# ADVANCED MEMORY CONTROLLERS

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

# Memory Controller

- Memory controller connects CPU and DRAM
- Receives requests after cache misses in LLC
  - Possibly originating from multiple cores
- Complicated piece of hardware, handles:
  - DRAM refresh management
  - Command scheduling
  - Row-buffer management policies
  - Address mapping schemes

# DRAM Control Tasks

- Refresh management
  - Periodically replenish the DRAM cells (burst vs. distributed)
- Address mapping
  - Distribute the requests to destination banks (load balancing)
- Request scheduling
  - Generate a sequence of commands for memory requests
    - Reduce overheads by eliminating unnecessary commands
- Power management
  - Keep the power consumption under a cap
- Error detection/correction
  - Detect and recover corrupted data

# Address Mapping

- A memory request

| Type | Address | Data |
|---|---|---|

- Address is used to find the location in memory
  - Channel, rank, bank, row, and column IDs
- Example physical address format

| Row ID | Channel ID | Rank ID | Bank ID | Column ID |
|---|---|---|---|---|

- A 4GB channel, 2 ranks, 4 banks/rank, 8KB page

# Address Mapping

- A memory request

| Type | Address | Data |
|---|---|---|

- Address is used to find the location in memory
  - Channel, rank, bank, row, and column IDs
- Example physical address format

| Row ID | Channel ID | Rank ID | Bank ID | Column ID |
|---|---|---|---|---|
| 16 | 0 | 1 | 2 | 13 |

- A 4GB channel, 2 ranks, 4 banks/rank, 8KB page

# Example Problem

- Start with empty row buffers, find the total number of commands if all the request are served in order
  - Address= row(12):channel(0):rank(1):bank(3):column(16)

