Emerging Computing Architectures

Asif Ali Khan

Fall Semester 2024

Department of Computer Systems Engineering

UET Peshawar

Expectations

Prerequisite:

- Computer Architecture and/or Digital System Design

Expectations

Prerequisite:

- Computer Architecture and/or Digital System Design

What not to expect from this course:

- The fundamentals of computer architecture/digital logic and system design
- In-depth (cross-layer) details of a particular architecture
- Spoon-feeding

Expectations

Prerequisite:

- Computer Architecture and/or Digital System Design

What not to expect from this course:

- The fundamentals of computer architecture/digital logic and system design
- In-depth (cross-layer) details of a particular architecture
- Spoon-feeding

What to expect from this course:

- Introduction to the state-of-the-art research in computing systems
- The landscape of novel methods in computing
- A lot of self-reading (mostly research articles)

Von Neumann Architecture and the Main Memory Subsystem

Fixed program vs. stored program computers

- Historically, there have been two types of computers:

Fixed program vs. stored program computers

- Historically, there have been two types of computers:
  - Fixed program: fixed function, not reprogrammable (e.g., calculators)

Fixed program vs. stored program computers

- Historically, there have been two types of computers:
  - Fixed program: fixed function, not reprogrammable (e.g., calculators)
  - Stored program: Reprogrammable; applications are stored in memory, hence the name

Fixed program vs. stored program computers

- Historically, there have been two types of computers:
  - Fixed program: fixed function, not reprogrammable (e.g., calculators)
  - Stored program: Reprogrammable; applications are stored in memory, hence the name

- Modern systems are based on stored-program concept
  - Introduced by John Von Neumann

Von-Neumann architecture

- Consists of three basic components:
  1. CPU
  2. Memory
  3. I/O interfaces

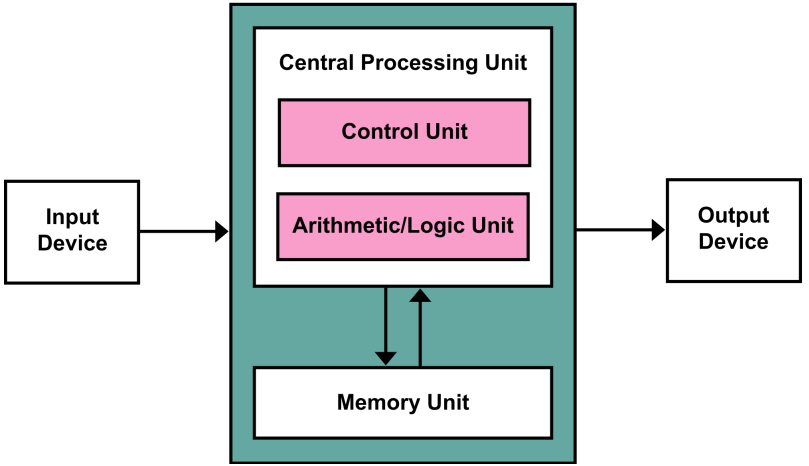


Image source: Wikipedia

Von-Neumann architecture

- Consists of three basic components:
  1. CPU
  2. Memory
  3. I/O interfaces
- All of them consists of multiple sub-components

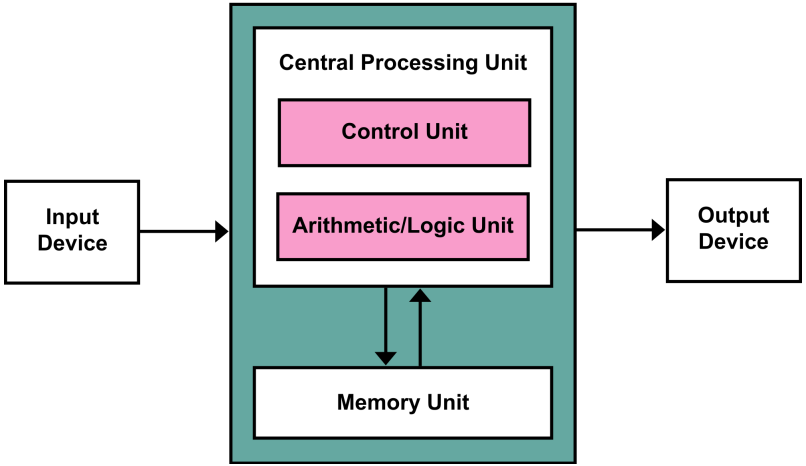


Image source: Wikipedia

Von-Neumann architecture

- Consists of three basic components:
  1. CPU
  2. Memory
  3. I/O interfaces
- All of them consists of multiple sub-components
- CPU: ALU, FPU, Control Unit, Registers

