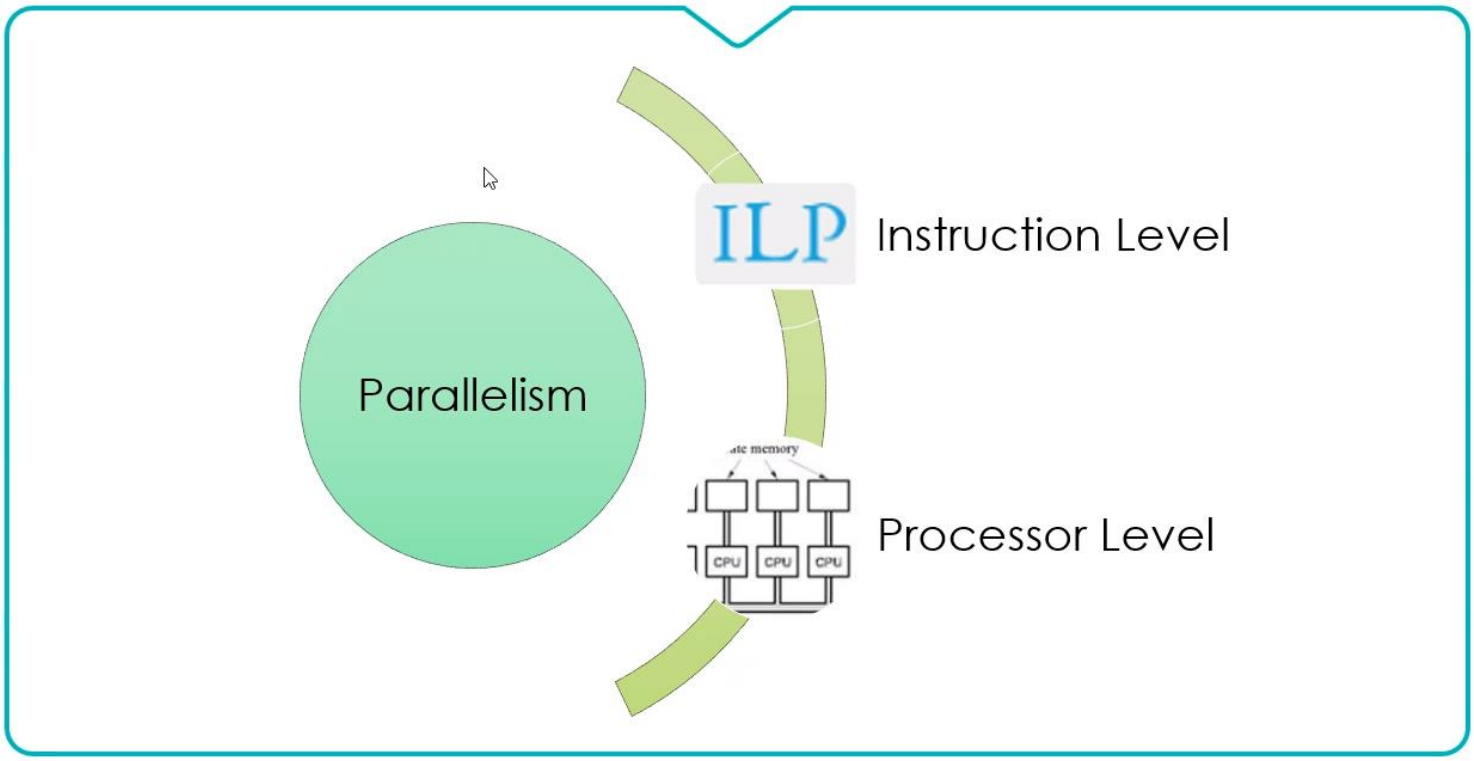
BYTE ADDRESSABLE EXAMPLE -SOLUTION

- A two-way set-associative cache has lines of 16 bytes and a total size of 8 KBytes. The 64-Mbyte main memory is **byte addressable**. Show the format of main memory addresses.

