Computer Organization

“Mikroişlemci Performans”

Dr. Cahit Karakuş
Esenyurt Üniversitesi

Performans Artırma

- CPU’da temel eleman transistor. Atomik yapıda üretilmektedir. Üretim teknolojisi final yaptı.
- Clock hızı final yaptı. Clock hızını aşmak için quantum bilgisayar (ALU – Devasa büyük boyutlarda hesaplama, yapay zeka algoritmaları)

Klasik bilgisayarlarda performans artırmak için

- Otonom – Kendini kendini otomatik olarak test etme ve performans izleme; düzeltme (Bilgisayar, Araba)
- Yüksek performanslı konut seti oluşturma.
- Pipelining: Ardışık düzende komut işleme; çok sayıda komutun parçalara ayrılıp farklı komtuların her bir parçasının aynı anda işlenmesi

CPU’nun Performansını Artırıcı Çözümleri

Bir CPU'nun performansını artırmak için:

1) **Çoklu İşlemci (Multi Core):** Tek yongada çoklu işlemci kullanarak ve daha hızlı devreler ekleyerek donanım geliştirilir. Tek yonga birden fazla CPU (çekirdek - core) içerir. Aynı anda çalışırlar. İş bölümü yaparlar. Çoğu yeni bilgisayarın en az iki çekirdeği vardır.
2) **Çoklu Görev (Multi Tasking)**: Geliştirilmiş donanımda, aynı anda birden fazla komut işlenecek şekilde görev paylaşımı yapılır.
3) **İşlemci Hızı (Clock Hızı):** Transistörlerin saniyede milyonlarca ve hatta milyarlarca kez açılıp kapanması dolayısıyla, makine çevrimi tekrarları baş döndürücü bir hızla gerçekleşir. İşlemci Mimarisinde Gigahertz - GHz mertebelerinde yüksek clock hızı kullanılması. Örneğin Clock frekansı, 2.4GHZ ise, Peryod=1/frekans=1/(2*10^9)=5*10^(-10) saniye=0.5 Nanosaniye.
4) **Ön Bellek (SRAM):** Mikroişlemcinin sonraki adımlarda işleyeceği verilerin önceden transfer edilip hazırlandığı kendi ön belleğidir (Cache Bellek). Bilgisayar sisteminde önbellek veri depolayan bir donanım veya yazılım bileşenidir, böylece gelecekteki veri talepleri daha hızlı bir şekilde yerine getirilebilir; Bir önbellekte depolanan veriler daha önceki bir hesaplamanın sonucu veya başka bir yerde depolanan verilerin bir kopyası olabilir. Mikroişlemcinin daha yavaş bir veri deposundan veriyi okumadan daha hızlı olan önbellekten okuyarak işler; bu nedenle, önbellekten ne kadar fazla istek yapılabiliyorsa, sistem o kadar hızlı çalışır.
5) **Ram Kapasitesi:** Günümüzde ana bilgisayarların ve süper bilgisayarların bellekleri GB ve hatta TB seviyesindedir.
6) **İşlemci Kapasitesi (Data yolu hat sayısı):** CPU iç saklayıcılarında(register) saklanabilen ve bir defada işlenebilen ve merkezi işlem birimi, bellek ve saklayıcıları birbirine bağlayan iç (yerel) sistem bus’daki veri yolu üzerinden bir defada gönderilebilen bit sayısıdır. 32 bit bir bilgisayar bir defada 32/8=4 byte işleyebilir. Bu da, 32 bit bilgisayarın 8 bit bilgisayardan yaklaşık 4 kat daha hızlı olması anlamına gelir.
7) **Komut İşleme Hızları:** Günümüzde milyonlar düzeyinde olan, saniyede işlenen komut sayısına göre de ölçülebilir. MIPS (Millions of Instructions Per Second – saniyedeki milyon komut sayısı) bilgisayar işleme hız ölçüsüdür. Günümüz ana bilgisayarlar 10000 ve üstünde MIPs hızlarında çalışmaktadır.

CPU’nun Genel Performansı Artırma İşlevleri

- **Ardışık düzende komut işleme (Döngüsel işlevlerin eş zamanlı yürütülmesi - Pipelining):** İşlemci verileri veya komutları kavramsal eş zamanlı olarak düzenlenmesi ve döngüsel işlevlerin yürütülmesidir. CPU'nun donanım öğelerinin genel performansı artırılacak şekilde düzenlenmesi sürecidir. Komutlar dizisinin eşzamanlı düzenlenmesi ve ardışık yürütülmesidir. İşlevsel döngülerin eşzamanlı yürütüldüğü bir işlemci mimari yapısıdır.
- **Dallanma tahmini (Branch Prediction):** Komut bellekten (ROM) bir sonraki döngüde işlenecek komutun dallanma durumlarına göre nasıl işleneceğini öngörülmesidir. Sonraki komutun işlevsel döngüsünün öngörülmesidir.
- **Veri akışı analizi (Data flow analysis):** İşlemcinin, optimize (eniyileme) edilmiş komutlar çizelgesini oluşturmak için hangi komutların birbirlerinin sonuçlarına bağlı olduğunu analiz etmesidir.
- **Spekülatif komutların işlevsel yürütülmesi (Speculative execution):** İhtiyaç duyulmadan ya da gerekli olmayan bazı döngüsel işlevleri gerçekleştirdiği bir optimizasyon (eniyileme) tekniğidir. Dallanma ve veri akışı analizine bakılarak, ihtiyaç duyulup duyulmadığı bilinmeden bazı döngüsel işlevlerin önce yapılmasıdır, böylece işin gerekli olduğu bilindikten sonra performansı artırmaya katkı verilmesi.

Tek veri yolu mimarisine kıyasla çok veri yolu mimarisi kullanmanın faydası nedir?

- Tek veri yolu mimarisi ile karşılaştırıldığında, çoklu veri yolu mimarisinin kullanılması hız açısından büyük bir avantaja sahiptir ve elbette performansı da etkileyecektir.
- Tek veri yolu mimarisi kullanmak yerine çoklu veri yolu mimarisi kullanmak daha uygundur. (Von Nuaman mimarisi yerine Harvard Mimarisi kullanma)
- Çoklu veri yolu mimarisinin kullanılması, her cihazın kendi veri yoluna bağlanmasını sağlar, bu da her cihazın kendi veri yoluna sahip olacağı anlamına gelir. Bu şekilde, her bir cihazın veri aktarımı daha hızlı olacaktır, böylece veri aktarımı, birçok cihazın tek bir veri yoluna bağlı olduğu tek veri yolu mimarisinde olduğu gibi, sonunda cihazın kapasitesine ulaşacak şekilde takılıp kalmak zorunda kalmaz. bus ve böylece verileri "kuyruk" yapacaktır.
- Tabii ki, birden fazla veri yoluna sahip olmak daha pahalıya mal olacak, ancak maliyet, bu tek veri yolu mimarisine kıyasla daha yüksek hız ihtiyacını karşılamayacaktır.

Ön Bellek – Cache Bellek

- Ön Belleğe Yazma /Okuma: Verilerin ana belleğe veya bir I/O modülüne yazılmasından önce, verimliliği artırmak amacıyla veriler ön bellekte geçici olarak saklar.
- Ön Bellek (SRAM): Mikroişlemcinin sonraki adımlarda işleyeceği verilerin önceden transfer edilip hazırlandığı kendi ön belleğidir (Cache Bellek). Bilgisayar sisteminde önbellek veri depolayan bir donanım veya yazılım bileşenidir, böylece gelecekteki veri talepleri daha hızlı bir şekilde yerine getirilebilir; Bir önbellekte depolanan veriler daha önceki bir hesaplamanın sonucu veya başka bir yerde depolanan verilerin bir kopyası olabilir. Mikroişlemcinin daha yavaş bir veri deposundan veriyi okumadan daha hızlı olan önbellekten okuyarak işler; bu nedenle, önbellekten ne kadar fazla istek yapılabiliyorsa, sistem o kadar hızlı çalışır.

Multitasking - Multi processing

- **Çoklu görev (multiasking):** Kullanıcılar sistemde aynı anda birden fazla işlem çalıştırabilirler. Örneğin; Aynı anda hem yazı yazarken hemde müzik dinleyebilmek.

- **Görev paylaşımı:** işlemci tek, aynı anda çalıştırılacak donanım ve yazılım fazla. Bu işlemlerin yerine getirilebilmesi için donanımlarında kendi işlemcilerine sahip olması gerekiyor. Örneğin; grafik kartı, TV kartı, sayısal işaret işlemi, sayısal görüntü işlemi vb.

- **Çoklu işlemci (multi processing):** çok büyük veri işlenmesi durumunda veya çok hızlı veri transferinde tek bir işlemci yetersiz kaldığında birden fazla mikro işlemci kullanılır. Örneğin; iş istasyonları

- **Çok Parçalı Kodlar(multiheading):** program ihtiyaç halinde işletim sistemi tarafından küçük parçalara ayrılır ve çalıştırılır. Örneğin; video transferi. Kablosuz iletişim sistemlerinde 3G diye adlandırılan teknolojide mobil kullanıcıya yüksek veri hizmeti vermek için aynı anda noktadan noktaya yüksek hızda veri göndermek yerine farklı noktalardan aynı noktaya veriyi parçalara ayırıp göndermek maliyet açısından daha uygundur.

Multitasking

- Bilgi işlemde çoklu görev, belirli bir süre boyunca birden fazla görevin (süreçler olarak da bilinir) eşzamanlı olarak yürütülmesidir.
- Yeni görevler, bitmesini beklemek yerine, halihazırda başlamış olanları bitmeden kesintiye uğratabilir.
- Sonuç olarak, bir bilgisayar birden fazla görevin bölümlerini aralıklı olarak yürütürken, görevler merkezi işlem birimleri (CPU'lar) ve ana bellek gibi ortak işlem kaynaklarını paylaşır.
- Çoklu görev, çalışan programı otomatik olarak kesintiye uğratır, durumunu kaydeder (kısmi sonuçlar, bellek içerikleri ve bilgisayar kayıt içerikleri) ve başka bir programın kayıtlı durumunu yükler ve kontrolü ona aktarır.
- Bu "bağlam anahtarı" sabit zaman aralıklarında başlatılabilir (önleyici çoklu görev) veya çalışan program, kesintiye uğradığında denetleyici yazılıma sinyal gönderecek şekilde kodlanabilir (işbirlikçi çoklu görev).
- Çoklu görev, birden fazla görevin tam olarak aynı anda paralel olarak yürütülmesini gerektirmez; bunun yerine, belirli bir süre içinde birden fazla görevin ilerlemesine olanak tanır.
- Çok işlemcili bilgisayarlarda bile çoklu görev, CPU'lardan çok daha fazla görevin çalıştırılmasına olanak tanır.

