Thread

Plan

- CPU ?

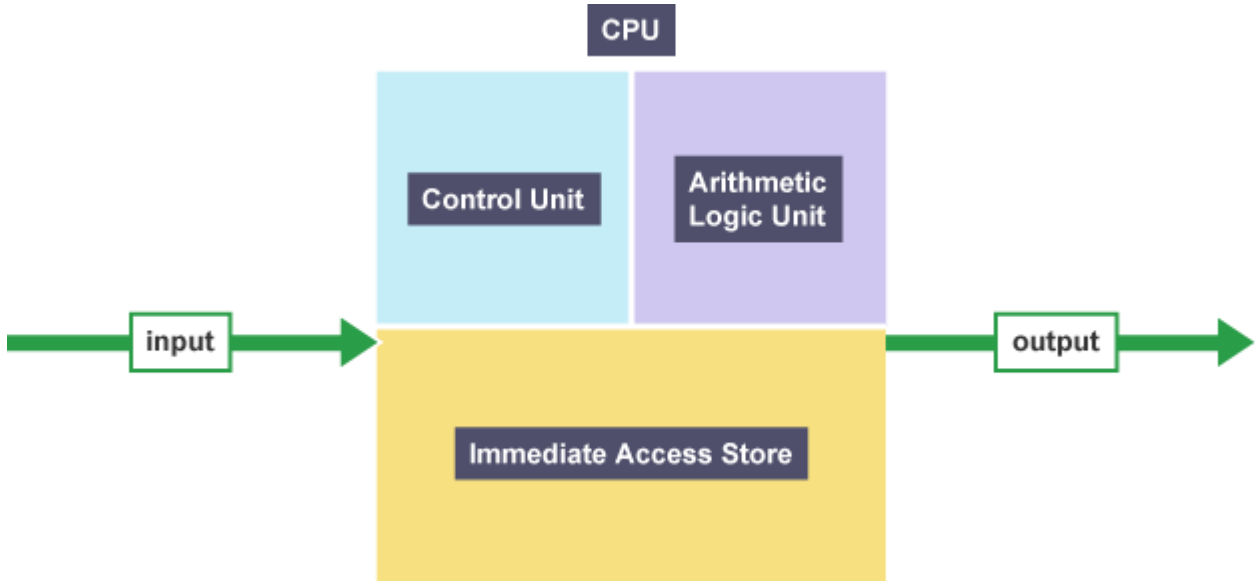
Before starting

- What is CPU?

CPU 1

- The CPU is where processes such as calculating, sorting and searching take place. Whatever is done on our computers, such as checking emails, playing games and doing homework, the CPU has processed the data we use.
- The CPU is made up of three main components, the control unit, the immediate access store and the arithmetic and logic unit.
- CPU - Central processing unit - the brain of the computer that processes program instructions.
- CPU hisoblash, saralash va qidirish kabi jarayonlar sodir bo'ladigan joy. Elektron pochta xabarlarini tekshirish, o'yin o'ynash va uy vazifasini bajarish kabi kompyuterlarimizda nima qilinmasin, protsessor biz foydalanadigan ma'lumotlarni qayta ishlaydi.
- Protsessor uchta asosiy komponentdan iborat: boshqaruv bloki, tezkor kirish do'koni va arifmetik va mantiqiy blok.
- Markaziy protsessor - dastur ko'rsatmalarini qayta ishlaydigan kompyuterning miyasi.