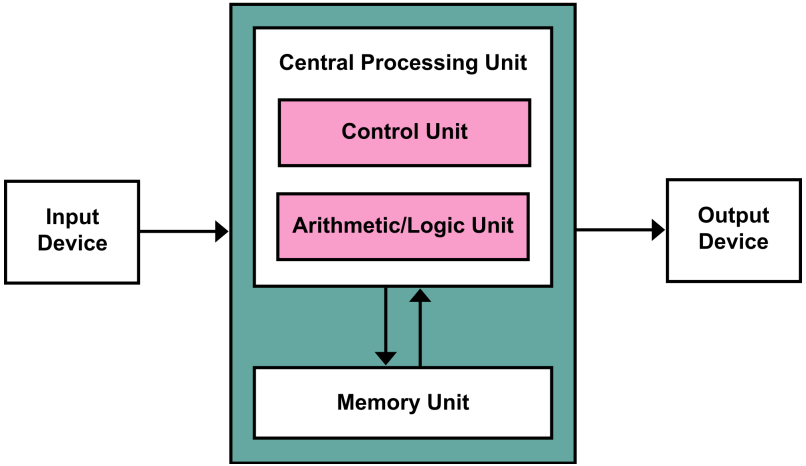


Image source: Wikipedia

Von-Neumann architecture

- Consists of three basic components:
  1. CPU
  2. Memory
  3. I/O interfaces
- All of them consists of multiple sub-components
- CPU: ALU, FPU, Control Unit, Registers

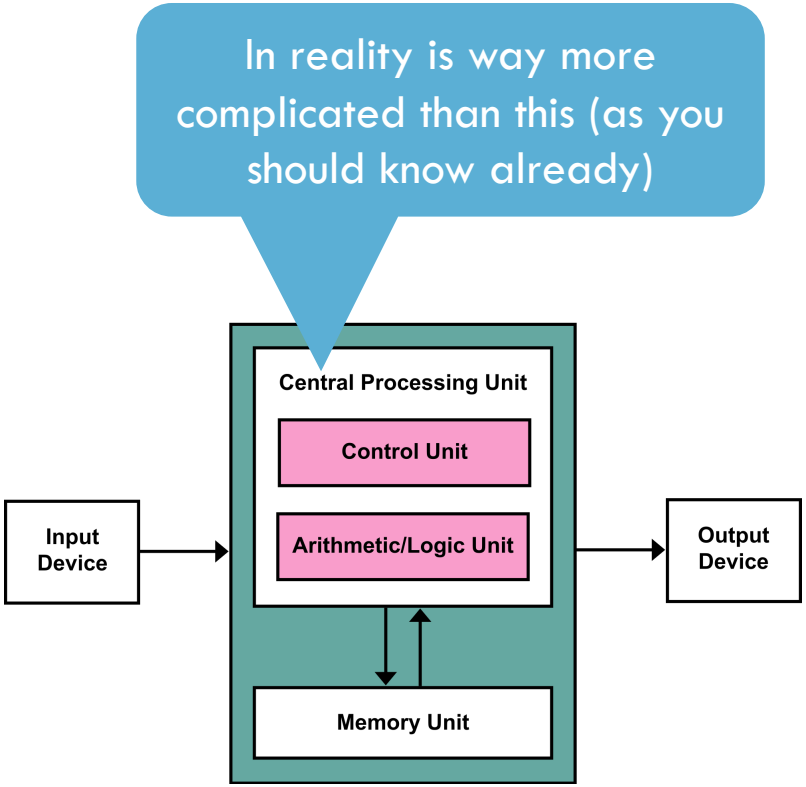


Image source: Wikipedia

Von-Neumann architecture

- Consists of three basic components:
  1. CPU
  2. Memory
  3. I/O interfaces
- All of them consists of multiple sub-components
- CPU: ALU, FPU, Control Unit, Registers
- Buses, represented with arrows, work as data/instruction carrying pathways

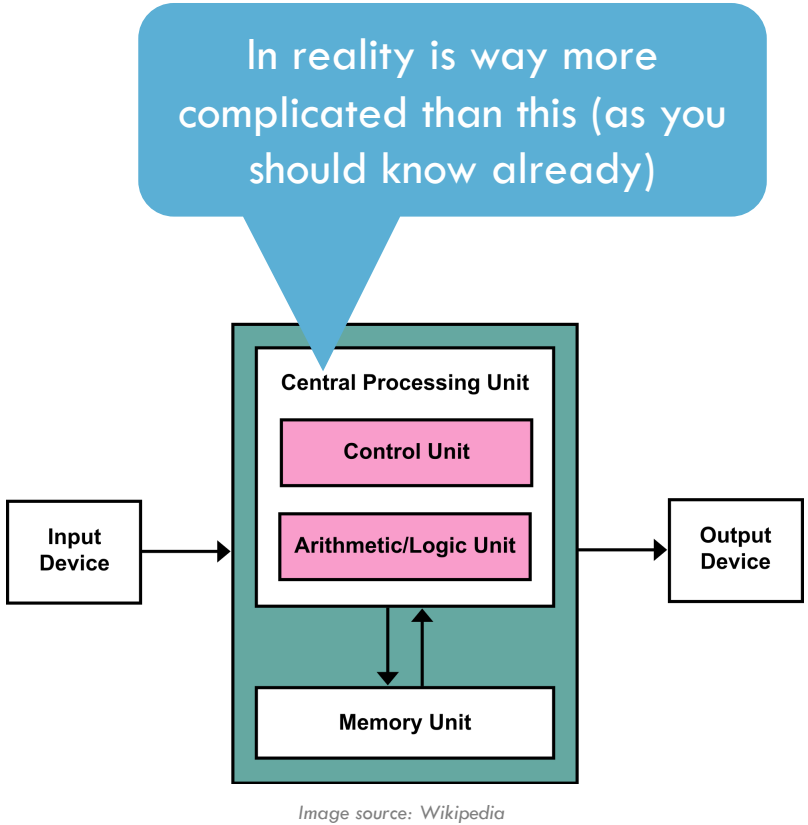


Image source: Wikipedia

Alternative architectures

Harvard architecture

- Separates data and instruction memory and their buses
- Both can be accessed simultaneously
- Used in embedded systems and digital signal processors
- Limitations: Complexity, non-flexibility, limited code size

