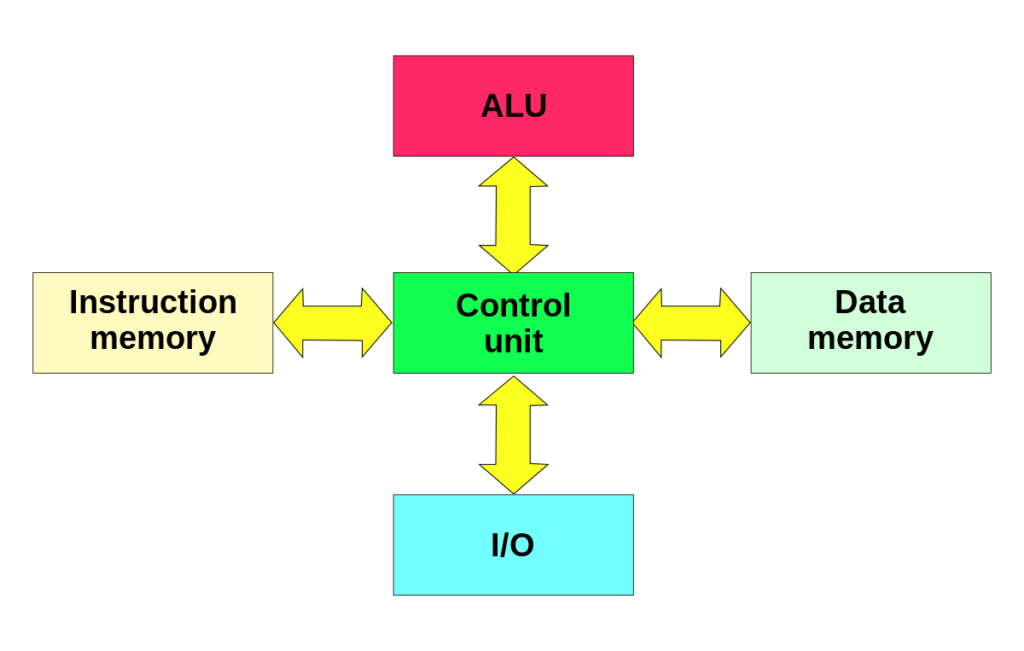
**Von-Neumann architecture(Von-Neumann Mimarisi)**

In a Von-Neumann architecture, the same memory and bus are used to store both data and instructions that run the program.

// Von-Neumann mimarisinde aynı bellek ve veri yolu, programı çalıştıran verileri ve talimatları depolamak için kullanılır.

Since you cannot access program memory and data memory simultaneously, the Von Neumann architecture is susceptible to bottlenecks and system performance is affected.

// Program hafızasına ve veri hafızasına aynı anda erişemediğiniz için, Von Neumann mimarisi tıkanıklıklara karşı hassastır ve sistem performansı etkilenir.

[](https://rh6stzxdcl1wf9gj1fkj14uc-wpengine.netdna-ssl.com/wp-content/uploads/2018/03/Harvard_architecture.svg_.png)

**Figure 2: The Harvard architecture has a separate bus for signals and storage. (Image: Wikimedia Commons)**

**//** **Şekil 2: Harvard mimarisinde sinyaller ve depolama için ayrı bir veri yolu bulunur. (Resim: Wikimedia Commons)**

**Harvard Architecture(Harvard Mimarisi)**

The Harvard architecture stores machine instructions and data in separate memory units that are connected by different busses.

// Harvard mimarisi, makine talimatlarını ve verilerini farklı veri yollarıyla birbirine bağlı ayrı bellek ünitelerinde saklar.

In this case, there are at least two memory address spaces to work with, so there is a memory register for machine instructions and another memory register for data.

// Bu durumda, çalışmak için en az iki bellek adres alanı vardır, bu nedenle makine talimatları için bir hafıza kaydı ve veriler için başka bir hafıza kaydı vardır.

Computers designed with the Harvard architecture are able to run a program and access data independently, and therefore simultaneously.

// Harvard mimarisiyle tasarlanan bilgisayarlar, bir programı çalıştırabilir ve bağımsız olarak ve dolayısıyla aynı anda verilere erişebilir.

Harvard architecture has a strict separation between data and code. Thus, Harvard architecture is more complicated but separate pipelines remove the bottleneck that Von Neumann creates.

// Harvard mimarisi veri ve kod arasında kesin bir ayrım yapar. Bu nedenle, Harvard mimarisi daha karmaşık ancak ayrı borular, Von Neumann'ın yarattığı tıkanıklıgı ortadan kaldırıyor.

**Modified Harvard Architecture(Modifiye Harvard Mimarisi)**

The majority of modern computers have no physical separation between the memory spaces used by both data and programs/code/machine instructions, and therefore could be described technically as Von Neumann for this reason.

// Modern bilgisayarların çoğunda, hem veri hem de programlar / kod / makine talimatları tarafından kullanılan bellek alanları arasında fiziksel bir ayrım yoktur ve bu nedenle teknik olarak Von Neumann olarak tanımlanabilir.

However, the better way to represent the majority of modern computers is a “modified Harvard architecture.”

// Bununla birlikte, modern bilgisayarların çoğunu temsil etmenin daha iyi yolu “değiştirilmiş bir Harvard mimarisi” dir.

Modern processors might share memory but have mechanisms like special instructions that keep data from being mistaken for code.

// Modern işlemciler hafızayı paylaşabilir, ancak verilerin kodla karıştırılmasını engelleyen özel talimatlar gibi mekanizmalara sahip olabilir.

Some call this “modified Harvard architecture.”