*addr*

00000010

20000001

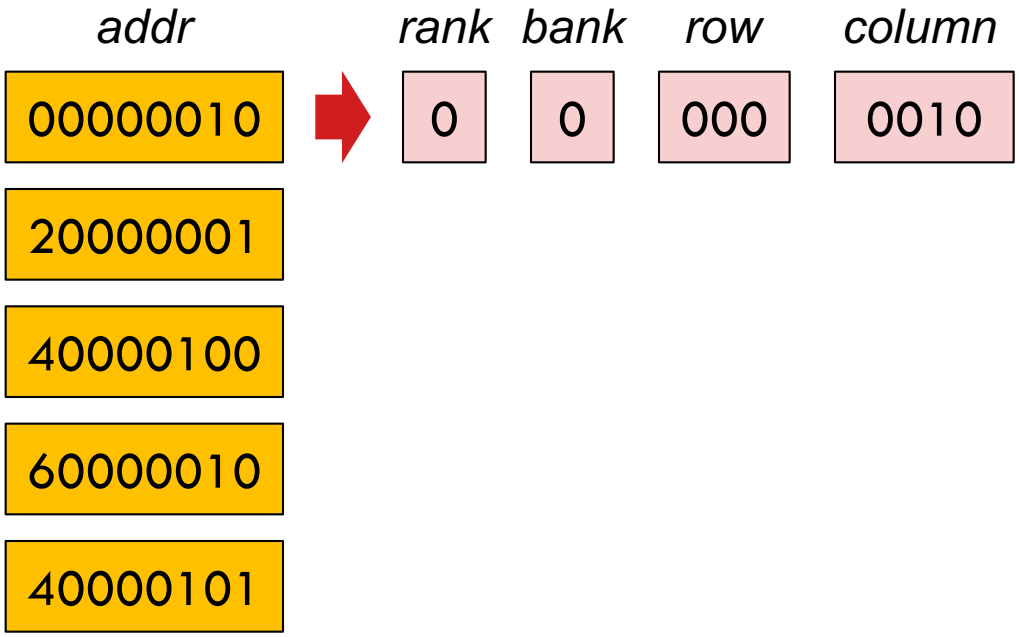
40000100

60000010

40000101

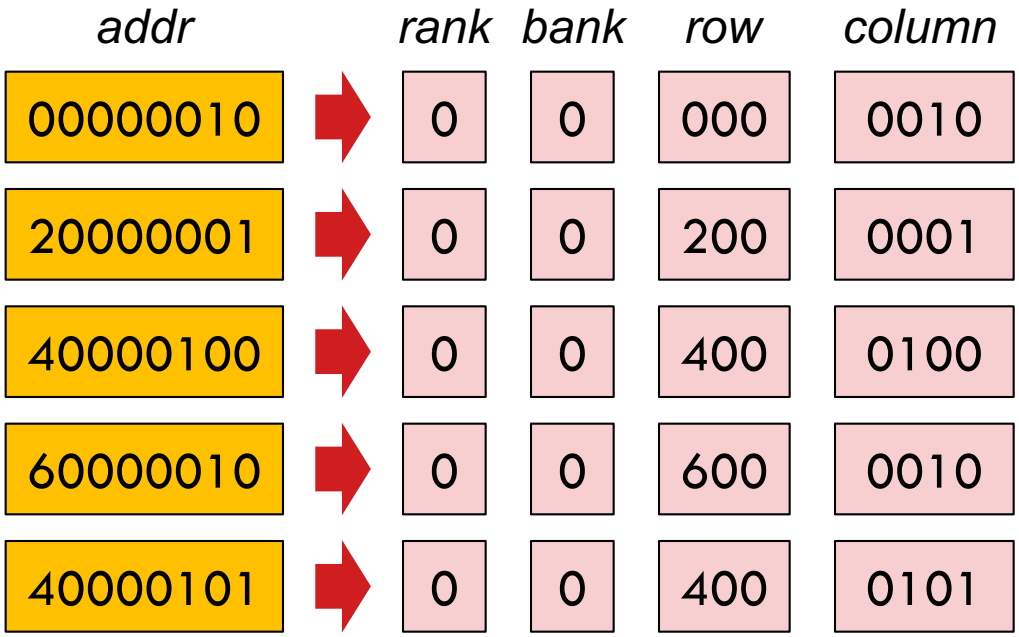
# Example Problem

- Start with empty row buffers, find the total number of commands if all the request are served in order
  - Address= row(12):channel(0):rank(1):bank(3):column(16)



