**Komut Döngüsü ve Pipelining nedir (mimarileri ve performansa etkisi)**

**2212721048 – Melike Çakmakoğlu**

## 1. GİRİŞ

Bu rapor, bilgisayar mimarisi ve organizasyonu dersi kapsamında işlemcilerin temel çalışma prensiplerinden biri olan komut döngüsü ve bu döngünün verimliliğini artıran pipelining (boruhattı) tekniğini incelemek amacıyla hazırlanmıştır. İşlemcilerin komutları nasıl işlediği ve bu süreci hızlandırmak için geliştirilen yöntemler, çalışmanın temelini oluşturmaktadır. Bu bağlamda, modern işlemcilerin pipelining mimarisini neden tercih ettiği, bu mimarinin performansa katkıları ve farklı mimari yaklaşımların sunduğu avantajlar ile karşılaştığı zorluklar ele alınmıştır. Ayrıca, literatürdeki akademik çalışmalardan örneklerle desteklenen bu inceleme, komut döngüsünün temel aşamalarını, pipelining'in çalışma mantığını ve farklı pipelining mimarilerinin işlemci performansına etkilerini derinlemesine anlamayı hedeflemektedir. Böylece, güncel yaklaşımların ve karşılaştırmaların değerlendirilmesiyle konuya geniş bir bakış açısı kazandırılması amaçlanmıştır.

## 2. KOMUT DÖNGÜSÜ NEDİR?

Komut döngüsü, işlemcinin (CPU) bellekteki komutları okuyarak yürütme sürecini ifade eder ve bilgisayarın temel işlevselliğini oluşturan bu mekanizma, her komutun belirli aşamalardan geçmesini sağlar. Bilgisayar mimarisinin temel taşlarından biri olan bu döngü, modern işlemcilerde genellikle beş ana aşamadan oluşur: getirme (fetch), çözme (decode), adres okuma, yürütme (execute) ve sonuç yazma (write-back). Bu aşamalar, işlemcinin komutları sistematik bir şekilde işleyerek verimli bir çalışma düzeni sağlamasına olanak tanır.

### 2.1 Komut Döngüsünün Aşamaları

Komut döngüsünün aşamaları, işlemcinin komutları sistematik bir şekilde işlemesini sağlayan temel adımları kapsar. İlk olarak, getirme (fetch) aşamasında, Program Sayacı (PC) bir sonraki komutun bellek adresini tutar ve bu komut, Bellek Adres Yazmacı (MAR) ile Bellek Veri Yazmacı (MBR) kullanılarak alınır, ardından Komut Yazmacına (IR) yüklenir. İkinci aşama olan çözme (decode) sürecinde, Komut Yazmacındaki komut, işlemcinin çözücü birimi tarafından analiz edilir; komutun hangi işlemi (örneğin, toplama) gerçekleştireceği ve hangi verilerin kullanılacağı belirlenir. Üçüncü aşamada, adres okuma süreci devreye girer; eğer komut belleğe erişim gerektiriyorsa (örneğin, dolaylı adresleme), gerekli verinin adresi hesaplanır. Yürütme (execute) aşamasında ise Aritmetik Mantık Birimi (ALU), komutun belirttiği işlemi, örneğin toplama veya mantıksal bir işlemi, gerçekleştirir. Son olarak, sonuç yazma (write-back) aşamasında, işlem sonucu komutun türüne bağlı olarak bir yazmaca veya belleğe kaydedilir. Bu aşamalar, işlemcinin komutları verimli ve düzenli bir şekilde işlemesini sağlar.

### 2.2 Örnek: ADD R, X Komutu

ADD R, X komutu, X adresindeki veriyi R yazmacındaki değere ekleyen bir işlemi ifade eder. Bu komutun yürütülmesi, komut döngüsünün aşamaları doğrultusunda gerçekleşir. İlk olarak, getirme aşamasında komut, Program Sayacı (PC) tarafından belirtilen bellek adresinden alınır ve Komut Yazmacına (IR) yüklenir. Çözme aşamasında, komut analiz edilerek X adresindeki verinin getirilip R yazmacındaki değere ekleneceği anlaşılır. Adres okuma aşamasında, X'in bellek adresi belirlenir.