CPU - UML


CPU
Control Unit
Arithmetic
Logic Unit
input
output
Immediate Access Store

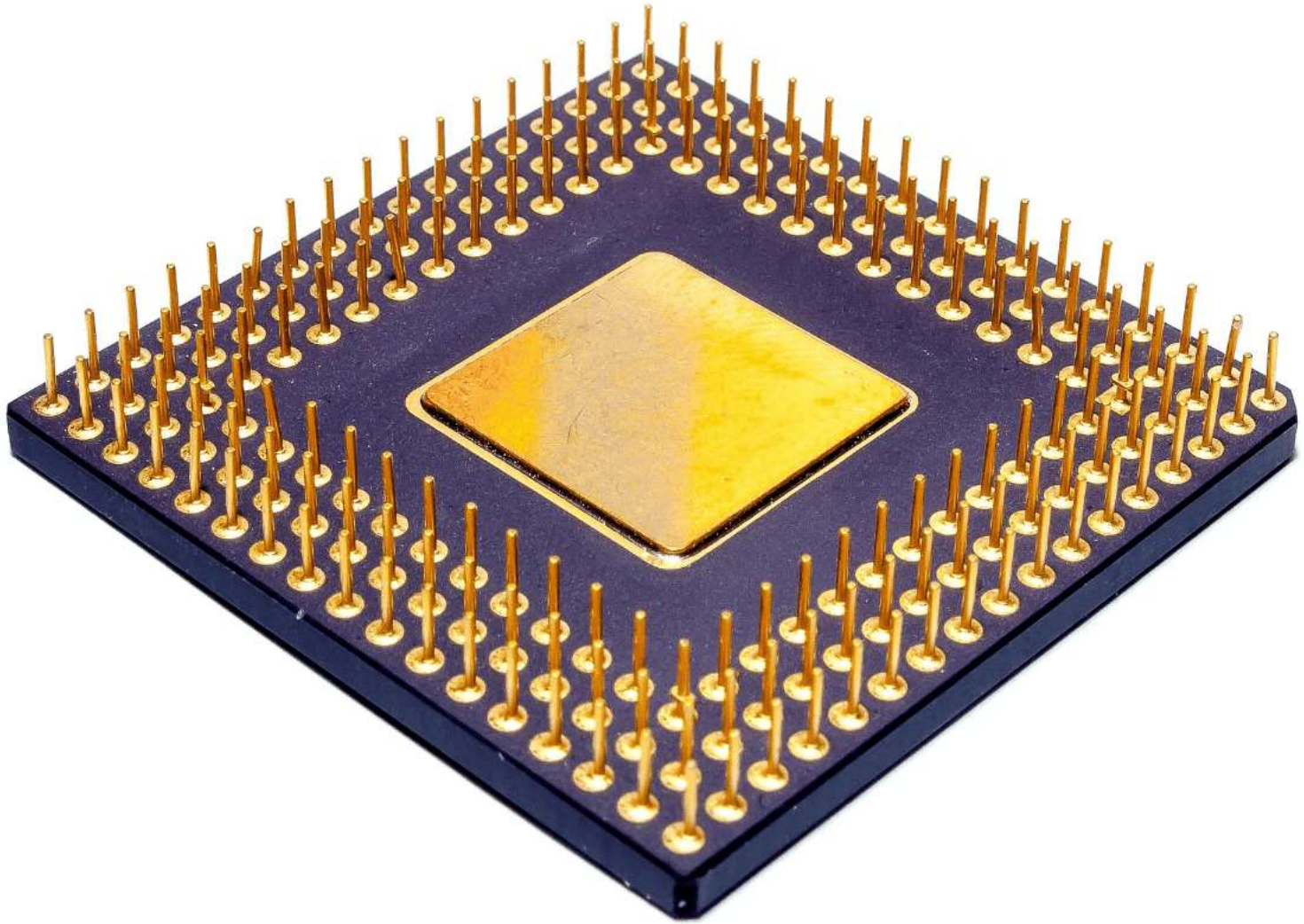
CPU 2

- The modern CPU usually contains a number of microprocessors (multi-core). It is often described as the 'brain of the computer'.
- The purpose of the CPU is to process data. It is where all the searching, sorting, calculating and decision making takes place in the computer.
- The CPU will issue instructions to other devices depending on the results of the processing. For example, if the user wants to print a document, the CPU will instruct the printer about the task.
- Zamonaviy CPU larni odatda bir qancha mikroprotsessorlarni (ko'p yadroli) o'z ichiga oladi. U ko'pincha "kompyuterning miyasi" deb ta'riflanadi.
- Protsessorning maqsadi ma'lumotlarni qayta ishlashdir. Bu erda barcha qidirish, saralash, hisoblash va qaror qabul qilish kompyuterda amalga oshiriladi.
- Protsessor qayta ishlash natijalariga qarab boshqa qurilmalarga ko'rsatmalar beradi. Misol uchun, agar foydalanuvchi hujjatni chop qilmoqchi bo'lsa, CPU printerga vazifa haqida ko'rsatma beradi.

CPU image 1

dasturlash.uz

CPU image 2



dasturlash.uz

CPU

- A computer’s speed is heavily influenced by the **CPU** it uses. There are three main factors that affect how quickly a CPU can carry out **instructions**:
  - clock speed
  - cores
  - cache
- Kompyuter tezligiga u foydalanadigan protsessor ta’sir qiladi. CPU ko‘rsatmalarni qanchalik tez bajarishiga ta’sir qiluvchi uchta asosiy omil mavjud:

CPU - Clock Speed 1

- Computers can only do one thing at a time. It may appear that many things are happening simultaneously, for example you may be listening to music while writing an essay and also downloading some software in the background.
- In reality, the computer can only process one instruction at a time. It is just that the machine is so fast, to you, everything seems to happen at once.
- Kompyuterlar bir vaqtning o'zida faqat bitta narsani qila oladi. Bir vaqtning o'zida ko'p narsalar sodir bo'layotgandek tuyulishi mumkin, masalan, insho yozish paytida musiqa tinglayotgan bo'lishingiz va fonda ba'zi dasturlarni yuklab olishingiz mumkin.
- Aslida, kompyuter bir vaqtning o'zida faqat bitta ko'rsatmani qayta ishlashga qodir. Bu shunchaki mashina juda tez, sizga hamma narsa bir vaqtning o'zida sodir bo'layotgandek tuyuladi.

CPU - Clock Speed 2

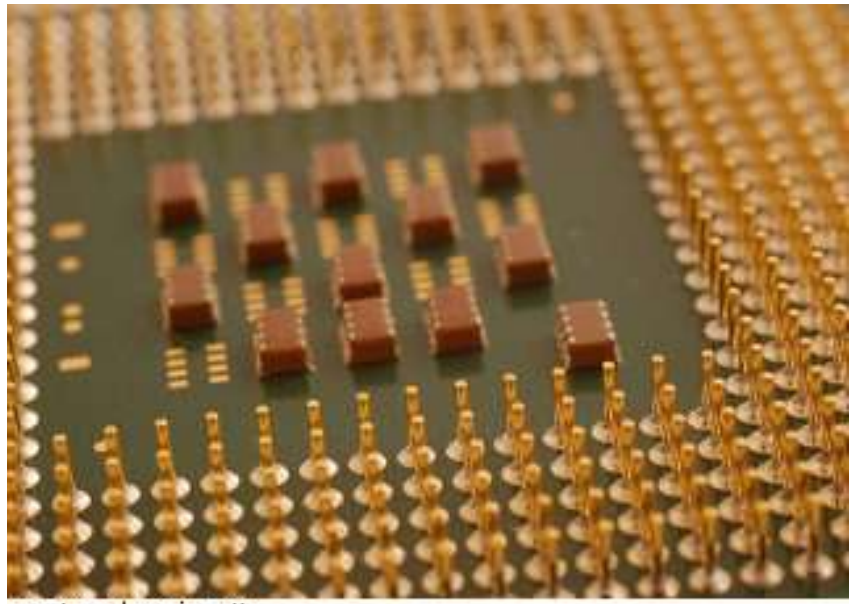
- A CPU processes digital data by taking each piece of data one-at-a-time and doing something with it. The amount of time that it has to process each piece of data is controlled by a quartz clock inside the CPU.
- CPU har bir ma'lumotni bir vaqtning o'zida olib, u bilan biror narsa qilish orqali raqamli ma'lumotlarni qayta ishlaydi. Har bir ma'lumotni qayta ishlash vaqti CPU ichidagi kvarts soati tomonidan boshqariladi.