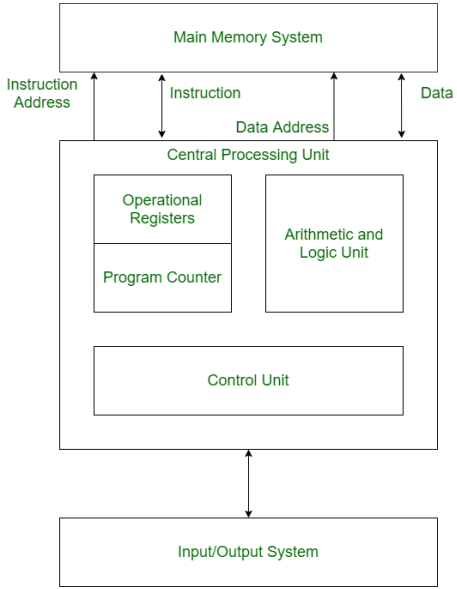


Image source: Wikipedia

Alternative architectures

Harvard architecture

- Separates data and instruction memory and their buses
- Both can be accessed simultaneously
- Used in embedded systems and digital signal processors
- Limitations: Complexity, non-flexibility, limited code size

Dataflow (DF) architecture

- Harvard and Von Neumann are control-flow driven
- In DF architectures, instructions are executed in data-flow order
  - Instructions are executed as soon as the operands are available
- In a DF machine, a program consists of DF nodes

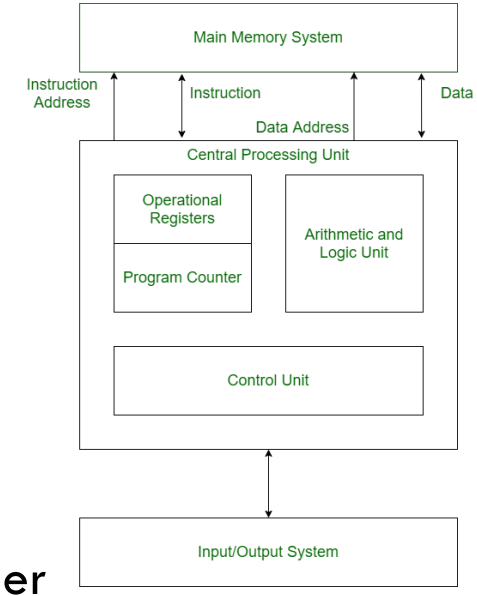


Image source: Wikipedia

Alternative architectures

Harvard architecture

- Separates data and instruction memory and their buses
- Both can be accessed simultaneously
- Used in embedded systems and digital signal processors
- Limitations: Complexity, non-flexibility, limited code size

Dataflow (DF) architecture

- Harvard and Von Neumann are control-flow driven
- In DF architectures, instructions are executed in data-flow order
  - Instructions are executed as soon as the operands are available
- In a DF machine, a program consists of DF nodes
- More details: Dennis and Misunas, "A Preliminary Architecture for a Basic Data Flow Processor," ISCA 1974.

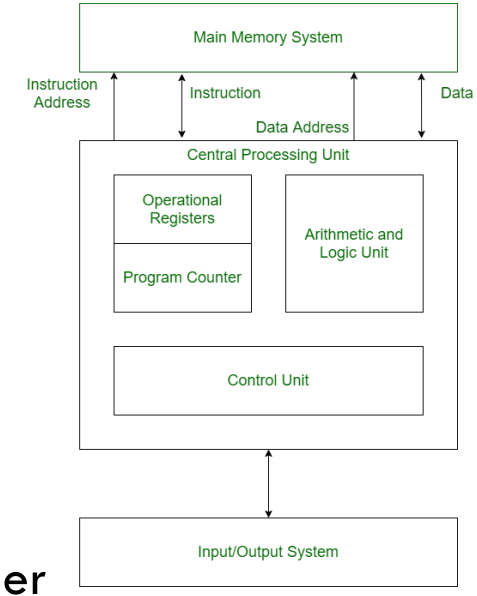
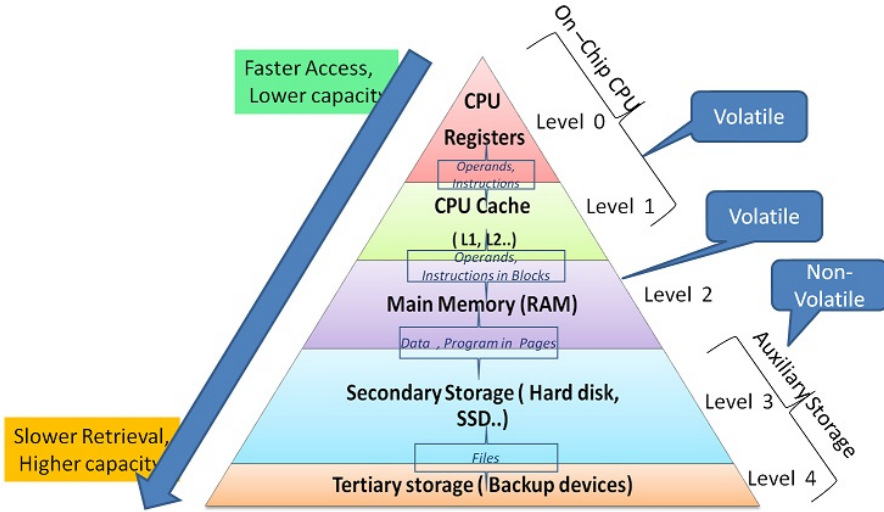