- Size of cache = 8 KB and Each block can have 16 Bytes .
- Number of cache lines = 512
- Since it's a 2 way – Number of sets =256 (2^8)
- Size of Memory = 64 M Byte To Map 64 MB (2^{26}) to 16 Byte blocks = 2^{26} / 16 = 2^{22} Blocks
- Tag + index =22 bits + offset
- Tag + 8 = 22 Tag =14 ,index =8 and offset =4

Your microphone is muted.

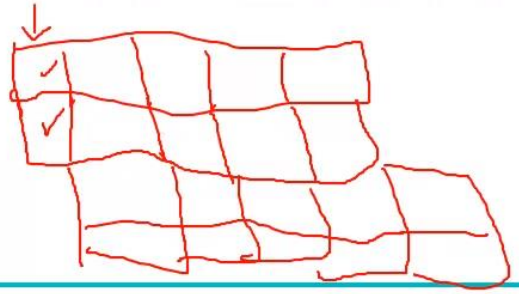
4/28/20



Parallelism = Instruction Level || Processor Level

Instruction Level Parallelism

- ***Instruction-level parallelism***: The parallelism among instructions.
  - Increasing the depth of the pipeline to overlap more instructions. Increase the number of stages.
  - Replicate the internal components (multiple issue). A scheme whereby multiple instructions are launched in one clock cycle.
  - **Multiple Issue**
    - Static multiple issue
    - Dynamic multiple issue



-**Instruction level** = Increasing Depth & Replicate internal components.
-Multiple Issue = Static | Dynamic multiple issue... Static = Compiling & Dynamic = Execution

Multiple issue pipeline

- Two primary and distinct responsibilities with in a multiple-issue pipeline:
  - Packaging instructions into *issue slots*:
    - How does the processor determine how many instructions and which instructions can be issued in a given clock cycle?
    - In static issue processors, this process is at least partially handled by the compiler;
    - In dynamic issue designs, it is normally dealt with at runtime by the processor,
  - Dealing with data and control hazards:
    - In static issue processors, the compiler handles .
    - dynamic issue processors , the hardware supports

-issue Slots -Data & Control hazards

Instruction Level Parallelism

- Terms
  - Single issue - One instruction in one clock cycle.
  - Multiple issue – Multiple instruction in one clock cycle.
  - Issue slots – Position available for the issue instruction in a given clock cycle.
  - Static issue – Decision made by compiler before execution.
  - Dynamic issue – Decision are made during execution.

-issue Slots -Data & Control hazards

ISSUES = Single / Multiple / Issue Slots / Static / Dynamic

Instruction Level Parallelism
Static multiple-issue

- Static Multiple Issue
  - ***Issue packet***: The set of instructions that issues together in one clock cycle; the packet may be determined statically by the compiler or dynamically by the processor.

  - ***Very Long Instruction Word*** (***VLIW***): A style of instruction set architecture that launches many operations that are defined to be independent in a single wide instruction, typically with many separate opcode fields.

- Issue Packet & Very Long Instruction Word (VLIW) = 64 Bit Address Instructions

Instruction Level Parallelism
Static multiple-issue

- Example
  - ALU or Branch Instruction in one slot.
  - Load or Store Instruction in another slot.