- With every tick of the clock, the CPU is able to process one piece of data or execute
- Soatning har bir belgisi (tiqqildashi) bilan protsessor bitta ma'lumotni qayta ishlashga yoki bajarishga qodir

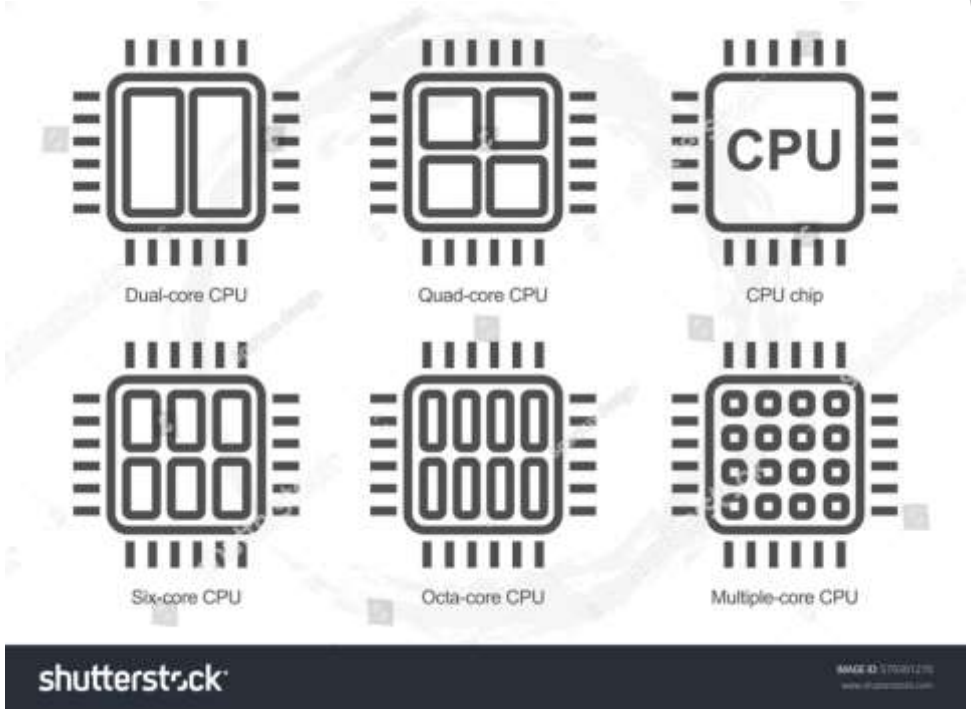
CPU - Clock Speed 3

- The CPU clock speed is measured in cycles per second. 1 cycle per second is also known as 1 Hertz
- A computer running at 1GHz can carry out a thousand million instructions per second. (billion)
- The clock on a modern desktop computer runs extremely quickly, typically three thousand million times a second (3 GHz).
- CPU soat tezligi soniyada aylanishlarda o'lchanadi. 1 sekundiga 1 sikl 1 Gerts deb ham ataladi.
- 1 gigagertsli chastotada ishlaydigan kompyuter soniyada 1 milliard ko'rsatmalarni bajarishi mumkin. (1,000,000,000)
- Zamonaviy desktop kompyuteridagi soat juda tez ishlaydi, odatda soniyada uch milliart marta (3 gigagertsli).

CPU - Cores



courtesy Lemsipmatt



CPU - Cores 1

- Now imagine that the clock cannot be sped up any more for technical reasons. How would you improve performance?
- A sensible way forward is to use two CPUs at the same time. The job in hand is shared between the two CPUs.
- In an ideal world, running two CPUs would give you twice the performance. In real life this is not quite the case because many applications have not been written to take advantage of extra processors.
- Oldinga borishning oqilona yo'li bir vaqtning o'zida ikkita protsessordan foydalanishdir. Qo'ldagi ish ikki protsessor o'rtasida taqsimlanadi.
- Endi tasavvur qiling-a, soatni texnik sabablarga ko'ra endi tezlashtirib bo'lmaydi. Ishlashni qanday yaxshilagan bo'lardingiz?
- Ideal dunyoda ikkita protsessorni ishga tushirish sizga ikki baravar ko'p ishlash imkonini beradi. Haqiqiy hayotda bu unchalik emas, chunki qo'shimcha protsessorlardan foydalanish uchun ko'plab ilovalar yozilmagan.

CPU - Cores 2

- However, by having more than one processor, you would certainly see an improved performance.
- A modern processing device may contain two or even four CPUs. Some chip-making companies call these CPUs 'cores'. So a dual-core device means it contains two CPUs and a quad-core contains four.