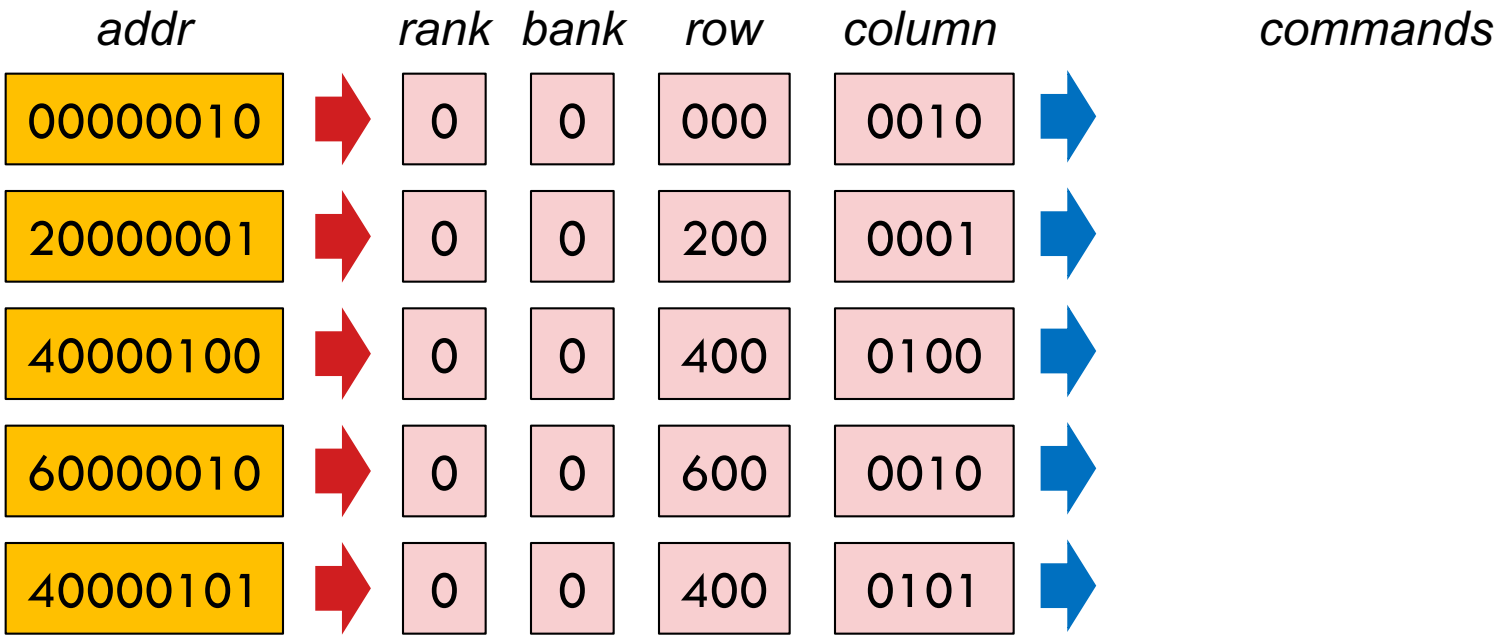
# Example Problem

- Start with empty row buffers, find the total number of commands if all the request are served in order
  - Address= row(12):channel(0):rank(1):bank(3):column(16)



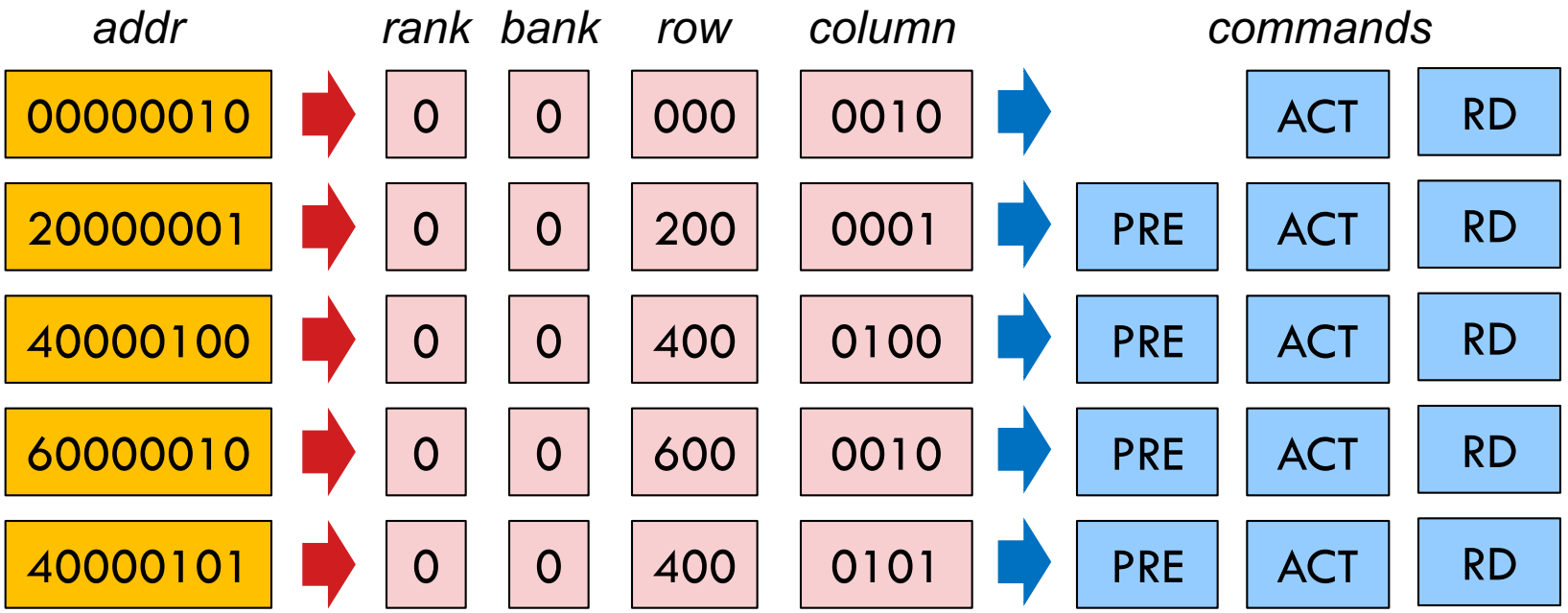
# Example Problem

- Start with empty row buffers, find the total number of commands if all the request are served in order
  - Address= row(12):channel(0):rank(1):bank(3):column(16)