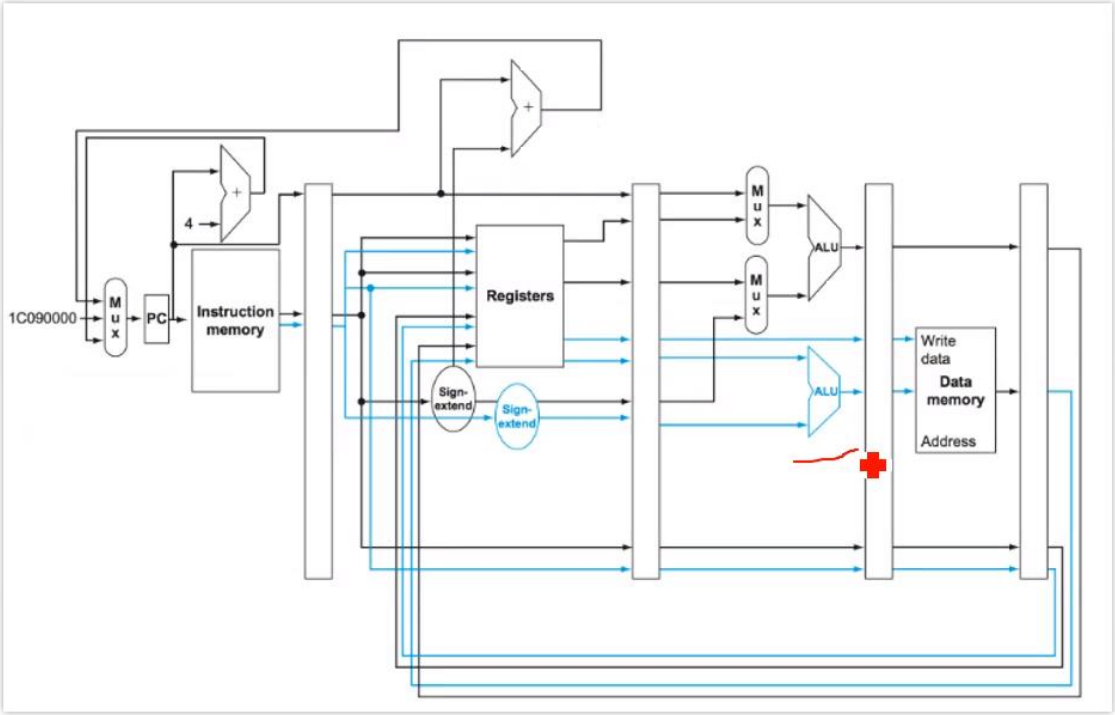
Instruction type	Pipe stages							
ALU or branch instruction	IF	ID	EX	MEM				
Load or store instruction	IF	ID	EX	MEM				
ALU or branch instruction		IF	ID	EX				
Load or store instruction		IF	ID	EX				
ALU or branch instruction			IF	ID				
Load or store instruction			IF	ID				
ALU or branch instruction				IF				
Load or store instruction				IF				

Instruction Level Parallelism
Static multiple-issue

- Example
  - ALU or Branch Instruction in one slot.
  - Load or Store Instruction in another slot.

Instruction type	Pipe stages							
ALU or branch instruction	IF	ID	EX	MEM	WB			
Load or store instruction	IF	ID	EX	MEM	WB			
ALU or branch instruction		IF	ID	EX	MEM	WB		
Load or store instruction		IF	ID	EX	MEM	WB		
ALU or branch instruction			IF	ID	EX	MEM	WB	
Load or store instruction			IF	ID	EX	MEM	WB	
ALU or branch instruction				IF	ID	EX	MEM	WB
Load or store instruction				IF	ID	EX	MEM	WB

-DataPath



2 Sign Extentions for Operations: 1.Load/Store & 2.Branch

Instruction Level Parallelism
Static multiple-issue

- Multiple Issue
- ***Use latency***: Number of clock cycles between a load instruction and an instruction that can use the result of the load without stalling the pipeline.
- How would this loop be scheduled on a static two-issue pipeline for LEGv8?

```
Loop: LDUR X0, [X20,#0]  // X0=array element
      ADD  X0,X0,X21     // add scalar in X21
      STUR X0, [X20,#0]  // store result
      SUBI X20,X20,#8     // decrement pointer
      CMP  X20,X22       // compare to loop limit
      BGT  Loop          // branch if X20 > X22
```

	ALU or branch instruction	Data transfer instruction	Clock cycle
Loop:			1
			2
			3
			4
			5