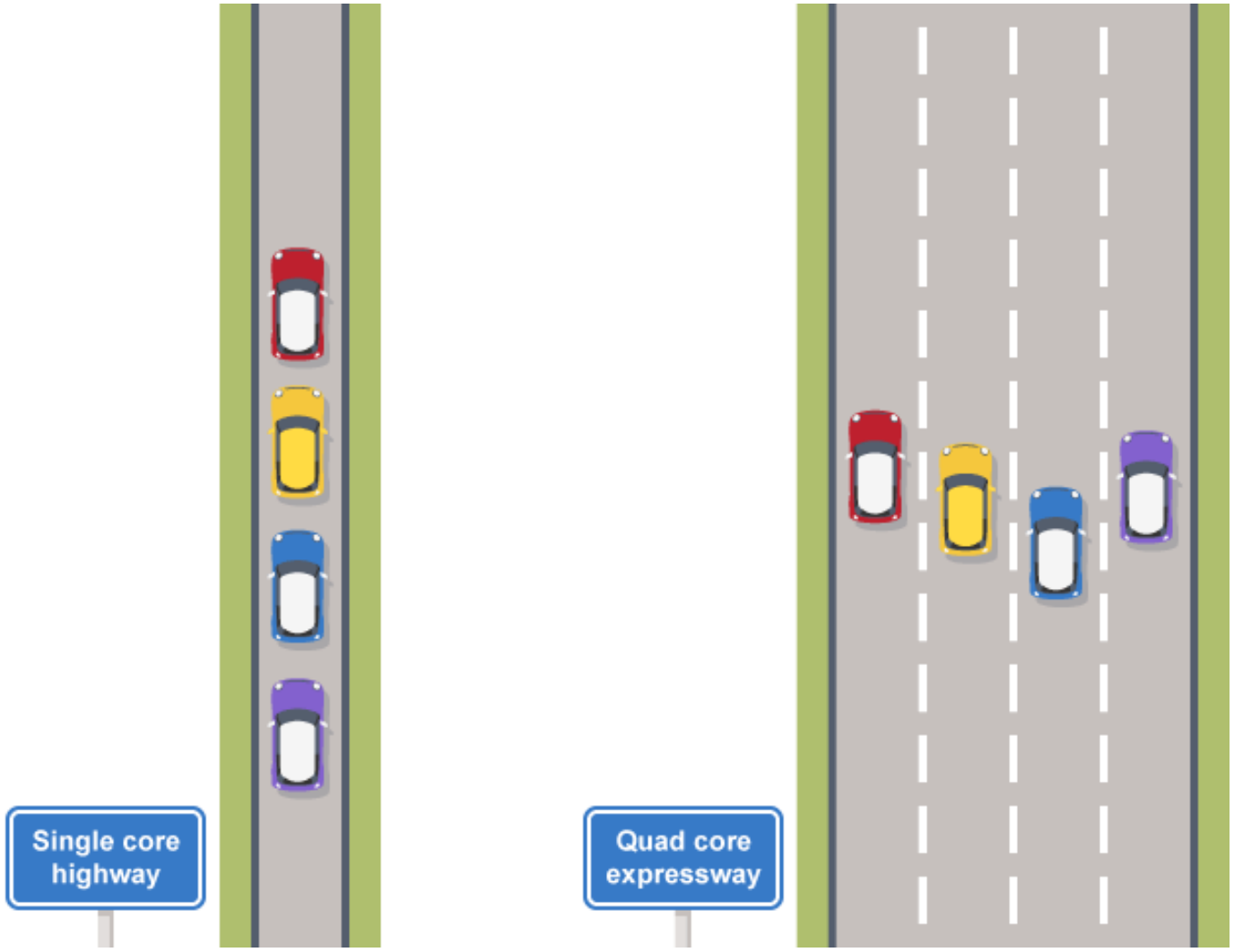
- Biroq, bir nechta protsessorga ega bo'lsangiz, siz, albatta, yaxshilangan ishlashni ko'rasiz.
- Zamonaviy ishlov berish qurilmasi ikkita yoki hatto to'rtta protsessorni o'z ichiga olishi mumkin. Ba'zi chip ishlab chiqaruvchi kompaniyalar ushbu protsessorlarni "yadro" deb atashadi. Shunday qilib, ikki yadroli qurilma ikkita protsessorni va to'rt yadroli to'rt yadroni o'z ichiga oladi.

CPU - Cores 3

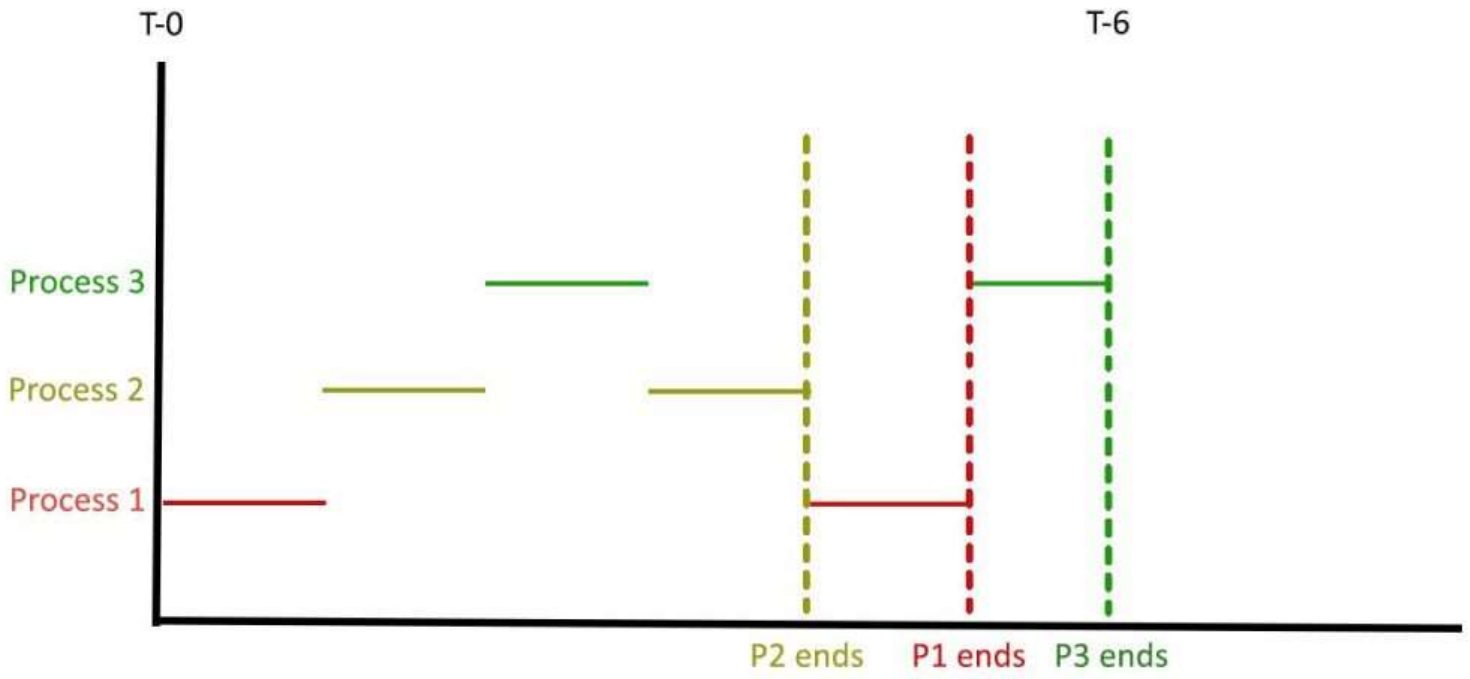
- A CPU is traditionally made up of a processor with a single core. Most modern CPUs have two, four or even more cores.
- A CPU with two cores, called a dual core processor, is like having two processors in one. **A dual core processor can fetch and execute two instructions in the same time it takes a single core processor to fetch and execute just one instruction**. A quad core processor has four cores and can carry out even more instructions in the same period of time.