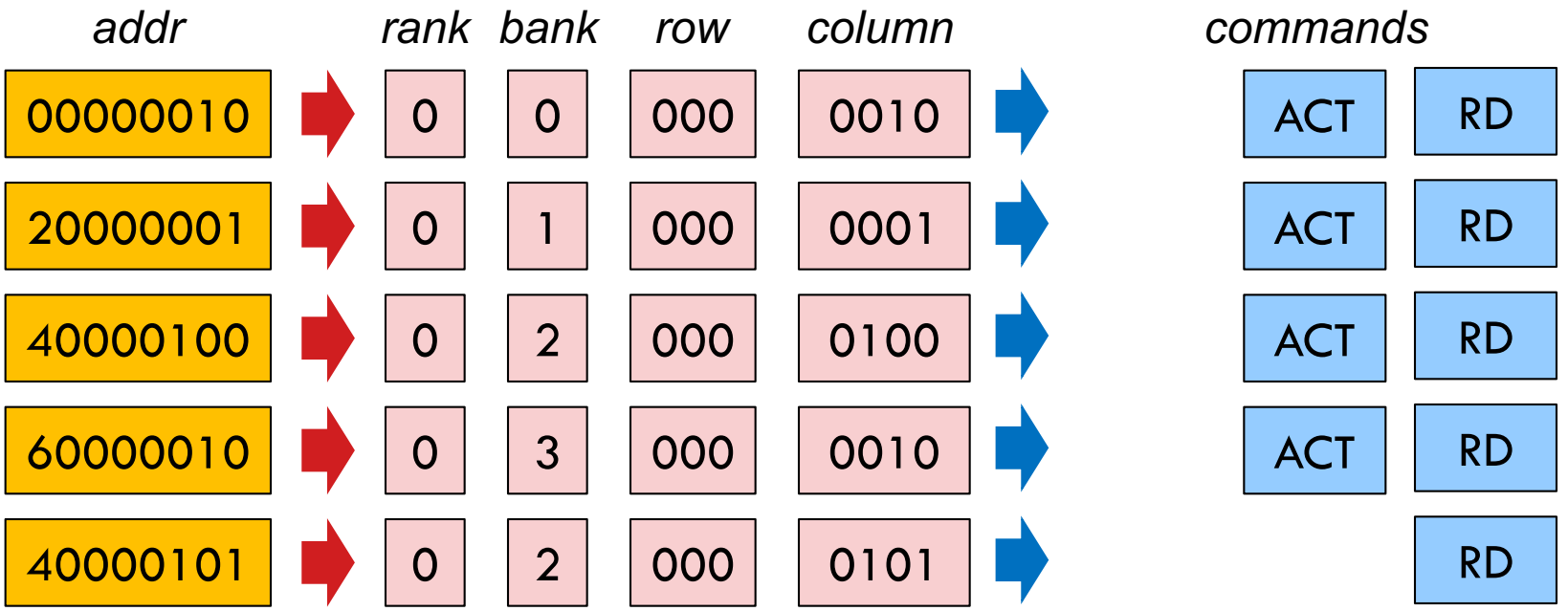
# Example Problem

- Start with empty row buffers, find the total number of commands if all the request are served in order
  - Address= row(12):channel(0):rank(1):bank(3):column(16)



# Example Problem

- Find the total number of commands using the following address mapping scheme
  - Address= bank(3):rank(1):channel(0):row(12):column(16)



# Example Problem

- Find the total number of commands using the following address mapping scheme
  - Address= bank(3):rank(1):channel(0):row(12):column(16)