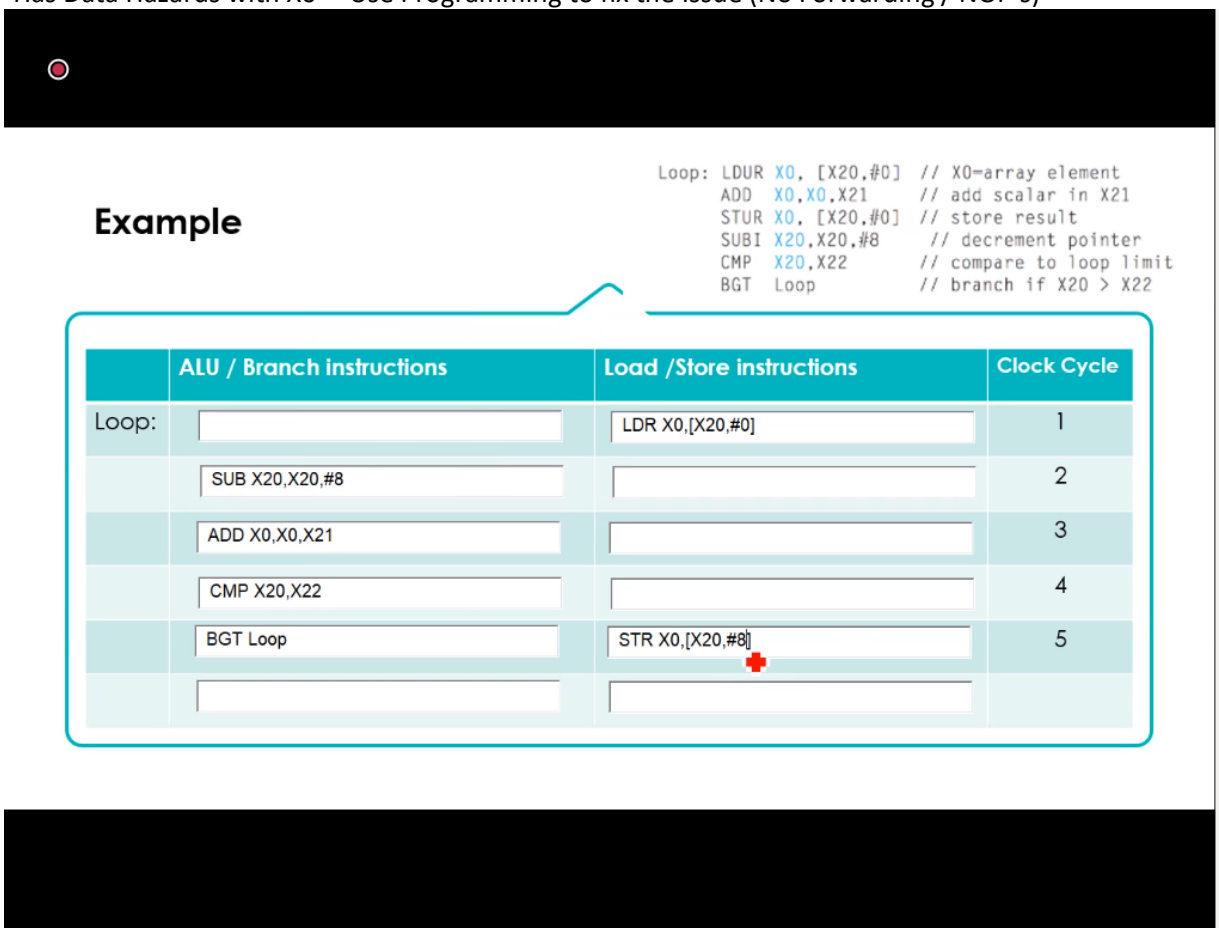
-Latency

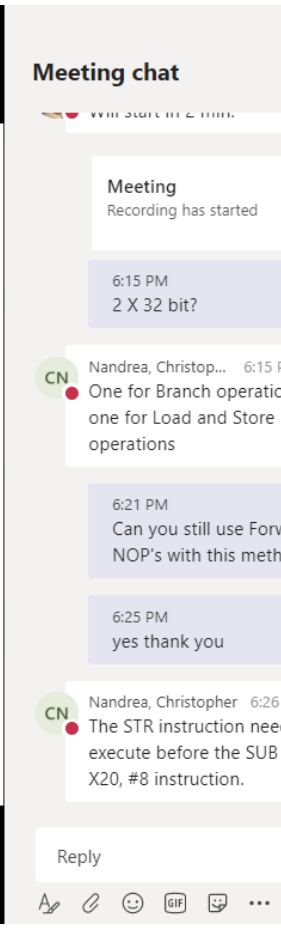
Example

```
Loop: LDUR X0, [X20,#0]  // X0=array element
      ADD  X0,X0,X21     // add scalar in X21
      STUR X0, [X20,#0]  // store result
      SUBI X20,X20,#8     // decrement pointer
      CMP  X20,X22       // compare to loop limit
      BGT  Loop          // branch if X20 > X22
```