Multi processing

- Çoklu işleme, iki veya daha fazla işlemcinin bir araya getirilerek, işlenmesi gereken buyruğun daha hızlı bir şekilde işlenmesini sağlamaya yönelik bir tasarımdır.
- Çoklu işlemcilerin tek başına bir işlemciden daha hızlı olması beklenir. İşlemci tasarımında oluşan zorluklar çoklu işlemcileri zorunlu kılmıştır.

CPU’da İşlevsel Adımlar

Komutları Yorumlama İşlevleri

Hangi işlevsel döngünün gerekli olduğunu belirlemek için program komutlarının yorumlanması ve çözülmesi.

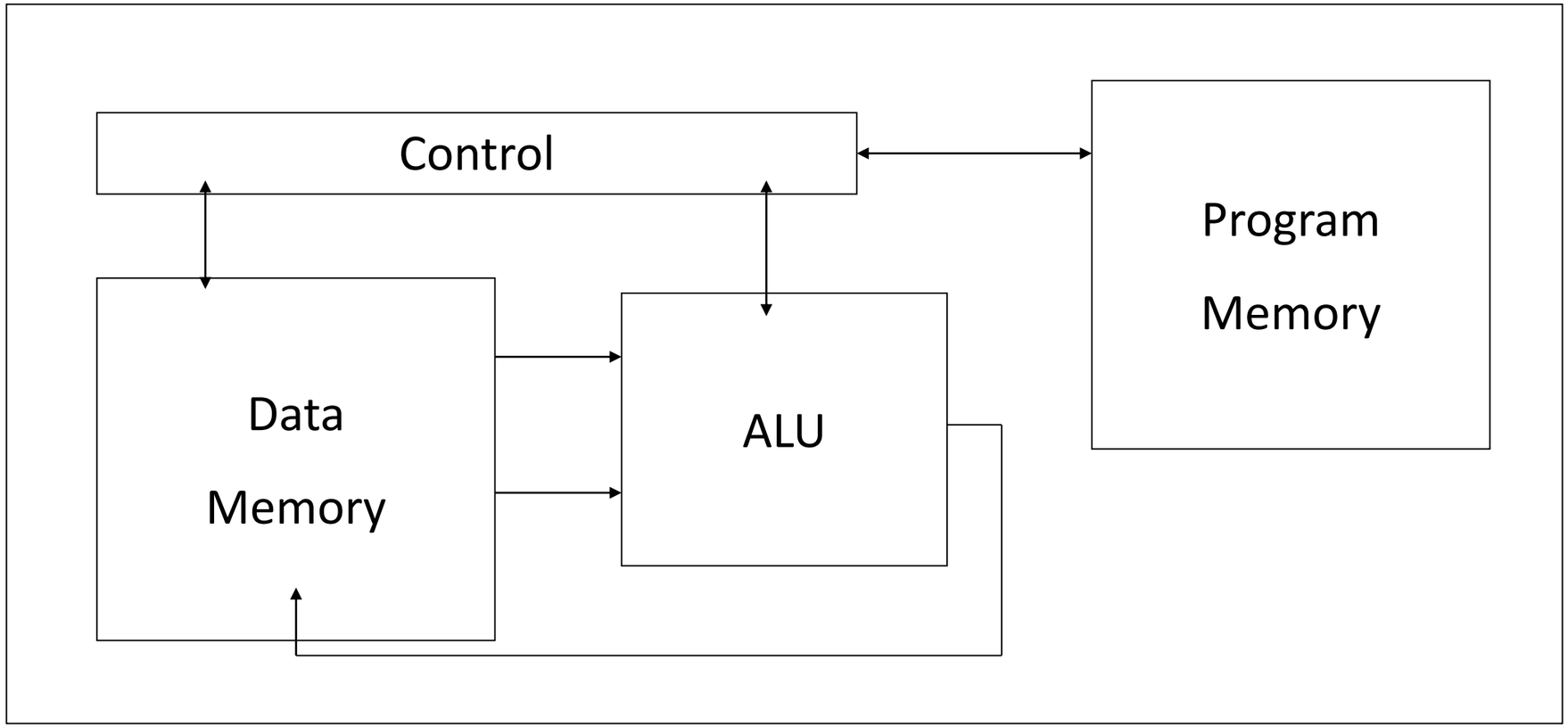
- Bellek ya da I/O adres erişim
- Bellek ya da I/O birimi seçilir.
- Bellek gözü seçilir.
- Veri yazılır ya da okunur
- Aritmetik ve Lojiksel işlemler
- Data Transfer
- Karşılaştırma
- Dizi
- Alt Programlara dallanma
- **Kesme**

- Data transfer instructions
- Addressing modes
- Data processing (arithmetic and logic)
- Program flow instructions

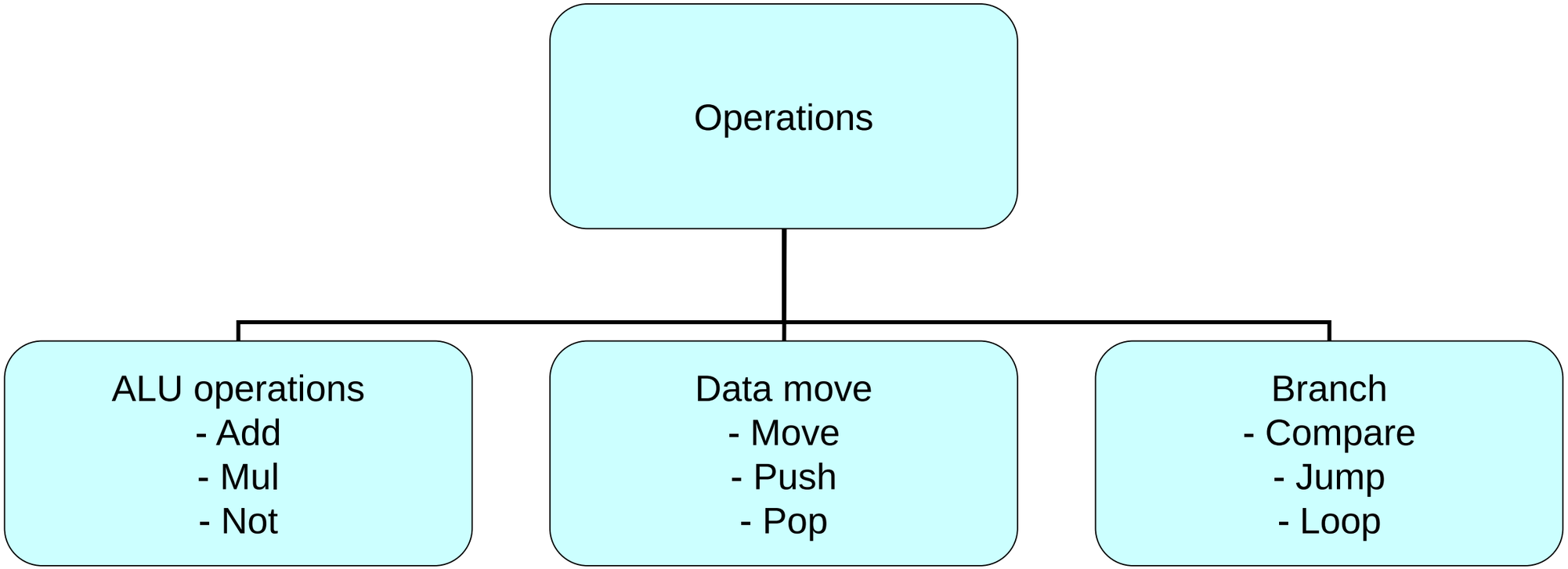
Komut:

- Movement Instructions
- Arithmetic and Logic Instructions
- Shift and Rotate Instructions
- Flow of Control