- CPU an'anaviy ravishda bitta yadroli protsessordan iborat. Aksariyat zamonaviy protsessorlarda ikkita, to'rt yoki undan ko'p yadro mavjud.
- Ikki yadroli protsessor deb ataladigan ikkita yadroli protsessor birida ikkita protsessorga o'xshaydi. Ikki yadroli protsessor bir vaqtning o'zida ikkita ko'rsatmalarni olishi va bajarishi mumkin, faqat bitta buyruqni olish va bajarish uchun bitta yadroli protsessor kerak bo'ladi. To'rt yadroli protsessor to'rt yadroga ega va bir vaqtning o'zida ko'proq ko'rsatmalarni bajarishi mumkin.

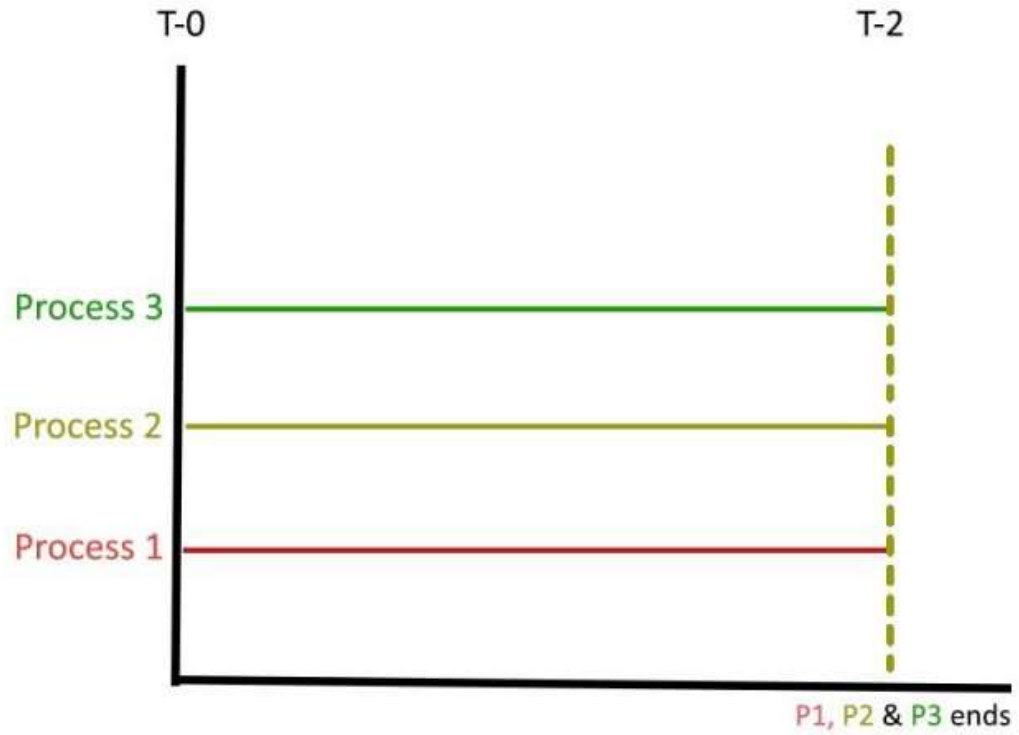
CPU - Cores 4



Single-core Processor



Multi-core Processor



CPU - Cores 4

- The main downside of using quad core processors is that they are more expensive to design and make, and they also use more power than single or dual core processors. Another disadvantage is that the instructions have to be split up to decide which core will execute them and the results have to be merged together again at the end, which slows the processor down a little.
- To'rt yadroli protsessorlardan foydalanishning asosiy salbiy tomoni shundaki, ularni loyihalash va ishlab chiqarish qimmatroq, shuningdek, ular bitta yoki ikki yadroli protsessorlarga qaraganda ko'proq quvvat sarflaydi. Yana bir kamchilik shundaki, ko'rsatmalar qaysi yadro ularni bajarishini hal qilish uchun bo'linishi kerak va natijalar oxirida yana birlashtirilishi kerak, bu esa protsessorni biroz sekinlashtiradi.

CPU - Cache

- A cache (pronounced ‘cash’) is a tiny block of **memory** built right onto the processor. The most commonly used instructions and **data** are stored in the cache so that they are close at hand. The bigger the cache is, the more quickly the commonly used instructions and data can be brought into the processor and used.

- Kesh ("naqd" deb talaffuz qilinadi) - bu protsessorga o'rnatilgan kichik xotira blokidir. Eng ko'p ishlatiladigan ko'rsatmalar va ma'lumotlar keshda saqlanadi, shuning uchun ular yaqin bo'ladi. Kesh qanchalik katta bo'lsa, tez-tez ishlatiladigan ko'rsatmalar va ma'lumotlar protsessorga tezroq kiritilishi va ishlatilishi mumkin.

CPU links

- https://www.teach-ict.com/gcse_new/computer%20systems/cpu/miniweb/index.htm
- https://www.bbc.co.uk/bitesize/guides/zws8d2p/revision/1
- https://www.bbc.co.uk/bitesize/guides/zws8d2p/revision/2
- https://emeraldforhome.com/cpu-cores-vs-threads/
- https://www.temok.com/blog/cores-vs-threads/

Concurrency vs Parallelism

- Concurrency means executing multiple tasks at the same time but not necessarily simultaneously.
- Parallelism means that an application splits its tasks up into smaller subtasks which can be processed in parallel (all at the same time), for instance on multiple CPUs at the exact same time.
- Parallelism requires hardware with multiple processing units, essentially. In single-core CPU, you may get concurrency but NOT parallelism. Parallelism is a specific kind of concurrency where tasks are really executed simultaneously.