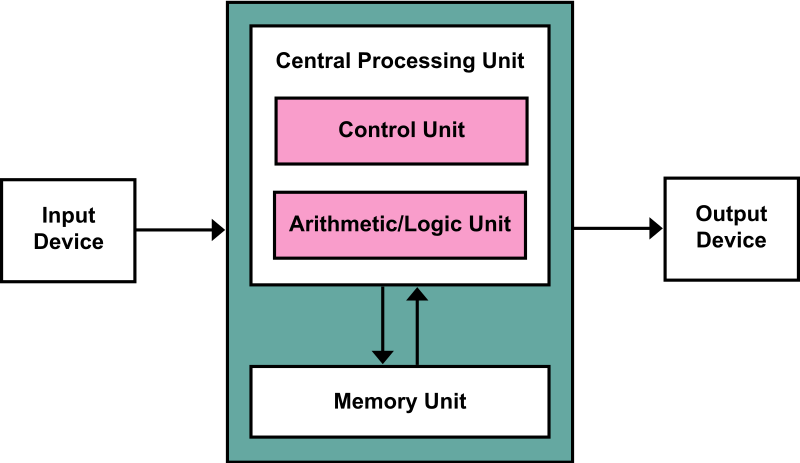
// Bazıları buna “değiştirilmiş Harvard mimarisi” diyor.

However, modified Harvard architecture does have two separate *pathways (busses)* for signal (code) and storage (memory), while the memory itself is one shared, physical piece.

// Bununla birlikte, değiştirilmiş Harvard mimarisinde sinyal (kod) ve depolama (bellek) için iki ayrı yol (veri yolu) bulunurken, belleğin kendisi paylaşılan bir fiziksel parçadır.

The memory controller is where the modification is seated, since it handles the memory and how it is used.

// Hafıza kontrolü, modifikasyonun yapıldığı yerdir, çünkü hafızayı ve nasıl kullanacağını tutar.

[](https://rh6stzxdcl1wf9gj1fkj14uc-wpengine.netdna-ssl.com/wp-content/uploads/2018/03/800px-Von_Neumann_Architecture.png)

**Figure 1: The Von Neumann architecture has been around since the 1940s. A clarifying trait is that a single bus used for both signal and storage. (Image: Wikimedia Commons)**

**//** **Şekil 1: Von Neumann mimarisi 1940'lardan bu yana kullanılıyor. Netleştirici bir özellik, hem sinyal hem de depolama için kullanılan tek bir veri yoludur. (Resim: Wikimedia Commons)**

**The Von Neumann Bottleneck(Van Neumann Tıkanıklıgı)**

If a Von Neumann machine wants to perform an operation on some data in memory, it has to move the data across the bus into the CPU.

// Bir Von Neumann makinesi hafızada bulunan bazı veriler üzerinde işlem yapmak istiyorsa, veriyolu üzerinden veri yolunu CPU'ya taşımak zorundadır.

When the computation is done, it needs to move outputs of the computation to memory across the same bus.

//Hesaplama yapıldığında, hesaplamanın çıktılarını aynı veri yolu üzerinden belleğe taşıması gerekir.

The amount of data the bus can transfer at one time (speed and bandwidth) plays a large part in how fast the Von Neumann architecture can be.

// Veriyolunun bir seferde transfer edebildiği veri miktarı (hız ve bant genişliği), Von Neumann mimarisinin ne kadar hızlı olabileceği konusunda büyük bir rol oynuyor.

The throughput of a computer is related to how false the processors are as well as the rate of data transfer across the bus.

// Bir bilgisayarın çıkışı, işlemcilerin ne kadar yanlış olduğu ve aynı zamanda veriyolu üzerinden veri aktarımı oranı ile ilgilidir.

The processor can be idle while waiting for a memory fetch, or it can perform something called speculative processing, based on what the processor might next need to do after the current computation is finished (once data is fetched and computations are performed).

// İşlemci, bir bellek getirisini beklerken boşta kalabilir ya da işlemcinin, mevcut işlem tamamlandıktan sonra işlemcinin yapması gerekenlere dayanarak spekülatif işlem adı verilen bir şey yapabilir (veriler alındıktan ve hesaplamalar yapıldıktan sonra).

The Von Neumann bottleneck occurs when data taken in or out of memory must wait while the current memory operation is completed.

// Von Neumann tıkanıklığı, mevcut hafıza işlemi tamamlandığında hafızaya giren veya çıkan veriler beklemek zorunda olduğunda meydana gelir.

That is, if the processor just completed a computation and is ready to perform the next, it has to write the finished computation into memory (which occupies the bus) before it can fetch new data out of memory (which also uses the bus).

// Diğer bir deyişle, işlemci yeni bir hesaplama tamamladıysa ve bir sonraki işlemi gerçekleştirmeye hazırsa, yeni veriyi bellekten (veriyolunu kullanan) çekebilmesi için bitmiş hesaplamayı belleğe (veri yolu kullanan) yazmak zorundadır.

The Von Neumann bottleneck has increased over time because processors have improved in speed while memory has not progressed as fast.

// Von Neumann darboğazı zaman içinde artmıştır, çünkü hafızalar hızlı bir şekilde ilerlememişken işlemciler hızda iyileşmiştir.

Some techniques to reduce the impact of the bottleneck are to keep memory in cache to minimize data movement, hardware acceleration, and speculative execution.

// Tıkanıklığın etkisini azaltmak için bazı teknikler, veri hareketini, donanım ivmesini ve spekülatif yürütmeyi en aza indirmek için belleği önbellekte tutmaktır.

It is interesting to note that speculative execution is the conduit for one of the latest security flaws discovered by Google Project Zero, named Spectre.

// Spekülatif uygulamanın, Spectre adlı Google Project Zero tarafından keşfedilen en yeni güvenlik hatalarından birinin borusu olduğunu belirtmek ilginçtir.