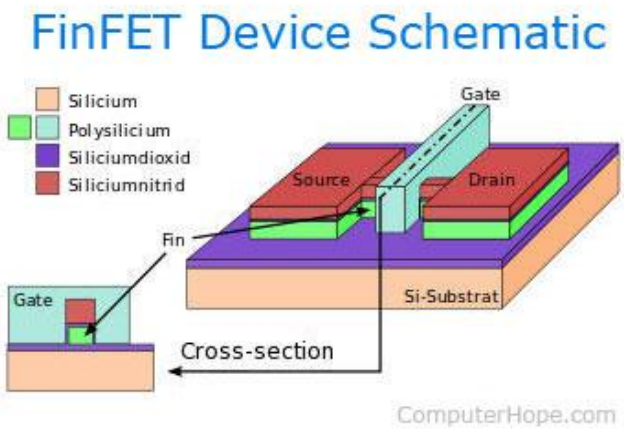
Silicon

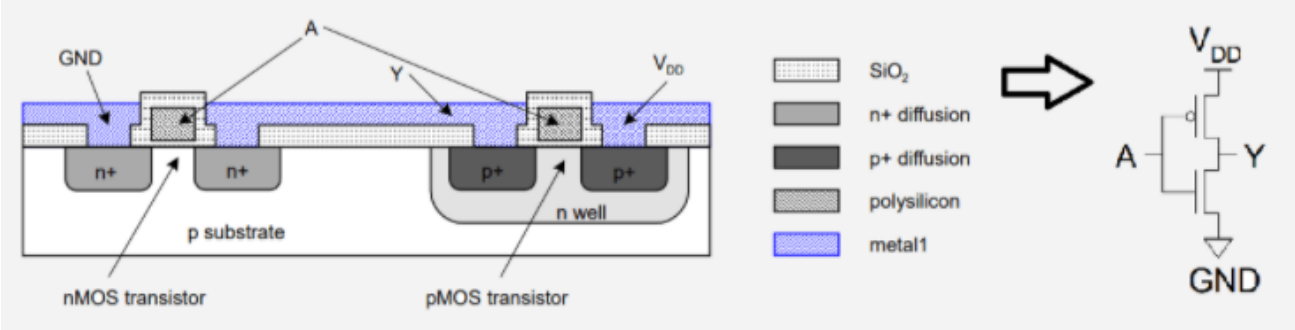
- silicon is a semiconductor, it conducts energy when a voltage is applied
- pure silicon is doped with small concentrations of certain other elements to increase conductivity and adjust its electrical response
- silicon rich sand is melted and cooled to form a silicon ingot
- ingots of silicon are then cut into thin wafers that make processor chips
- silicon has four electrons in its valence shell (outer shell)
    - atoms want eight electrons in its outer shell to be stable, so silicon readily forms covalent bonds with four neighboring silicon atoms
    - if an impurity such as Phosphorus which has five electrons is added to the silicon the extra electron is left free to roam and the total structure is now an N type (negatively charged)
    - if Boron which has three electrons is added to silicon than this structure wants to gain electrons and will steal electrons from other atoms leading to a hole (positive charge)
    - NPN or N-type, P-type, N-type transistors exploit these differences in charge
        - normally electrons from the N-type section flow to the P-type section which creates a barrier of negative charge and prevents further electrons from passing through
        - if a positive charge is applied to the P-type section than the barriers disappear and electrons can flow through freely

Transistors

FinFET transistor

- transistor where voltage determines the conductivity of the device
- three terminal device with a source, gate, and drain
- this is the currently used transistor design
- modern transistors are 1000 x 1000 x 1000 atoms
    - the fin (gate) is 120 atoms wide
- quantum effects begin to occur around the size of 10 atoms

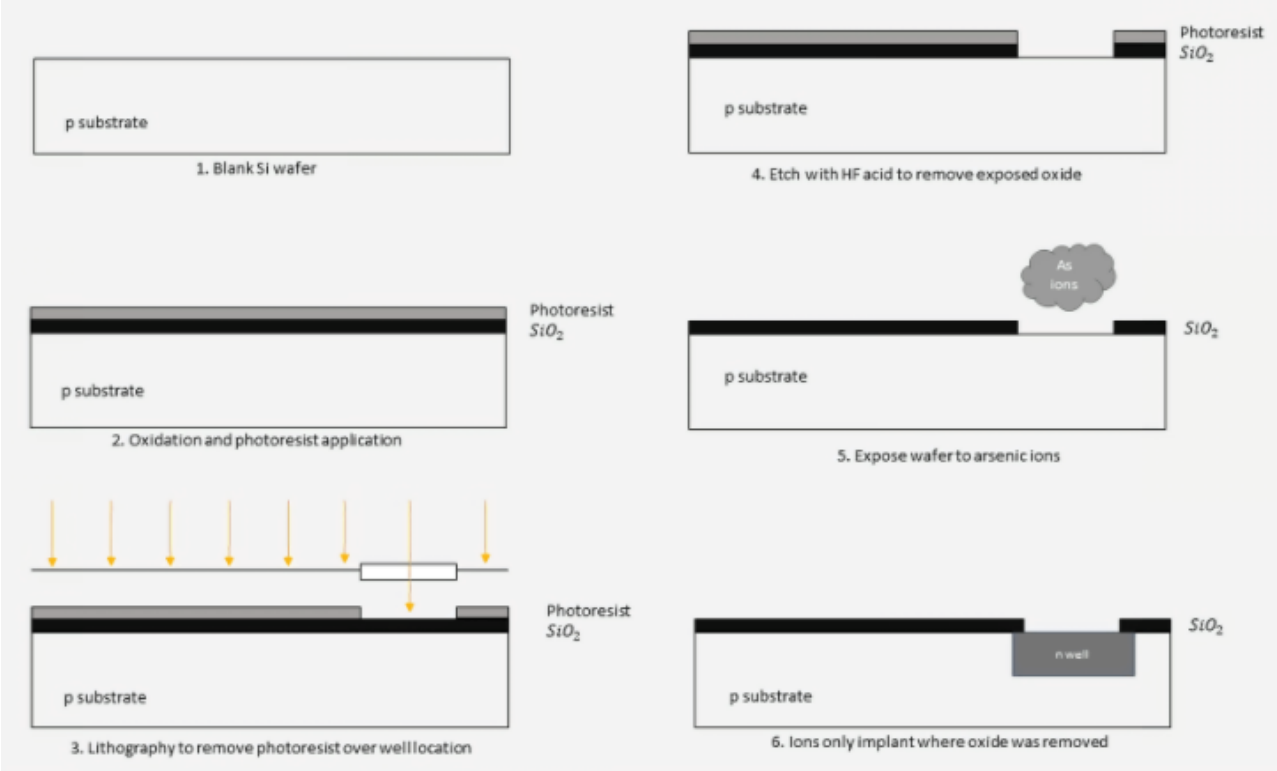




CPU Production

- lithography is used to etch silicon wafers
- lithography is akin to photography in that it uses light to transfer images onto a substrate
- silicon is the traditional substrate used in chipmaking
- light is directed onto a mask. A mask is like a stencil of the circuit pattern. The light shines through the mask and then through a series of optical lenses that shrink the image down. This small image is then projected onto a silicon wafer
- this etches specific patterns in the wafer and exposes various sections
- the exposed sections of the silicon wafer are bombarded with ions during the process of doping

- a pattern of hard material is applied to the top of the silicon via another phase of lithography

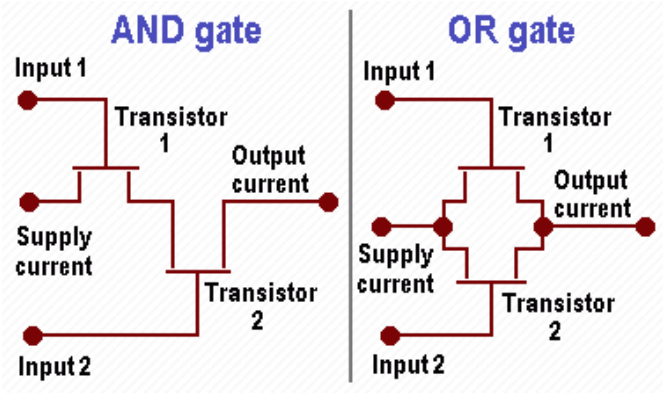


- electroplating: an insulation layer is applied to the surface of the chip
- layering interconnects: all the transistors are now connected in an architecture which allows the chip to function like a processor. There can be over 30 layers of metal connections in a single processor
    - the complexity of interconnects is so high that computers are generally used to design the interconnect
    - the interconnects are what make logic gates out of multiple transistors
- chips are packaged with a substrate and heat spreader, and assume the familiar form of a desktop processor. The heat spreader conducts heat away from the silicon and into the heatsink mounted on top of it
- the single biggest factor preventing us from making bigger and bigger chips are defects in the manufacturing process. Modern chips have billions of transistors and if a single part of one is broken, the whole chip may need to be thrown away. As we increase the size of processors, the chance that a chip will be faulty increases

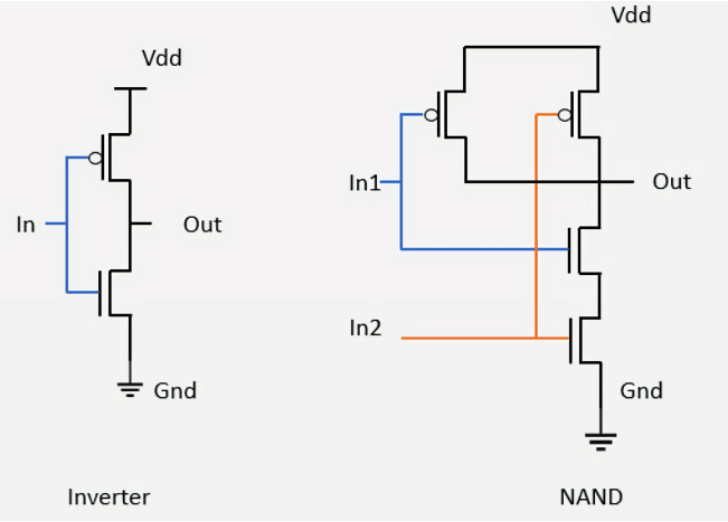
Logic Gates

- transistors are organized together to build logic gates
- the processor uses gates in combination to perform arithmetic function and trigger data in memory
- logic gates are built on the base of Boolean logic

