**52 – 64 - How DRAM works - transistor, gate and capacitor. DRAM. Capacitors and charge leakage.**

*DRAM* - Data is stored in capacitors. Capacitors that store data in DRAM gradually discharge energy, no energy means the data has been lost. So, a periodic refresh of power is required in order to function. DRAM is called dynamic as constant change or action (change is continuously happening) i.e. refreshing is needed to keep the data intact.

*Capacitors and charge leakage* - The process of updating data in DRAM is associated with the peculiarities of storing information in memory of this type. In DRAM, each bit of data is stored in a capacitance (capacitor), and this capacitance tends to lose charge over time due to leaks.

To store data in DRAM, capacities need to be refreshed periodically. The update process involves recharging the capacity to the maximum level to prevent data loss.

*Transistor* - The transistor in a DRAM cell is typically a metal-oxide-semiconductor field-effect transistor (MOSFET). This transistor acts as a switch that controls the flow of charge between the capacitor and the bitline.

*Gate* - The gate is a part of the transistor and controls the flow of charge between the capacitor and source. When Gate is open and we have charge source, capacitor will get charged.

*Drain* - The drain is the terminal through which current exits the transistor. It is the endpoint for the flow of charge carriers in the channel.

In summary, the source terminal of the transistor is connected to the bitline in a DRAM cell, and the wordline is used to select a specific row of memory cells for read or write operations, facilitating the transfer of charge to or from the bitline. These components work together to enable the functionality of a DRAM memory array.

Horizontal lines are wordline, vertical lines are bit line.

**56 - How DRAM works. Wordlines and bitlines. Describe the process of reading data from single cell.**

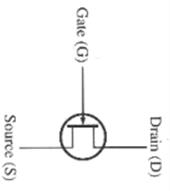
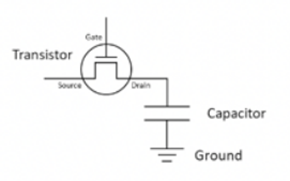
DRAM cells store a single bit.

*How we write information:* In order to write information to capacitor, we open the gate and give voltage to source, and it will be stored inside capacitor.

When Gate is open and we have charge source, capacitor will get charged. Then drain (is the terminal) through which current exits the transistor. (It is the endpoint for the flow of charge carriers in the channel). When we close the gate, source Sources blocked and if something in capacitor, it will stay there.

Gate is door between source and capacitor.

Horizontal lines are wordline, vertical lines are bit line.



*How we read information from cell:* In order to read information, we have to charge bitline half of the voltage that is provided for RAM, then we activate wordline and open the gates. After opening gates, there are two situations: In **First situation**, if capacitor is empty, it will take a little bit voltage from bitline. **In Second situation.** If capacitor has some charge the bitline will get a little bit charge to itself.

**67 - How DRAM works. Sense amplifier.**

*Sense amplifier* - measure difference between original voltage and resulting voltage. If it is less, then capacitor was empty.

**70 - SCOTT CPU. How DRAM works. What is destructive read.**

Destructive read means when we read information, then information is lost, and we need restore it. Because when we read information, it changed. Every read operation is followed by write operation.

**CPU Infrastructure. SCOTT CPU.**

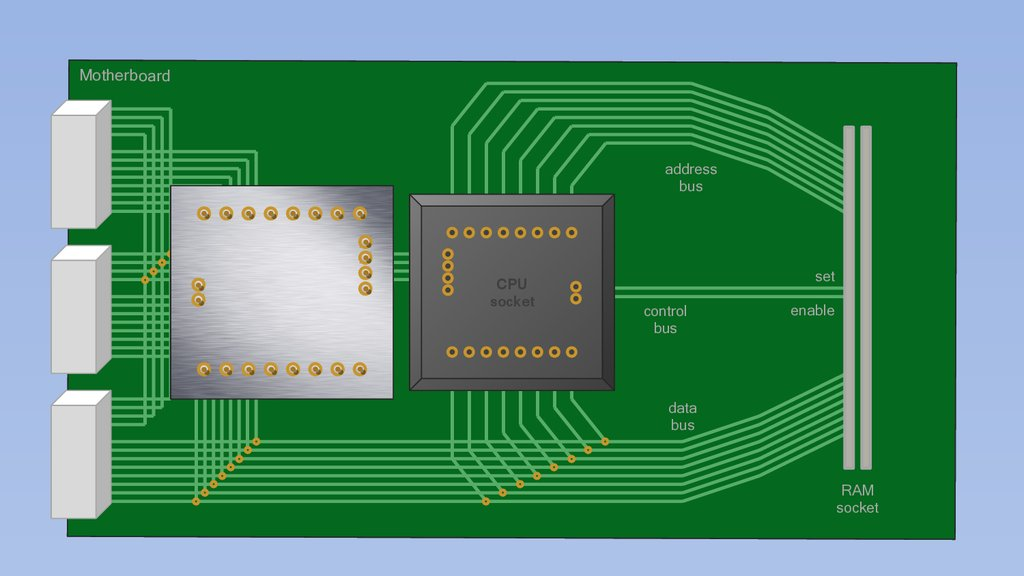
The CPU is a crucial component of the computer and that performs the majority of the processing tasks. it has several key elements.

