2GA3 Tutorial #10

DATE: November 26th, 2021

TA: Jatin Chowdhary

I’m Back

- Guess who’s back
  - Back again
    - Shady’s back
      - Tell a friend
- We’re gonna catch up on missed content
  - Lucky for us, the stuff is easy peasy
    - I love ¢a¢h£
      - Doesn’t everyone?

So What Happened?

- **Question:** Why did we cancel last week (November 19th, 2021)?
- **Answer:** My left side (arm, shoulder, wrist, etc.) was *killing* me
  - It's (slowly) getting better now
  - But last week it was horrible
    - Couldn't even walk

Black Friday

- Today is Black Friday
- Go out, get some deals
  - Good deals
- Now's a good time to take a break (Just before exams)
  - Get some rest this weekend
    - Do some shopping
- Personally, I'm getting a *new* phone and changing ISPs
  - ISP = Internet Service Provider
- **Starting Monday, November 29th, it's grind time**
  - 3 weeks 'till 2GA3 exam

Another Re-Structure

- We’re going back to the old format
  - Turns out, everyone prefers this format
- The new (old) format is:
  - Administrative stuff
  - Review questions/material
  - Tutorial questions
- We’re gonna take it nice and easy
  - “Doctor’s orders”

Recordings Are Back

- Someone deleted the lectures/tutorials, but I was able to restore them
  - I've downloaded the lectures; in case it happens again
- *We need to change read/write permissions*
- Took some time to recover everything
  - Today's tutorial will be slow
    - But then again, ''doctor's orders''

Videos Are Coming

- I’m working on the scripts right now
  - And it’s Black Friday so I need a break

```
- Pipelining problems: Hazards (i.e. Structural, data, control)

- Hazards prevent the instruction from executing in the next cycle

- 3 kinds of hazards:
    - Structural
        - i.e. single read/write port
    - Data
    - Control

- The register file is READ in the 2nd half of the clock cycle
- The register file is WRITTEN in the 1st half of the clock cycle
    - This allows overlapping without ANY problems

- Data hazard: Caused by a prior instruction that is already in the pipeline
```

More Content

- And more content is planned
- We got 3 weeks till the final exam
  - **Final exam = 50%**
- Now's the time to push through
  - Take a break this weekend
    - And then *floor it*, starting Monday