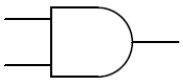
- AND gate outputs a 1 if and only if all the inputs are on
- inverter or NOT gate will turn its output on if the input is off
- combination of NOT and AND gate creates a NAND gate which outputs a 1 if and only if none of the inputs are on



- pMOS transistors allow current to flow when the gate is discharged or set low
- nMOS transistors allow current to flow when the gate is charged or set high
    - in the below figure pMOS transistors are drawn with a small circle connecting to their gate
    - in the inverter example: if the input is on electrons will flow to the nMOS gate and then to the drain, resulting in a 1 turning to a 0
    - if the input is off electrons flow to the pMOS gate and then to the output turning a 0 to a 1

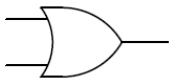


- the NAND logic gate requires four transistors and as long as one of the inputs is off then the output will be on



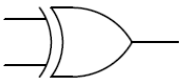
AND

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



OR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

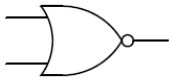


XOR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

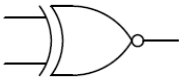
NAND

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



NOR

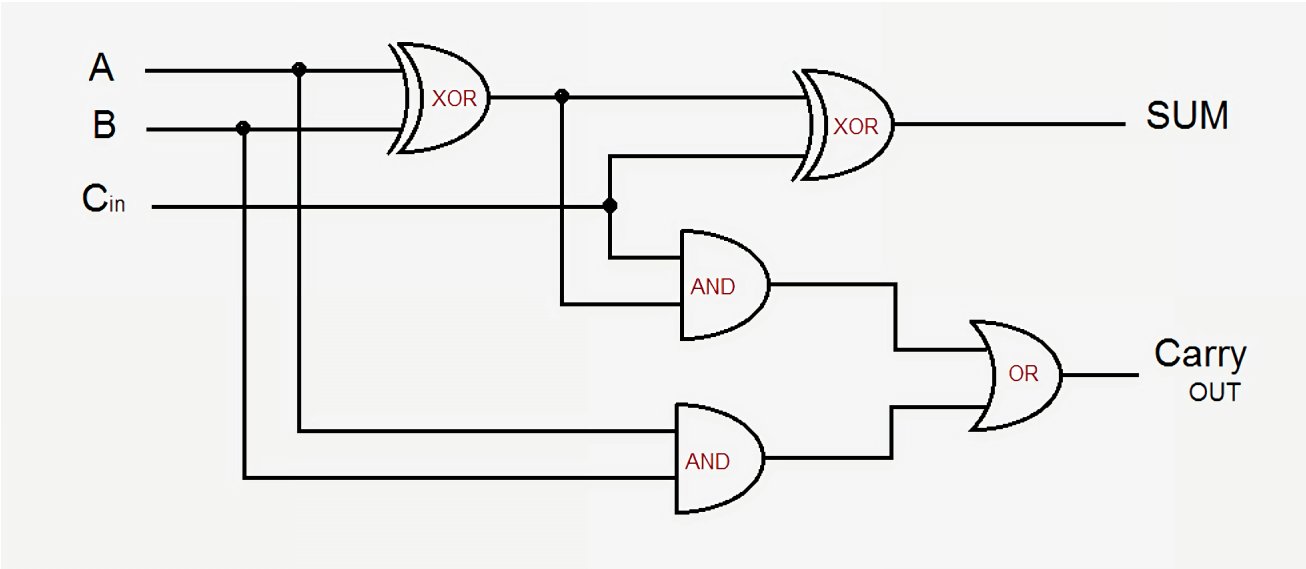
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



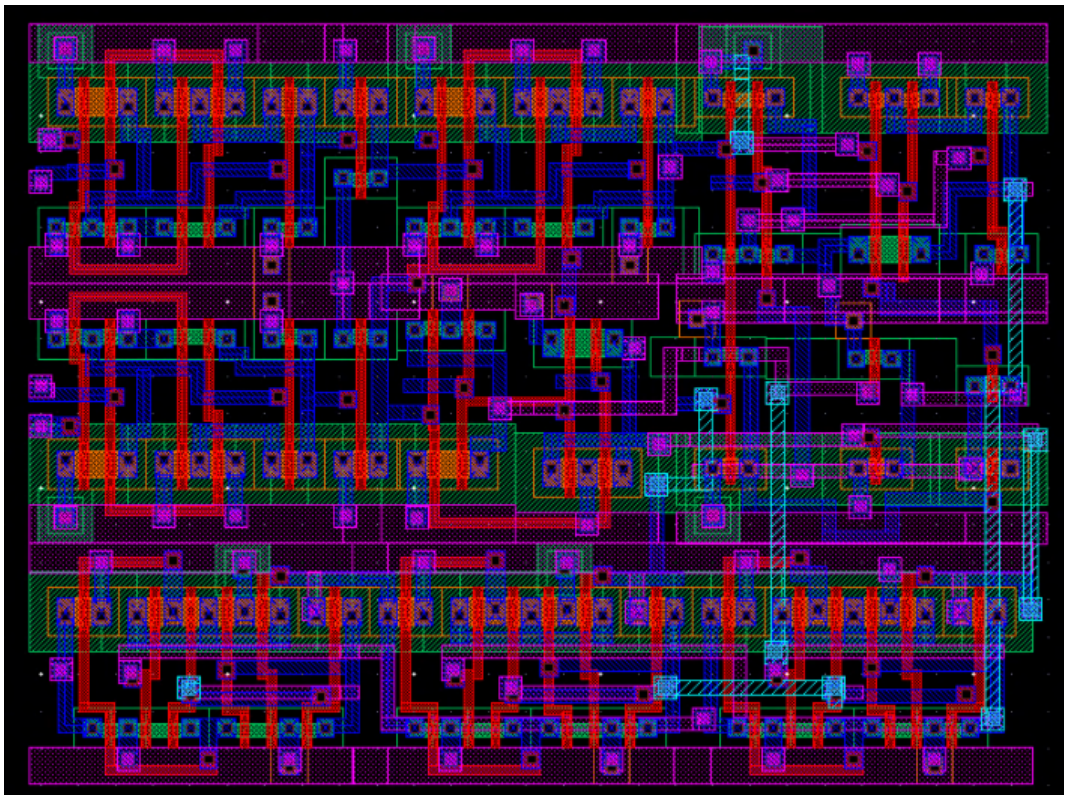
XNOR

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

- below is the logic gate design of a 1 bit adder. It uses five logic gates
- multiple 1 bit adders can be connected to form a wider adder by connecting the carry out of the previous bit to the carry in of the current bit.

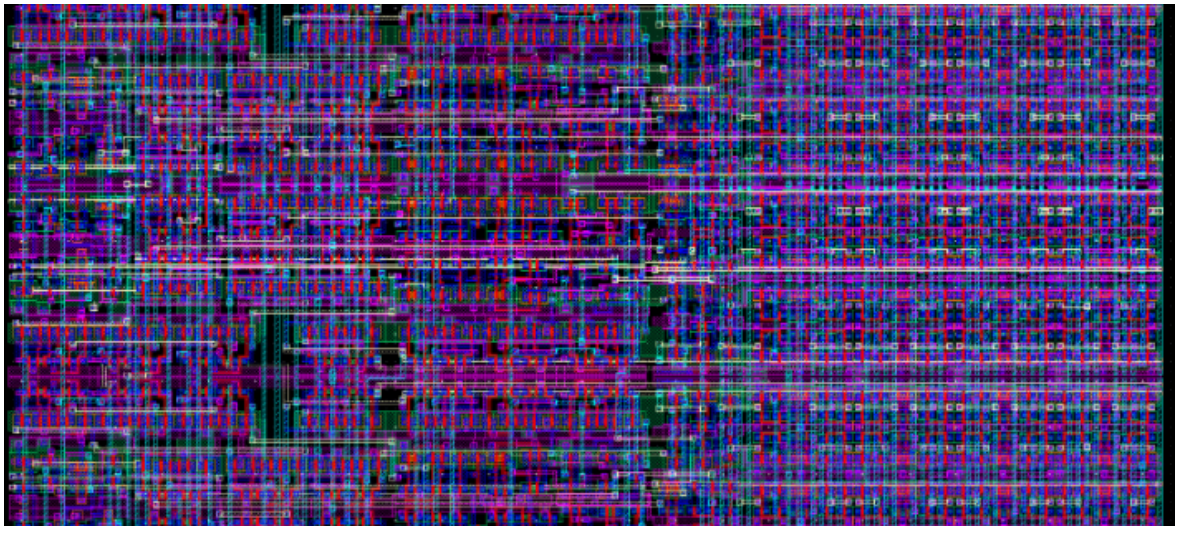


- below is a 1-bit adder unit made up of several logic gates as it appears on the CPU



- green rectangles are pMOS and nMOS transistors, blue and purple areas are metal connectors, red areas are polysilicon gates

- below, putting together many cells and 2,000 transistors, we have a basic 4-bit processor with 8-bytes of RAM on four metal layers. Looking at how complex this is, one can only imagine the difficulty of designing a 64-bit CPU with megabytes of cache, multiple cores, and 20+ pipeline stages. High-performance CPUs today can have upwards of 5-10 billion transistors and a dozen metal layers