	ALU / Branch instructions	Load /Store instructions	Clock Cycle
Loop:			1
			2
			3
			4
			5

-Has Data Hazards with X0 -Use Programming to fix the Issue (No Forwarding / NOP's)





-Change STUR X0,[X20,#0] into STUR X0[X20,#8] to Increment the Address

Instruction Level Parallelism
Static multiple-issue

- **Loop Unrolling :** A technique to replicate loop body Multiple Times.
- Example:
  - for(i=0;i< 100 ;i++)
    - {X[i] =X[i] + K;
  - for(i=0; i<100; i+=4)
  - How will you modify the code without changing the result ?

-Loop Unrolling - J

Instruction Level Parallelism
Static multiple-issue

- ***Register renaming***: The renaming of registers by the compiler or hardware to remove name dependences (***Antidependence)***.

	ALU or branch instruction	Data transfer instruction	Clock cycle
Loop:	SUBI X20, X20,#32	LDUR X0, [X20,#0]	1
		LDUR X1, [X20,#24]	2
	ADD X0, X0, X21	LDUR X2, [X20,#16]	3
	ADD X1, X1, X21	LDUR X3, [X20,#8]	4
	ADD X2, X2, X21	STUR X0, [X20,#32]	5
	ADD X3, X3, X21	STUR X1, [X20,#24]	6
	CMP X20, X22	STUR X2, [X20,#16]	7
	BGT Loop	STUR X3, [X20,#8]	8

-Register Renaming = Antiindependence

Instruction Level Parallelism
Static multiple-issue

- **Loop Unrolling :** A technique to replicate loop body Multiple Times.
- Example:
  - for(i=0;i< 100 ;i++)
    - {X[i] =X[i] + K;
  - for(i=0; i<100; i+=4)
  - How will you modify the code without changing the result ?

```
for (i=0;i<100;i++)
{
  x[i]=x[i]+k;
}
```

Your microphone is muted.

Do 4 registers at once to remove data hazards.

Instruction Level Parallelism
Dynamic multiple-issue processors

- ***Superscalar***: An advanced pipelining technique that enables the processor to execute more than one instruction per clock cycle by selecting them during execution.
- ***Dynamic pipeline scheduling***: Hardware support for reordering the order of instruction execution to avoid stalls.

-Superscalar / Dynamic pipeline scheduling

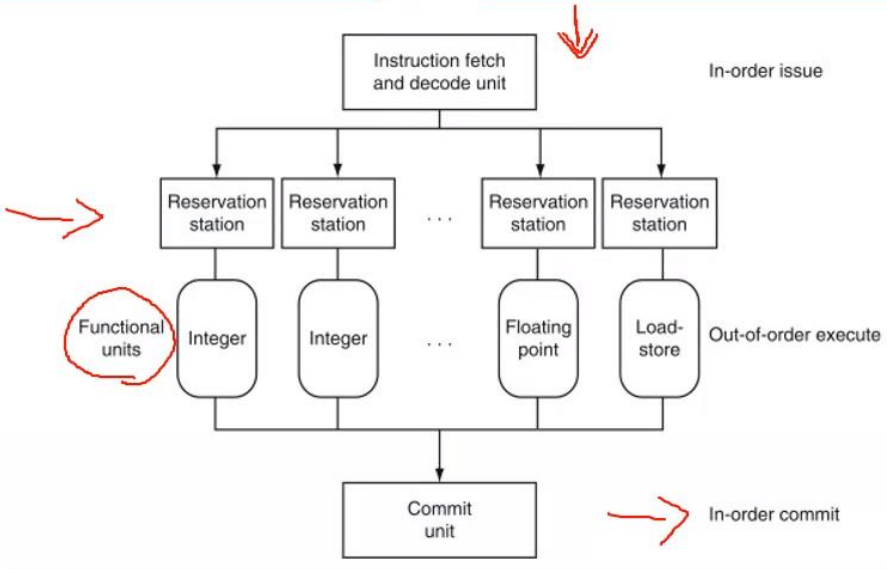
Instruction Level Parallelism
Dynamic multiple-issue processors

- ***Commit unit***: The unit in a dynamic or out-of-order execution pipeline that decides when it is safe to release the result of an operation to programmer-visible registers and memory.
- ***Reservation station***: A buffer within a functional unit that holds the operands and the operation.
- ***Reorder buffer***: The buffer that holds results in a dynamically scheduled processor until it is safe to store the results to memory or a register.