Synchronous vs Asynchronous 1

- Synchronous (Sync) and asynchronous (Async) programming can be done in one or multiple threads. The main difference between the two is when using synchronous programming we can execute one task at a time, but when using asynchronous programming we can execute **multiple** tasks at the same time.

- Synchronous (Sync) va asynchronous (Async) dasturlassh bir yoki birnecchta Thread larda bajarilishi mumkin. Ularning asosiy farqlari synchronous dasturlashda bir vaqtni o’zida bitta Thread ishlaydi. Ammo asynchronous dasturlashda birvaqtni o’zida bir nechta Thread ishlaydi.

Sync

- Sync:
  - Single thread: I start to boil an egg, after it is boiled I can start to toast the bread. I have to wait for a task to be done in order to start another.
  - Multi-thread: I start to boil an egg, after it is boiled my mom will toast the bread. The tasks are performed one after another and by different persons(threads).
- Sync :

Async

- Async:
  - Single thread: I put the egg to boil and set a timer, I put the bread to toast and start another timer, when they are done, I can eat. In asynchronous I don't have to wait for a task to be done in order to start one other task.
  - Multi-thread: I hire 2 cooks and they will boil the egg and toast the bread for me. They can do it at the same time, neither they have to wait for one to finish in order for the other one to start.

- https://www.cognizantsoftvision.com/blog/async-in-java/

Thread

Thread

- A thread is a unit of execution on concurrent programming.
- Multithreading is a technique which allows a CPU to execute many tasks of one process at the same time.
- Threads allows a program to operate more efficiently by doing multiple things at the same time.
- These threads can execute individually while sharing their resources.
- Threads can be used to perform complicated tasks in the background without interrupting the main program.

- Thread - bu parallel dasturlashning bajarish birligi.
- Multithreading - bu CPU ga bir vaqtning o'zida bir prosessni (dasturni) ko'plab vazifalarini bajarishga imkon beradigan texnikadir.
- Threads bir vaqtning o'zida bir nechta ishlarni bajarish orqali dasturga yanada samarali ishlash imkonini beradi.
- Bu thread lar alohida ishlashi va shu paytning o’zida resource larni almashishi mumkin.
- Thread asosiy dasturni to'xtatmasdan, orqa fonda murakkab vazifalarni bajarish uchun ishlatilishi mumkin.

Thread in JVM

- The Java Virtual Machine allows an application to have multiple threads and execute them concurrently.
- Java Virtual Mashina dasturga bir vaqtning o'zida bir nechta Thread/oqim larni ega bo'lish va ulanri bir vaqtni o’zida ishga tushurish imkonini beradi.

MultiThreading

- Multi

Procces based multitasking

Thread based multitasking

Defining a Thread

- Thread yaratish.
- We can define a thread in two ways:
  - 1. By extending Thread class.
  - 2. By implementing Runnable interface.
- Biz Thread ni 2ta yo’l bilan yaratishimiz mumkin.

Defining a Thread - extending Thread 1

```
public class MyThread extends Thread {

    @Override
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println("child Thread");
        }
    }
}
```

Defining a Thread - extending Thread 12

```
public class ThreadDemo {

    public static void main(String[] args) throws InterruptedException {
        MyThread myThread = new MyThread(); //< instantiation
        myThread.start(); //< START_CHILD_THREAD

        for (int i = 0; i < 10; i++) {
            System.out.println("main Thread");
        }

    }
}
```

Defining a Thread – implement Runnable 1

```
public class MyRunnable implements Runnable {
    @Override
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println("child class");
        }
    }
}
```

Defining a Thread – implement Runnable 2

```
public class RunnableDemo {
    public static void main(String[] args) {
        MyRunnable myRunnable = new MyRunnable();

        Thread thread = new Thread(myRunnable);
        thread.start();

        for (int i = 0; i < 10; i++) {
            System.out.println("main Thread");
        }
    }
}
```

Thread run() method

- run() method it is a job of the Thread.
- run method bu Thread ning bajarishi kerak bo’ladigan ishi hisoblanadi. Thread ishga tuganida u run() methodni chaqiradi va methodda yozgan ishlarni bajaradi.

Thread Scheduler

Thread Scheduler 1

- It is a part of JVM.
- It is responsible to schedule threads that is if multiple threads are waiting to get a chance to executing than in which order threads will be executed is decided by thread scheduler.
- We can’t expect exact algorithm followed by thread schedule. It is varied from JVM to JVM. Hence we can not expect thread execution order and exact output.
- Bu JVM ning bir qismi dir.
- U Thread larni rejalashtirish uchun javobgardir. Yani agar bir nechta Thread CPU da ishlash uchun kutub turgan bo’lsa ular qaysi tartibda CPU da ishlashi mumkin ligini ThreadScheduler hal qiladi.
- Biz ThreadSchedule ni qani ishlash algoritmni ayta olmaymiz. Bu JVM ga qarab farq qiladi. Shu sababdan biz Thread larni ishlash tartibini va aniq natijani ayta olmaymiz.

Thread Scheduler : link

- https://www.javatpoint.com/thread-scheduler-in-java

Thread start() method 1

- Thread class start() method is responsible to register the thread with thread schedule and all other mandatory activities.
- Hence without executing start() method there is no change od starting a new Thread in Java.
- Start method tasks:
  - Register this thread with thread schedule.
  - Perform all other mandatory activities.
  - Invoke run() method.
- Thread class ning start() metodi shu thread ni Thread Schedule (thread jadvali) da ro'yhatdan o'tkazadi va qolgan muhim ishlarni bajaradi.
- Javada start() – metodini chaqirmasdan yangi Thread boshlash iloji yo'q.
- Start metodi bajaradigan ishlari:
  - Thread schedule ga registratsiya qiladi.
  - Qolgan muhim ishlarni bajaradi.
  - run() methodni ishga tushuradi.