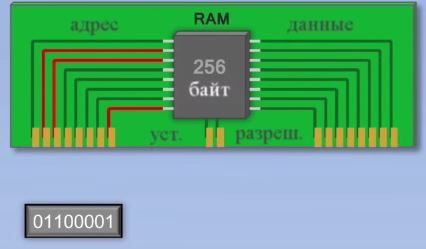
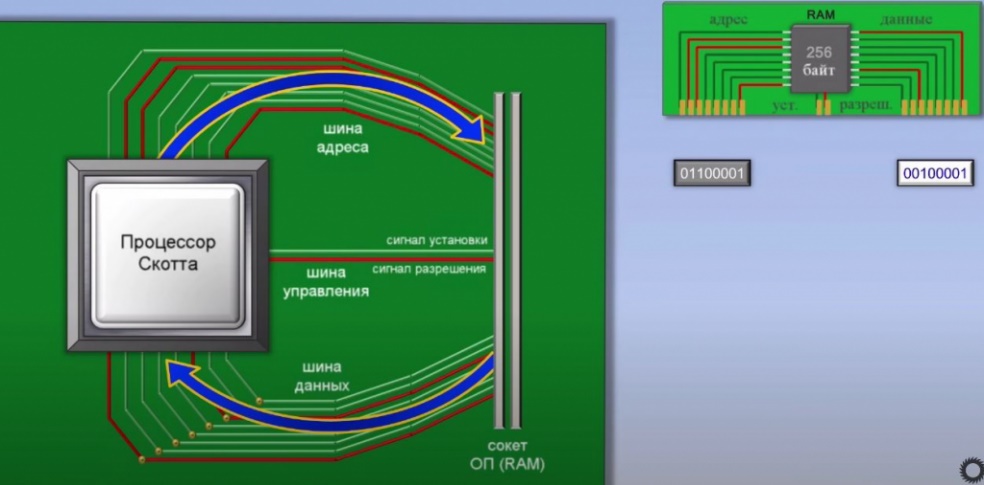
1. *Arithmetic Logic Unit* **-** the ALU is responsible for performing arithmetic and logical operations such as addition subtraction and comparison operations.
2. *Control Unit* **-** the control unit manages and coordinates the activities of the CPU. It fetches instructions from memory decodes them and controls them the flow of data between the CPU and other parts of the computer.
3. *Cache Memory* **–** high speed cache memory is faster than the main memory. It locates inside the CPU to store frequently access data and instructions.
4. *Registers* - are small rooms inside the CPU that using for storage then locations instructions and data there are some types of registers.

5) *Bus Interface* **–** Bus interface manage their communication between the CPU and other components, such as I/O devices use data buses and address buses.

Scott CPU is a theoretical CPU for general using educational purpose. It has the same features as the real CPU and the working system is the same.

**CPU and RAM communication.**





**51 - CPU Infrastructure. CPU Registers.**

*Registers* - are small rooms inside the CPU that using for storage then locations instructions and data there are some types of registers:

*Accumulator* - it’s a general-purpose register used for arithmetic and logical operations. Generally, it used to store the result of many operations in the CPU.

*Instruction register* - tells to the control unit which instruction to perform doesn’t have enabled wire.

*Instruction address register* – save address of next instruction.

*Memory data register* - it holds the data that is read from or written to the memory.

*Memory address register* - it holds the memory address where data is to be written or read from.

**53 - CPU Infrastructure. Instruction set.**

Instruction set - is a set of commands that a processor can execute. These commands represent basic operations that the processor performs within its capabilities.

Load - is using generally for loading data from RAM.

Add - add number.

Store - to store additional results in the RAM.

Compare - to compare the numbers.

Jump if - jump to address based on the condition of compare.

Out - output data to the device.

In - input data from device.

**55 - What is CPU data bus. CPU and RAM communication.**

Inside the computer we used the buses to communicate the CPU.