*addr*

00000010

20000001

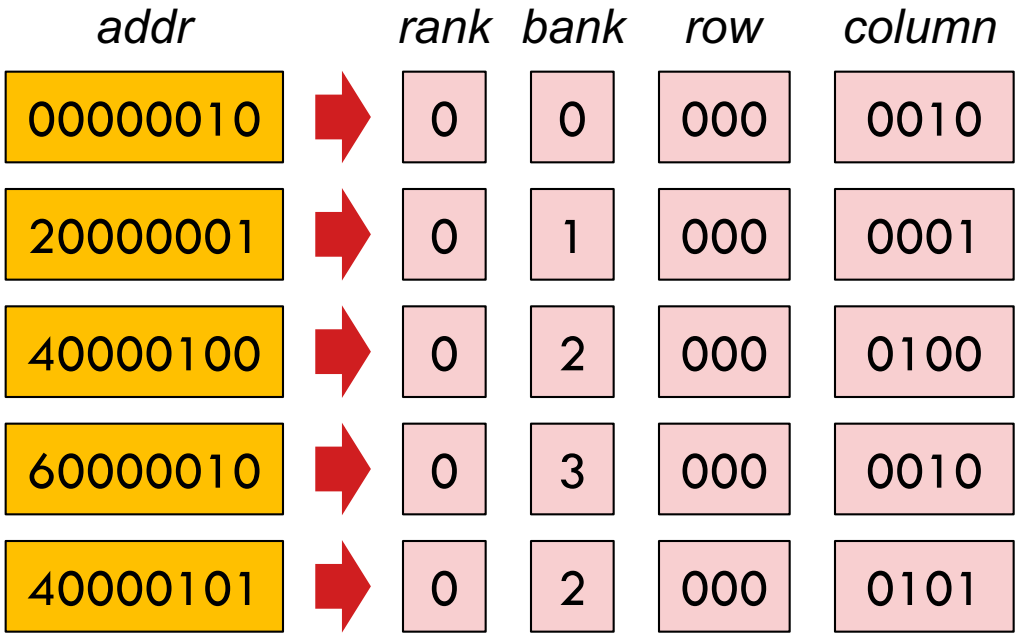
40000100

60000010

40000101

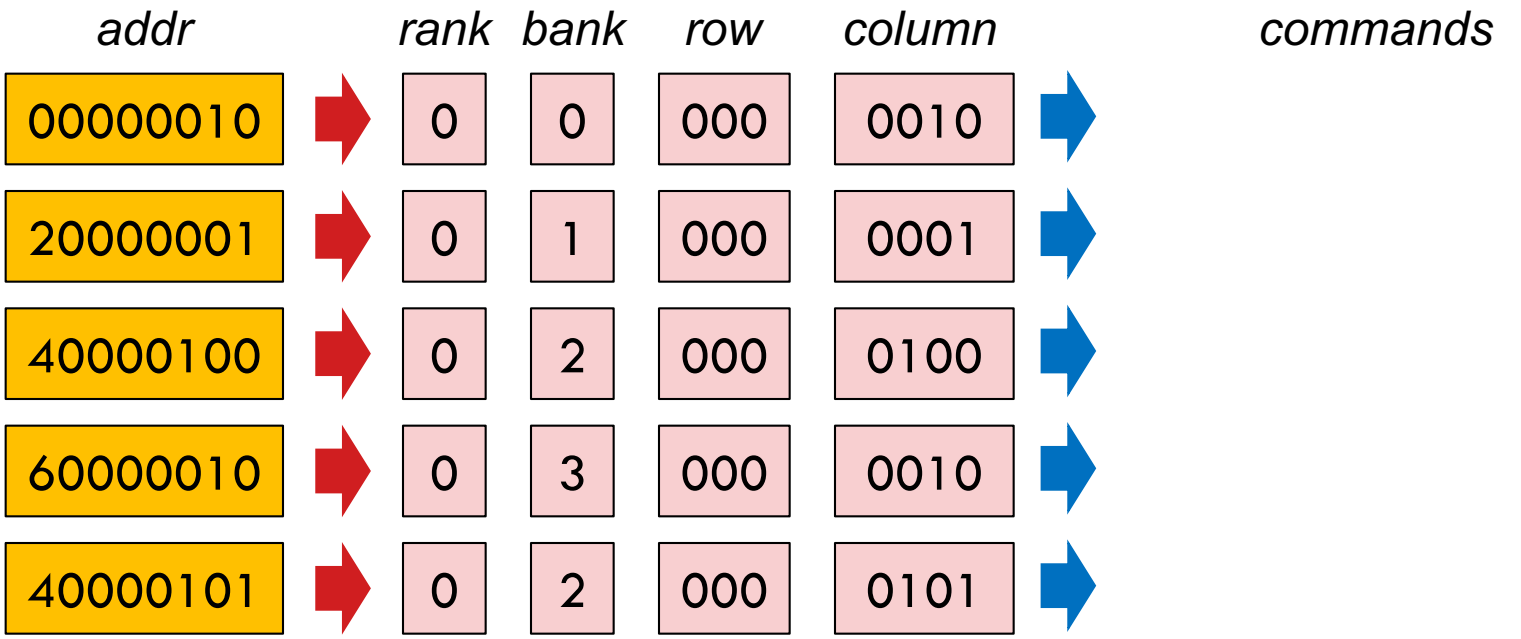
# Example Problem

- Find the total number of commands using the following address mapping scheme
  - Address= bank(3):rank(1):channel(0):row(12):column(16)