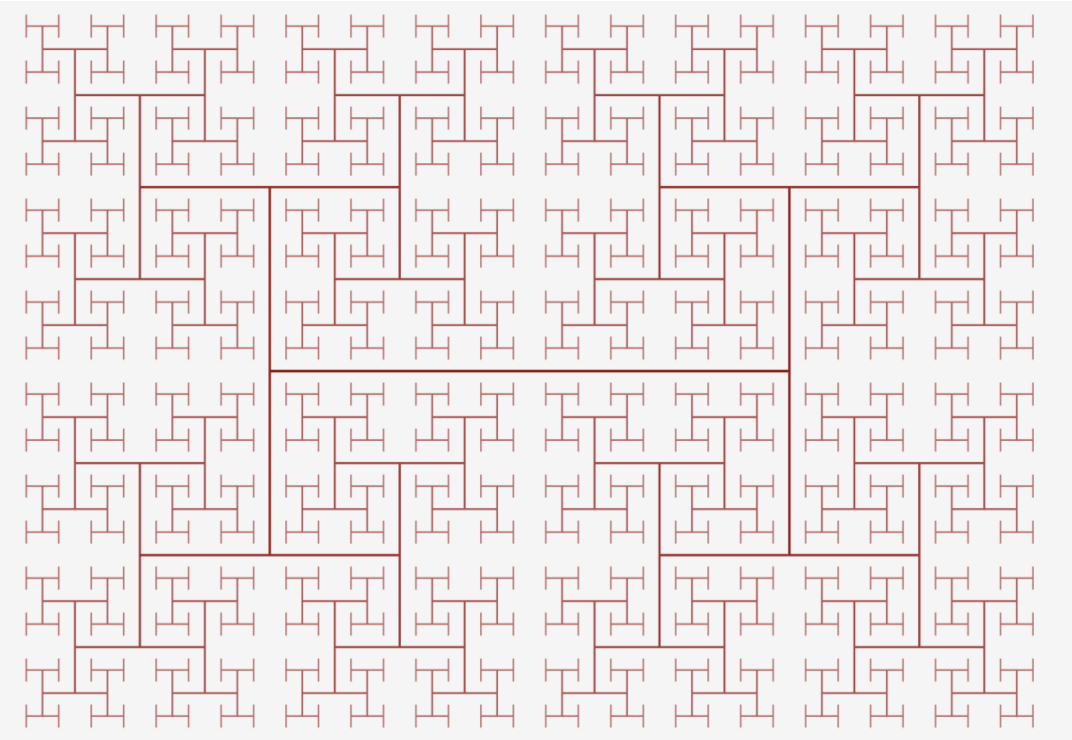


- combining a series of gates to produce an arithmetic output such as an adder is called combinational logic
- in order to store data we need **sequential logic**
    - sequential logic is built by connecting inverters and other logic gates so that their outputs feed back into the inputs of the gates
    - these feedback loops store one bit of data and are called static RAM or SRAM
    - SRAM is used to build super fast caches inside of processors
    - SRAM is very stable but requires 6 to 8 transistors to store each bit of data
- dynamic RAM of DRAM is stored in a capacitor and only required a single transistor per bit, the capacitor being charged represents bid value of and a discharged capacitor represents bid value 0
- the charge in the capacitor is so small that it needs to be constantly refreshed, thus when you turn off your computer DRAM is wiped
- SRAM also needs constant power to run
- SRAM is on chip while DRAM is off chip

CPU Architecture and Operation

Clock Signal

- all key components in a processor are connected to a clock signal
- the clock signal alternates between high and low at a predefined interval known as the frequency
- the logic processor typically performs calculations when the clock goes from low to high
- synchronizing everything ensures that data always arrives at the correct time
- overclocking involves turning up the frequency so that there are more cycles and more work can be done
- clock speed is measured in gigahertz GHz (higher number = higher clock speed)
    - hertz measures numbers of time the clock turns on in a second and giga means billions, CPUs operate at 3-5 billion ticks per second
    - modern processors typically run between 3 and 4.5 GHz
- by the end of the clock cycle every component in the processor must have finished its operation
    - if any parts of the operation aren't finished the clock is running too fast. This dictates the max clock speed
- the clock signal runs up against issues of physics
    - even though the speed of light is fast, it's not quick enough to send the signal to all the parts of the chip at the same time so that operations occur in sync
    - to keep all portions of the chip in time an H-tree is designed so that all endpoints are the exact same distance from the center



- chips are too complex to design manually, thus automation is used
    - automation tools will typically take a high level description of what the component should do, and determine an optimal hardware configuration to meet those requirements. There has been a recent trend towards a technique called High Level Synthesis which allows developers to specify the functionality they want in code, and then have computers figure out how to optimally achieve it in hardware
    - languages such as Verilog and VHDL allow hardware designers to express the functionality of whatever circuit they are making. Simulation and verification is done on these designs and if everything passes, they can be synthesized down into the specific transistors that will make up the circuit

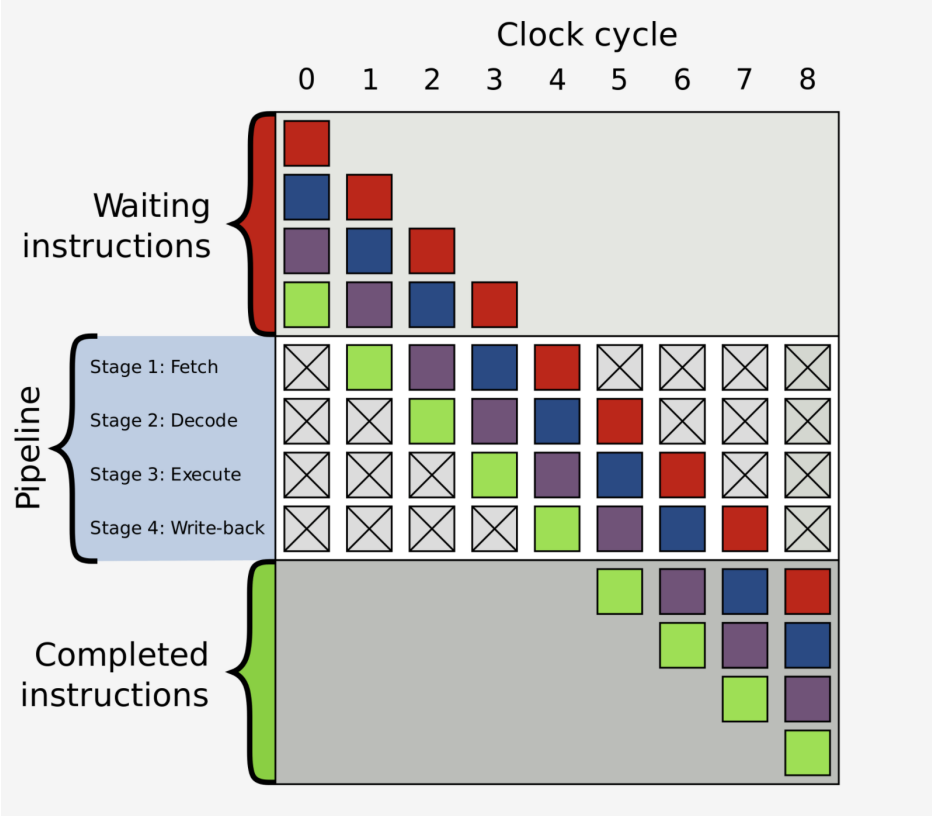
CPU Operation

- execution of an instruction has several parts that are broken down through the many stages of a processor
    - the first step is to fetch instructions from memory into the CPU to begin execution
    - the second step is for instructions to be decoded into binary
        - there are many types of instructions including arithmetic instructions, branch instructions, and memory instructions
    - once the CPU knows the type of instruction, the operands for the instructions are extracted from memory. To add A and B the CPU has to extract the values of A and B first

    - results are calculated and stored in memory if necessary, after this the CPU moves on to the next instruction

Pipeline Execution

- most modern processors will break these few stages up into 20 or more smaller stages to improve efficiency
- although the processor will start and finish several instructions each cycle, it may take 20 or more cycles for any one instruction to complete from start to finish
- this model is typically called a pipeline since it takes a while to fill the pipeline and for liquid to go fully through it, but once it's full, you get a constant output



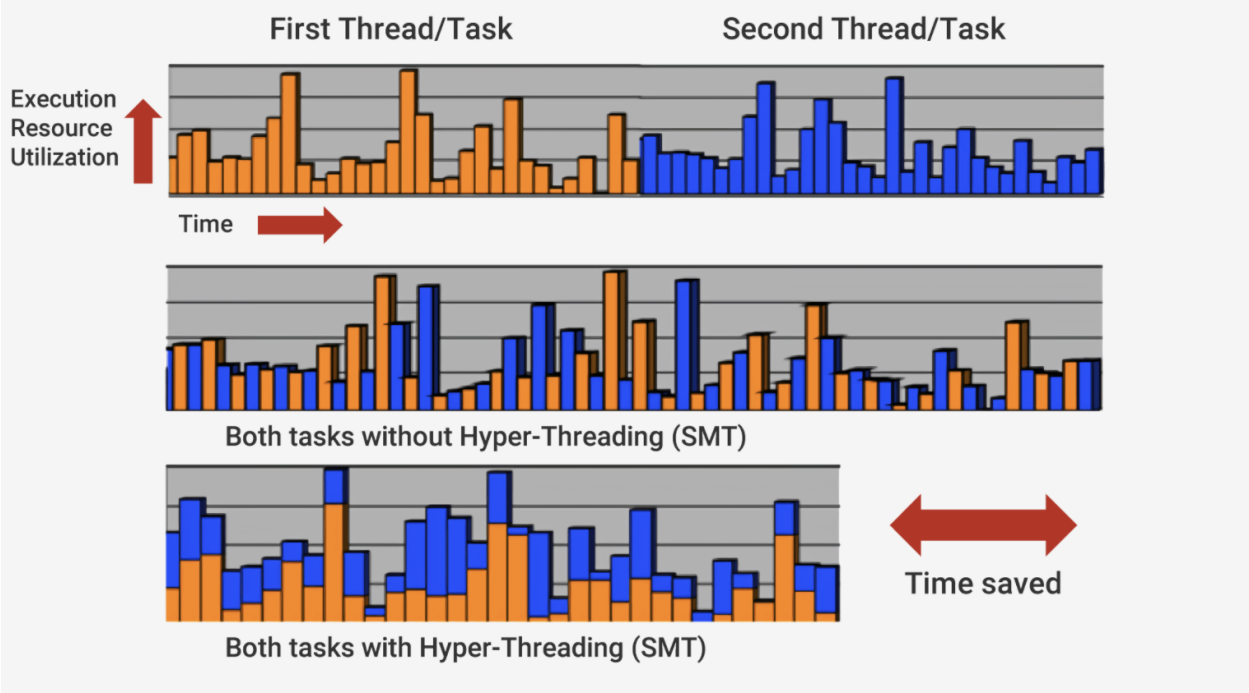
- colored boxes represent instructions independent of each other