Image source: Wikipedia

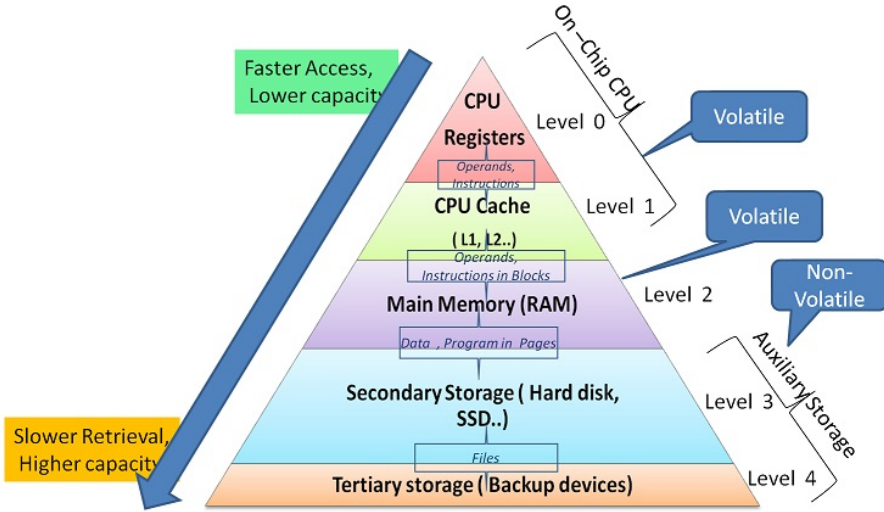
The memory hierarchy



Bill Dally, MICRO, 2019

The memory hierarchy

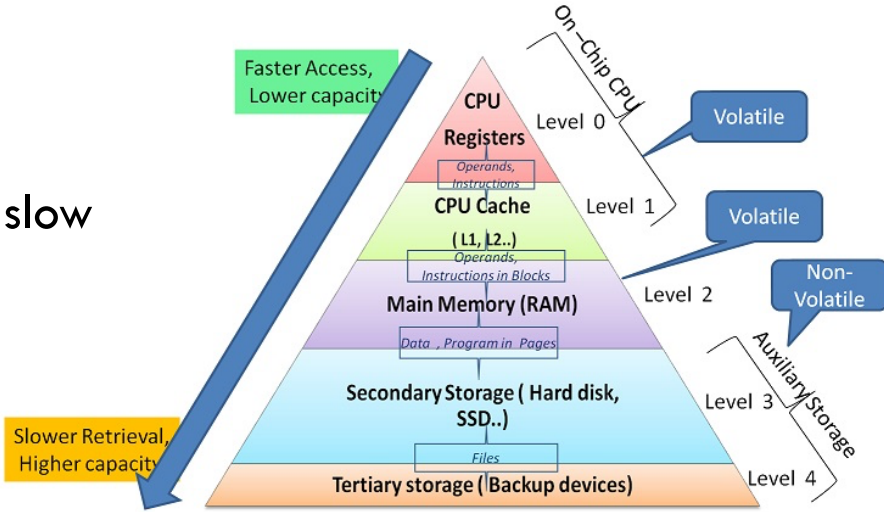
- CPU registers are fastest but extremely small size
  - Usually a few bytes
  - Single cycle access



Bill Dally, MICRO, 2019

The memory hierarchy

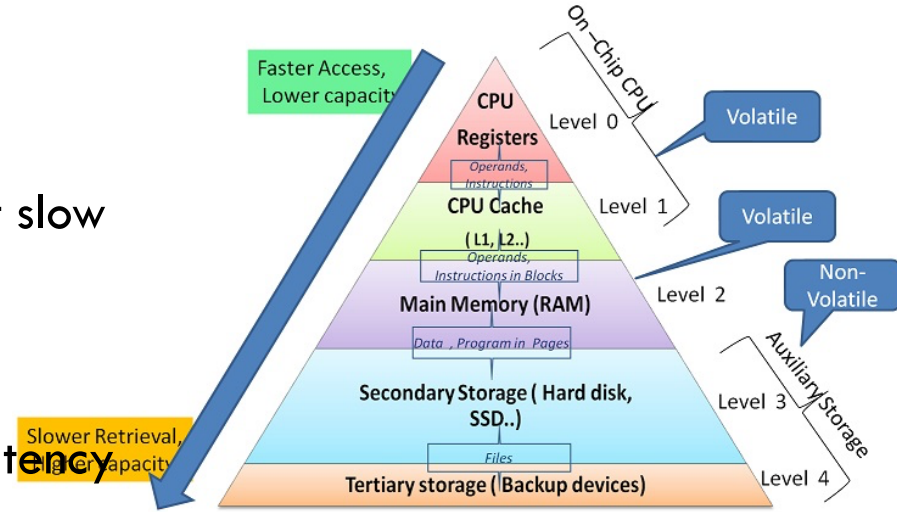
- CPU registers are fastest but extremely small size
  - Usually a few bytes
  - Single cycle access
- Storage devices are extremely dense but slow
  - Sizing in tera/peta bytes
  - Access time is usually in millisconds