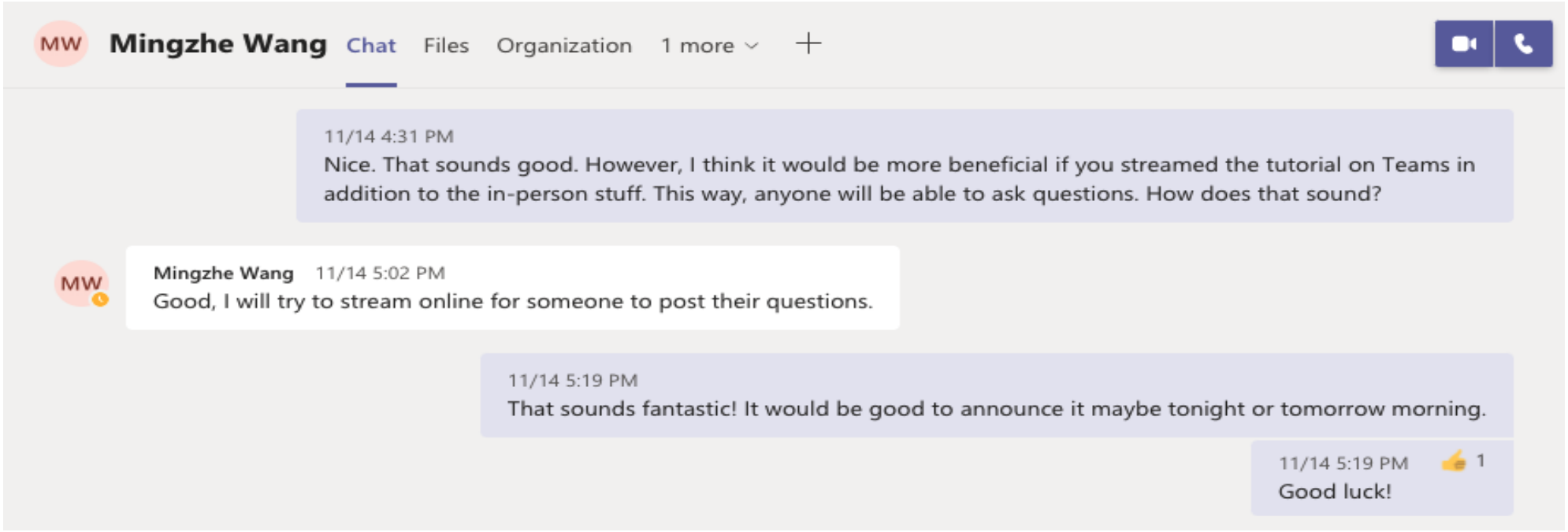
Ground Up

- Tutorials stand on their own 2 feet
  - So even if you're behind in lecture, you should still come to tutorial, because you'll learn something
    - And you don't need to know anything
- Also:
  - Slides are gonna be updated to be "whole"
    - This means going back and doing all the tutorial questions and then putting them in the slides
  - Everything will be done from the ground up

Listen Carefully

Always ASK!

- If you want something done a particular way, **ASK!**
  - And if I can't get it done, I'll find someone or something else to get it done
    - i.e.



Always ASK!

- **Life lesson:** Everything in life can be renegotiated and reneged
  - *Everything***.**
    - i.e. Loans, payments, due dates, deadlines, etc.
- So don't be afraid to ask
  - "You don't get what you don't ask for"
    - This applies to all facets of life
      - i.e. Getting a loan, applying for a job, etc.

Next Week

- The last tutorial for 2GA3 is:

  Friday, December 3rd, 2021

  - That's next week
    - 7 days from now
- Should you come?
  - **I promise I'll make it worth your while**
    - Yes you should
      - *Can't say more...*

Questions?
Comments?
Concerns?

Quick Review

- I need a general idea of where you're all at
  - Who looked at last week's content (Tutorial #9)?
  - How comfortable are you with the stuff on cache?
    - Not ca$h, but cache
      - C.R.E.A.M
  - How do you feel about the 50% final exam?
    - Other than pumping out slides and videos, what else can I do?
- Now that I know what your story is, I can navigate the swamp better
  - "You can never underestimate how wrong you can be"

Cache Review

- Why do we need cache?
  - Pretend your cooking pizza
    - Is it better if you have all the ingredients on the table where you can easily access them?

      OR
    - Is it better if you have to run to the pantry everytime you need an ingredient (i.e. Sauce, cheese, etc.)

      OR
    - Is it better if you have to run to the store for each ingredient, a different store each time?

Cache Example

- Obviously, it's better if the ingredients are immediately available and accessible on the table
  - Less time spent on acquiring ingredients, the better
    - We can spend more time on cooking
- Generally, the further we have to travel FROM the CPU, the L _ _ _ _ _ it takes to retrieve data
  - i.e.
    - Ingredients on the table = Registers
    - Ingredients in the pantry = Cache
    - Ingredients from neighbours = RAM
    - Ingredients from grocery store = HDD
      - HDD/SDD = S_ _ _ _ _ _ _ _ _ storage
    - Import ingredients from Italy = Magnetic tapes stored in a sketchy basement

L1/L2/L3 Cache

- **Question:** What is the difference between L1, L2, and L3 cache?
  - Somebody in lecture
- **Answer:** The simple answer is we want to cut down on ‘‘travel time’’. The less we need to use RAM/HDD, the better
  - Generally the rule is: L1 » L2 » L3
    - L1 is faster than L2, which is faster than L3
  - But they also serve a special purpose for parallel processing and concurrency
    - But this is way beyond the content of this course