Execution Out of Order

- not all instructions finish at the same time, addition is very fast while division or loading from memory may take hundreds of cycles
- to avoid stalling the entire processor while one slow instruction finishes, processors execute out of order
- if the current instruction isn't ready yet, the processor will jump forward in the code to see if anything else is ready

Superscalar Architecture

- the processor is executing many instructions at one time in each stage of the pipeline
- to execute many instructions at once, processors will have several copies of each pipeline stage inside. If a processor sees that two instructions are ready to be executed and there is no dependency between them, rather than wait for them to finish separately, it will execute them both at the same time
    - parallelism takes a task that requires 500 instructions, runs all 500 instructions at the same time and then completes the task by synthesizing the 500 outputs
- a superscalar processor is a CPU that implements a form of **parallelism** called instruction-level parallelism within a single processor. In contrast to a scalar processor that can execute at most one single instruction per clock cycle, a superscalar processor can execute more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to different execution units on the processor
    - **found parallelism**: the CPU has to figure out the dependency graph between instructions and then based on this graph compute many instructions in parallel
    - **forced parallelism**: the GPU is given the dependency graph and simply knows how to compute many instruction
- one common implementation of this is called Simultaneous Multithreading (SMT), also known as Hyper-Threading
- SMT splits physical cores into virtual cores which can perform two operations simultaneously

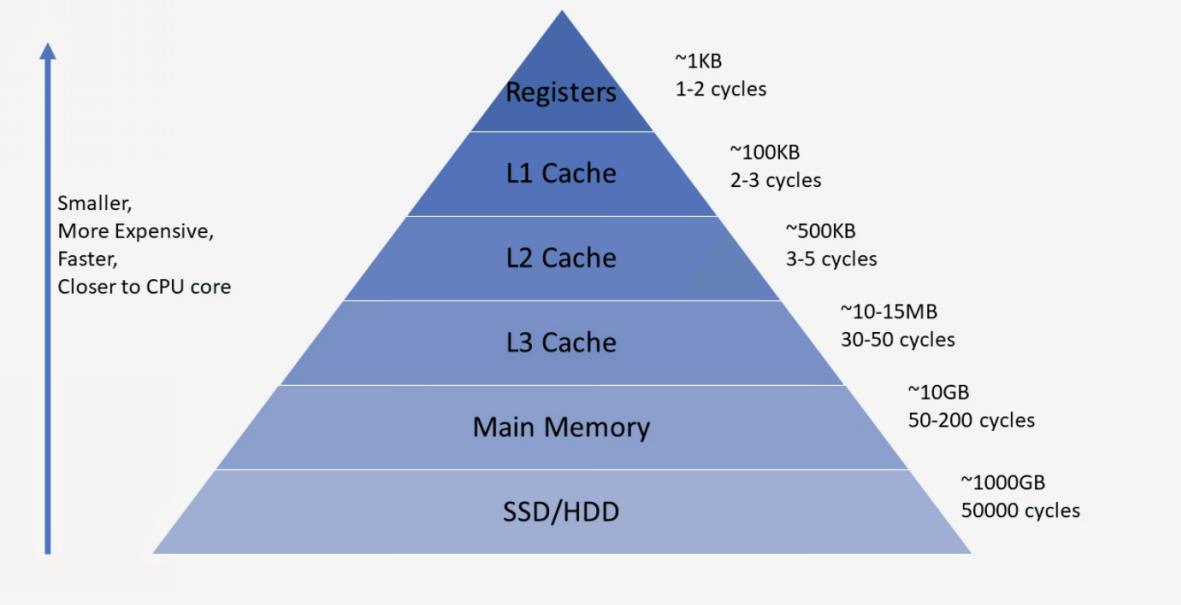


Cores

- a CPU can have multiple cores that perform operations separately from each other
    - multiple cores can work together to perform parallel operations on a shared set of data in the CPU's memory cache
- to accomplish the carefully choreographed execution of parallelism and superscalar architecture, a processor has many extra elements in addition to the basic core
    - the two biggest elements are the caches and branch predictor

Caches

- made up of static SRAM
- even though DRAM is very fast, it is orders of magnitude too slow for the CPU
- it may take hundreds of cycles for DRAM to respond with data and the processor would be stuck with nothing to do. If the data isn't in the DRAM, it can take tens of thousands of cycles for data on an SSD to be accessed
- processors typically have three levels of cache that form what is known as a memory hierarchy.
    - L1 cache is the smallest and fastest, L2 is in the middle, and L3 is the largest and slowest of the caches
    - above the caches in the hierarchy are small registers that store a single data value during computation. These registers are the fastest storage devices in your system by orders of magnitude
    - when a compiler transforms high-level program into assembly language, it will determine the best way to utilize these registers.



- each core typically has two L1 caches: one for data and one for instructions
- there is typically an L2 cache for each core though it may be shared between two cores

- there is a single L3 cache shared between all cores

- When a processor is executing code, the instructions and data values that it uses most often will get cached.

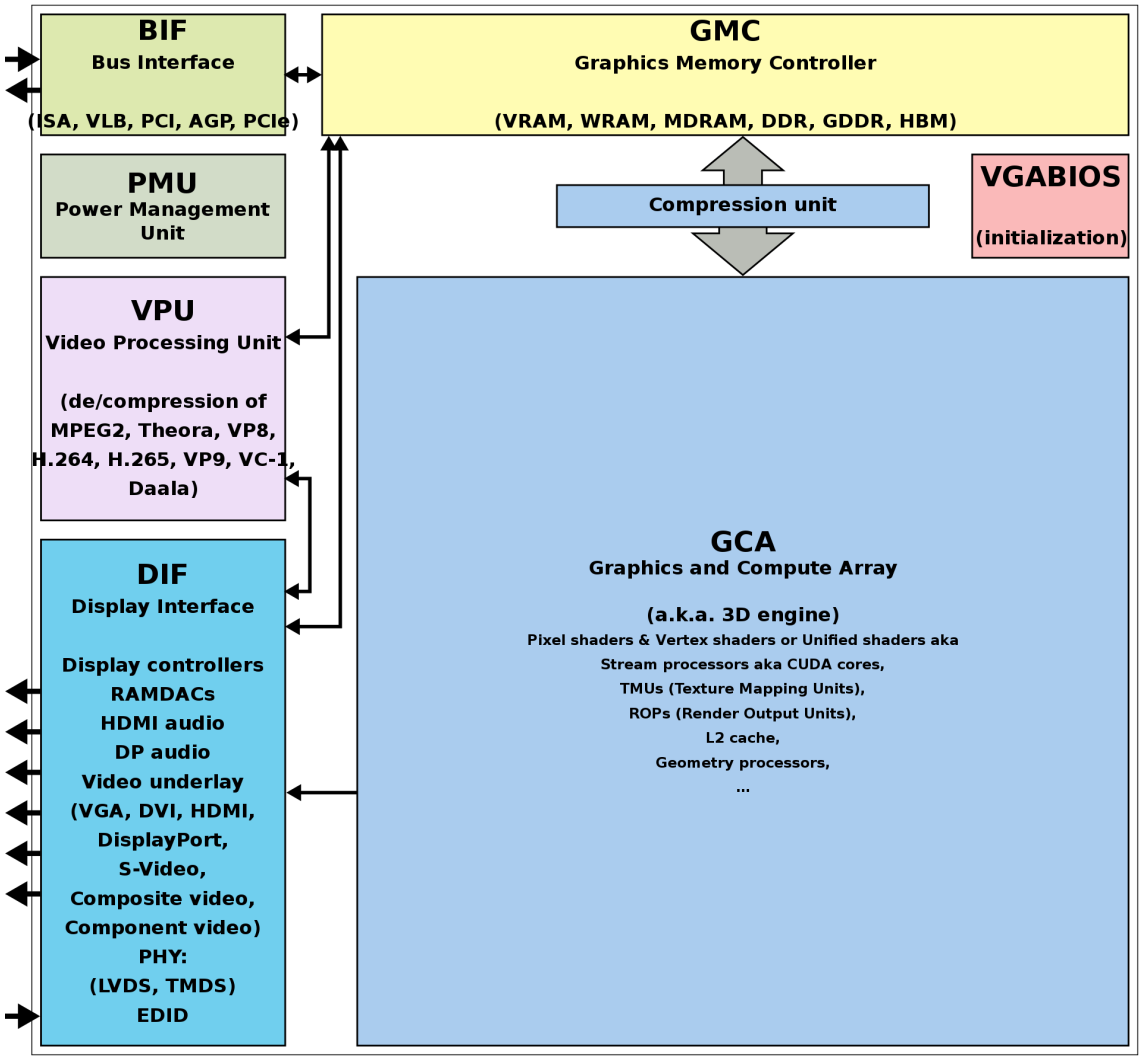
Branch Predictor

- branch instructions are similar to “if” statements, one set of instructions will execute if the condition is TRUE and another set of instructions will execute if the condition is FALSE
- at any one time the CPU may be in the process of executing ten or twenty instructions at once, thus it is very important to know which instructions to execute
    - it may take 5 cycles to determine if the current instruction is a branch and another 10 cycles to determine if the condition is true
    - in that time, the processor may have started executing dozens of additional instructions without even knowing if those were the correct instructions to execute
- branch prediction attempts to guess the outcome of a conditional operation and prepare for the most likely result
    - when a conditional operation such as an if…else statement needs to be processed, the branch predictor "speculates" what condition is most likely to be met. It then executes the operations required by the most likely result ahead of time so that they are already complete if and when the guess was correct
    - if the prediction is incorrect, the processor stops execution, removes all incorrect instructions that it has started executing, and starts over from the correct point
- these branch predictors are some of the earliest forms of machine learning since the predictor learns the behavior of the branches as it goes
    - if it predicts incorrectly too many times, it will begin to learn the correct behavior. Decades of research into branch prediction techniques have resulted in accuracies greater than 90% in modern processors
    - modern branch prediction resembles a neural net