Bill Dally, MICRO, 2019

The memory hierarchy

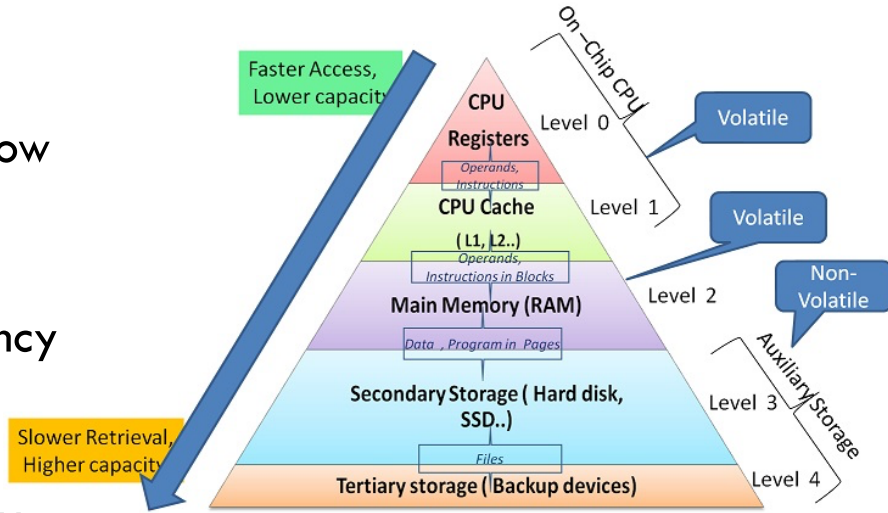
- CPU registers are fastest but extremely small size
  - Usually a few bytes
  - Single cycle access
- Storage devices are extremely dense but slow
  - Sizing in tera/peta bytes
  - Access time is usually in millisconds
- Main memory hides the storage device latency
  - Typical access latency is in microseconds
  - Size is typically giga/tera bytes



Bill Dally, MICRO, 2019

The memory hierarchy

- CPU registers are fastest but extremely small size
  - Usually a few bytes
  - Single cycle access
- Storage devices are extremely dense but slow
  - Sizing in tera/peta bytes
  - Access time is usually in millisconds
- Main memory hides the storage device latency
  - Typical access latency is in microseconds
  - Size is typically giga/tera bytes
- Caches hide the main memory access latency
  - Typically in kilo/mega byte ranges
  - Access latency in nanoseconds



Bill Dally, MICRO, 2019

Caches

- Caches are used to exploit *memory locality*

Caches

- Caches are used to exploit *memory locality*
- Memory locality is the principle that consecutive memory accesses are closed to each other
  - Temporal locality: Close in time
  - Spatial locality: Close in space

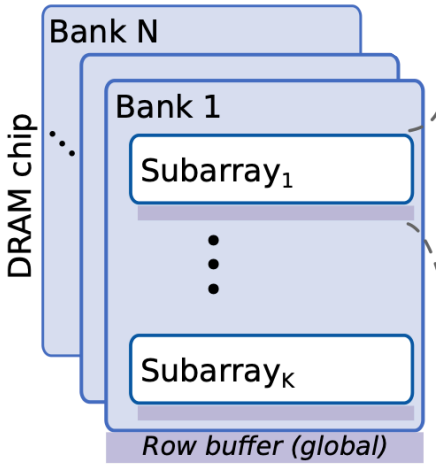
Caches

- Caches are used to exploit *memory locality*
- Memory locality is the principle that consecutive memory accesses are closed to each other
  - Temporal locality: Close in time
  - Spatial locality: Close in space
- Cache contents depend on the cache structure and its management policies
  - Reading material: *Jaleel et al, Achieving Non-Inclusive Cache Performance with Inclusive Caches: Temporal Locality Aware (TLA) Cache Management Policies, MICRO 2010*

Main memory organization

- Consists of channels, ranks, and banks

- Each bank can have one or more subarrays



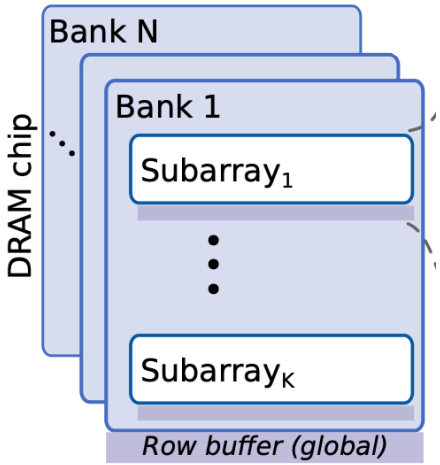
Sishadri et al, MICRO 2017
Lima et al., Arxiv, 2024