Programmable state machine



Instructions

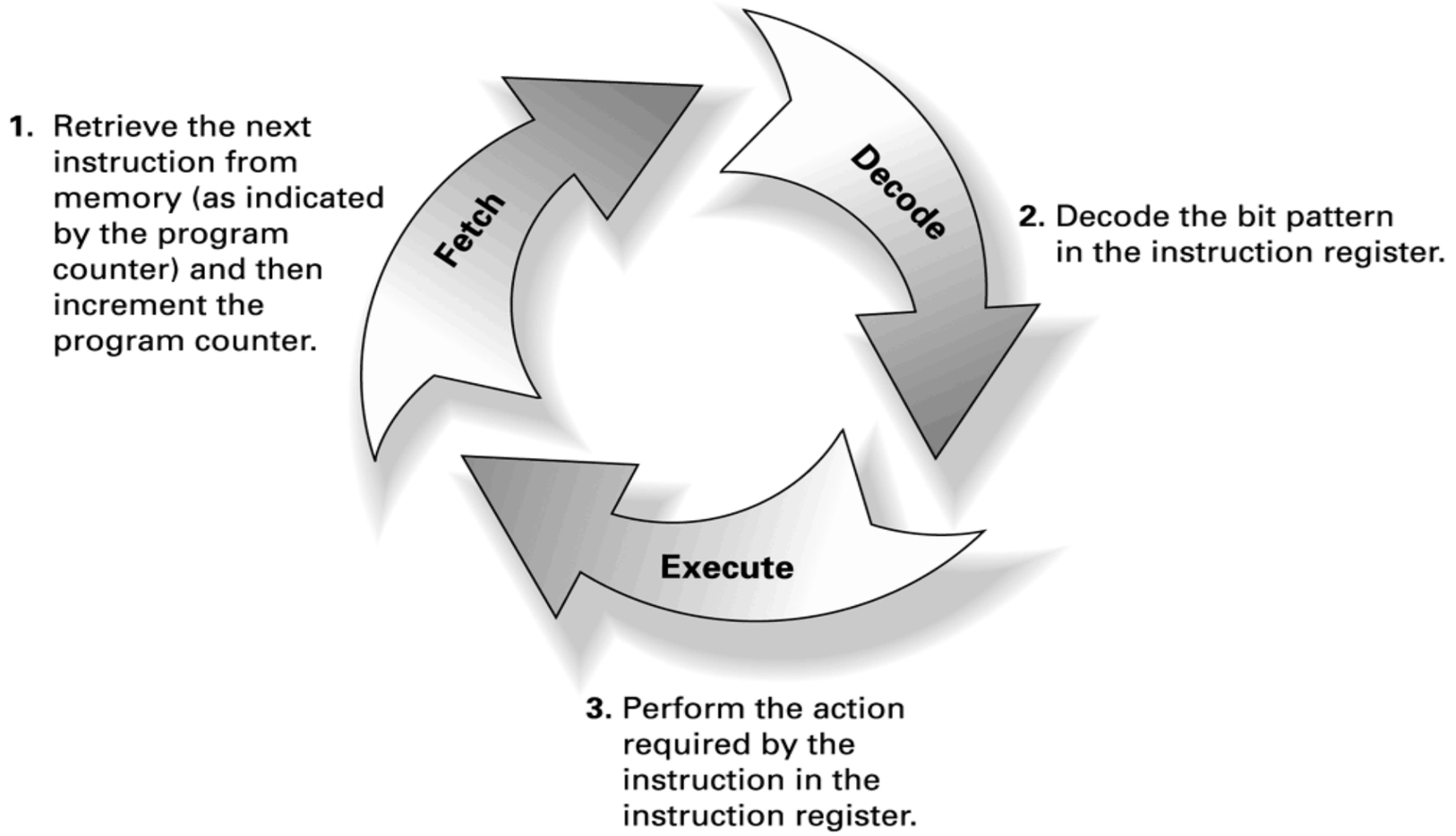


CPU Döngüsel İşlevler

CPU, işlevsel çevrimde, Adres bus ve Data bus üzerinde aynı anda işlem yapılmaz. Önce ilgili belleğin ve bellek gözünün ya da giriş-çıkış birimim adresi seçilir. Sonra seçilen adrese veri yazılır ya da okunur. Bu nedenle data yolu hatları ile adres yolu hatları aynı hatlar olabilir. Döngüsel işlevler:

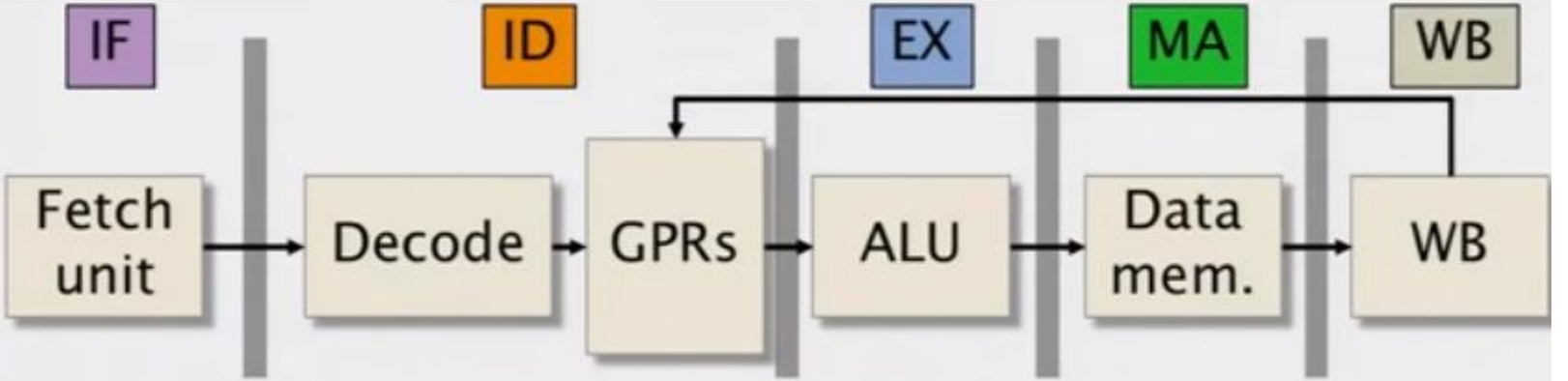
- **Komutu alıp getirme (Fetch Instruction):** Komut Bellekteki (ROM) komut işlevinin yürütülmesi, komut ne istiyor? Komutun çözümlenmesi..
- **Address Decoding Unit:** Bellek gözlerinin çakışmasını önlemek amacıyla bellek seçilmesi
- **Veriyi alıp getirme/alıp götürme (Fetch Data):** İşlenecek verinin register'dan ana RAM belleğe veya I/O birimine transfer edilmesi ya da Ram bellekten veya I/O biriminden register'a transfer edilmesi işlevi.
- **Data Yorumlama (Interpret Instruction):** Hangi işlemin gerekli olduğunu (Aritmetik, mantık, karşılaştırma, …) belirlemek için program komutlarının yorumlanması
- **Veri İşleme (Process data):** Bir komutun yürütülmesi aşamasında, veriler üzerinde aritmetik veya mantıksal işlemlerin gerçekleştirilmesi
- **Ana Belleğe Yazma /Okuma:** Verilerin I/O birimlerine ya da ana belleğe yazılması; I/O birimlerinden ya da bana bellekten okunması
- **Ön Belleğe Yazma /Okuma:** Verilerin ana belleğe veya bir I/O modülüne yazılmasından önce, verimliliği artırmak amacıyla veriler ön bellekte geçici olarak saklar.

Mikroişlemci Ana İşlev Döngüsü



- Bir komut çevrimi, işlenen komutların karmaşıklığına göre bir kaç clock çevriminde tamamlanır.
- Çevrim: Fetch, Execute (Komutları işlevsel yürütülmesi) ve adres decoding yapısından oluşur.

Block Diagram of 5-Stage Processor



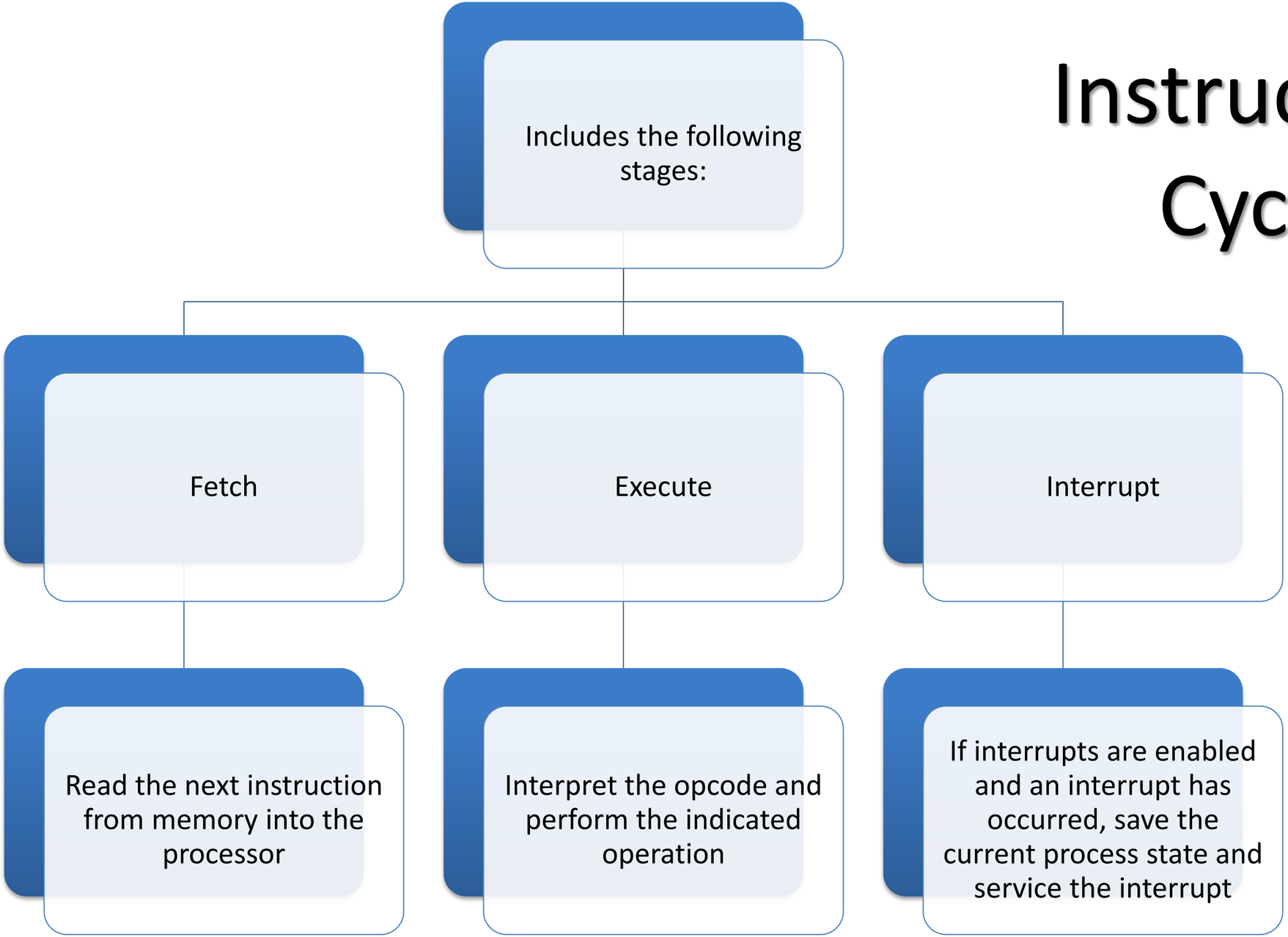
Each instruction is executed through 5 stages:

1. Instruction fetch (IF): Read instruction from memory.
2. Instruction decode (ID): Determine which units to use to execute the instruction, and extract the register arguments.
3. Execute (EX): Perform ALU operations.
4. Memory (MA): Read/write data memory.
5. Write back (WB): Store result into registers.