GPU Architecture

- GPUs are similar to CPUs but are designed specifically to perform complex mathematical and geometric calculations necessary for graphics rendering
    - used heavily for deep learning as well
- as the GPU creates images it needs to store the information somewhere
    - it stores data about each pixel, its color, and location in RAM
    - RAM acts as a frame buffer, holding completed images until it is time to display them

    - video RAM (VRAM) is typically dual ported, so it can be read from and written to at the same time
- GPUs break complex problems into hundreds of tasks and complete them all at once with forced parallelism
- while a CPU is composed of a few cores and lots of cache memory that can handle a few software threads at a time, a GPU is composed on hundreds of cores that act simultaneously

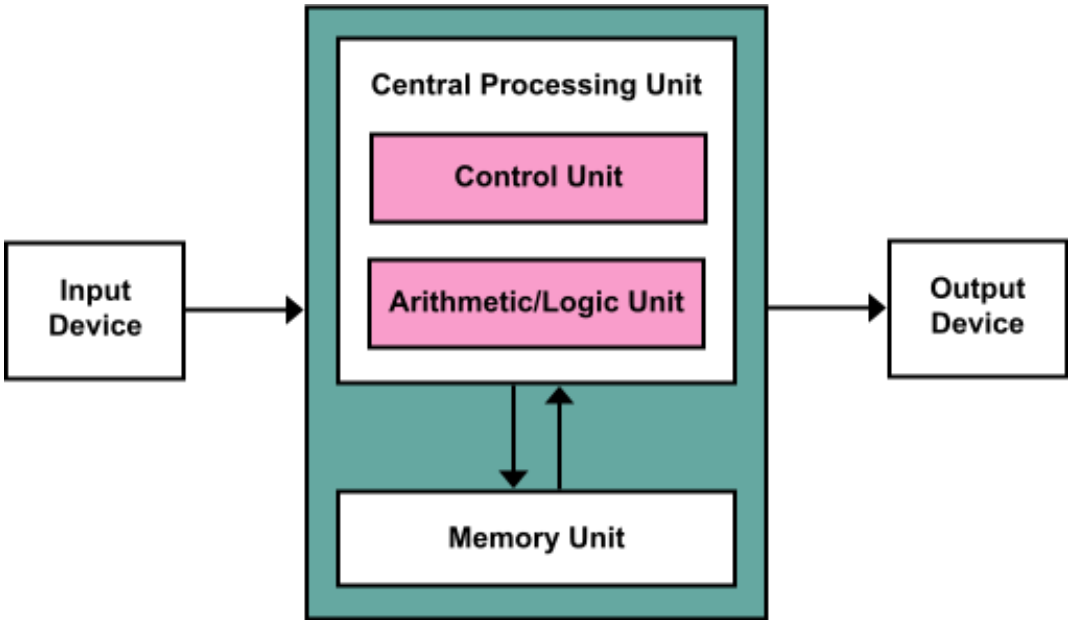


CPU Components

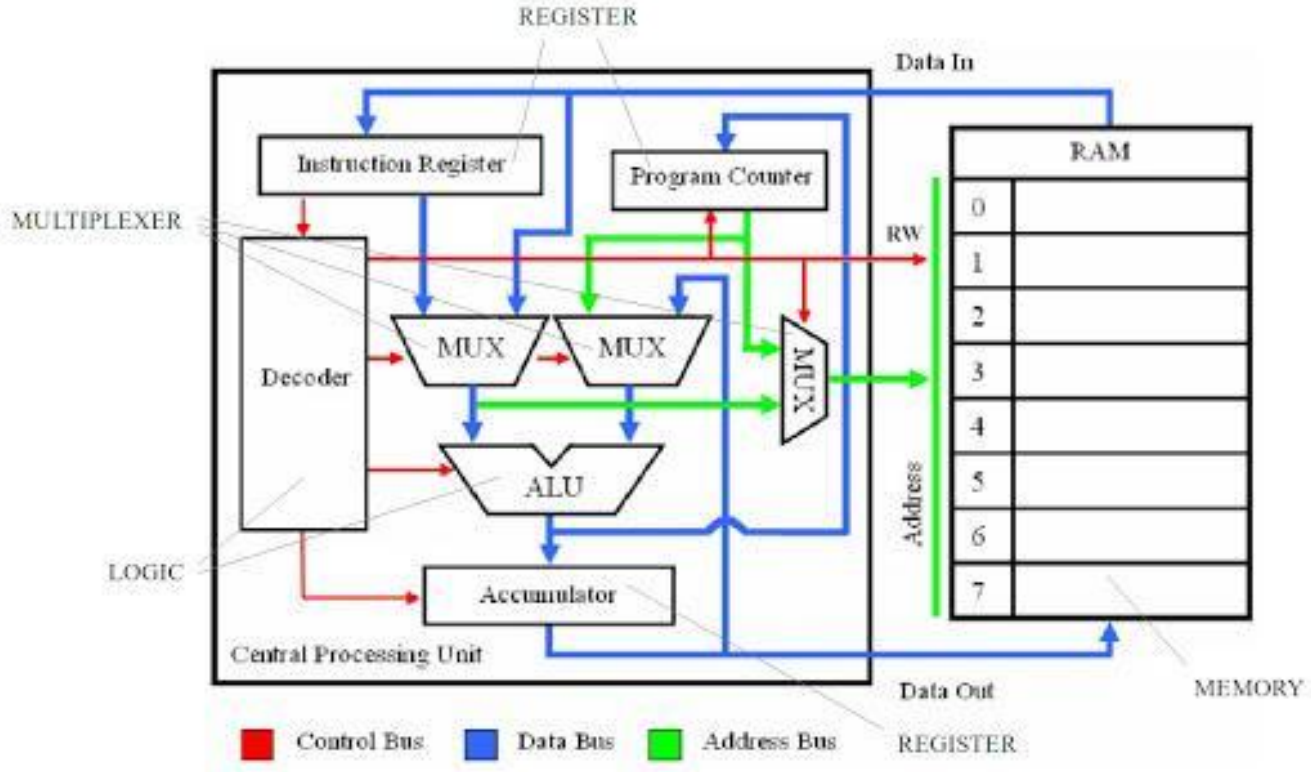
Three High Level Components

- CU (control unit) - directs operations within a CPU
- ALU (arithmetic logic unit) - circuit used to perform arithmetic and logic operations

- Register / Memory Unit - used to push and pull data



- input devices include mouse and keyboard
- output device include monitor



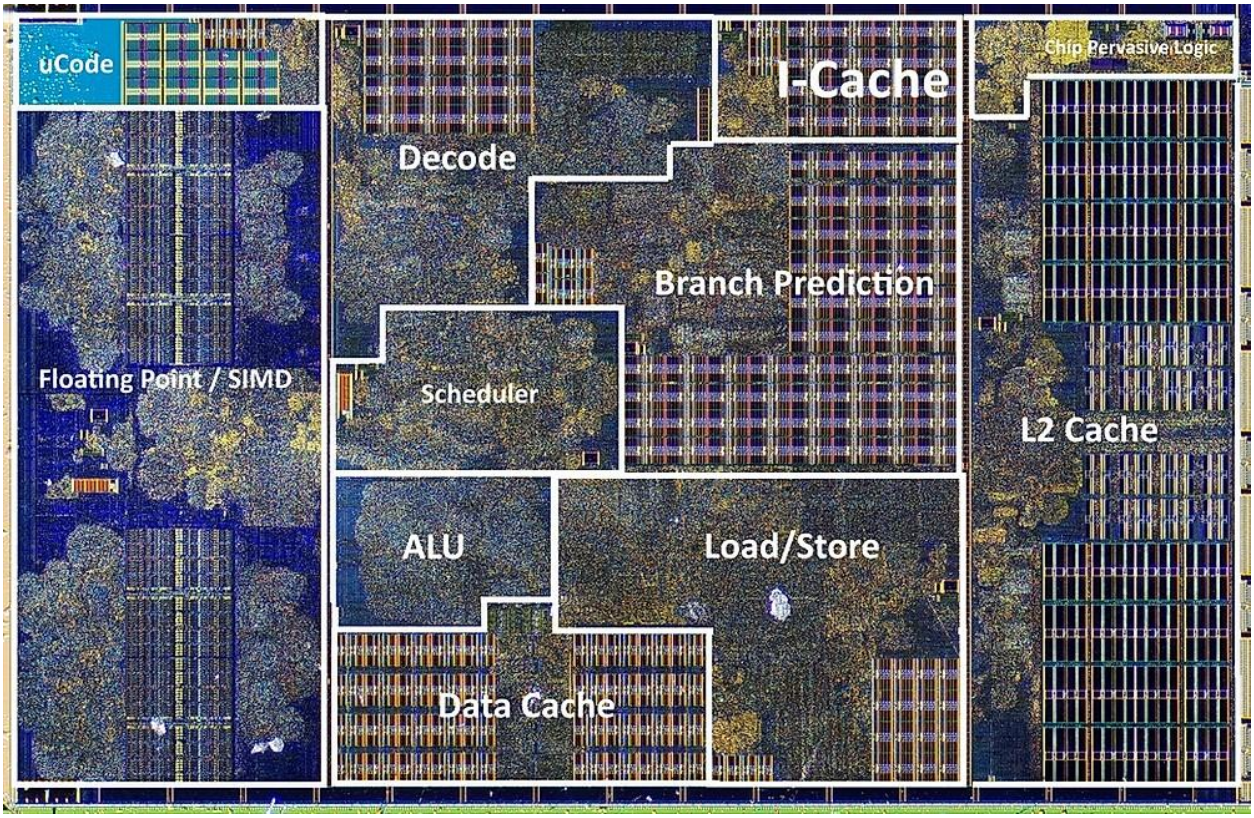
- Instruction Register - updated at each clock tick with instructions to be processed (decoded and executed)
- Accumulator - general purpose data register providing data (operands) to be processed by the ALU, also used to store results
- MUX (Multiplexers) - switches allowing the processor to select information from multiple sources and route it to a single destination
- Program Counter - register used to store the address in memory of the current instruction being executed