Main memory organization

- Consists of channels, ranks, and banks

- Each bank can have one or more subarrays

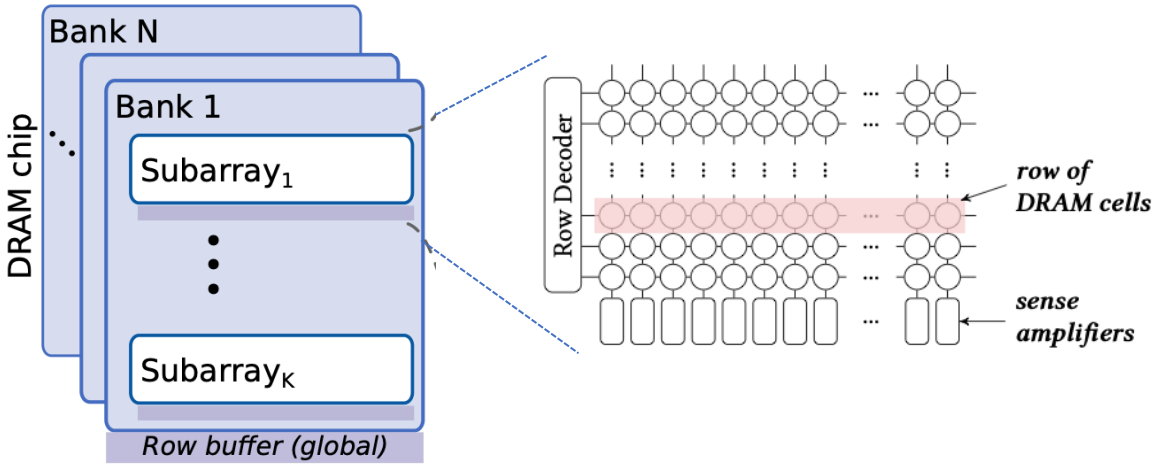
- A subarray is a grid of rows (wordline) and columns (bitline)



Sishadri et al, MICRO 2017
Lima et al., Arxiv, 2024

Main memory organization

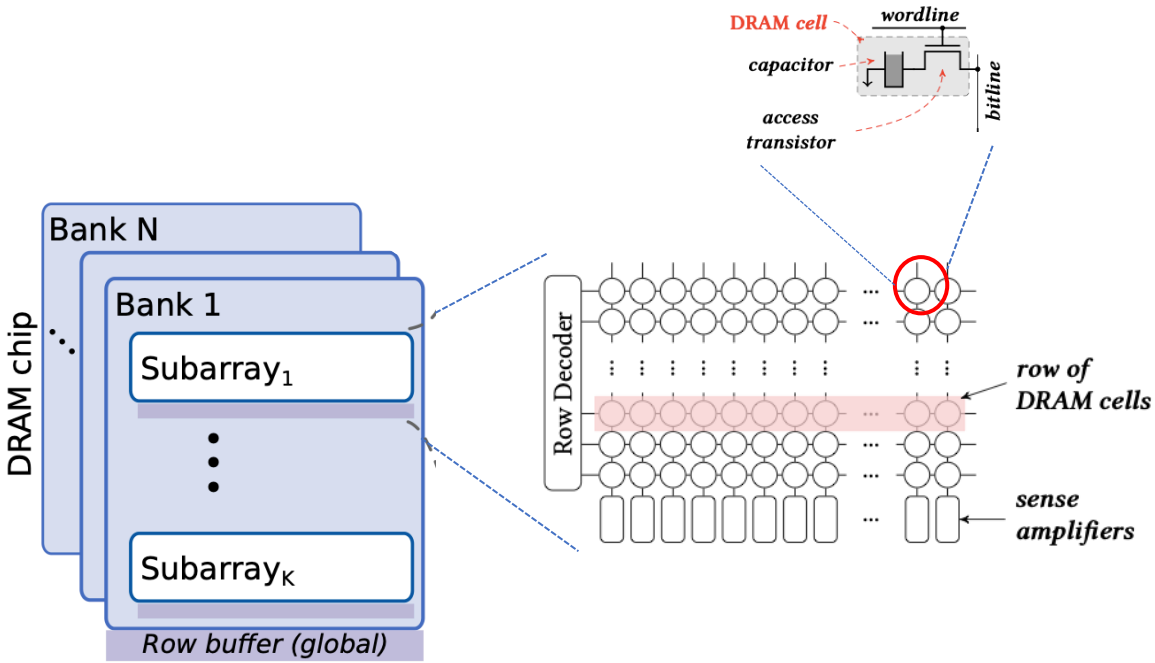
- Consists of channels, ranks, and banks
- Each bank can have one or more subarrays
- A subarray is a grid of rows (wordline) and columns (bitline)