Fetch-Execute Döngüsü

- *Fetch-Execute Döngüsü, Yazılan programlar çalışırken*
  - Hafızadan komutlar alır
  - Neyi temsil ettiğini görmek için komut kodun çözülür.
  - İstenen işlemler gerçekleştirilir.
  - Bir sonraki komut getirilir, kod çözülür ve yürütülür. Köpürtün, durulayın, tekrarlayın!
- *Gerçekleşen İşlemler*
  - Dış birimden ya da bellekten veri oku; dış birime ya da belleğe veri yaz.
  - Dallanma
  - Aritmetiksel, Mantıksal işlemleri gerçekleştir. (ALU)
  - Bayraklar
  - Pipelining: Ardışık düzende komut işleme; Parelel çok sayıda komutun aynı anda işlenmesi
  - Zamanlama (Clock Timing) ve Kontrol Birimi
  - Kesme işlemleri (Interrupt)

Instruction Cycle



Instruction Cycle

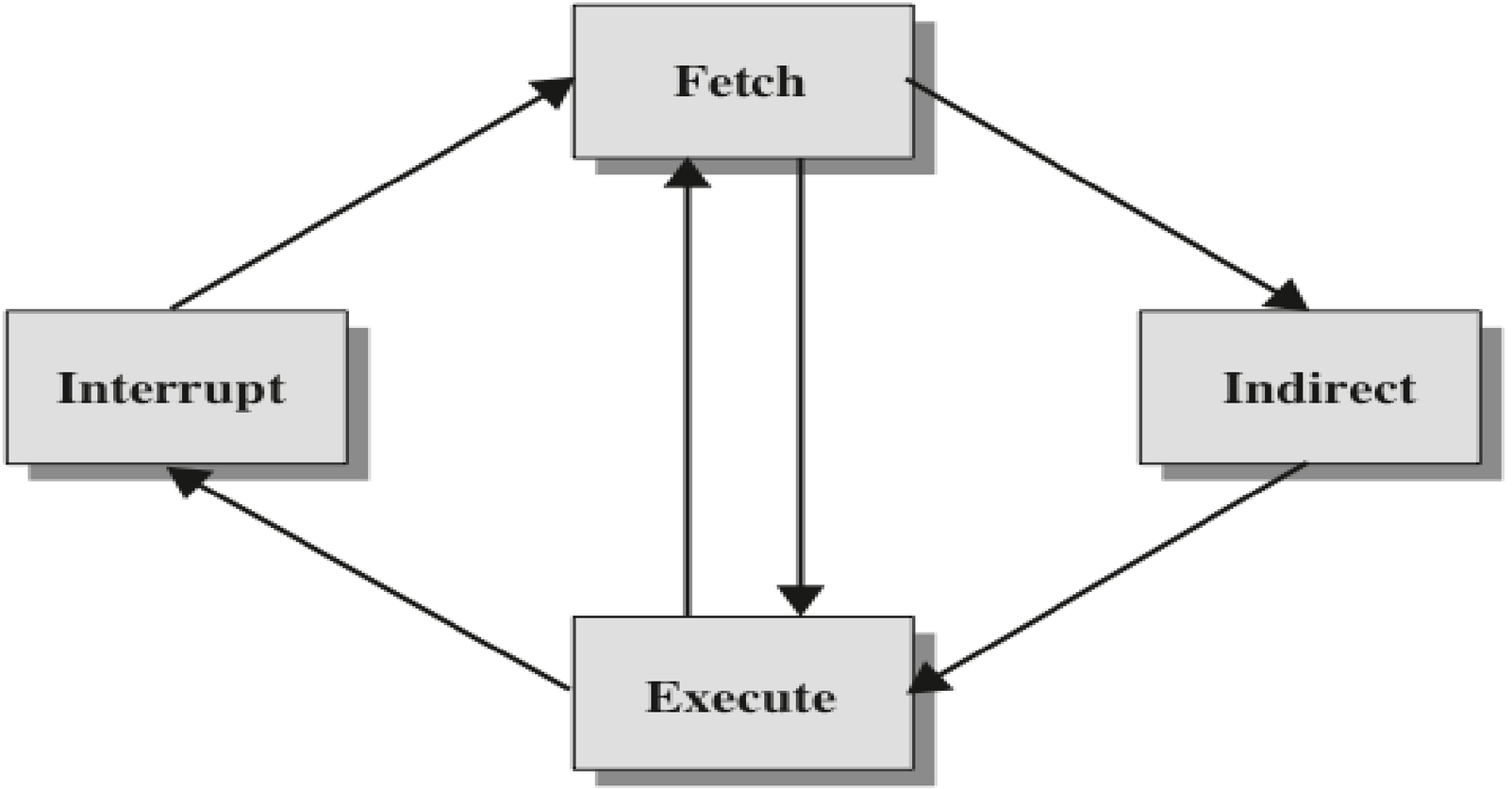


Figure 14.4 The Instruction Cycle

Kesme (Interrupt)

- CPU'nun dikkatini döngüsel işlevini yürüttüğü programdan başka bir işleve (bir G/Ç aygıtı, işletim sistemi) çevirebilmesi için kesintiye uğratılmasıdır.
- Her getir-yürüt-götür döngüsünün sonunda, CPU bir kesinti olup olmadığını kontrol eder.
- I/O Cihazları, kontrol veri yolu üzerinden CPU'ya kesme sinayi gönderir. Komut işlevi bir kesmeye neden olursa, Kesme Bayrağı (durum bayraklarında) ayarlanır.
- Bir kesinti meydana geldiyse, kesinti aşağıdaki gibi işlenir.
  - CPU o ana yaptıklarını kaydeder. Program sayacı (PC) ve diğer önemli kayıtlar yığın belleğe kaydedilir
  - CPU, kesmeyi kimin yükselttiğini ve bu tür bir kesmeyi işlemek için bir kesme işleyicisini çalıştırdığını anlar. Kesme işleyicisi, o kesme isteğine ilişkin bellekte depolanan bir kod kümesini çalıştırılır.
  - Kesme işlevinde istenilenler yerine getirildikten sonra, CPU, çalışma zamanında bellekte kaydedilen değerleri alarak kesintiye uğrayan programı geri yükler. Kaldığı yerden çalışmasına devam eder.

Perormans Artırıcı Çözümler

CPU’yu Oluşturan Temel Bileşenler

- Registers
- Zamanlama (Clock Timing) ve Kontrol Birimi
- ALU ve Bayraklar
- Ön Bellek (Cache Bellek) – SRAM
- Veri transferi

Architectural Improvements

Historically, computer architects have aimed to improve processor performance by two means:

- Exploit parallelism by executing multiple instructions simultaneously.
  - Examples: instruction-level parallelism (ILP), vectorization, multicore.
- Exploit locality to minimize data movement.
  - Example: caching.

Exploit: faydalanmak, kullanmak
Data Movement: Time and Path

“Mikroişlemcilerde Verimi Artırma”

Pipelining: Ardışık düzende komut parçalarını işleme

Süper ardışık düzen(pipelined) makine:

- Ardışık düzen, bir işlemciyi oluşturan birimlerinde bir anda bir komuta ait tek bir iş yapılması yerine çok sayıda komutun belirli işlevlerinin çok bölüme (aşamaya) ayırma ve komutların bununla örtüşmesine izin verme eylemidir.
- Söz gelimi, bir mikro işlemci 5 birimmden oluşsun (Register, ALU, Adres dekoding, WR/RD, Bellek hücre seçme) bir komut register'da iş yaparken diğer beş birim neden boş beklesin. Farklı 5 komutun belirli işlevleri bu 5 birimde ayndı yapılabilir.
- Süper disiplinli makineler ardışık düzen aşamalarını üst üste bindirir. Sonuncusu tamamlanmadan önce işlemlere başlayabilecek aşamalara dayanır.

Süper Ölçekleme Makinesi:

- Bir Superscalar makine, birden çok bağımsız komutu paralel olarak yürütmek için birden çok bağımsız ardışık düzen kullanır.
- Özellikle genel komutlar (aritmetik, yükleme / saklama, koşullu dallanma) bağımsız olarak yürütülebilir.
- CPU'nun işlevsel döngüsü; adres dekoding circuit ve adres hatları; okuma ya da yazma işlevleri; data bus aktif etme; ALU; veri transfer etme işlevleri. Ayı anda işlevsel döngüyü oluşturan işlemler yapılır. Sıralı işlevlerde parelel komutlar yürütülür.

CPU Temel İşlev Döngüsü

CPU İşlev Döngüsü:

- Address Decoding: Mikroişlemcide bellek gözlerinin çakışmasını ya da üst üste gelmesini önlemek amacıyla bellek ya da I/O birimlerini seçer?.
- Fetching cycles: Mikroişlemcide veriyi alıp getirme işlevlerdir.
- Execution cycles: Veir işleme gibi manipülasyon işlevlerdir.
- Clock and Timing Signals: Mikroişlemcide verilerin senkronize (eş zamanlı) işlenmesini sağlar. Mikroişlemcide verilerin senkronize (eş zamanlı) işlenmesini ve iletilimesini clock sinyali sağlar.
- Pipelining – Ardışık düzende komut işleme: Mikroişlemcide çalışma performansını artırmak amacıyla, komutlar dizisinin ardışık düzenlendiği ve eşzamanlı parelel yürütüldüğü işlevlerdir. Aynı anda 4 – 5 komutun farklı bölümleri CPU tarafından işlenir. A, B, C, D, E komutlar ise
- A komutun parçacıkları A1,A2,A1, A4, A5;
- B komutu: B1, B2, B3, B4 ,B5; diğer komutlar benzer biçimde ....
- E1,D2,C3,B4,A5 komut parçalarını aynı anda işleyebilmektedir. E1: E komutunun 1. parçacığı, D2: D komutunun 2. parçacığı, ...

CPU Temel İşlev Döngüsü

Komut	Adres Bulma	Read	Kod Çözm	İşlem	Yaz
A	A1	A2	A3	A4	A5
B	B1	B2	B3	B4	B5
C	C1	C2	C3	C4	C5
D	D1	D2	D3	D4	D5
E	E1	E2	E3	E4	E5
1. Clock	A1				
2. Clock	B1+A2				
3. Clock	C1+B2+A3				
4. Clock	D1+C2+B3+A4				
5. Clock	E1+D2+C3+B4+A5				

Mikroşlemci farklı komutlarının farklı işlevsel adımlarını parallel yerine getirir.