- **BUS** - connection between components in a computer
    - bus operate at regular interval measured in MHz
        - a computer with 200 MHz completes 200 million data transfers per second
    - bus contains multiple wires, each carrying a bit
    - address buses are generally 36 bit
    - a 36 bit bus can access 2^36 addresses or 64GB of data

- external bus communicates between internal and external hardware
    - ex USB, thunderbolt

- internal bus communicates between internal hardware such as cpu and memory
    - **address bus** transfers memory addresses
        - unidirectional
        - bus width determines the amount of memory a system can address
    - **data bus** transfers data among components
        - bidirectional
        - bus width determines rate at which data can be transferred
    - **control bus** carries commands from the CPU and returns status signals

- below, AMD Zen2 CPU



Binary Code

Binary Numbers

- what is the 8 bit binary number 01101000

Exponent:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Value:	128	64	32	16	8	4	2	1
ON/OFF:	0	1	1	0	1	0	0	0

- 64 + 32 + 8 = 104

- what is the 8 bit binary number 11111111

Value:	128	64	32	16	8	4	2	1
ON/OFF:	1	1	1	1	1	1	1	1

- 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 256
- this is the maximum 8 bit value

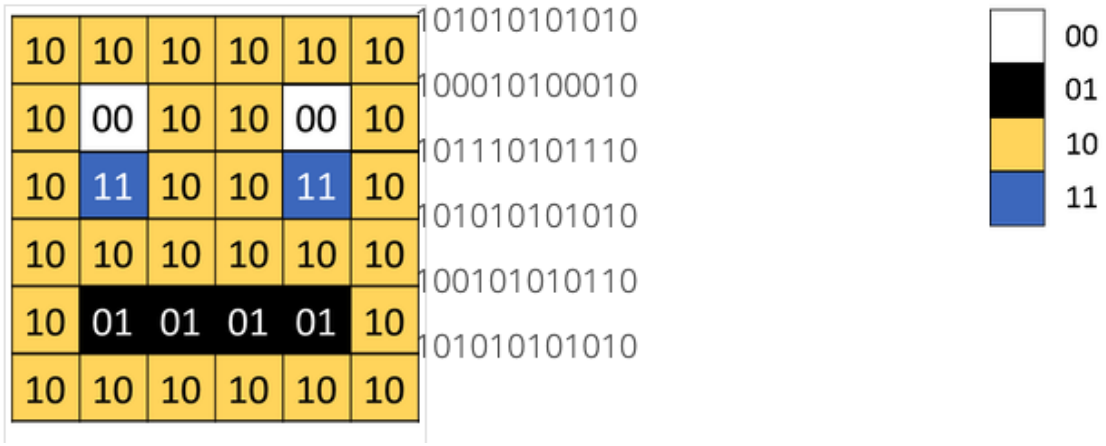
Characters

- ASCII converts binary numbers to characters
- to spell “hi” you would add 104 and 105
    - in binary this command would be: 0110100001101001

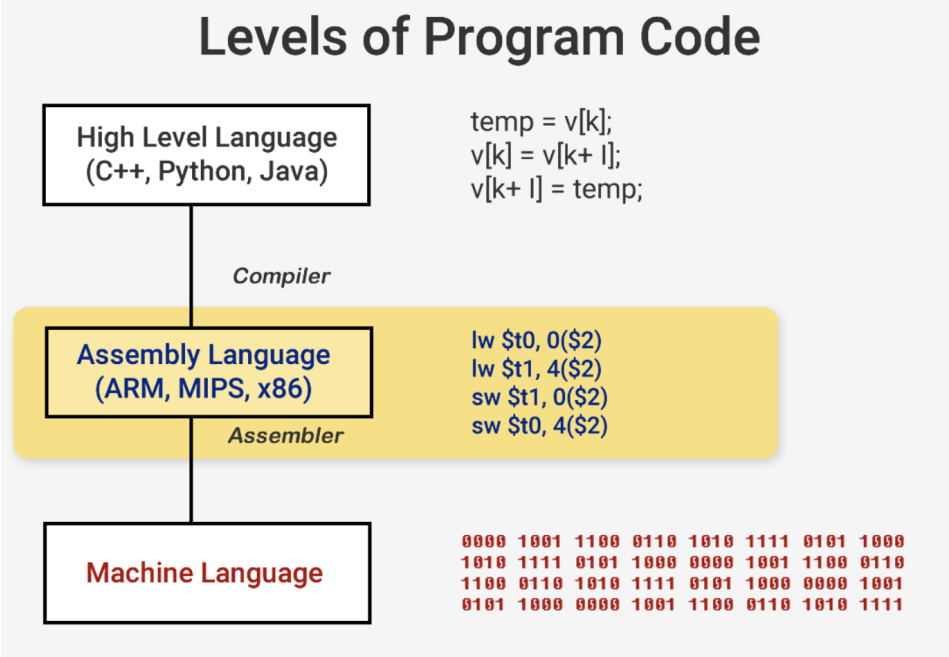
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t

Images

- each pixel and pixel color is represented by a binary code



Instruction Set Architecture



Assembler

- program that converts low level language (ISA) into binary code
- since each assembly language (ISA) is designed for a specific processor, assembling a program is performed using a simple one-to-one mapping from assembly code to machine code

- compilers, on the other hand, must convert generic high-level source code into machine code for a specific processor

ISA

- programs are compiled into low-level instructions by the ISA
    - this is the set of instructions the CPU understands and executes
- processors with different microarchitectures (different transistor count, layout, etc) can share a common instruction set

- some of the most common ISAs are x86, MIPS, ARM, RISC-V, and PowerPC
- ISAs can be broken up into two main categories: fixed-length and variable-length
    - RISC-V uses fixed length instruction so a predefined number of bits in each instruction determines what type of instruction it is
    - x86 uses variable length instruction, instructions can be encoded in different ways with different numbers of bits for different parts, because of this complexity the instruction decoder in x86 CPUs is typically the most complex part of the whole design
- fixed length instruction allows for easier decoding but limit the total number of instructions an ISA can support
    - RISC-V has about 100 open source instructions
    - x86 has a few thousand instructions

- below are some instructions from the RISC-V ISA

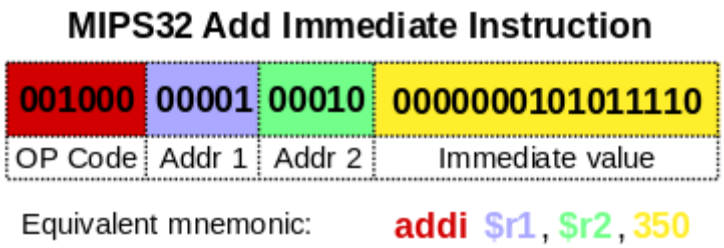
RV32I Base Instruction Set

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH

Common Instructions in an Instruction Set

- LOAD a number from RAM into the CPU
- ADD two numbers together
- STORE a number from the CPU back out to RAM
- COMPARE one number with another
- JUMP IF condition to jump to another address in RAM
- JUMP to another address in RAM
- OUTput to another device such as a monitor
- INput from a device such as a keyboard

- below is an opcode which sources bits from addr1, adds these bits to the immediate value, and pushes the output to addr2



- instructions include an opcode (the actual operation, adding above) and operands which specifies registers, memory locations, on data

- instruction vary in size and length from 8 to 64 bits
- most modern processors are 64 bit which means that each data value is 64 bits
    - 64-bit refers to the width of a CPU register, data path, and/or memory address. That means how much information a computer can handle at a time
    - 64-bit = the instruction length and memory that can be read and written with each data fetch

- fixed length instruction sets like the RISC (reduced instruction set computer) sacrifice code density to try and increase speed with higher clock frequency and more registers
    - RISC computers often require longer instructions to implement a task because of this and don't optimally use BUS bandwidth and cache space
- CISC (complex instruction set computers) have variable length code have high code density and complex instructions (loops, operands, etc)

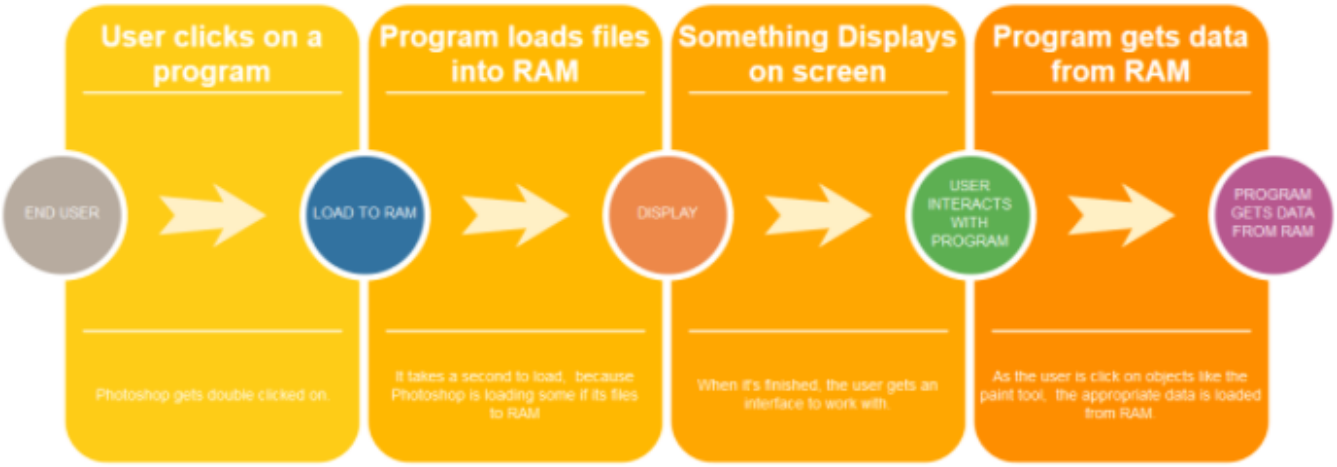
Instruction Set Implementation

- the instruction decoder interprets the binary instruction it receives and puts the necessary tasks in motion
- low level language that has been compiled into binary by the assembler can be implemented (run) by the hardware in many ways
- when designing the microarchitecture of a processor, engineers use blocks of "hard-wired" electronic circuitry (often designed separately) such as adders, multiplexers, counters, registers, ALUs, etc.
- from troubleshooting with this hardware a register transfer language is created to describe the decoding and sequencing of each instruction of an ISA using this physical microarchitecture
    - some computer designs "hardwire" the complete instruction set decoding and sequencing (just like the test microarchitecture)
    - some employ microcode routines or tables typically as on-chip ROM