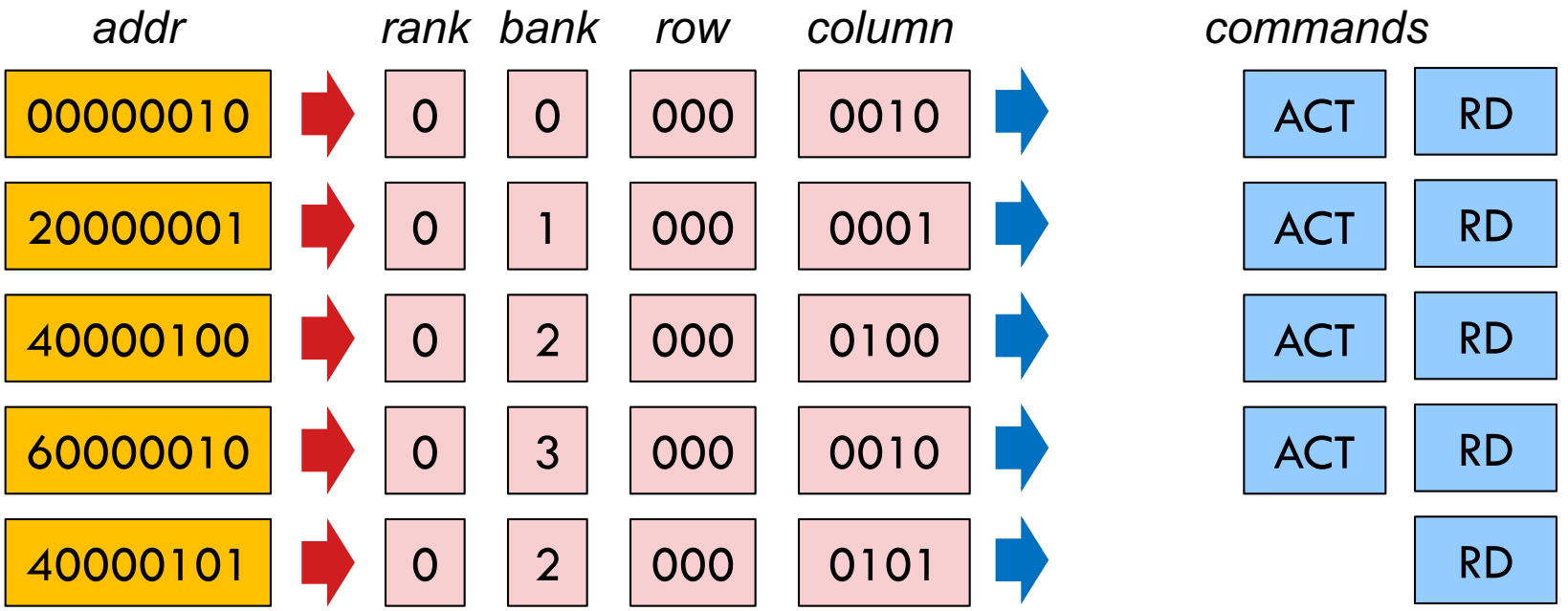
# Example Problem

- Find the total number of commands using the following address mapping scheme
  - Address= bank(3):rank(1):channel(0):row(12):column(16)



# Example Problem

- Find the total number of commands using the following address mapping scheme
  - Address= bank(3):rank(1):channel(0):row(12):column(16)



# Command Scheduling

- Write buffering
  - Writes can wait until reads are done
- Controller queues DRAM commands
  - Usually into per-bank queues
  - Allows easily reordering ops. meant for same bank
- Common policies
  - First-Come-First-Served (FCFS)
  - First-Ready First-Come-First-Served (FR-FCFS)