Main Idea

- The main idea is: Going into memory is a pain
  - It is expensive, slow, and should be avoided at all costs
- Hence, programs and data are loaded ahead of time to avoid going back and forth between CPU and storage
  - Waiting for stuff to be loaded requires stalls
    - Memory stalls = Wasted CPU cycles
      - L _ _ _ performance
- From a high level, this is all you need to know about cache

Learning Time

(Straighten Up)

Outline

- Quickly review last week's content to bring everyone up to speed
  - I'm not gonna assume everyone reviewed tutorial #9
    - **I now know better; do not ASSUME**
      - "You can never underestimate how wrong you can be"
- First question from Tutorial #10
  - *It's gonna make so much sense!*

Questions?
Comments?
Concerns?

Tutorial Question #1

- **Question:** Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 64-bit memory address references, given as word addresses.

  0x03, 0xb4, 0x2b, 0x02, 0xbf, 0x58, 0xbe, 0x0e, 0xb5, 0x2c, 0xba, 0xfd

  - For each of these references, identify the binary word address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list whether each reference is a hit or a miss, assuming the cache is initially empty.

Precursor

- Important things to know:
  - Direct-mapped cache
  - Cache is initially empty
  - In total: 16 one-word blocks
- Given the word address, the question wants the binary word address, the tag, the index, and whether each reference is a hit or a miss.
  - *Next slide*

Binary Address

- Given the word address, the question wants the binary word address, the tag, the index, and whether each reference is a hit or a miss.
  - The binary word address is the word address, but in binary
    - Convert word address to binary (0x → 0b)

Word Address	Binary Address
0x03	0b00000011
0xb4	0b10110100
0x2b	0b00101011

Index

- **Question:** How to calculate the index size?
  - We have to use the cache size
  - Recall: *given a direct-mapped cache with 16 one-word blocks*
    - *So what's the size of the cache? Rather, how many blocks are in this direct-mapped cache?*
- **Answer:** The size of the index is: 4
  - $16 = 2^n$ (What is *n*?)
    - n = index size (in bits)
      - Start from least significant bits

Tag

- **Question:** So what's the size of the tag?
  - We have to use the index size to compute the tag size
    - Recall: *The size of the index is 4*
      - *And if we only have 8 bits to work with, and 4 are used for the index, then the remaining 4 are used for the tag*
- **Answer:** The size of the tag is: 4
  - 8 – 4 = 4
    - Start tag from where index left off

Putting It All Together

- So the index is the 4 bits from the right, or the least significant bits
  - The tag is 4 bits after the index

Word Address	Binary Address	Tag	Index
0x03	0b00000011	0	3
0xb4	0b10110100	b	4
0x2b	0b00101011	2	b

Hits/Misses

- **Question:** How do we know if something is a cache hit or cache miss?
  - Cache hit means requested data was found in the cache and can be quickly brought back
  - Cache miss means that the data was not found in the cache and now we need to go to the next level in the memory hierarchy
- **Answer:** Look at the index and tag
  - Index tells you where the data is located
    - Tag tells you if there is a ‘‘match’’
  - *Homework: Figure it out!*

Tutorial Question #2

- **Question:** Consider an Intel microprocessor with a 16 Kbyte unified L1 cache. The miss rate for this cache is 3% and the hit time is 2 CCs. The processor also has an 8 Mbyte, on-chip L2 cache. 95% of the time, data requests to the L2 cache are found. If data is not found in the L2 cache, a request is made to a 4 Gbyte main memory. The time to service a memory request is 100,000 CCs. On average, it takes 3.5 CCs to process a memory request. How often is data found in main memory? *Assume we also know L2 hit time = 15 CCs and Main hit time = 200 CCs.*

Pizza Time!