Memory and Motherboard

RAM

- external to the CPU
- stores short term memory from web browser data to image editing software data
- newer RAM offers higher speeds through higher mHZ rating
- RAM also has a clock speed like a CPU
- when a program is opened its files are loaded into RAM



- memory cells are etched onto a silicon wafer in an array of columns (bitlines) and rows (wordlines). The intersection of a bitline and wordline constitutes the address of the memory cell
- a single transition and capacitor make up each memory cell

- the transistor allows the state of the capacitor to be changed
- DRAM works by sending a charge through the appropriate column (CAS) to activate the transistor at each bit in the column
- a capacitor acts like a bucket that stores electrons. It is either filled with electrons (1) or empty (0)
- the bucket leaks over time so it must be constantly refreshed, to do this the memory controller reads the memory and then writes it back every 70 nanoseconds
- memory cells have a support infrastructure of circuits that do the following:
    - identify each row and column (row address select and column address select)
    - keep track of the refresh sequence (counter)
    - read and restore the signal from a cell (sense amplifier)
    - tell a cell whether it should take a charge or not (write enable)

ROM

- nonvolatile data source, data is not lost when power is removed
- ROM is located on the motherboard
- data stored on ROM is unchangeable
    - this necessitates the programming of perfect and complete data when the chip is created
    - new types of ROM such as flash memory are re-programmable
- ROM uses a grid of columns and rows similar to RAM
- ROM uses a diode to connect lines if the value is 1, if the value is 0 the lines are not connected
    - a diode allows current to flow forward in only one direction if the current is above a certain threshold

Flash Memory

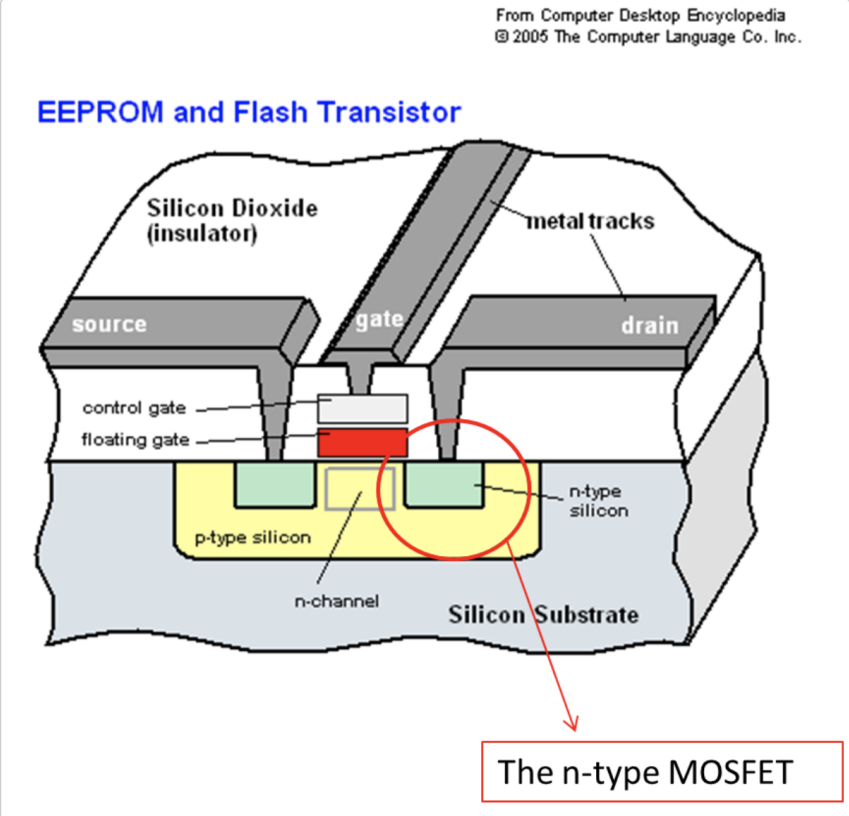
- one style of ROM is flash memory
- solid state storage device, no moving parts
- examples of flash memory include BIOS and memory sticks for cameras
- type of EEPROM chip, which stands for Electronically Erasable Programmable Read Only Memory
- two transistors are separated from each other by a thin oxide layer. One of the transistors is known as a floating gate, and the other one is the control gate
- the floating gate's only link to the bitline, or wordline, is through the control gate. As long as this link is in place, the cell has a value of 1. To change the value to a 0 requires Fowler-Nordheim tunneling
- tunneling is used to alter the placement of electrons in the floating gate. An electrical charge, usually 10 to 13 volts, is applied to the floating gate

- this charge causes the floating-gate transistor to act like an electron gun
    - the excited electrons are pushed through and trapped on the other side of the thin oxide layer, giving it a negative charge
    - negatively charged electrons act as a barrier between the control gate and the floating gate

- a device called a cell sensor monitors the level of the charge passing through the floating gate. If the flow through the gate is above the 50 percent threshold, it has a value of 1. When the charge passing through drops below the 50-percent threshold, the value changes to 0
- flash memory uses in-circuit wiring to apply the electric field either to the entire chip or to predetermined sections known as blocks
- this erases the targeted area of the chip, which can then be rewritten

SSD

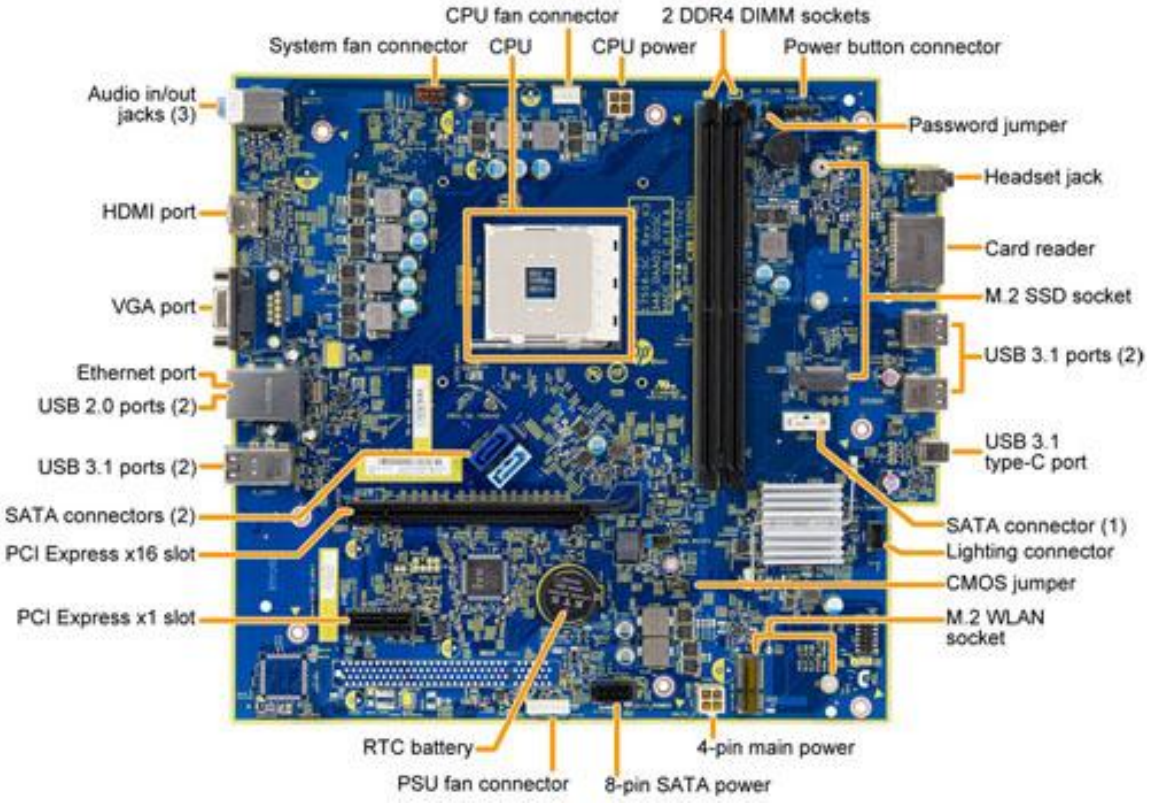
- SSD memory is rewritable and non-volatile
- there are two types of SSD flash memory: NOR and NAND
    - NOR is arranged in parallel and is very fast for reads, but slow for writes
        - often used in places where code is written once and read a lot
    - NAND is arranged in serial, faster for writes, and takes up less space than NOR
- most computers use a NAND SSD
- operates similarly to flash memory
- NAND flash has transistors arranged in a grid with columns and rows



- flash memory and SSDs work by adding a second floating gate to each cell
- when the transistor is in the “on” state electrons pass through and some get stuck in the floating gate and remain stored there as a 1 with or without power
- to delete data a large positive voltage is sent to the cell
    - the positive charge on the source makes the negative charged electrons move from the negative charged floating gate to the source, resulting in the floating gate losing its charge and reverting to a 0

Motherboard

- allows all hardware component to receive power and communicate with each other
- the socket for the microprocessor dictates what type of CPU can be used
- the chipset is made of two parts, the southbridge and northbridge which connect the CPU to other hardware components
    - northbridge connects directly to the processor via the front side bus (FSB) and gives the CPU fast access to memory
    - southbridge connects to USB ports and hard drives
- real time clock chip is a battery operated chip that maintains basic settings and system time