For communication CPU and the RAM, we use three main types of the bus:

*Address bus* - is to store address of data in the RAM. In each address there is one information stored. We tell RAM which address do we want to work and generally these addresses consist of 1 and 0.

*Data bus* - is used for carrying the main content of the data that it can be instruction number or letter.

*Control bus* - Inside the CPU there is a control bus that is provided to communication between the input output devices and RAM. This control bus has two wires which one of them is called set and another one is enable. Set is using for writing if you want to store some data in the RAM. Enable is using for reading if you want to retrieve some data from RAM.

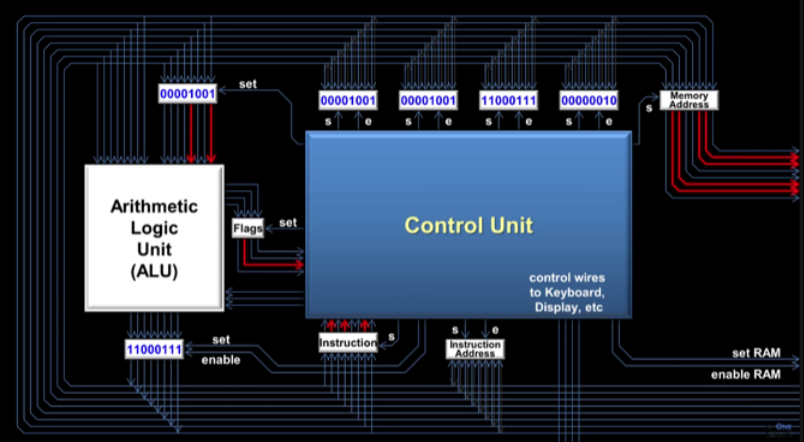
For example, when we want to store data into ram we have to activate the set wire, but when we want to retrieve the data from a RAM we have to activate the enable wire.

**63 - CPU Registers. Set and enable wires.**

*Set Wires:* In the context of digital circuits and CPU design, "set wires" typically refer to control signals or lines that are used to set the state of certain elements within the circuit. These elements might include flip-flops or latches that store binary values (0 or 1).

*Enable Wires:* "Enable wires" are control lines in digital circuits that allow the activation or deactivation of specific elements. These wires are often used to enable or disable the functionality of certain components within a circuit.

**57 - Inside SCOTT CPU. Arithmetic Logic Unit, its inputs and output.**

****

**69 - SCOTT CPU. Set and enable wires. CPU Registers without enable wire.**

There are three registers without enable wire:

There is a register that don’t have enabled wire:

*ALU input register* – output directly to ALU.

*Instruction register* - tells to CU which instruction to perform. Doesn’t have enabled wire.

*Memory address register* - don’t have enabled wire because it directly outputs to ram their output is predefined.

**68 - SCOTT CPU. Number guessing game using CPU instruction set.**

Number guessing - game using CPU instruction set. As we know the CPU instruction set is consist of different instructions and these instructions help the CPU to work and done all tasks easily. For number guessing game we have to use 5 different instruction.

1) Load number for example nine into CPU and stored in the RAM.

2) *In* instruction to get number from user followed by keyboard addresses.

3) *Compare* input with the number stored in the RAM.

4) *Jump if* - if the number is equal guessing number jump to another address otherwise continue.

5) *Out* - out followed by monitor address and letters

**73 - CPU infrastructure. Control Unit.**

*Control Unit* **-** the control unit manages and coordinates the activities of the CPU. It fetches instructions from memory decodes them and controls them the flow of data between the CPU and other parts of the computer.

**74. CPU infrastructure. Moore's Law. CPU clock tick measurements.**

*Moore’s law -* describes the regularity that the number of transistors on integrated circuit doubles every two years. This advancement is important for other aspects of technological process in computing such as processing speed on the price of computers.

*CPU clock ticks* - clock every time flash on and off. It means it performs operation. CPU speed measured number of on and off times per second.

If CPU speed measured in gigahertz it flashes 1 billion times. Clock is a special wire that turns on and off to keep everything synchronized. When measuring CPU performance, it is essential to consider not only the clock speed but also the number of cores architecture, cache size, how efficient the CPU executes and instructions.

**75. Process execution. CPU context and memory image.**

Process has unique identifiers PID. In task manager we can check PID for each process. Hip is memory. Bullet open that's bad. Stack is for holding function called stack. Heap is memory. Process states: (*Running* - currently executing), (*Ready* - waiting for execution/be scheduled), (*Block* - waiting for event (Reading data from disk) (If it takes some time to take data when we waiting data to be downloaded from Internet the process goes to Block state, if it happens so fast it directly goes to ready state), (*New state* - Being created), (*Terminate* - it when process terminated or killed)).