# Command Scheduling

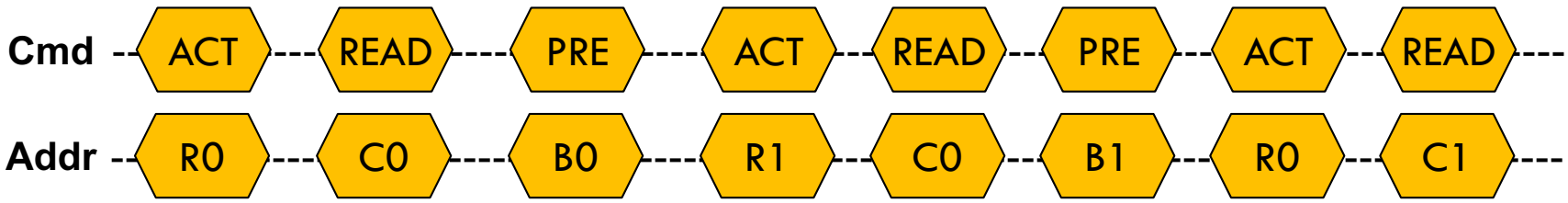
- First-Come-First-Served
  - Oldest request first
- First-Ready First-Come-First-Served
  - Prioritize column changes over row changes
  - Skip over older conflicting requests
  - Find row hits (on queued requests)
    - Find oldest
    - If no conflicts with in-progress request → good
    - Otherwise (if conflicts), try next oldest

# FCFS vs. FR-FCFS

- READ(B0,R0,C0) READ(B0,R1,C0) READ(B0,R0,C1)
  - FCFS

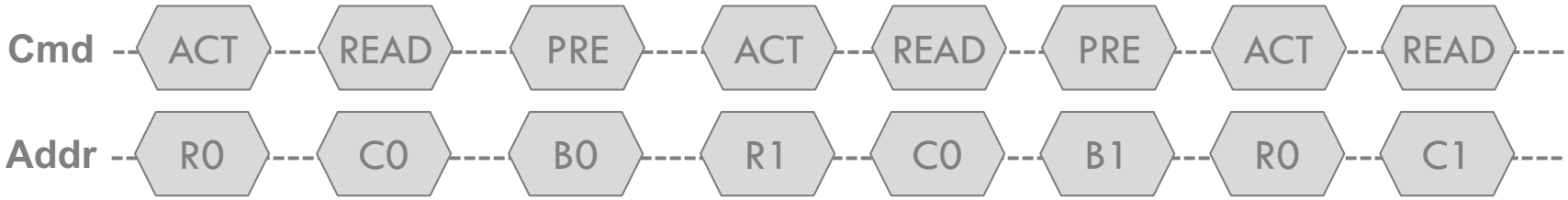
# FCFS vs. FR-FCFS

- READ(B0,R0,C0) READ(B0,R1,C0) READ(B0,R0,C1)
  - FCFS



# FCFS vs. FR-FCFS

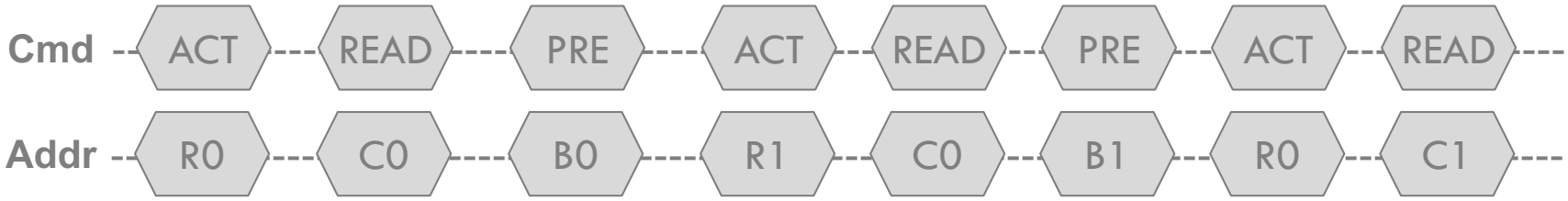
- READ(B0,R0,C0) READ(B0,R1,C0) READ(B0,R0,C1)
  - FCFS