Yürütme aşamasında, Aritmetik Mantık Birimi (ALU), R yazmacındaki değer ile X adresindeki veriyi toplayarak işlemi gerçekleştirir. Son olarak, sonuç yazma aşamasında, elde edilen sonuç R yazmacına kaydedilir. Bu süreç, komut döngüsünün pratikte nasıl işlediğini örneklemektedir.

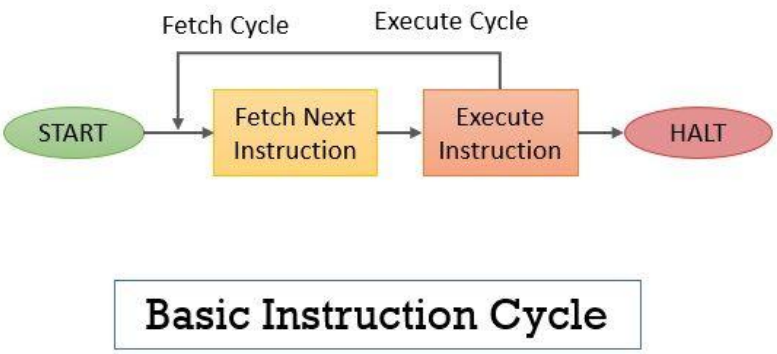
### 2.3 Komut Döngüsünün Önemi ve Sınırlamaları

Komut döngüsü, işlemcinin (CPU) komutları doğru ve düzenli bir şekilde yürütmesini sağlayan temel bir mekanizmadır. Bu döngü, bilgisayarın sistematik çalışmasının temelini oluşturur. Ancak, sıralı yürütme modeli, her komutun tamamlanmasını beklediği için işlem hızını sınırlamaktadır. Bu durum, işlemcinin aynı anda birden fazla komutla çalışmasını engelleyerek performansı olumsuz etkiler. Modern işlemci mimarilerinde bu sınırlamayı aşmak amacıyla pipelining teknikleri kullanılarak komut döngüsü paralel hale getirilmiş ve böylece işlem verimliliği artırılmıştır.

### 2.4 Komut Döngüsü için Literatür İncelemesi

Komut döngüsü, işlemcinin bellekteki komutları sırayla okuyarak işlemesini sağlayan temel bir süreçtir ve bilgisayar mimarisinin en önemli unsurlarından biri olarak kabul edilir. Literatürde, bu döngünün aşamaları ve mikro işlemleri ayrıntılı şekilde incelenmiştir. Singh, komut döngüsünü beş temel aşamaya ayırır: getirme (fetch), çözme (decode), adres okuma, yürütme ve sonuç yazma (write-back) [1]. Getirme aşamasında, Program Sayacı (PC) çalıştırılacak komutun adresini belirtir; komut, Bellek Adres Yazmacı (MAR) ve Bellek Veri Yazmacı (MBR) aracılığıyla Komut Yazmacına (IR) yüklenir. Çözme aşamasında ise komut analiz edilerek işlemcinin gerçekleştireceği işlem belirlenir. Singh, ADD R, X komutunu örnekleyerek mikro işlemleri şu şekilde açıklar: ilk olarak MAR, komutun adresini alır; ardından MBR, bellekten veriyi getirir ve son olarak R yazmacındaki değer ile MBR'deki veri toplanır [1]. Bu örnek, komutun işlemci içinde izlediği adımları net bir şekilde ortaya koyar.

Flynn ve Hung ise komut döngüsünü daha geniş bir perspektiften ele alır ve süreci getirme (IF), çözme (ID), adres oluşturma (AG), veri getirme (DF), yürütme (EX) ve sonuç yazma (WB) aşamalarına böler [2]. Ayrıca, klasik komut döngüsünün sınırlı verimlilik sunduğunu vurgulayarak, komutların sıralı işlenmesinin işlemci kaynaklarının tam olarak kullanılmasını engellediğini belirtirler. Bu nedenle, işlemcilerin daha verimli çalışabilmesi için pipelining gibi paralel yürütme tekniklerine ihtiyaç duyulduğuna dikkat çekerler [2].