- **Precursor:** Think back to our Pizza making example
  - Assume that ingredients for the pizza are scattered across: The table, pantry, basement, neighbour's house, and the grocery store
    - The quickest is: Table
    - The slowest is: Grocery store
  - So, if I wanted to calculate the time it takes to make a pizza, minus cooking time, what do I do?
    - Time spent on gathering ingredients is included
      - *Next slide!*

Pizza Time!

- **Precursor:** Example of cooking time:
  - Opening packaging
  - Spread sauce on the base
  - Grate cheese, but the grater is in the pantry
    - Run to the pantry, and then grate cheese
  - Cut onions, mushrooms, etc.
  - Decide to add pineapples
    - We don't have any, so we ask the neighbour
      - Run to the neighbour's house
      - He doesn't have any, so we run to the grocery store
- **Question:** Out of all this stuff what slows us down the most?

Pizza Analogy!

- **Precursor:** Think about it this way
  - Working on the table/counter is *like* working with registers
    - i.e. Cutting vegetables, spreading sauce, etc.
  - Running to the pantry to grab the grater is *like* fetching data from L1 cache
    - Takes time, but generally very quick
  - Checking with the neighbour's if they have pineapples is like checking L2 cache if the data we need is there
    - If the neighbour's don't have pineapples, then we gotta run to the grocery store, which is the most time consuming
      - Fetching data from main memory is also time consuming

Time Is Of The Essence

- **Precursor:** Total time?
  - So, the total time to make a pizza is:
    - Time spent unwrapping stuff, spreading sauce
    - Time spent running to the pantry
    - Time spent grating cheese
    - Time spent checking with the neighbour
    - Time spent running to Whole Foods
  - Similarly, if something isn't available in L1 cache, we check L2 cache, then we check main memory (RAM)

Tutorial Precursor #2

- **Precursor:** Whenever you do a question like this, you wanna separate the information for L1, L2, LN, main memory (RAM), etc.
  - L1
    - Hit time = 2
    - Miss rate = 3% = *0.03*
  - L2
    - Hit time = 15
    - Hit rate = 15% = *0.15*
    - Miss rate = 1 – 0.15 = 0.85 *(or 85%)*
  - RAM
    - *Next slide*

Tutorial Precursor #2

- **Precursor:** Whenever you do a question like this, you wanna separate the information for L1, L2, LN, main memory (RAM), etc.
  - RAM (Main Memory)
    - Hit time = 200
    - Miss penalty = 100,000
  - We know that the average memory request is 3.5
- And the equation we need is:

  Average memory access time **=** *Hit Time* **+**

  (Miss Rate ***** *Miss Penalty)*

Tutorial Precursor #2

- **Precursor:** But before we solve this, you need to understand that if there's a miss in L1, then we need to check L2.
  - So the *Miss_Penalty* of L1 is the *AMAT* of L2
    - Does everyone understand this?
  - Recap:
    - If there is no hit in L1, then we need to check L2. So the time it takes to check L2 is the *Miss_Penalty* of L1

Tutorial Answer #2

- **Answer:** We need our equation:
  - Note:
    - AMAT = Average Memory Access Time
    - HT = Hit Time
    - MR = Miss Rate
    - MP = Miss Penalty
- AMAT = HT + (MR x MP)

Tutorial Answer #2

- **Answer:** Let's color it up!
  - $AMAT = HT_{L1} + (MR_{L1} \times \mathbf{MP_{L1}})$
  - $\mathbf{MP_{L1}} = AMAT_{L2}$
  - $AMAT_{L2} = HT_{L2} + (MR_{L2} \times \mathbf{MP_{L2}})$
  - $\mathbf{MP_{L2}} = AMAT_{RAM}$
  - $AMAT_{RAM} = HT_{RAM} + (MR_{RAM} \times MP_{RAM})$

Tutorial Answer #2

- **Answer:** Let's do it ourselves

THE END