-Commit unit / Reservation Station / Reorder buffer

Instruction Level Parallelism
Dynamic multiple-issue processors

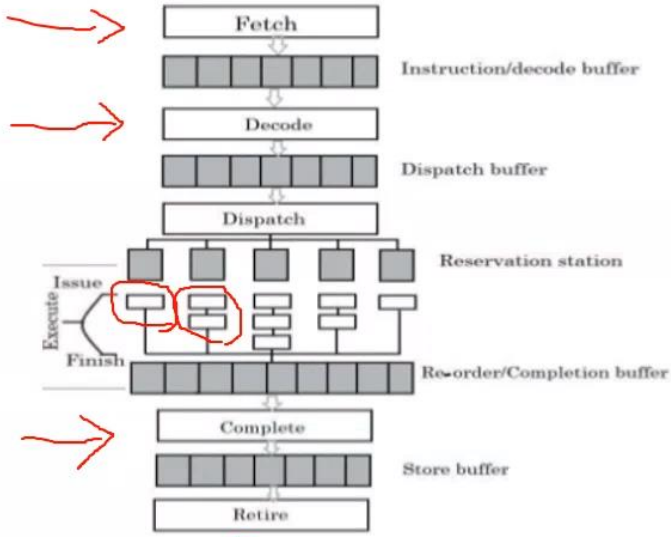
- Dynamically scheduled pipeline



Dynamic = Hardware

Instruction Level Parallelism
Dynamic multiple-issue processors

- Super Scalar



Parallel Processors

- ***Multiprocessor***: A computer system with at least two processors.
- ***Task-level parallelism*** or ***process-level parallelism***: Utilizing multiple processors by running independent programs simultaneously.
- ***Parallel processing program***: A single program that runs on multiple processors simultaneously.
- ***Cluster***: A set of computers connected over a local area network that function as a single large multiprocessor.
- ***Multicore microprocessor***: A microprocessor containing multiple processors ("cores") in a single integrated circuit.

Parallel Processors

- Execution time after improvement
- $= \frac{Execution\ time\ \ affected\ by\ improvement}{Amount\ of\ improvement}$ + Execution time unaffected

Q

- Suppose you want to perform two sums: one is a sum of 10 scalar variables, and one is a matrix sum of a pair of two-dimensional arrays, with dimensions 10 by 10. For now let's assume only the matrix sum is parallelizable; What speed-up do you get with 10 versus 40 processors?
- Calculate the speed-ups assuming the matrices grow to 20 by 20.

10 scalar variables[Can NOT apply parallelism] 10 X 10 =100 vector [Can apply parallelism]

For one processor => Total time= 10 for scalar + 100 for matrix = 110t
For 10 processors => Ex Time after impr. = Ex time affected /Amount of impr + unaffected = 100 t /10 + 10t = 20t.
Speed = 110t / 20t = 5.5

Solution Cont .

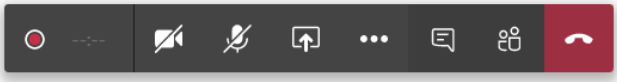
- If Matrices grow to 20 by 20 ?
- The sequential program now takes $10t + 400t = 410t$.
- For 10 processors and 40 processors?
- Execution time after improvement $= \frac{Execution\ time\ affected\ by\ improvement}{Amount\ of\ improvement}$ + Execution time unaffected
- $= \frac{400t}{10} + 10t = 50t$ $= \frac{400t}{40} + 10t = 20t$

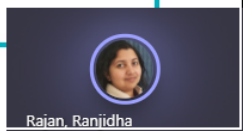
10 processors is $410t/50t$ **= 8.2** 40 processors is $410t/20t$ = **20.5**.

- Thus, for this larger problem size, we get 82% of the potential speed-up with 10 processors and 51% with 40.