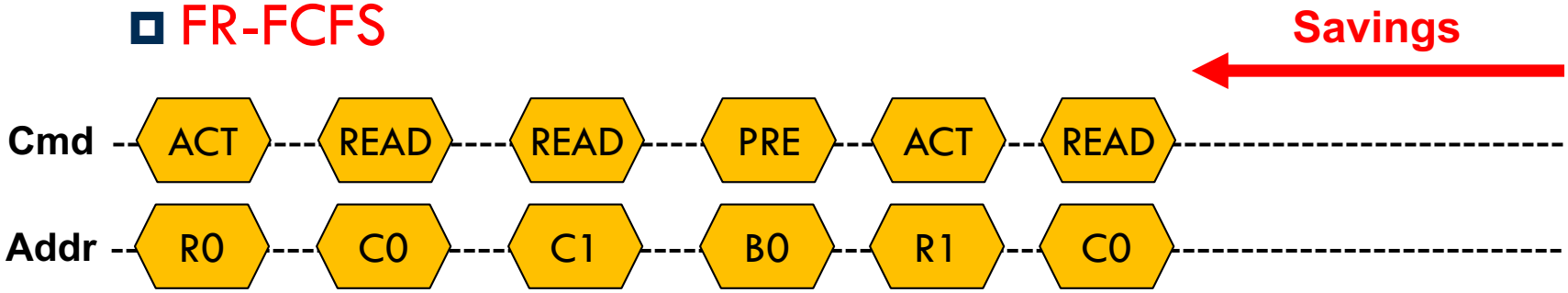

  - FR-FCFS

# FCFS vs. FR-FCFS

- READ(B0,R0,C0) READ(B0,R1,C0) READ(B0,R0,C1)
  - FCFS




  - FR-FCFS



# Row Buffer Management Policies

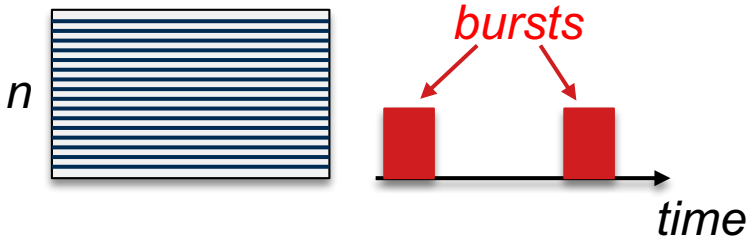
- Open-page policy
  - After access, keep page in DRAM row buffer
  - If access to different page, must close old one first
    - Good if lots of locality
- Close-page policy
  - After access, immediately close page in DRAM row buffer
  - If access to different page, old one already closed
    - Good if no locality (random access)

# DRAM Refresh Management

- DRAM requires the cells' contents to be read and written periodically

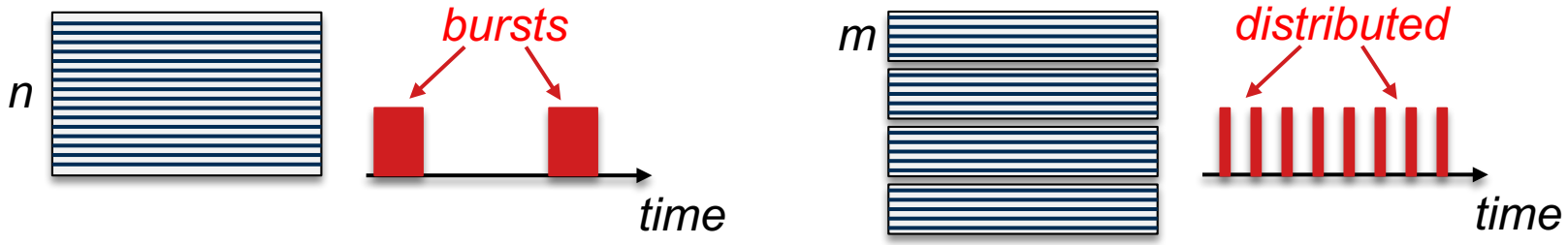
# DRAM Refresh Management

- DRAM requires the cells' contents to be read and written periodically
  - Burst refresh: refresh all of the cells each time
    - Simple control mechanism



# DRAM Refresh Management

- DRAM requires the cells' contents to be read and written periodically
  - Burst refresh: refresh all of the cells each time
    - Simple control mechanism
  - Distributed refresh: a group of cells are refreshed
    - Avoid blocking memory for a long time



# DRAM Refresh Management

- DRAM requires the cells' contents to be read and written periodically
  - Burst refresh: refresh all of the cells each time
    - Simple control mechanism
  - Distributed refresh: a group of cells are refreshed
    - Avoid blocking memory for a long time
- Recently accessed rows need not to be refreshed
  - Smart refresh