Thread start() method 2

- What if we are not overriding run method?
  - Thread class run method will be executed which has empty implementation.
  - Hence we will not get any output.
- What if we override start method?
  - Thread will not start and start() method will execute as a normal method.
  - It is not recomented to override start method.

Thread life cycle

- Hayot tarzi. Tug’ilishidan to o’lgunichia bo’lgan jarayon bosqichlari.

Thread life cycle – Simple Life cycle

- Oddiy life cycle namuna:
- Thread life cycle also known as Thread States
- 1. New - Born
  - t = new Thread()
- 2. Ready – Runnable
  - t.start()
- 3. Running
  - If Thread Schedule allocates procces
- 4. Dead
  - If run() method completed

Thread life cycle : New

- **New:** Whenever a new thread is created, it is always in the new state. For a thread in the new state, the code has not been run yet and thus has not begun its execution.
- Yangi Thread yaratilganda u hardoim New state(hola) da bo’ladi. Thread ning stati (holati) New bo’lsa bu hali thread o’z ishini boshlamadi degani.

Thread life cycle : Ready – Runnable

- When a thread invokes the start() method, it moves from the new state to the Ready/Runnable state.
- Thread da start() metodi chaqirilgandan keyin uning stati(holati) new dan Ready/Runnable state ga o’tadi.

Thread life cycle : Running

- When the thread gets the CPU, it moves from the runnable to the running state.
- Generally, the most common change in the state of a thread is from runnable to running and again back to runnable.
- ThreadSchedule thread ga CPU ni bo'shatib berganidan keyin, thread runnable state dan running state ga o'tadi.
- Ko'p holatda Thread ning stati (holati) Runnable dan Running ga va Running dan Runnable ga o'zgaradi.

Thread life cycle : Dead

- A thread reaches the Dead state because of the following reasons:
  - When a thread has finished its job, then it exists or terminates normally.
  - **Abnormal termination:** It occurs when some unusual events such as an unhandled exception or segmentation fault.

- Thread Dead state ga quyidagi holarlarda o’tishi mumkin.
  - Qachon Thread o’z ishini tugatganida. Bunda odatiy holatda o’z ishini tugatadi.
  - Noodatiy tugatish: qachon Thread ishlashi jarayonida ushlanmagan hatoliklar sodir bo’lsa.

Thread life cycle will be continued

Thread name

- Thread nomi.

Thread name 1

- The Thread class provides methods to change and get the name of a thread. By default, each thread has a name, i.e. thread-0, thread-1 and so on. By we can change the name of the thread by using the setName() method. The syntax of setName() and getName() methods are given below:
- Thread klassi yordaminda Thread ning nomini o'zgartirish va berilgan nomi olishimkoni bor.
- **public** String getName(): is used to **return** the name of a thread.
- **public void** setName(String name): is used to change the name of a thread.

Thread name example

```
public static void main(String[] args) {

        MyThread myThread = new MyThread();
        myThread.start();
        System.out.println(myThread.getName());

        Thread main = Thread.currentThread();
        main.setName("BigMan");
        System.out.println(Thread.currentThread());

    }
```

Thread Priority

- Thread ustuvorligi.

Thread Priority 1

- Every Thread in java has some Priority.
- It may be default priority , generated by JVM or Custom priority provided by programmers.
- Priority can be from 1 to 10.
- 1 – min priority.
- 10 – max priority.

Thread Priority 2

- Thread schedule will use priority while allocating processor.
- That thread which is having heights priority will get chance first.
- If two thread having same priority then we can’t expect exact executing order. It depends from Thread schedule.
- If we try set big number as priority we get RE: IllegalArgumentException
- The default priority for main Thread is 5.
- But for all Remaining Thread priority will be inherited from parent priority.

Thread Priority example

```
public static void main(String[] args) {
        MyThread myThread = new MyThread();
        myThread.setPriority(10);
        myThread.start();
    }
```

Prevent Thread Execution

- Thread ni bajarilishini oldini olish

Prevent thread execution ways

- We can prevent thread execution by using the following methods:
- yield() - Yo'l bering
- join() – kutmoq deb tarjima qilsak to’g’ri bo’ladi.
- sleep() – uxlash.

Thread yield()

- yield (yo'l bering) method causes to pause current executing thread to give chance for waiting threads of same priority
- If there is no waiting thread or all waiting thread have lower priority then same thread can continue it is execution.
- If multiple thread are waiting with same priority then which waiting thread will get chance we can not expect. It depends on thread scheduler.
- The Thread which is yield(), it will leave processor .

Thread yield() example

```
public class MyThread extends Thread {
    @Override
    public void run() {
        Thread.yield();
        for (int i = 0; i < 10; i++) {
            System.out.println("Child thread");
        }
    }
}

public static void main(String[] args) {
        MyThread myThread = new MyThread();
        myThread.start();
        for (int i = 0; i < 10; i++) {
            System.out.println("main thread");
        }
    }
```

Thread join()

- join (kutish)
- Thread class join method permits/makes one thread to wait until the other thread to finish its execution.
- If a thread wants to wait until finishing another thread then we should use join method.
- For example id a Thread t1 wants to wait until completing t2 then t1 has to call t2.join();
- It t1 executes t2.join() then immediately t1 will be enter into waiting state until t2 completes (finishes it is job) t1 can continue it is executing .
- Hayotiy misol:
  - https://youtu.be/1V2FC5aomtY

Thread join() methods

- **public final void** join() **throws** InterruptedException
- **public final synchronized void** join(**long** mls) **throws** InterruptedException,
  - where mls is in milliseconds
- **public final synchronized void** join(**long** mls, **int** nanos) **throws** InterruptedException,
  - where mls is in milliseconds.
  - where Nanos is in nanosecunds
- Join method throws InterruptedException (which is checked exception)