BIOS (Basic Input Output System)

- uses flash memory
- located on the motherboard
- makes sure all chips, ports, hard drives, and the CPU function together
- the most important role of the BIOS is to load the operating system
    - when the computer turns on the CPU tries to execute its first instruction, but it can't get the instruction from the OS because the OS is located on the SSD and the CPU can't get to the SSD without instructions on how to do so
- BIOS also provides low level routines the OS uses to interface to hardware devices
    - manages the keyboard, screen, and ports especially when the computer is booting

Bios Bootup Sequence

1. check **CMOS** setup for custom settings
    a. check for info stored in a tiny (64 byte) amount of RAM located on a complementary metal oxide semiconductor (CMOS) chip
    b. CMOS Setup provides detailed information particular to your system and can be altered as your system changes
2. load **interrupt handlers** and **device drivers**
    a. interrupt handlers are small pieces of software that act as translators between the hardware components and the operating system, when you press a key on your keyboard the signal is sent to the keyboard interrupt handler which tells the CPU what it is and passes it on to the operating system
    b. device drivers identify the base hardware components such as keyboard, mouse, hard drive and floppy drive. Since the BIOS is constantly intercepting signals to and from the hardware, it is usually copied, or shadowed, into RAM to run faster
3. check for a video card / GPU
    a. most GPUs have their own bios that will boot up the GPU
4. check if it is a cold boot or reboot
    a. BIOS checks to see if this is a cold boot or a reboot. It does this by checking the value at memory address 0000:0472. A value of 1234h indicates a reboot, and the BIOS skips the rest of POST. Anything else is considered a cold boot
5. if it is a cold boot
    a. BIOS verifies RAM by performing a read/write test of each memory address
    b. checks the PS/2 ports or USB ports for a keyboard and a mouse
6. BIOS then looks for boot devices
    a. storage devices identified as boot devices in the CMOS setup are used to launch the OS

Operating System

- OS is a collection of software that manages hardware and provides services for programs
- major components of an OS are file system, scheduler, and device driver
- All user software needs to go through the operating system in order to use any of the hardware, whether it be as simple as a mouse or keyboard or as complex as an Internet component

Process Management

- a process is a program in execution
- the Process Control Block is a data structure maintained by the OS for every process
- the PCB keeps track of all the following information
    - process state - whether the process is ready, waiting, or running
    - process privileges - what system rescounces a program can access
    - program counter - pointer to the address of the next instruction to be executed
    - CPU registers - CPU registers where processes need to be stored for execution
    - CPU scheduling information - priority about which processes will run first
    - memory management information - information on memory limits, etc
- the OS also manages threads when a process is split into multiple threads to take advantage of parallelism

Memory Management

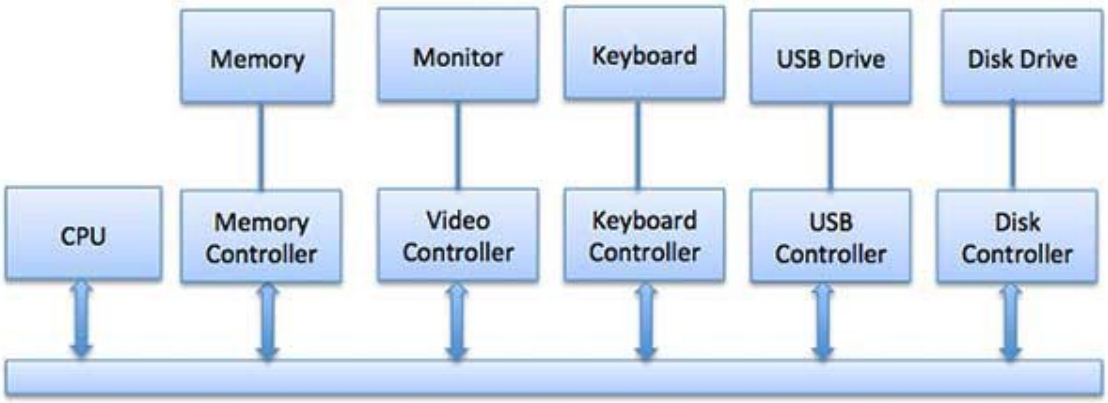
- OS moves processes back and forth between registers, caches, RAM, and SSD
- OS decides how to allocate memory

Inter Process Communication

- some processes run independently, others called cooperating processes can be affected by other processes
- OS can coordinate shared memory, message sharing, etc between processes

Input / Output Management

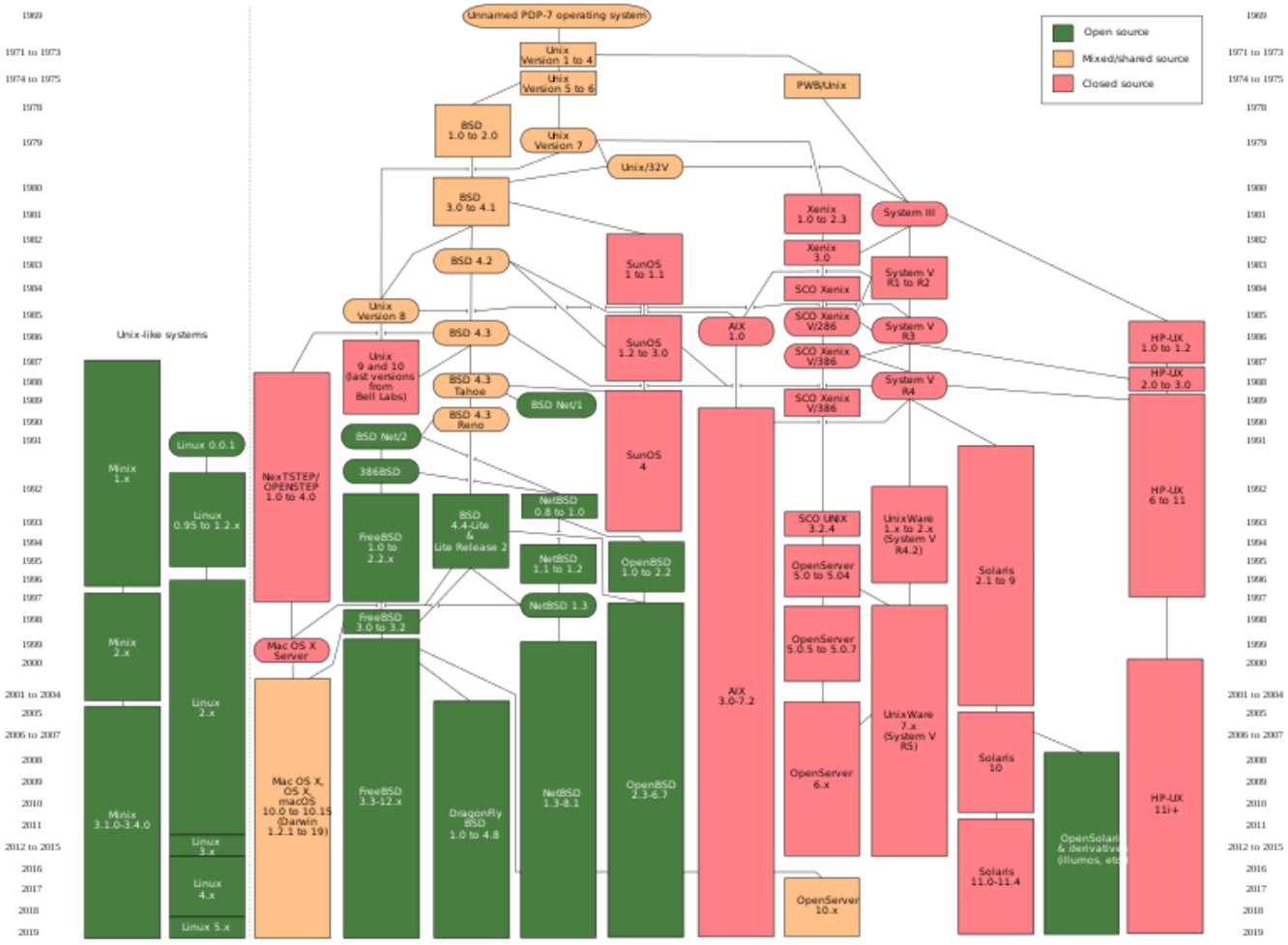
- OS manages mouse, keyboard, usb, etc and the interaction of these devices with the CPU



- Windows, MacOS, and Linux are all written mostly in C with some parts in assembly language
- application code is usually executed directly by the hardware, but it frequently makes system calls to the OS or is interrupted by the OS
    - for hardware functions such as input, output, and memory allocation the OS acts as an intermediary between programs and computer hardware

History of Operating Systems

- Unix was the first major operating system and most operating systems are derived from Unix
    - Unix was originally written in assembly language before being rewritten in C



- Unix like operating systems include Linux and MacOS
- macOS, a replacement for Apple's earlier (non-Unix) Mac OS, is a hybrid kernel-based BSD variant derived from NeXTSTEP, Mach, and FreeBSD

Higher Level Languages

C

- general purpose, procedural language
- released in 1972
- C provides constructs that map efficiently to typical machine instructions
    - thus is frequently used to build applications that were previously coded in assembly language
- C compilers are available for the majority of computer architectures and operating systems
- it was designed to be compiled to provide low-level access to memory and language constructs that map efficiently to machine instructions, all with minimal runtime support
- a C program written with portability in mind can be compiled for a wide variety of computer platforms and operating systems with few changes to its source code
- many higher level language such as python are built on C

Java

- general purpose programming language, class based, object oriented
- designed to have as few implementation dependencies as possible
    - WORA - write once run anywhere
- Java scripts are typically compiled in bytecode that can be run on any Java Virtual Machine regardless of underlying computer architecture
- can be used to create everything from desktop applications to smartphone apps

Higher Higher Level Languages (R, python, etc)

C++

- extension of C with very similar syntax and compilers

Python

- written in C
- object oriented approach

R

- written mostly in C with packages being written mostly in R