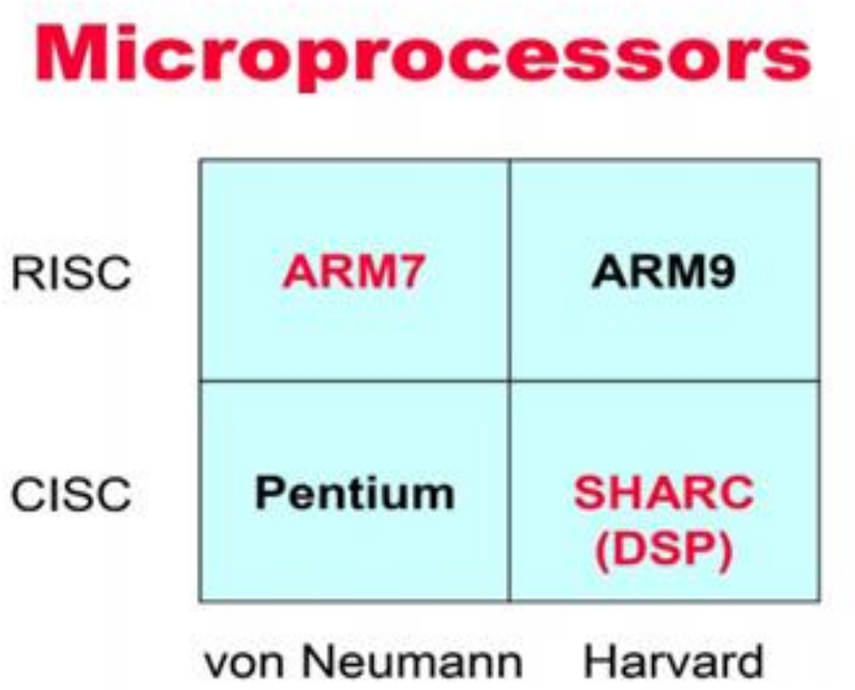
Machine Language Philosophies

- Bilgisayar tasarımında önemli noktalardan birisi işlemcinin komut kümesinin belirlenmesidir. Belirli bir bilgisayar için seçilen komut kümesi bu bilgisayarın makine diliyle yazılımlanmasını belirler.
- Sayısal donanım ucuzlamaya başlayıp, tüm devreler daha ileri bir seviyeye ulaşınca bilgisayar komutları da hem sayı hem de karmaşıklık olarak arttı. Bazı bilgisayarlar 100 hatta 200'ün üzerinde komut kümesine sahip oldular.
- Bu bilgisayarlar çok farklı veri tiplerini kullanabiliyorlar ve çok sayıda adresleme kipi bulunuyordu.

- Reduced Instruction Set Computing (RISC)
  - Few, simple, efficient, and fast instructions
  - Examples: PowerPC from Apple/IBM/Motorola and ARM
- Complex Instruction Set Computing (CISC)
  - **Çok sayıda komutları bulunan bir bilgisayar CISC olarak adlandırılır.**
  - Many, convenient, and powerful instructions
  - Example: Intel

Mikroişlemcilerin Sınıflandırılması

- By the width of the data format they process
- By their instruction set
  a) RISC (Reduced Instruction Set Computer)
  b) CISC (Complex Instruction Set Computer)
  c) EPIC (Explicitly Parallel Instruction Computing)



RISC İşlemciler

- RISC işlemciler, yüksek düzeyli yazılım dillerinin oluşturduğu veriler dikkate alınarak merkezi işlem birimlerinin verimlerini artırmak amacıyla
  - daha az bellek erişimi yapan,
  - daha az sayıda komut,
  - daha az sayıda adresleme kipi,
  - sabit uzunlukta komut yapısı (komut çözme işi kolaydır),
  - doğrudan bellek üzerinde işlem yapan komutlara sahip olmayıp, işlemlerin iç saklayıcılarda yapılması
  - Belleğe sadece okuma/yazma işlemleri için erişme
  - Tek çevrimde alınıp yürütülebilen komutlar (komut işhattı sayesinde)
  - Devrelendirilmiş (hard wired) donanım birimi özelliklerine sahiptir.

  Bazıları tüm RISC makinelerde bulunmayan bazıları ise CISC makinelerde de rastlanılabilen RISC işlemciler için özellikle önemli özellikler ise:
  - Çok sayıda geçici saklayıcı (register file)
  - Kesişimli saklayıcı penceresi (overlapped register window)
  - Komutlar için optimize edilebilen işhattı(pipeline: ardışık düzende komut işleme)
  - Derleyici desteği olarak sayılabilir.

Komut İşleme Teknikleri Açısından: Mikroişlemci Mimari Kavramları

CISC- Complex Instruction Set Computer (Karmaşık Komut Kümeli Bilgisayar):

- Bu mimaride mikroişlemci çok sayıda komut içerir ve her eylem için bir komut tanımlanmıştır. Böylece yüzlerce komut arasından seçilen komutlarla yazılan program daha kısa olmaktadır.
- RISC- Reduced Instruction Set Computer (Azaltılmış Komut Kümeli Bilgisayar): Bu mimaride ise daha basit komutlar kullanarak tümdevre karmaşıklığı azaltılmaktadır. Ancak komutların daha kısa olması belli bir görevin tamamlanabilmesi için daha fazla komuta ihtiyaç duyulur.
- RISC mimarileri PC sektöründe olmasa da SUNUCU sektörünü tamamen işgal etmiştir: SUN ve IBM tarafından üretilen sunucuların işlemcileri RISC mimarisinde tasarlanmıştır. PC piyasasında ise CISC mimariler popüler olarak kullanılmaktadır (Intel, AMD).
- RISC, "Azaltılmış Komut Seti Bilgisayarları" anlamına gelir. komutlar, kullanıcıların kendi işlemlerini tasarlamalarına izin vermek için olabildiğince çıplaktır. Gömülü sistemlerde RISC mimari kullanılır.
- CISC, "Karmaşık Komut Seti Bilgisayarları" anlamına gelir. Her biri aynı işlemin farklı bir permütasyonunu gerçekleştiren çok sayıda komut.

Komut İşleme Teknikleri Açısından RISC / CISC

Mimari olarak:

- “RISC (Reduced Instruction Set Computers)” Power PC, Alpha, MIPS
- “CISC (Complex Instruction Set Computers)” x86, Pentium, Itanium, Athlon, Duron

RISC	CISC
+Daha anlaşılır ve basit kod kullanımı	-Karmaşık assembly komutları
-Aynı işlev için daha uzun program kodu	+Daha az saat çeviriminde daha çok iş
+Daha hızlı kod işletimi	-Uzun süren kod çevirim aşamaları
+Daha az donanım	-Daha çok donanım

Classification of Microprocessors - RISC

- Bilgisayarda, çok sayıda karmaşık ve uzmanlaşmış komut yerine, basit ve genel yönergeler bulunur.
- Komut setinin son derece düzenli komut özel grupsal (pipe line) hat akışı için optimize edilmesidir.
- Komutun bir parçası olarak değil, belirlikomutlar yoluyla belleğe erişildiği yükleme / saklama mimarisidir.
- RISC (Reduced Instruction Set Computer) Azaltılmış Komut Seti Bilgisayarı aşağıdaki özelliklere sahiptir:
  a) Basit ilkel komutlar ve adresleme modları
  b) Komutlar bir saat döngüsünde yürütülür
  c) Düzgün uzunluk komutları ve sabit komut formatı
  d) Kablolu kontrol
  e) Sabit mekanizmalar aracılığıyla bellekle komut arayüzü
  f) Karmaşıklık derleyiciye aktarılma
  g) Daha az komutu destekler
  h) Örneğin Pentium1, AMD K6, Intel P6 vb.

RISC

- RISC ifadesi “Reduced Instruction Set Computer”, yani azaltılmış komut kümeli bilgisayar anlamına gelmektedir. Adından da anlaşılacağı üzere, RISC işlemciler CISC işlemcilere göre daha az sayıda komut içermektedir. Daha az sayıda ve daha basit komutlar ile işlem süresi azaltılabilmektedir. RISC işlemcilerin temel ve ortak bazı özellikleri şunlardır:
- Az sayıda komut ve adresleme kipi
- Sadece LOAD ve STORE komutları ile bellek erişimi gerçekleştirilmesi
- Sabit uzunluklu, kolay çözülebilen komut yapısı
- Mikroprogramlı denetim yerine donanımsal denetim gerçekleştirilmesi
- komutların tek evrede icra edilmesi
- RISC işlemcilerdeki önemli özelliklerden biri komut uzunluğunun sabit olması ve yapısının kolay çözülmeye imkân sağlayabilmesidir.
- Günümüzde RISC ailesine ait bazı önemli işlemciler ARM, AMD 29k, SPARC ve PowerPC’dir.

Classification of Microprocessors - CISC

CISC (Karmaşık Komut Seti Bilgisayarı) basit ya da karmaşık yüzlerce komut setini destekler. Değişken boyutlardaki komutların yürütülmesi genellikle 1 clock peryodundan fazla sürer.

- CISC (Complex Instruction Set Computer) aşağıdaki özelliklere sahiptir:
  a) Yüzlerce komutu destekler
  b) Daha zengin komut seti, bazıları basit, bazıları çok karmaşık
  c) komutların yürütülmesi genellikle 1clock peryodundan fazla sürer
  d) Değişken boyutların komutları
  e) Örneğin Intel 386, Intel 486, Pentium III, Pentium Pro vb.

CISC

- Karmaşık komut kümesi bilgisayar mimarisidir.
- Bellekten yükleme, aritmetik işlem ve bellek depolama gibi tek bir komut dahilinde adresleme modları gibi çok adımlı işlemler yapabilen bir bilgisayardır.