Scaling

- ***Strong scaling***: Speed-up achieved on a multiprocessor without increasing the size of the problem.
- ***Weak scaling***: Speed-up achieved on a multiprocessor while increasing the size of the problem proportionally to the increase in the number of processors.
- **Load Balancing**
- If one processor has 5% of the parallel load, then it must do 5% × 400 or 20 additions, and the other 39 will share the remaining 380.





Rajan, Ranjidha

-Scaling – Strong Scaling / weak scaling / load balancing
-Process management = Threads

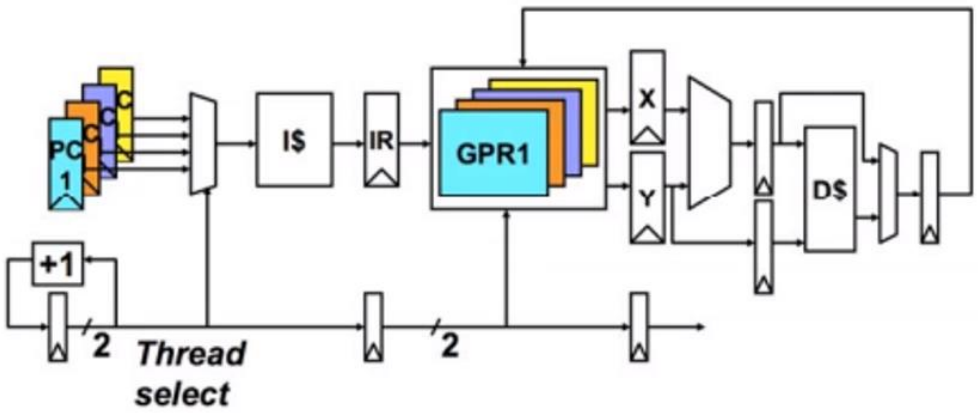
Processor Level -Hardware Multithreading

- ***Hardware multithreading***: Increasing utilization of a processor by switching to another thread when one thread is stalled.
- Process: Unique address space
- Thread : PC + Registers +Stack
  - Share address space

-Hardware Threading – OS manages

Processor Level -Hardware Multithreading

- Multithreaded Pipeline



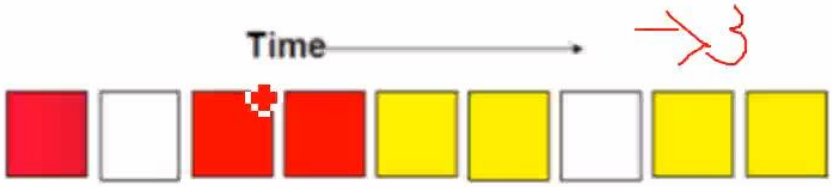
Hardware multithreading

Fine-grained multithreading: A version of hardware multithreading that implies switching between threads after every instruction.

-Fine-grained Multithreading

Hardware multithreading

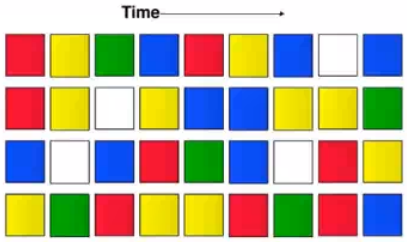
- ***Coarse-grained multithreading***: A version of hardware multithreading that implies switching between threads only after significant events, such as a last-level cache miss.
- Consider start-up overhead.