Sishadri et al, MICRO 2017
Lima et al., Arxiv, 2024

Main memory organization

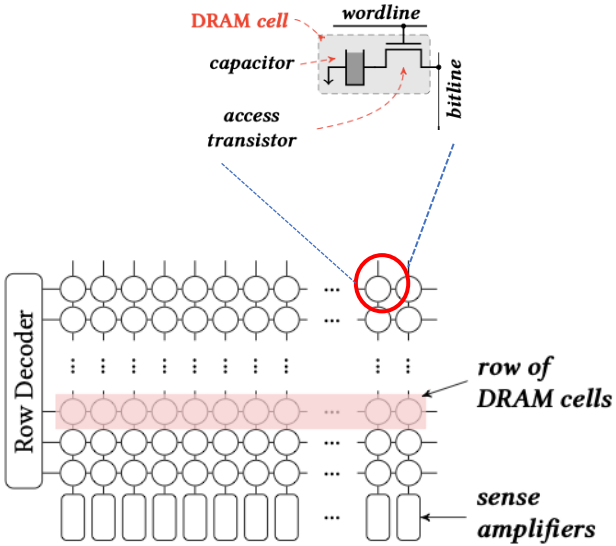
- Consists of channels, ranks, and banks
- Each bank can have one or more subarrays
- A subarray is a grid of rows (wordline) and columns (bitline)
- Each row-col intersection has a memory cell



Sishadri et al, MICRO 2017
Lima et al., Arxiv, 2024

Main memory organization

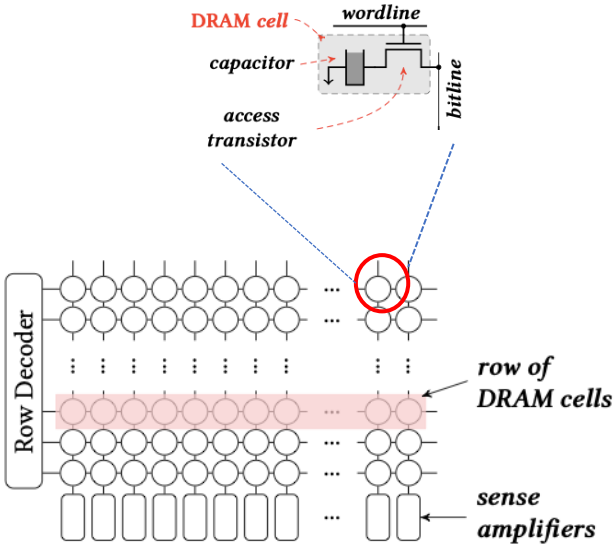
- Each subarray also consists of sense amplifiers connected to bitlines



Sishadri et al, MICRO 2017
Lima et al., Arxiv, 2024

Main memory organization

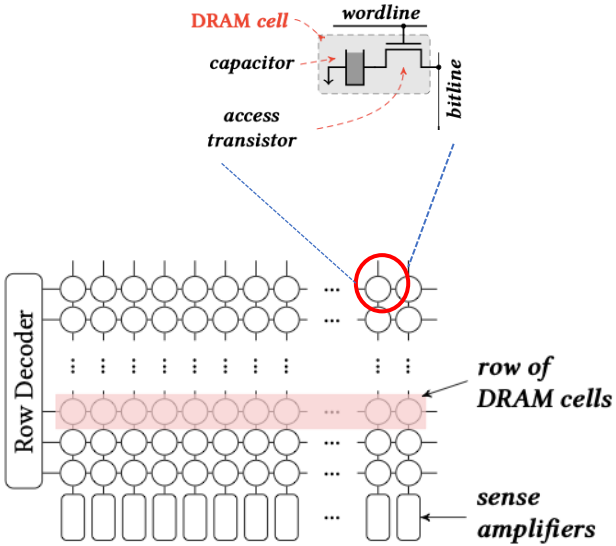
- Each subarray also consists of sense amplifiers connected to bitlines
- To access data from the memory:
  - A row needs to be activated first, i.e., the wordline needs to be enabled
  - The data appears on the bitlines (in the form voltages)
  - The sense amplifiers amplify the voltage signal
  - The amplified signal is then compared to a reference voltage to infer the final value



Sishadri et al, MICRO 2017
Lima et al., Arxiv, 2024

Main memory organization

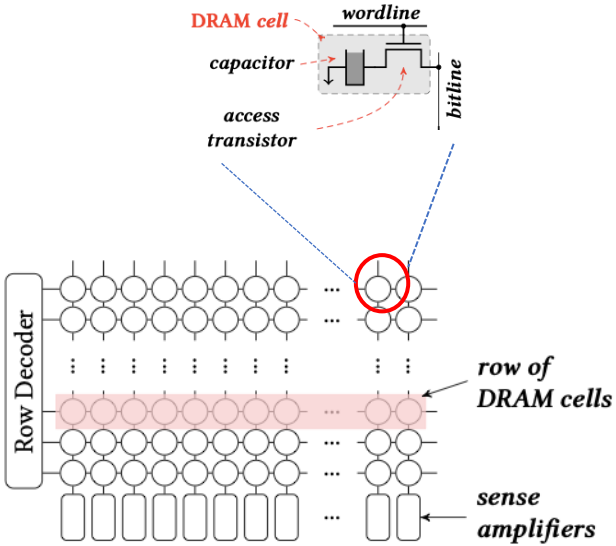
- Each subarray also consists of sense amplifiers connected to bitlines
- To access data from the memory:
  - A row needs to be activated first, i.e., the wordline needs to be enabled
  - The data appears on the bitlines (in the form voltages)
  - The sense amplifiers amplify the voltage signal
  - The amplified signal is then compared to a reference voltage to infer the final value
- The senseamp is also referred to as the row-buffer