- Kompleks yönerge seti bilgisayarlarında (complex instruction set computers, CISC) yıllarca yapılan tasarım çabalarının sonuçları
- CISC tasarımının mükemmel örneği
- Bir zamanlar yalnızca ana bilgisayarlarda ve süper bilgisayarda bulunan sofistike tasarım ilkelerini bir araya getirir
- İşlemci tasarımına alternatif bir yaklaşım, azaltılmış yönerge seti bilgisayarı (reduced instruction set computer, RISC)
- ARM mimarisi çok çeşitli gömülü sistemlerde kullanılmaktadır ve piyasadaki en güçlü ve en iyi tasarlanmış RISC tabanlı sistemlerden biridir
- In terms of market share Intel is ranked as the number one maker of microprocessors for non-embedded systems

CISC

- Karmaşık komut kümesi bilgisayar mimarisidir. Bellekten yükleme, aritmetik işlem ve bellek depolama gibi tek bir komut dahilinde adresleme modları gibi çok adımlı işlemler yapabildiği bir bilgisayardır.
- Terim geriye dönük olarak azaltılmış komut seti bilgisayarının (RISC) aksine üretilmiştir ve bu nedenle RISC olmayan her şey için, büyük ve karmaşık ana bilgisayarlardan bellek yükü ve depolama işlemlerinin yapıldığı basit mikrodenetleyicilere kadar aritmetik komutlardan ayrılmamış bir şemsiye terim haline gelmiştir. Bu nedenle modern bir RISC işlemci, örneğin elektronik devrelerinin karmaşıklığında değil, aynı zamanda kodlama modellerinin komutlarında veya karmaşıklığında CISC etiketli bir komut seti kullanan modern bir mikrodenetleyiciden çok daha karmaşık olabilir. Tek tipik ayırt edici özellik, çoğu RISC tasarımının neredeyse tüm komutlar için tek tip komut uzunluğu kullanması ve kesinlikle ayrı yük / mağaza komutları kullanmasıdır.

CISC

- CISC işlemcilerde kullanılan komutlar, yüksek seviyeli dillerin sağladıkları özelliklere benzer işlevleri gerçekleştirmek üzere tasarlanmışlardır. CISC işlemcilerin bazı belirgin özellikleri;
- 100-250 arasında değişen çok sayıda komut
- 5-20 arasında değişen çok sayıda adresleme kipi
- Değişken uzunluklu komut yapıları
- Bellekten veri kullanabilen komutlar
- Yukarıda verilen özelliklerde belirtildiği üzere, CISC işlemcilerde komut uzunlukları sabit olmayabilmektedir. Bu tip işlemcilerde komut çözme işlemi aşama aşama gerçekleştirilir. Komutun bütünüyle çözülmesinin kaç aşamada tamamlanacağı farklı komutlar için farklı olabilmektedir. Bellekten okunacak verinin ve ya komutun boyutu, her aşamada komuta bakılarak tekrar değerlendirilebilir.

EPIC: Explicitly Parallel Instruction Computing

- Bilgisayarlarda perormans artırılmasında, işlemci clock sinyali etkin parametredir. Komutların işlenme adımlarını düzenleyerek performans artırılabilir mi? Yazılımların derlenmesi aşamasında komut işleme adımları öyle düzenlenmeli ki, performans artmış olsun. Bu da parelel komut işleme hesaplanması ile mümkündür.
- EPIC, 1980'lerin başından beri araştırmacıların araştırdığı bir bilgi işleme paradigmasını tanımlar.
- 1997 yılında HP – Intel tarafından kullanılan bir terimdir.
- EPIC, mikroişlemcilerin paralel komut yürütmesini kontrol etmek için karmaşık kalıp üstü devre yerine derleyiciyi kullanarak yazılım komutlarını paralel olarak yürütmelerine izin verir. Bu, daha yüksek clock peryodlarına başvurmadan basit performans ölçeklendirmesine izin vermek için tasarlanmıştır.
- 1989'da HP'deki araştırmacılar, azaltılmış komut seti bilgisayar (RISC) mimarilerinin döngü başına bir komutta bir sınıra ulaştığını fark ettiler.
- EPIC'in bir amacı, komut çizelgelemenin karmaşıklığını CPU donanımından yazılım derleyicisine taşımaktı ve bu da çizelgelemeyi statik olarak yapabiliyordu (izleme geri bildirimi bilgileri yardımıyla).
- Böylece, CPU'da, ek yürütme kaynağı da dahil olmak üzere diğer işlevler için yer ve güç tasarrufu sağlayan karmaşık zamanlama devresi ihtiyacını ortadan kaldırır.

EPIC

- Mikroişlemcilerin paralel komut yürütmesini kontrol etmek için karmaşık kalıp üstü devre yerine derleyiciyi kullanarak yazılım komutlarını paralel olarak yürütmelerine izin verir.

- EPIC (Explicitly Parallel Instruction Computing) aşağıdaki özelliklere sahiptir:

  a)Paralellik sağlar

  b) Mümkün olduğu kadar paralel sabit genişlikli komutlar kullanmaz

  c) Programlar, açıkça ortaya konmuş paralellikle sıralı anlamlar kullanılarak yazılmalıdır.

  d) Derleyici, yürütme planının tasarlanmasında kilit rol oynamalı ve mimari, bunu başarılı bir şekilde yapması için gerekli desteği sağlamalıdır.

ARM

- **ARM** mimarisi (orijinal adı **Acorn RISC Machine**) RISC tabanlı bir işlemci mimarisidir, 32 ve 64 bit versiyonları vardır, genel itibarıyla düşük güç tüketimi, diğer RISC tabanlı işlemcilere göre yüksek performanslı oluşu ve x86-x64 işlemcilere göre daha hesaplı olmasından dolayı gömülü sistemlerde, taşınabilir aygıtlarda kullanılan yongasetlerinde genelde ARM işlemci tercih edilir.
- ARM firması kendi başına işlemci üretmez, dizayn ve lisansı satar, CISC tabanlı işlemcilere göre özelleştirilebildiğinden aynı jenerasyon işlemci farklı üreticilerden değiştirilmiş olarak çıkabilir, bağlı olarak da performans farklılıkları görülür. Bu yüzden işlemci jenerasyonları ve karakteristikleri incelenirken ARM referans tasarımı ele alınır.
- CPU mimarisi, cep telefonlarında, iPod'larda, uzaktan sensör ekipmanlarında ve diğer birçok cihazda kullanılan gömülü işlemcidir. Aslında azaltılmış bir komut seti bilgisayardır.
- Gömülü işlemci: CPU + Bellek + I/O aynı entegre içindedir.

Gerçek Mimari

- Intel – 8086 ile 1979'da bir CISC olarak başladı, zamanla karmaşıklık ve yetenek açısından ilerledi, orijinal olarak 16 bit olan 4 genel amaçlı yazmaç,
- 80386 ile 32 bit'e genişletildi ve pentium, pentium II'de kayan nokta işlemleri ve daha sonra birçok RISC içerir pipelinening , süperskalar ve spekülatif yürütme gibi özellikler
- MIPS – RISC mimarisi, başlangıçta 32 bit, şimdi 64 bit, 32 bit register, süperskalar ve spekülatif yürütme ile gelişmiş pipelining mimari

“CPU Execution Time”

Örnek:
CPU Performans: Clock Hızı ve Data Bus

- Bir belleğin data bus hat sayısı 128 bit, clock frekansı 2GHz ise bir saniyede bu belleğe kaç bit yazılır? 1 clock peryodunda 128 bit (16 byte) yazılır.
- Clock peryodu, T=1/f=1/2GHz=1/(2*10^9Hz)=0.5*10^-9 saniye
- 0.5*10^-9 saniyede 128 bit yazılır ise 1 saniyede kaç bit yazılır?
- 1 saniyede yazılacak veri miktarı bit olarak =128/(0.5*10^-9)= 256*10^9bit = 256Gbit/sec.
- 1 saniyede yazılacak veri miktarı byte olarak =256*10^9bit /8= 32Gbyte/sec.
- 1 saniyede yazılacak veri miktarı word olarak =256*10^9bit /16= 16Gword/sec.

CPU Execution Time: Example

- Bir Program, aşağıdaki parametrelerle belirli bir makinede çalışıyor:
  - Toplam Komut Sayısı(TIC:Total Instruction Count): 10,000,000 komut
  - Komut Ortalama Clock Döngüsü (CPI: Cycles Per Instruction) : 2.5 Cycles/Komut.
  - CPU Clock Hızı (f): 2 GHz.
- Bu program için CPU Yürütme Zamanı (CPU-ET: Execution Time) nedir?
- Clock Peryod, T=1/f.

CPU Program Yürütme Zamanı, CPU-ET=TIC x CPI x T.

CPU-ET= Total Instruction count x CPI x Clock Peryod

= 10,000,000 x 2.5 x 1 / Clock Hızı

= 10,000,000 x 2.5 x 0.5×10^{-9}

= .0125 seconds

Örnek: Komut Tipleri & CPI

- Bir komut setinin üç komut sınıfı vardır:

Komut Sınıfı	Komut Dizisi	CPI
K1	2	1
K2	1	2
K3	2	3

Örnek: 1. Dizi için CPU Clock Döngü Sayısı= A1*K1+B1*K2+C1*K3=2 x 1 + 1 x 2 + 2 x 3 = 10 cycles

1. Dizi için CPI = Clock Döngüsü Sayısı / Komut Sayısı = 10 /(A1+B1+C1)=10/5 = 2

- İki kod dizisi aşağıdaki komut sayılarına sahiptir:

	Komut sayıları			
Kod Dizisi	Ai	Bi	Ci	i=1,2.
1	2	1	2	
2	4	1	1	

Örnek: 2. Dizi için CPU Clock Döngü Sayısı= A2*K1+B2*K2+C2*K3=4 x 1 + 1 x 2 + 1 x 3 = 9 cycles

2. Dizi için CPI = 9 /(A1+B1+C1)=9/6 = 1.5

“Fetching & Execution Cycles”

