Review Full Semester

+ a little bit of new information on caching

What do y’all remember about hardware?

Nikesh

Transistors/Logic Gates

Transistors are the smallest part of a computer. There are many different types of transistors however cpus/gpus use cmos transistors.

Transistors work by holding a current of electricity at one terminal, and only allowing that current to flow to the opposite terminal when the middle terminal receives a sufficient amount of current.

Logic gates are collections of transistors that have been organized in a way to create very basic logical concepts such as OR, AND, and NOR.

Latch/Flip Flops

Latches are a hardware construct that takes an input and feeds itself the same input until it is reset with a fresh input.

Latches have issues due to how basic they are however, such as difficult-to-work with delay, which is a delay error with changing the output and input signals.

Flip Flops aim to fix the problems that latches have by adding additional gates to fix the problems with latches.

Both of these are sequential memory elements meaning that they store their inputs until changed.

Registers/Register Files

Registers are collections of latches or flip flops. A 64 bit register is a collection of 64 latches or flip flops right next to each other. Registers are the most used memory element. They are heavily centralized in relation to other notable hardware and so can easily work with them.

Register Files are a collection of registers that have been put together into a grid or memory array. They are used for their energy efficiency because registers are built incredibly energy inefficient.

Transistors, gates, latches/flip flops, registers, and register files all designed off of each other. Can you describe how?

Answer -
Transistors are used to build logic gates. Logic gates are used to build latches/flip flops. Latches/flip flops are put side by side to make registers. Registers are placed in a memory array to make register files.

Sanjay

Cache/RAM/Disk

The cache is a portion of memory that is meant to keep a copy of a part of memory close to the notable hardware.

RAM is similar to the cache however it is bigger and meant to handle full programs.

Disk memory is a type of memory that handles permanent storage of files and other memory.

How much do you remember abouts OS?

Sanjay

CPU OS basics

The main purpose of an OS is to handle file systems, assign hardware resources and use specific hardware constructs such as threads.

An OS assigns hardware resources to processes. Processes are instances of programs. A program is just code in disk memory.

An OS handles processes in a lifecycle. These life cycles can happen in a different number of stages such as a 3 stage or 5 stage.

Process lifecycle

A process starts with the ready state. In this stage the process is in a queue and is ready to be used.

When a process is ready to be run it is put into the running state and is given to the cpu. Processes have an address space which is a range of memory locations the process can use.

If a process needs a memory element that it does not have easily (i.e. not in a register, etc.) it goes into the blocked/waiting state. When it gets the memory element it returns to the ready state.

When the process is done it enters the terminated state.

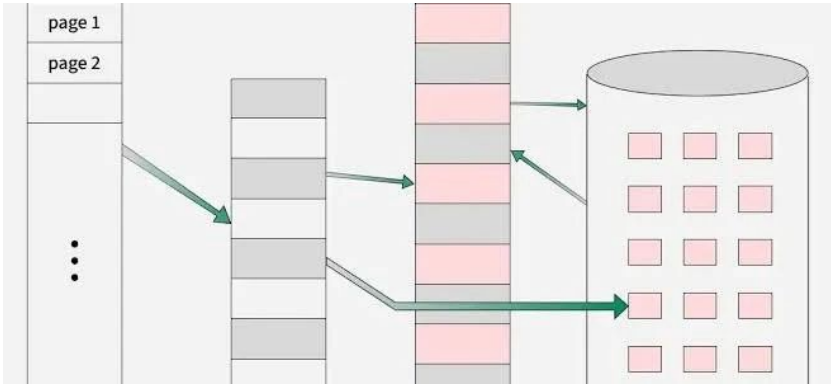
OS scheduling/pages

Processes that are ready are put in a queue. The OS choose which to schedule by weighing a number of factors such as when they arrived, which is smallest, how long it is estimated to execute, etc depending on the specific scheduling system.

The way that a processes address space is organized allows for something called paging, meaning that the address space can be divided into chunks and split across the memory.

Virtual Memory

Virtual memory works by associating a physical page in memory to a virtual memory address in a list of addresses in a module in the cpu (MMU). RAM is one place where memory addresses are stored. RAM has limited in size however so pages that are not loaded into RAM are instead represented on disk by a virtual address in a list of addresses in a module in the cpu (MMU). When a process calls a page that is not in RAM then the page is loaded from disk. This is extremely slow however, up to 1000x.



Kernel/User mode and misc.

Processes are separated into kernel and user mode processes. Kernel mode has full control of the OS. User mode doesn’t and is used for security. In user mode a program can call a syscall in order to access a kernel mode functionality.

The switching of control of processes is called a context switch. The time it takes to switch between processes and load them in is called dispatch latency.

During the running of a process a special command call an interrupt can be used to perform a specific action.

What do you remember about CPUs?

Nikesh

CPU

The cpu runs off an instruction cycle. The instruction cycle is split into the fetch, decode, and execute stages. The fetch associates a memory location stored in a register called the program counter (in intel).

The decode step associates the instruction with a place on the cpu that has a certain tool. If you have an add instruction for example you would send it to the alu(which has the add module).

The execute actually executes the instruction in the place the decode step chose.

CPU

A cpu usually has 3 levels of caches. The L1 being the smallest and fastest, the L2 being bigger and slower and the L3 being the biggest and slowest.

The L1 cache usually has an i-cache and a d-cache. The i-cache is used for parallelism. The d-cache is used for data that needs to be on hand.

Parallelism is anything that makes a computer execute multiple useful things simultaneously.

An example in a cpu is when a cpu executes the fetch, decode and execute stages at once.