**Şekil 1:** Temel Komut Döngüsü Şeması [6]

Şekil 1, işlemcinin bir komutu nasıl aldığı ve çalıştırdığı sürecini adım adım görselleştirir. Bu döngü, işlemci çalıştırıldığında (START) başlar ve işlemci kapanana (HALT) kadar sürekli devam eder. İlk olarak, getirme döngüsü (fetch cycle) devreye girer; bu aşamada Program Sayacı (PC), çalıştırılacak komutun bellek adresini belirtir. Bu adres, Bellek Adres Yazmacına (MAR) gönderilir ve komut, bellekten alınarak Bellek Veri Yazmacı (MBR) aracılığıyla Komut Yazmacına (IR) yüklenir. Ardından,

yürütme döngüsü (execute cycle) başlar. Bu süreçte komut çözülerek ne yapılacağı belirlenir, Aritmetik Mantık Birimi (ALU) tarafından ilgili işlem (örneğin, toplama) gerçekleştirilir ve sonuç bir yazmaca veya belleğe kaydedilir. Şekil 1, temel komut işleme yapısını sade bir şekilde açıklamakla birlikte, modern işlemcilerde bu yapının daha karmaşık bir hale geldiğini de yansıtır. Bu görselleştirme, komut döngüsünün mantığını anlamayı kolaylaştırır.

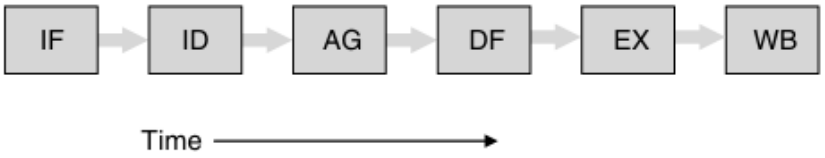



**Figure 4.** Instruction execution sequence.

**Şekil 2:** Temel Komut Döngüsü Şeması [2]

Şekil 2, modern işlemcilerde bir komutun nasıl işlendiğini adım adım görselleştirir ve işlemcinin karmaşık işlemleri hızlı ve doğru bir şekilde gerçekleştirmesini sağlayan yapıyı ortaya koyar. Komut yürütme süreci altı aşamaya ayrılmıştır. İlk olarak, komut getirme (Instruction Fetch - IF) aşamasında, komut bellekten alınarak Komut Yazmacına (IR) yüklenir. Ardından, komut çözme (Instruction Decode - ID) aşamasında, komut analiz edilerek gerçekleştirilecek işlem belirlenir. Adres oluşturma (Address Generation - AG) aşamasında, komutun veriyle çalışması durumunda, ilgili verinin bellek adresi hesaplanır. Veri getirme (Data Fetch - DF) aşamasında, hesaplanan adresteki veri bellekten alınır. Yürütme (Execute - EX) aşamasında, Aritmetik Mantık Birimi (ALU) tarafından toplama veya mantıksal işlem gibi belirtilen işlem gerçekleştirilir. Son olarak, sonuç yazma (Write Back - WB) aşamasında, işlem sonucu yazmaçlara veya belleğe kaydedilir. Flynn ve Hung'un belirttiği üzere, bu adımlar geleneksel olarak sıralı bir düzende ilerler; ancak modern işlemcilerde pipelining teknikleri kullanılarak işlemler eşzamanlı hale getirilir [2]. Bu yaklaşım, bir komut yürütülürken diğer komutların hazırlanmasını sağlayarak işlemci hızını önemli ölçüde artırır.

## 3. PIPELINE MİMARİSİ

### 3.1.1 Pipeline'ın Tanımı ve Temel Amacı

Pipeline, işlemci performansını artırmak için geliştirilmiş bir tekniktir. Geleneksel komut döngüsünde, bir komut tamamen işlenmeden diğerine geçilemez; bu da işlem süresini uzatır. Pipeline tekniği ise işlemcinin aynı anda birden fazla komut üzerinde eşzamanlı olarak çalışmasını sağlar. Bu süreç, bir üretim bandına benzetilebilir: her aşama belirli bir görevi yerine getirir ve komutlar ardışık olarak bu aşamalardan geçer. Böylece, işlemci kaynakları daha verimli kullanılır, işlem süresi kısalır ve genel performans önemli ölçüde artar.

### 3.1.2 Pipeline Aşamaları ve İşleyişi

Pipeline, işlemcinin komutları eşzamanlı olarak işlemek için kullandığı bir yapı olup genellikle beş temel aşamadan oluşur. İlk olarak, komut getirme (Instruction Fetch - IF) aşamasında, komut bellekteki adresinden alınır. Ardından, komut çözme (Instruction Decode - ID) aşamasında, komutun neyi ifade ettiği analiz edilerek içeriği anlaşılır. Yürütme (Execute - EX) aşamasında, aritmetik veya mantıksal işlemler gerçekleştirilir. Bellek erişimi (Memory Access - MEM) aşamasında, gerektiğinde veriler

belleğe yazılır veya bellekten okunur. Son olarak, sonuç yazma (Write Back - WB) aşamasında, hesaplanan sonuç yazmaçlara veya belleğe kaydedilir. Bu aşamalar sıralı bir şekilde ilerler; ancak pipeline yapısı sayesinde, her aşama bir önceki komutla eşzamanlı olarak çalışabilir, böylece işlemci verimliliği artar.