-Course-grained Multithreading

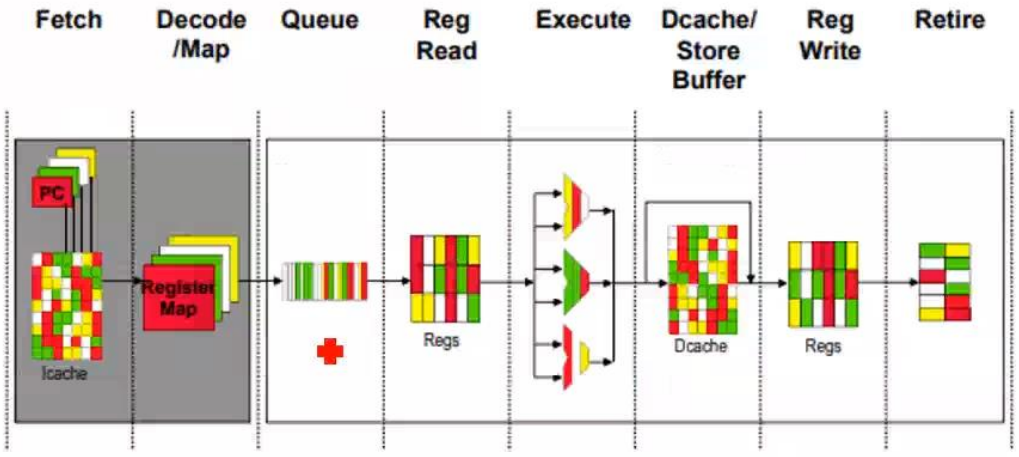
Hardware multithreading

- ***Simultaneous multithreading*** (***SMT***): A version of multithreading that lowers the cost of multithreading by utilizing the resources needed for multiple issue, dynamically scheduled microarchitecture.
- SMT is *always* executing instructions from multiple threads, leaving it up to the hardware to associate instruction slots and renamed registers with their proper threads.



-Simultaneous Multithreading (SMT)

SMT Pipeline



SMT Pipeline

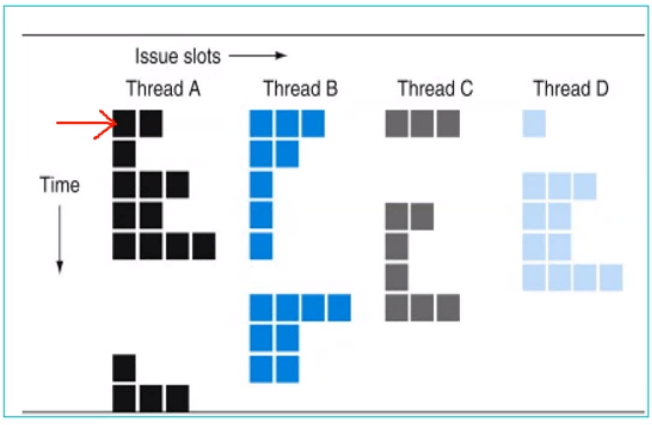
- Replicated resources
  - Program counter
  - Register map
  - Return address stack
  - Global history register
- Shared resources
  - Register file (size increased)
  - Instruction queue (scheduler)
  - First and second level caches
  - Translation lookaside buffers
  - Branch predictor

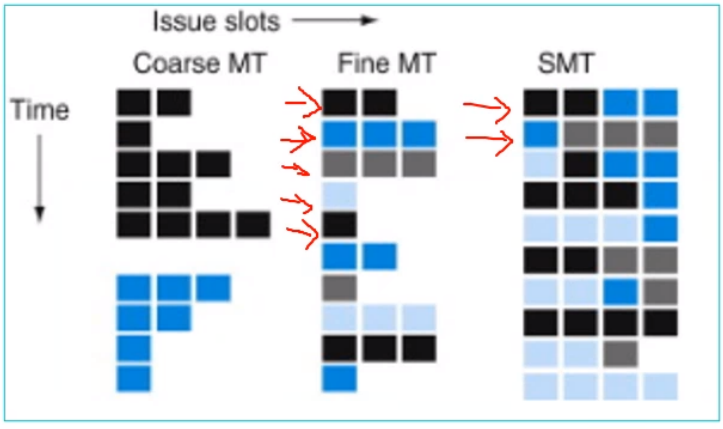
Which Thread to Fetch From?

Static
  Round Robin
Dynamic
  Threads with minimum outstanding misses

How to select threads to co-schedule ?
Snavely and Tullsen,
"Symbiotic Job scheduling for a Simultaneous Multithreading Processor,"
ASPLOS 2000.

SMT





Press Esc to exit full screen

Online Presentations – May 5th and 7th

- Sign Up sheets is available in Moodle ,Sign Up your Topic in class.
- All topics are from text book, You can refer any resources outside text.
- Presentation must be done using a power point presentation .
- Submit the name and topics each team members will be doing today (can change with consent of team members.
- Use MS Teams to team discussions on the presentation
- All team members participation in presentation will be counted.
- Prepare a rubric in an excel sheet for the peer evaluations. [You have to submit it with PPT after the presentation]
- Extra References will be counted.