When a process is executed in a computer system, it involves the coordination of various components, including the CPU, memory, and other resources. The concept of "CPU context" and "memory image" is essential in understanding how processes are managed during execution. CPU Context:

*The CPU context* refers to the information stored in the CPU registers and other control registers that define the current state of a process being executed.

This context includes the values of the program counter (PC), stack pointer, general-purpose registers, and other relevant control registers.

*The memory image* of a process is the snapshot of its state in the computer's memory at a specific point in time during execution. It includes the contents of the program code, data, heap, and stack areas in the process's virtual address space.

**72 - CPU Privilege modes. User mode vs Kernel mode.**

CPU privilege modes, also known as processor modes, refer to the different levels of privilege or access rights that the central processing unit (CPU) provides to software running on a computer system. These modes are commonly referred to as "User mode" and "Kernel mode.

*User mode* is where regular applications run with limited privileges while kernel mode is reserved for the operating systems core functionality running with full privileges to manage hardware and critical system tasks the distinction enhanced system security and stability.

*User mod:*

Privileges: Limited access to system resources. Restricted set of instructions. Cannot perform privileged operations directly.

Purpose: User mode is designed to provide a secure and isolated environment for applications. User-level processes are not allowed to directly manipulate critical system resources or execute privileged instructions.

*Kernel mod:*

Full access to system resources. Unrestricted set of instructions. Ability to execute privileged instructions.

Purpose: Kernel mode is used for executing the core operating system functions, managing system resources, and handling privileged operations. The kernel has the authority to control hardware, manage memory, and execute critical system tasks.

**54 - Process management.** **Process abstraction. Process states.**

The same with 12 - Process Management. Process states and lifecycle.

Process abstraction - is the concept of representing a running program as an independent and isolated entity known as a "process." Processes operate independently, have their own memory space, and are managed by the operating system. They provide isolation, independence, and resource allocation, allowing for concurrent execution and efficient utilization of system resources. Processes can communicate with each other through inter-process communication, and their creation and termination are dynamically managed by the operating system. This abstraction is fundamental to modern operating systems for managing and controlling the execution of applications.

**65 - IPC - Interprocess communication. Signals.**

Interprocess communication through shared memory is a concept where two or more process can access the common memory and communication is done via this shared memory where changes made by one process can be viewed by another process.

IPC becomes essential for processes to exchange data and coordinate their activities. There are several methods of IPC, and one of them involves the use of signals.

Signals are a form of IPC used in Unix-like operating systems to notify a process that a specific event has occurred. These events could include a request to terminate, an error, or some other condition.

SIGTERM (15): This signal is used to request a process to terminate gracefully. The process has the opportunity to perform cleanup operations before shutting down.

SIGKILL (9): This signal immediately terminates a process without giving it a chance to perform any cleanup. It is a forceful way to stop a process.

SIGINT (2): Sent by the terminal interrupt character (usually Ctrl+C) to interrupt a running process.

SIGSEGV (11): Sent when a process tries to access restricted memory, resulting in a segmentation fault.

**66 - Process management. Interrupts.**

Interrupts are signals sent to the processor by external devices or internal mechanisms to request its attention. When an interrupt occurs, the processor stops its current execution and transfers control to a specific interrupt handler.

Interrupts play a crucial role in multitasking and real-time operating systems, allowing the system to quickly respond to external events without having to poll devices continuously.

Software Interrupts: Generated by software, often through system calls. Timer Interrupts: Generated by an internal timer to provide a mechanism for the operating system. Hardware Interrupts: Triggered by external devices.

**61 - System call. Trap instructions.**

User functions - when we write our own functions.

System functions when we reserve some memory we need to call operating system call (special functions that is available by operating system). Operating system will initialize trap instruction.

We will have user stack for user functions.

When we make system calls CPU will make for to high privilege level. Switch to kernel stack to user. We need to save our context in kernel stack.

Instruction description table each function from instruction set will have own address all system functions addresses will share in instruction description table.

*Trap instructions* happen during system call program fault or terminates interrupts in block states.

Context switch stock one process and take another one for execution.

*A system call* is a mechanism that allows a user-level program to request a service from the operating system's kernel. User-level applications typically run in a less privileged mode than the operating system kernel to ensure system integrity and security.

*The trap instruction* (or software interrupt instruction) is used to transition from user mode to kernel mode. It is a special instruction that triggers an exception or interrupt.