### 3.1.3 Pipeline'ın Performansa Etkisi

Pipeline kullanımı, işlemcinin performansını artırır; çünkü birden fazla komut, farklı aşamalarda eşzamanlı olarak işlenir. Bu yaklaşım, her komut için gereken döngü sayısını azaltarak toplam işlem süresini önemli ölçüde kısaltır. Özellikle sıralı komutların yoğun olduğu programlarda, pipeline tekniğinin sağladığı verimlilik artışı belirgin bir şekilde ortaya çıkar.

### 3.1.4 Pipeline'ın Karşılaştığı Sorunlar

Pipeline yapısı bazı teknik sorunlarla karşılaşabilir:

Pipeline yapılarında karşılaşılan temel sorunlar arasında veri tehlikeleri, kontrol tehlikeleri ve yapısal tehlikeler yer alır. Veri tehlikeleri, bir komutun, henüz tamamlanmamış bir önceki komutun sonucuna ihtiyaç duyması durumunda ortaya çıkar. Kontrol tehlikeleri, dallanma (branch) gibi komutların yürütme akışını değiştirmesiyle boru hattında aksamalara yol açabilir. Yapısal tehlikeler ise aynı anda birden fazla aşamanın aynı donanım kaynağını kullanmak istemesiyle oluşan çatışmalardan kaynaklanır. Bu problemleri çözmek için dallanma tahmini (branch prediction), atlama (bypass) ve duraklatma (stall) gibi teknikler geliştirilmiştir; bu yöntemler, boru hattının verimliliğini artırarak kesintisiz bir yürütme sağlar.

## 3.2 Pipeline için Literatür Taraması

Literatürde pipeline teknolojisi üzerine yapılan çeşitli çalışmalar, bu tekniğin işlemci performansına katkılarını farklı açılardan ele almaktadır. Flynn ve Hung, pipeline'ın komutları eşzamanlı çalıştırarak işlemci hızını artırdığını belirtir; statik pipeline'da aşamalar sabitken, dinamik pipeline'da Type 1 ve Type 2 gibi farklı türler bulunur. Ayrıca, süperskalar işlemcilerin birden fazla komutu aynı anda işleyebildiğini, ancak derin pipeline'ların güç tüketimini artırabileceğini vurgularlar [2]. Stiles ve McFarland, NexGen 8 işlemcisindeki pipeline yapısını inceleyerek, bu işlemcinin tek çevrimde komut yürütmesi ve out-of-order çalışma özelliğiyle yüksek hız sunduğunu ifade eder. Tıkanıklıkları önlemek için abort cycles gibi yöntemlerin kullanıldığını belirtirler [3]. Instruction Formats dokümanında, pipeline'ın getirme ve çözme aşamalarının komut formatlarına bağlı olduğu, düzenli komut formatlarının pipeline verimliliğini artırdığı vurgulanır [4]. Kambe ve Saituji ise değişken uzunlukta vektör pipeline (VVP) tasarımı üzerinde durur; bellek erişimini hızlandırmak için veri buffer'ları ve HMM RAM ayrımı gibi yöntemler kullanmış, döngü optimizasyonuyla pipeline performansını iyileştirmiştir. Bu yaklaşımla, konuşma tanıma sisteminde 1.127 ms'ye varan bir hız elde edilmiştir [5].

| arch. | mem. separ. | buffer | unroll | area (gates) | time (ms) |
|---|---|---|---|---|---|
| soft* | - | - | - | | 55.300 |
| Seq. | no | no | no | 48,385 | 20.341 |
| pip.(A) | no | no | no | 171,968 | 5.826 |
| pip.(B) | yes | no | no | 170,813 | 5.396 |
| pip.(C) | no | yes | no | 294,579 | 4.906 |
| pip.(D) | yes | yes | no | 293,706 | 4.891 |
| pip.(E) | no | no | yes | 316,725 | 5.127 |
| pip.(F) | no | yes | yes | 547,231 | 4.345 |
| pip.(G) | yes | no | yes | 439,783 | 1.226 |
| pip.(H) | yes | yes | yes | 546,080 | 1.127 |