Thread join() example 1

- Waiting of main thread until completing child thread.

```
public static void main(String[] args) throws InterruptedException {
        MyThread myThread = new MyThread();
        myThread.start();
        myThread.join();
    }
```

Thread join() example 2

- Waiting of child Thread until completing main Thread.

```
public class MyThread extends Thread {
    private Thread parent;

    public MyThread(Thread parent) {
        this.parent = parent;
    }

    @Override
    public void run() {
        try {
            parent.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Thread join deadlock 1

```
main{
    ch.join()
}

child{
  m.join()
}
```

- Then both threads will wait forever and program will be passed are stock.
- This is something like deadlock.

Thread join deadlock 2

```
main{
    Thread.currentThread.join()
}
```

- If a thread calls join method on the same thread. Then the program will be stock. This is something like deadlock
- In this case thread has to wait infinite amount of time.

Thread sleep

- sleep – uxlash.
- The method sleep() is being used to halt/pause the working of a thread for a given amount of time.
- If a thread do not want any operation for a particular amount time then we should go for sleep() method.
- Methods:
  - **public static void** sleep(**long** mls) **throws** InterruptedException
  - **public static void** sleep(**long** mls, **int** n) **throws** InterruptedException
- Methods throws Interrupted Exception (checked exception)

Thread sleep example

```
public static void main(String[] args) {
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
```

Thread Interrupt

- Interrupt - xalaqit bermoq , to'xtatish

Thread Interrupt 1

- A Thread can interrupt a sleeping thread or waiting thread by using interrupt() method of Thread class.

Thread Interrupt example

```
public class MyThread extends Thread {
    @Override
    public void run() {
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            System.out.println("I got interrupted");
        }
    }
}

public class InterruptDemoMain {
    public static void main(String[] args) {
        MyThread m = new MyThread();
        m.start();
        m.interrupt();
        System.out.println("End of main");
    }
}
```

Thread Interrupt example description

- In this case after starting child thread it will start it job. After interrupt() child Thread it immediately throws InterruptedException and output “I got interrupted” and child continue it is job after catch.
- When ever we are calling a interrupt method If target thread not in sleeping state or waiting state. Than there is no impact of interrupt call immediately. Interrupt call will be waited until target thread entered into sleeping or waiting state.
- If target thread entered into waiting or sleeping state then immediately interrupt call will interrupt the target thread.

Daemon Thread

- Background Thread

Daemon Thread 1

- The Threads which are executing in the background are called daemon Threads.
- Example:
  - Garbage Collector
  - Signal Dispatcher
  - Attach listener
- The main objective of Demon Threads is to provide support for non demon Threads(main Thread).
- Example: If main Thread runs with law memory then JVM runs GC to destroy useles objects so that number of bytes of free memory will be improved.
- Orqa fonda ishlaydigan Thread larga Daemon Thread deyiladi.
- Daemon Thread larni ishlatishdan maqsad asosiy (Daemon bo'lmagan) Thread larga yordamchi sifatida ishlatish.
- Masalan: Agar main Thread ishlash jarayonida hotiga yerishmasa JVM kerakmas ob'ektlarni o'chirish uchun GC ni ishga tushiradi.

Daemon Thread 2

- Usually Daemon Threads runs with law priority but JVM can change to high priority.
- We can change daemon nature of thread before starting Thread. Else we got IllegalThreadStateException

- Asosan Daemon Thread lar kam/past priority bilan ishlaydi, Ammo JVM uning prioritisini o’zgartirishi mumkin.
- Biz Thread ni daemon xususiyatini Thead ni ishga tushurmasdan oldin o’zgartirishimiz mumkin. Aks holsa hatolik bo’ladi. Mazgi.

```
public boolean isDeamon()

public void setDeamon(boolean b)
```

Daemon Thread 3

- By default main Thread is always non demon.
- Thread Daemon nature is inherited from parent to child.
- It is impossible to change Demon nature of main thread because it is already started by JVM at beginning.
- When ever last non demon Thread terminates automatically all daemon threads will be terminated.

- Default holatda main Thread hardoim non daemon.
- Daemon hususiyati Parent dan nasil olinadi.
- Thread ishga tushganidan keyin uni Daemon hususiyatini o’zgartirib bo’lmaydi.
- Oxirgi daemon bo’lmagan Thread o’lganidan keyin barcha daemon thread lar to’xtatiladi.

Daemon Thread Example

```
public static void main(String[] args) {
        MyThread myThread = new MyThread();
        Thread t = new Thread(myThread);
        t.setDaemon(true);
        t.start();
        System.out.println("Main Thread completed");
    }
```

Thread Stop

- Thread ni to’xtatish

Thread Stop

- We can stop a thread executing by using stop method of thread class.
- After we call stop() method then the immediately Thread will enter into dead state.
- Any where stop method is deprecated and not recommended to use.

- Biz thread ni Thread classini stop() metodi orqali to’xtatishimiz mumkin.
- stop() methodini chaqirgandan keyin Thread darhol Dead state ga o’tadi.
- Nima bo’lsa ham stop() method man qilingan va ishlatishga tavsiya qilinmaydi.

How to suspend and resume of a Thread

- Suspend - To'xtatib turish
- Resume - Qayta boshlash

Thread suspend and resume methods

- We can suspend a Thread by using suspend() method of Thread class, then immediately the Thread will be enter into suspended sate.
- We can resume a suspended thread by using resume method of thread class. Then suspended thread can continue it is executing.
- These 2 methods deprecated and not recommended to use.
- Thead ni suspend() metodi orqali to’xtatib turish mumkin. Bunda Thread suspended (to’xtatilgan) state ga tushib qoladi.
- Suspended (to’xtatilgan) Threadni resume() metodi orqali qayta ishga tushurishimiz mumkin. resume() metodi chaqirilgandan keyin Thread o’z ishini davom ettirishimi mumkin.
- Bu 2kala method man etilgan va ishlatishga tavsiya qilinmaydi.