**58 - Address translation. What is Memory Management Unit (MMU)?**

*Address Translation*: Address translation refers to the process of converting virtual addresses into physical addresses. In modern computer systems, programs use virtual addresses while the physical addresses correspond to the actual locations in the computer's physical memory (RAM). The purpose of address translation is to provide an abstraction layer, allowing programs to use virtual addresses without having to be aware of the underlying physical memory organization.

*The Memory Management Unit (MMU)* is a hardware component within the central processing unit (CPU) that is responsible for handling the address translation between virtual and physical addresses. Its primary function is to map virtual addresses generated by the CPU to corresponding physical addresses in the computer's memory.

Memory management unit use page table for translation. Table consists of two columns first one VA second one PA. VA is divided in memory chunks called page and PA is divided into chunks called frames. Each page is mapped to PA frame that is stored in Page table. Each process has its own page table.

**60 - Memory Management Unit. PA/VA cache. TLB misses.**

We use cash for storing recent VA PA translation. So, we won’t go to the page table each time.

*TLB misses* - if we won’t translate VA and it is not in the cache in this case we will go to page table so if there is something that is not in the cache it is called TLB misses

*PA/VA caches*, often referred to as Physical Address/Virtual Address caches, are used by the MMU to store recently used address translations. These caches help improve the efficiency of the address translation process by providing a faster way to retrieve frequently accessed mappings.

This cache stores recently used mappings between virtual addresses and their corresponding physical addresses. Virtual Address (VA) Cache: Some MMUs may also have a cache for virtual addresses. In this case, the MMU stores recently used virtual addresses and their corresponding physical addresses.

*The Translation Lookaside Buffer (TLB)* is a specific type of cache within the MMU that stores a subset of recently used virtual-to-physical address mappings. When the CPU generates a virtual address, the MMU first checks the TLB. If the virtual address is found in the TLB (a TLB hit), the corresponding physical address can be quickly retrieved.

**71. Memory management. Virtual address space (VAS).**

*Virtual Address Space (VAS)* is a concept in memory management that refers to the range of virtual addresses available to a process in a computer system. Each process running on an operating system is given its own isolated virtual address space, allowing it to operate as if it has exclusive access to the entire memory of the system.

Each memory has its own address. The way operating system provides abstraction for the memory is using virtual address space. Operating system will provide programs some memory space but this memory space will map to physical memory using virtual address space. Program will access virtual address space not directly memory instead.

Fragmentation data spread across different sectors.

Virtual address space maps all address to the real addresses in ram because in ram addresses are not in order they are spread.

Virtual address space used for isolation and security.

Virtual address space also used for programs are not mixed with each other. When we work directly with ram code of one program will access code of other program and this is security issue.

**59. Memory Management. Paging and page table.**

Memory management is a critical aspect of operating systems, and paging, along with page tables, is a key technique used in virtual memory systems to manage the mapping between virtual addresses used by programs and physical addresses in the computer's memory.

Paging is a memory management scheme that allows the operating system to use fixed-size blocks of memory called "pages" to store and manage data. Similarly, physical memory is divided into fixed-size blocks called "frames.

Memory management unit use page table for translation. Table consists of two columns first one VA second one PA. VA is divided in memory chunks called page and PA is divided into chunks called frames. Each page is mapped to PA frame that is stored in Page table. Each process has its own page table.

When a program accesses a virtual address, the page table is consulted to translate the virtual page number into a physical frame number.

**62 - Memory Virtualization. Heap and Stack.**

*Memory virtualization* is a technique used in computer systems to abstract and manage physical memory in a way that allows each process to have its own isolated address space. Memory virtualization is a key feature of modern operating systems

*Virtual Memory*: The concept of virtual memory allows a process to use more memory than is physically available by using a combination of RAM and disk space.

*Memory Paging*: Memory paging involves dividing the virtual address space and physical memory into fixed-size pages. The operating system swaps pages between physical memory and disk storage as needed.

*Page Tables*: Page tables are data structures used by the operating system to keep track of the mapping between virtual pages and physical pages.

*Heap:* The heap is a region of memory used for dynamic memory allocation. It is managed by the programmer and is typically used for storing data structures like linked lists, trees, and objects.

Memory allocation and deallocation in the heap are more manual and flexible. Programmers need to explicitly allocate memory (e.g., using functions like malloc in C) and release it when it is no longer needed (free in C).

*Stack:* The stack is a region of memory that is used for the execution of functions or procedures. It stores local variables, function parameters, return addresses, and other function-related information. Memory allocation and deallocation in the stack are automatic and follow a Last-In-First-Out (LIFO) order.