∗ : Julius software(CPU: Pentium ▪ 600[MHz])

**Şekil 3:** Farklı Pipeline Mimarilerinin Karşılaştırması [5]

Pipeline performansını artırmak amacıyla çeşitli yöntemler geliştirilmiş ve farklı mimariler karşılaştırılmıştır. Şekil 3, bu mimarilerin hız (ms) ve alan (gates) açısından performanslarını ortaya koyar [5]. Örneğin, (H) mimarisi, 1.127 ms ile en yüksek hıza ulaşırken, 546,080 gates ile en fazla alan kaplar. Buna karşılık, yazılım (soft) tabanlı yaklaşım 55.300 ms sürmüş, bu da pipeline tekniğinin sağladığı hız avantajını açıkça göstermektedir [5].

| No. | buffering | Gaussian | EL cal. |
|---|---|---|---|
| (A) | - | **5,731 | 290 |
| (B) | - | **5,333 | 228 |
| (C) | 3,391 | **4,835 | 214 |
| (D) | 1,070 | **4,820 | 228 |
| (E) | - | **4,851 | 228 |
| (F) | **3,357 | 1,344 | 290 |
| (G) | - | **1,135 | 228 |
| (H) | 1,070 | **1,115 | 228 |

∗∗ : the dominant stage.

**Şekil 4:** Pipeline Aşamalarının İşlem Süreleri (ns) [5]

Şekil 4, pipeline aşamalarının sürelerini nanosaniye (ns) cinsinden karşılaştırarak hangi aşamanın daha fazla zaman gerektirdiğini ortaya koyar [5]. Örneğin, (A) mimarisinde Gaussian aşaması 5,731 ns ile en uzun süreye sahipken, (H) mimarisinde bellek optimizasyonu sayesinde bu süre 1,115 ns'ye düşmüştür. Bu şekil, pipeline süreçlerinde zaman kullanımını analiz ederek optimizasyonun etkisini anlamayı kolaylaştırır [5].

## 4. SONUÇ

Bu raporda, bilgisayar mimarisinin temel unsurlarından biri olan komut döngüsü ile bu döngünün gelişmiş bir uygulaması olan pipelining (boru hattı) mimarisi ayrıntılı bir şekilde ele alınmıştır. Komut döngüsü, işlemcinin bir komutu bellekten alarak yürütmesine kadar geçen süreci tanımlar ve genellikle getirme, çözme ve yürütme gibi aşamalardan oluşur. Ancak, sıralı yürütme modeli performans açısından sınırlamalar getirir. Bu kısıtlamayı aşmak için modern işlemcilerde pipelining tekniği kullanılır; bu yöntem, birden fazla komutun eşzamanlı olarak farklı aşamalarda işlenmesini sağlayarak işlemci verimliliğini artırır. Bu çalışma, komut döngüsü ile pipelining mimarisinin yapısal farklarını ve performansa etkilerini karşılaştırmalı olarak sunarak, her iki yaklaşımın işlemci tasarımındaki önemini ortaya koymaktadır.

## 5. KAYNAKLAR

[1] A. Singh, “Instruction Cycle,” Univ. Lucknow, Comput. Sci. Eng. Dept., Lucknow, India, 2023.
[2] M. J. Flynn and P. Hung, “Computer Architecture,” in *Wiley Encyclopedia of Computer Science and Engineering*, B. Wah, Ed. John Wiley & Sons, Inc., 2007, pp. 1–18.
[3] D. R. Stiles and H. L. McFarland, “Pipeline Control for a Single Cycle VLSI Implementation of a Complex Instruction Set Computer,” in *Proc. IEEE Conf.*, 1989, pp. 504–508.
[4] “Instruction Formats on Computer Organization,” [Online]. Available: [PDF document].
[5] T. Kambe and M. Saituji, “A Variable Length Vector Pipeline Architecture Design Methodology,” in *11th EUROMICRO Conf. on Digital System Design Architectures, Methods and Tools*, 2008, pp. 665–668.
[6] “Instruction Cycle,” Binary Terms, [Online]. Available: https://binaryterms.com/instruction-cycle.html, Accessed: Apr. 30, 2025.