Sanjay

Can y’all describe how parallelism of the instruction cycle work in a cpu?
Bonus - Can you name any types of parallelism?

Answer - The cpu will fetch the instruction and then send it to the decode stage. The cpu will then run the fetch again while the decode is running. This will repeat for all stages.

Answer - Types of parallelism include instruction-level, data-level, and thread-level parallelism.

How much do y’all remember about gpus?

GPUs have terms that are provider specific. These slides will use some terms that are specific to NVIDIA gpus. There are AMD alternatives that are the same thing.

GPU

A gpu implements its instruction cycle slightly differently than a cpu.

A gpu fetches instructions the same however it uses simt for the decode stage and the execute stage uses simd.

Simt means single input multiple thread.

A thread is essentially another computer in the same address space. They share memory.

Simd means single input multiple data. It is based off vector registers which are multiple registers “packed” together into one.

GPU

Gpus schedule based on warps (groups of threads) instead of processes.

The simt module executes the same instruction across all threads in a warp. This happens until the threads run a conditional (if statement). When this happens the threads are split into the true and false categories a branch divergence happens.

The designers choose a group to go first and the threads that are in that category run to a point defined by the warp scheduler. Then the other threads run until they all are running the same instructions again and they have a branch convergence.

GPU

Whenever threads need to access registers they use a bank.

A bank is a collection of register files that are “stacked” on top of each other. They are split into lanes. If you imagine banks that stack 3x3 register files then the banks would have three lanes. The leftmost lane would take up all three of the leftmost registers in every lane. Each lane can only read or write to one thing in its range. Having lanes allows multiple threads to access the bank simultaneously.

In order to optimize the order that registers are requested an operand collector is used. Instructions are split into operands (the instruction to be run) and operands (whatever data element is being used).

GPU

Gpus have a unified L1 cache that combines a texture cache and an actual L1 cache. Whenever a gpu requests data from a cache it coalesces the data or makes it so all of the instructions that would be unveiled together are together.

When there is a cache hit (the value is in the cache) it is retrieved from the cache. When there is a cache miss (the value is not in the cache) it is sent to a process request table and retrieved from the lower memory. The instruction is repeated (this is called an instruction replay) until the cache is updated and then the value is unveiled.

When all threads are running the same instructions the simd unit runs and the instructions run simultaneously.

Sanjay

Can y’all name some CPU/GPU differences?

Some Answers

-a gpu has a unified cache (texture and regular cache are together) while cpu has just a regular cache.

-the gpu has L1 and L2 while a cpu usually has an L1, L2 and a L3 cache.

-using registers vs banks

-implementation of threading

-implementation of simd

-scheduling warps for gpu and processes for cpu

-gpus compute instructions in parallel while cpus are more general.

Caching basics

Caching works because two types of locality are common in computing.

Temporal Locality - memory locations are likely to be reused.

Spatial Locality - memory locations near one chosen are likely to be used next.

Caches are made up of frames. Frames are made up of data, a tag, an offset (occasionally) and a state bit. You can imagine frames as a 2d array with the rows being frames and the columns being the data and tag. The offset is used to find a specific piece of data.

The way that the cache replaces the instructions inside of itself is called the eviction policy.

Cache mapping

There are three ways to configure a cache.

Direct-mapping caching occurs when a requested frame is mapped to one tag. Direct mapping doesn’t have a replacement algorithm. If the element is not there then the value is replaced automatically.

Set-associative caching occurs when a requested frame maps to a number of tags (also called a set of tags).

Fully-associative caching occurs when there is one frame and each tag in the frame must be checked to see if it is the right one.

The types of mapping are different in their relations between _____ and ____.

Answer - the frames / the tags

Cache

When something is being looked up the frame is given. The state bit says whether there is a value in the data section of the cache. Depending on the way you are caching the frame is associated to a number of tags. All the tags are compared to the tag on the request and if there is a hit then all of the data is given back up the chain. If you are using an offset bit then the offset is used to get the exact piece of data instead of getting the whole data.

If not then there is a cache miss and the frame is full then the eviction policy goes into place. Otherwise the memory location is mapped onto a location that is not taken.

Cache write policies

The way the cache handles updating data in itself can be done in 3 main ways.

Write-through policy involves writing updated data both to the cache and to main memory.

Write-back policy involves updating cache and then after some time (often when the data is evicted).

Write-around policy involves updating the main memory only and letting the cache update after subsequent reads.

Cache allocation policies

Allocation policies are how a cache handle a write miss (trying to write to an element in cache that is not in the cache).

A fetch-on-write policy involves loading the block that was missed and then updating it with the data in cache.

A write-no-allocate involves updating the data in main memory and not touching the data in cache.

If I want a cache that maps one frame to one tag that updates data in cache only, handles write misses by loading data into the cache and updating it, and has a write policy that updates the cache (main memory is updated after the data is evicted).

Answer -
A direct-mapped cache,
With a write-back cache write policy,
And a fetch-on-write allocation policy

How does a fully associative cache that uses a write through policy and a fetch-on-write policy work?

Answer - The cache maps all tags to a single frame. The cache updates both the cache and main memory when updating a value and the cache handles write misses by loading the value into the cache and then updating it.

What is the smallest component of a computer?

Answer - The transistor. For cpus/gpus cmos transistors are mostly used.

A process is to a cpu as a _____ is to a gpu?

Answer - warp/wavefront. A cpu schedules by processes while a gpu schedules by warps/wavefronts.

When an OS has a bunch of processes that are ready to execute what does it use to tell which one to run next?
Bonus - Name a couple of factors that influence your answer.

Answer - scheduling algorithm.
Bonus Answer -
Turnaround time,
Cpu idle time,
Burst time.