Fetching & Execution Cycles

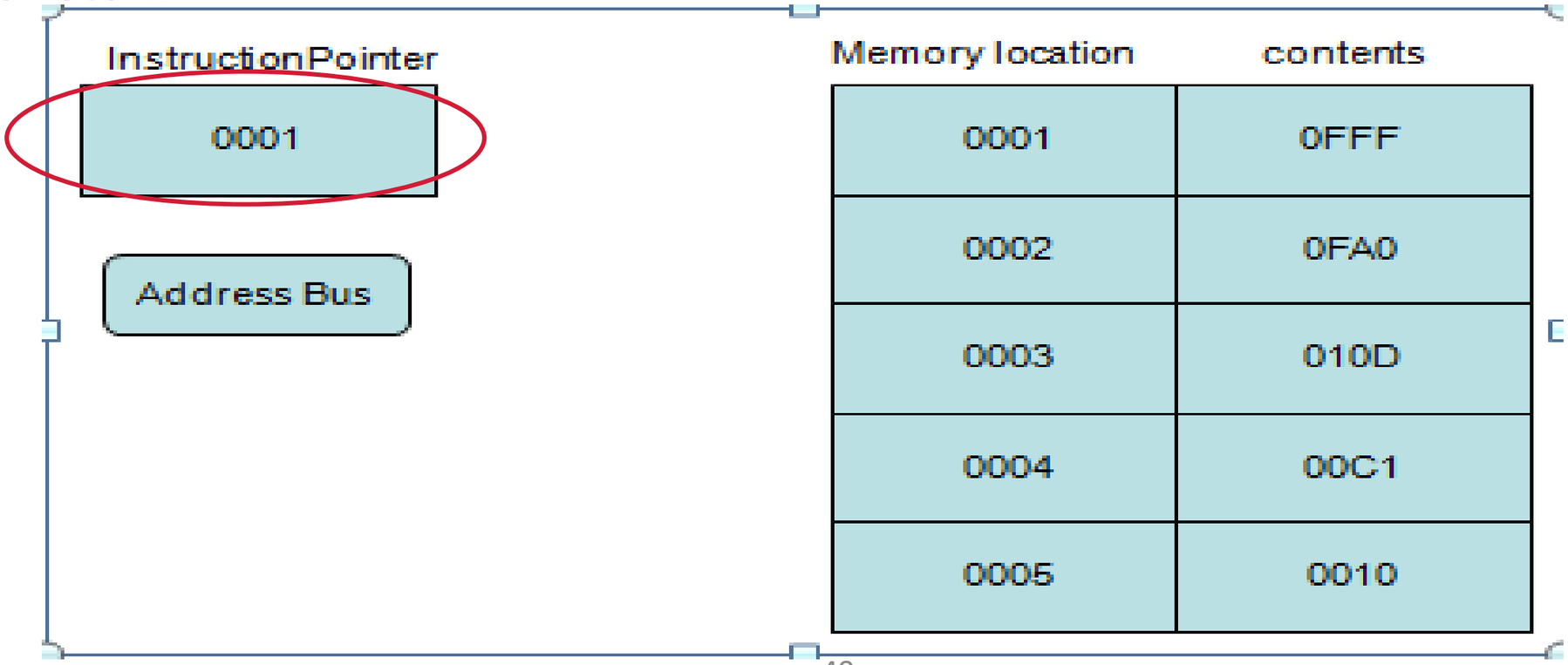
- **Fetching Cycles**
  - Getirme döngüsü, gerekli talimatı bellekten alır, talimat kaydında saklar ve program sayacını bir sonraki talimatı gösterecek şekilde hareket ettirir.

- **Execute cycle**
  - Bir talimatın yürütme döngüsü sırasında meydana gelen gerçek eylemler.
  - Hem talimatın kendisine hem de gerekli olabilecek verilere erişmek için kullanılmak üzere belirtilen adresleme moduna bağlıdır.

Fetching an instruction

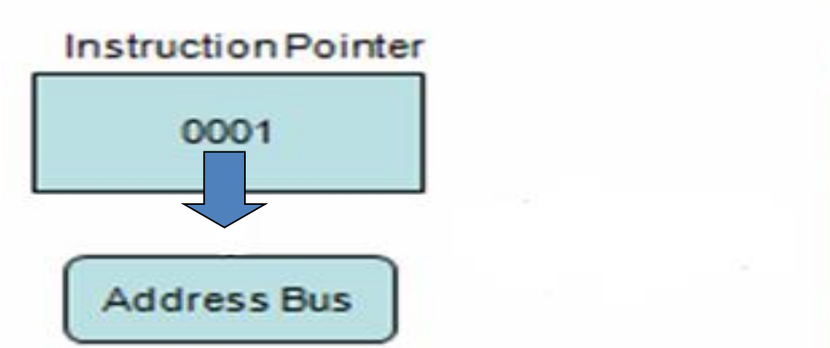
- **Step 1**

  Instruction pointer (program counter) hold the address of the next instruction to be fetch.



FETCHING AN INSTRUCTION (cont.)

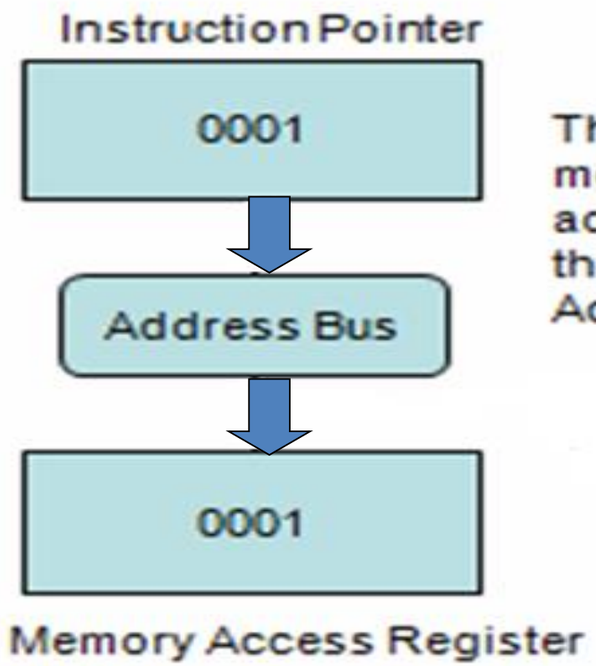
- **Step 2**



Memory location	contents
0001	0FFF
0002	0FA0
0003	010D
0004	00C1
0005	0010

FETCHING AN INSTRUCTION (cont.)

- **Step 3**

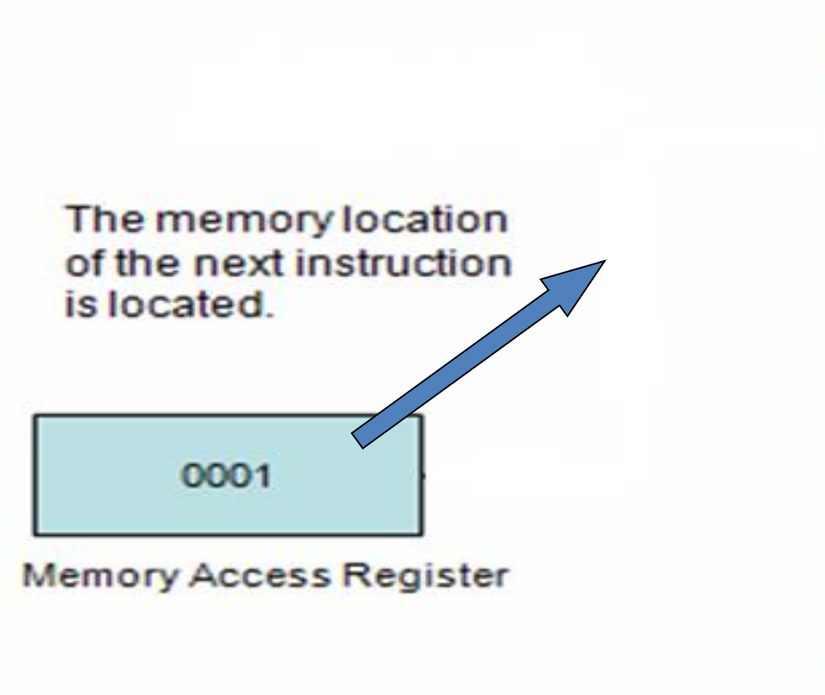


The address moves over the address bus to the Memory Access Register

Memory location	contents
0001	0FFF
0002	0FA0
0003	010D
0004	00C1
0005	0010

FETCHING AN INSTRUCTION (cont.)