Sishadri et al, MICRO 2017
Lima et al., Arxiv, 2024

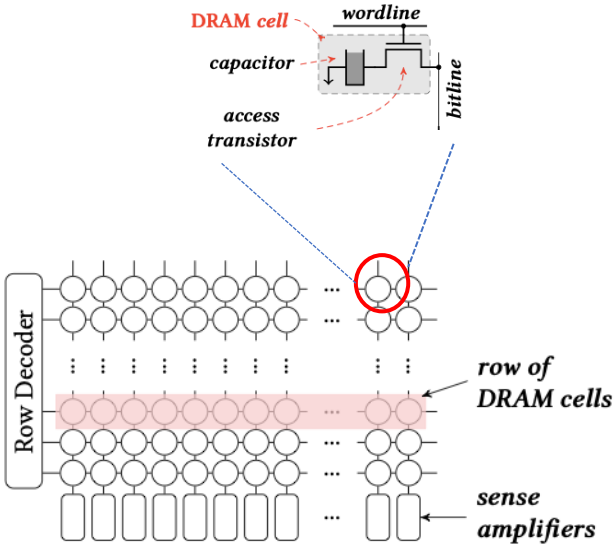
Main memory organization

- If the end-to-end flow is not clear (yet)
  - The CPU initiates a memory request to fetch data or instruction



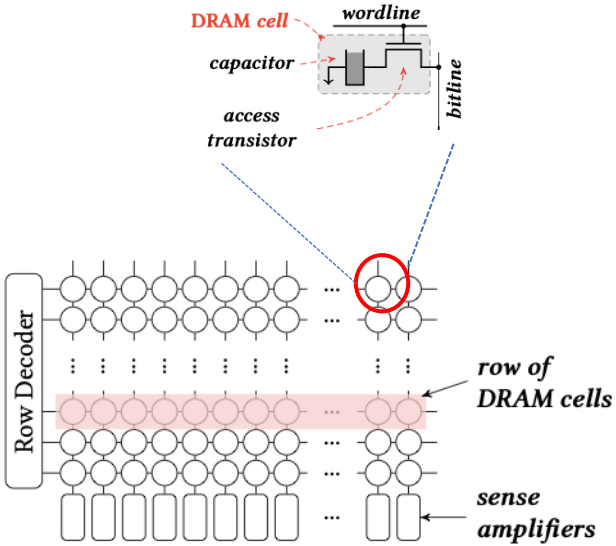
Main memory organization

- If the end-to-end flow is not clear (yet)
  - The CPU initiates a memory request to fetch data or instruction
  - If the operands are not in the reigsters, they are searched in the cache (staring from L1, all the way to the last level cache)



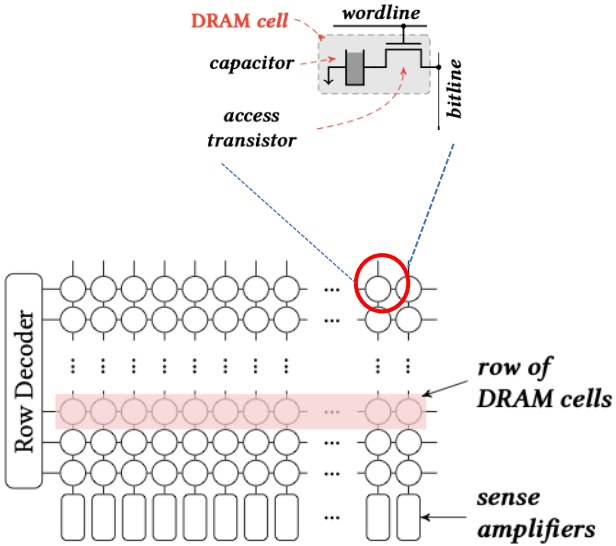
Main memory organization

- If the end-to-end flow is not clear (yet)
  - The CPU initiates a memory request to fetch data or instruction
  - If the operands are not in the reigsters, they are searched in the cache (staring from L1, all the way to the last level cache)
  - If the data is found, it is return to the CPU (ns range)



Main memory organization

- If the end-to-end flow is not clear (yet)
  - The CPU initiates a memory request to fetch data or instruction
  - If the operands are not in the reigsters, they are searched in the cache (staring from L1, all the way to the last level cache)
  - If the data is found, it is return to the CPU (ns range)
  - If not, it is fetched from the main memory
    - The memory controller receives the request, including request type, and address
    - The address decoder decode the address to find channel rank, bank, subarray, etc.
    - The row decoder decodes the row address
    - That row is activated, data comes to the row buffer
    - The column address decoder decodes the column address
    - Data is eventually put on the bus and the row is closed (precharged)



Reading material

- Kim et al., “A case for exploiting subarray-level parallelism (SALP) in DRAM”, ISCA 2012

- Backes et al, “The Impact of Cache Inclusion Policies on Cache Management Techniques”, Memsys 2019

- Escuin et al, “Compression-Aware and Performance-Efficient Insertion Policies for Long-Lasting Hybrid LLCs”, HPCA 2023

Thank you!

asif.ali@uetpeshawar.edu.pk