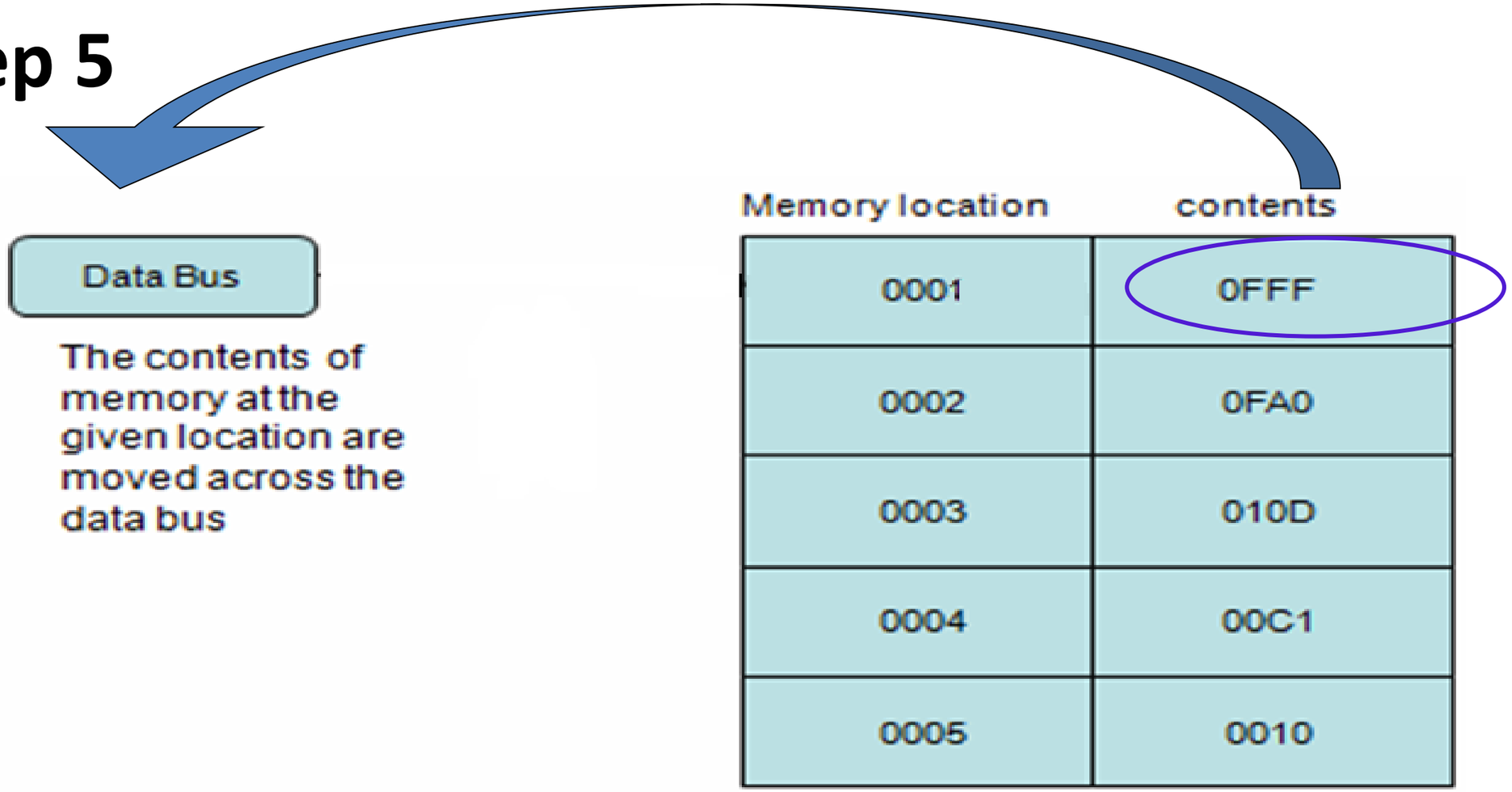
- **Step 4**



Memory location	contents
0001	0FFF
0002	0FA0
0003	010D
0004	00C1
0005	0010

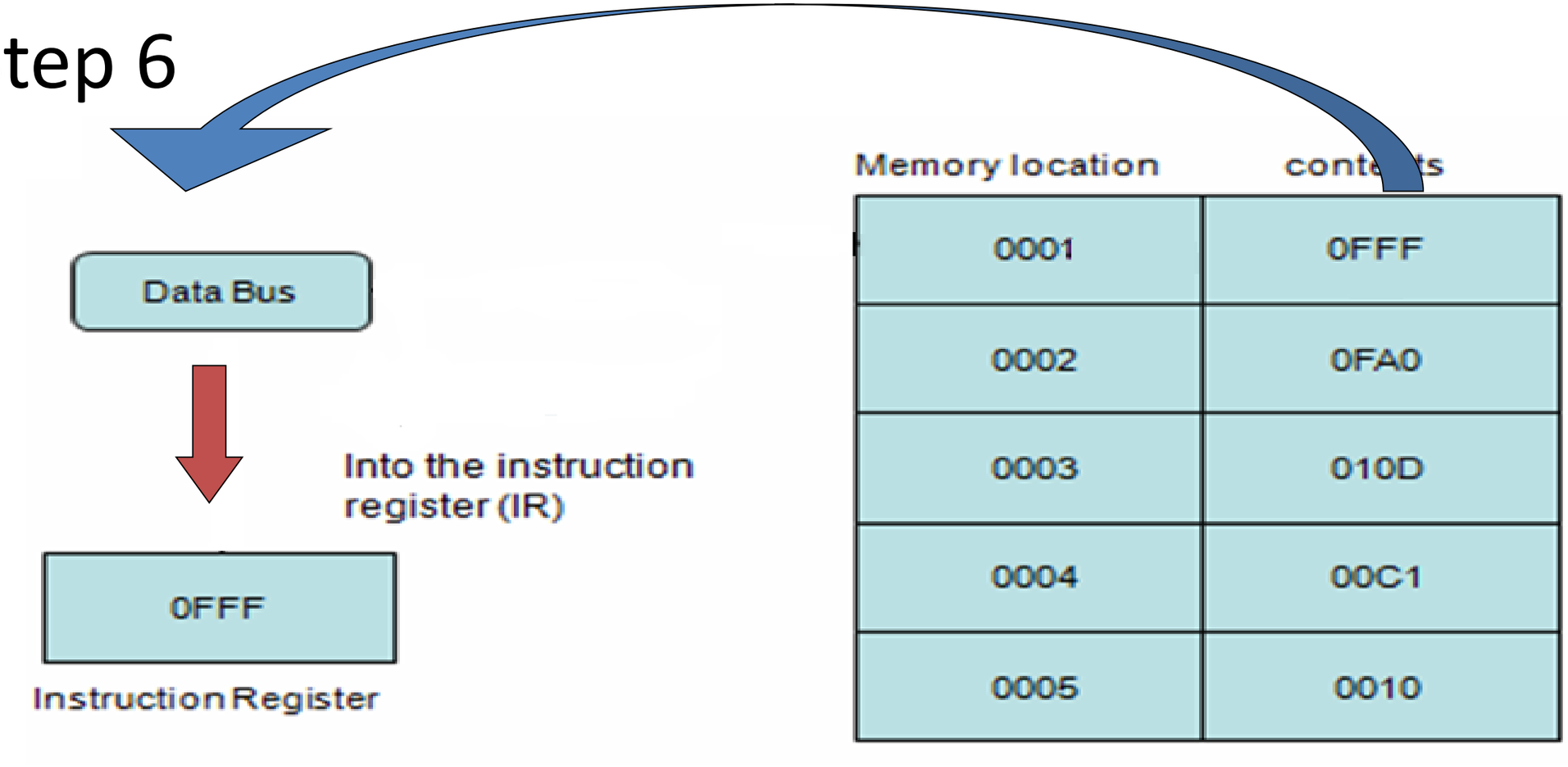
FETCHING AN INSTRUCTION (cont.)

- **Step 5**



FETCHING AN INSTRUCTION (cont.)

- Step 6



PIPELINE (Parelel İşlevsel Hatlar) MİMARİSİ

Ardışık düzende komut işleme

Pipelined Instruction Execution

Processor hardware exploits instruction-level parallelism by finding opportunities to execute multiple instructions simultaneously in different pipeline stages.

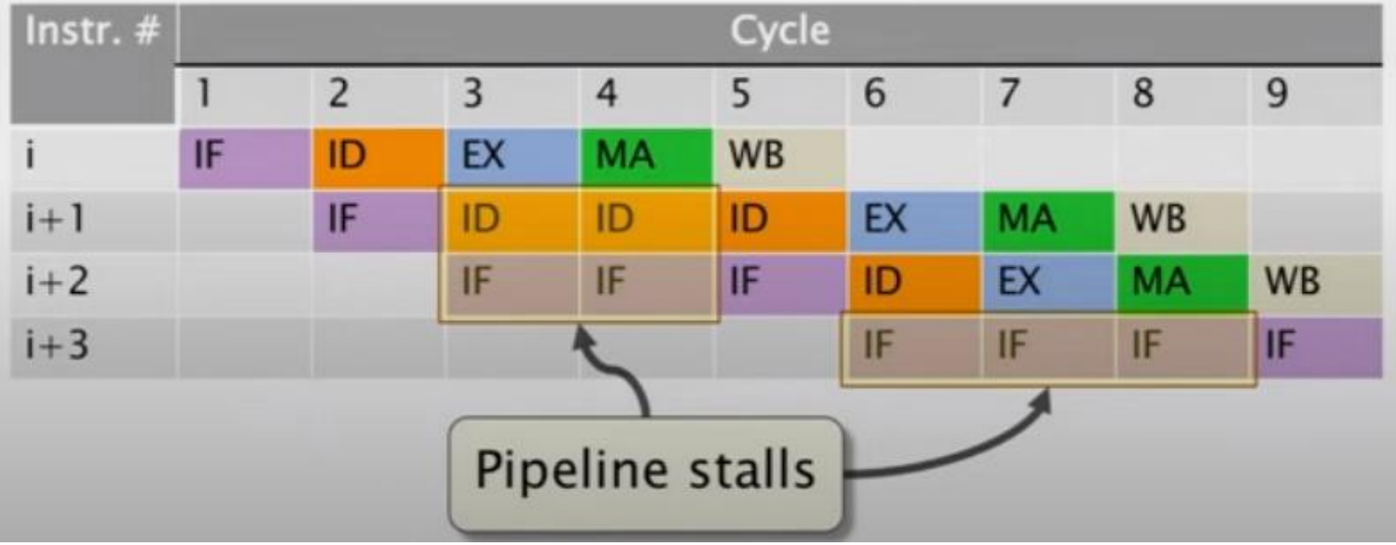
Ideal pipelined timing

Instr. #	Cycle								
	1	2	3	4	5	6	7	8	9
i	IF	ID	EX	MA	WB				
i+1		IF	ID	EX	MA	WB			
i+2			IF	ID	EX	MA	WB		
i+3				IF	ID	EX	MA	WB	
i+4					IF	ID	EX	MA	WB

Pipelining improves processor throughput.

Pipelined Execution in Practice

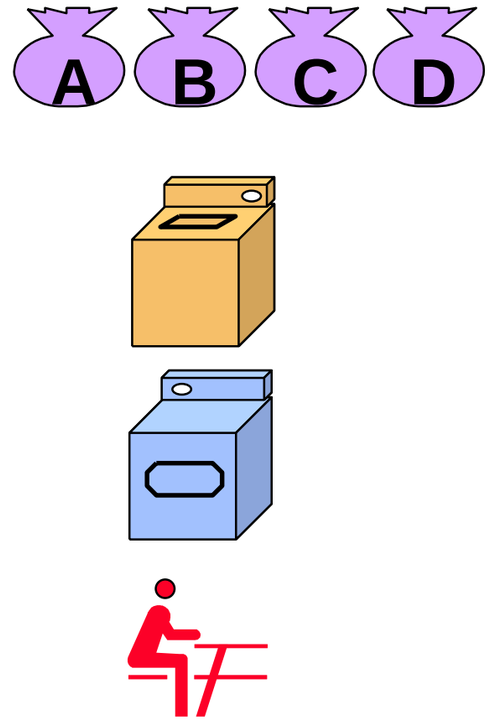
In practice, various issues can prevent an instruction from executing during its designated cycle, causing the processor pipeline to stall.



Designated: Belirlenmiş
Cause: Neden olmak, yol açmak
Stall: geçiktirmek

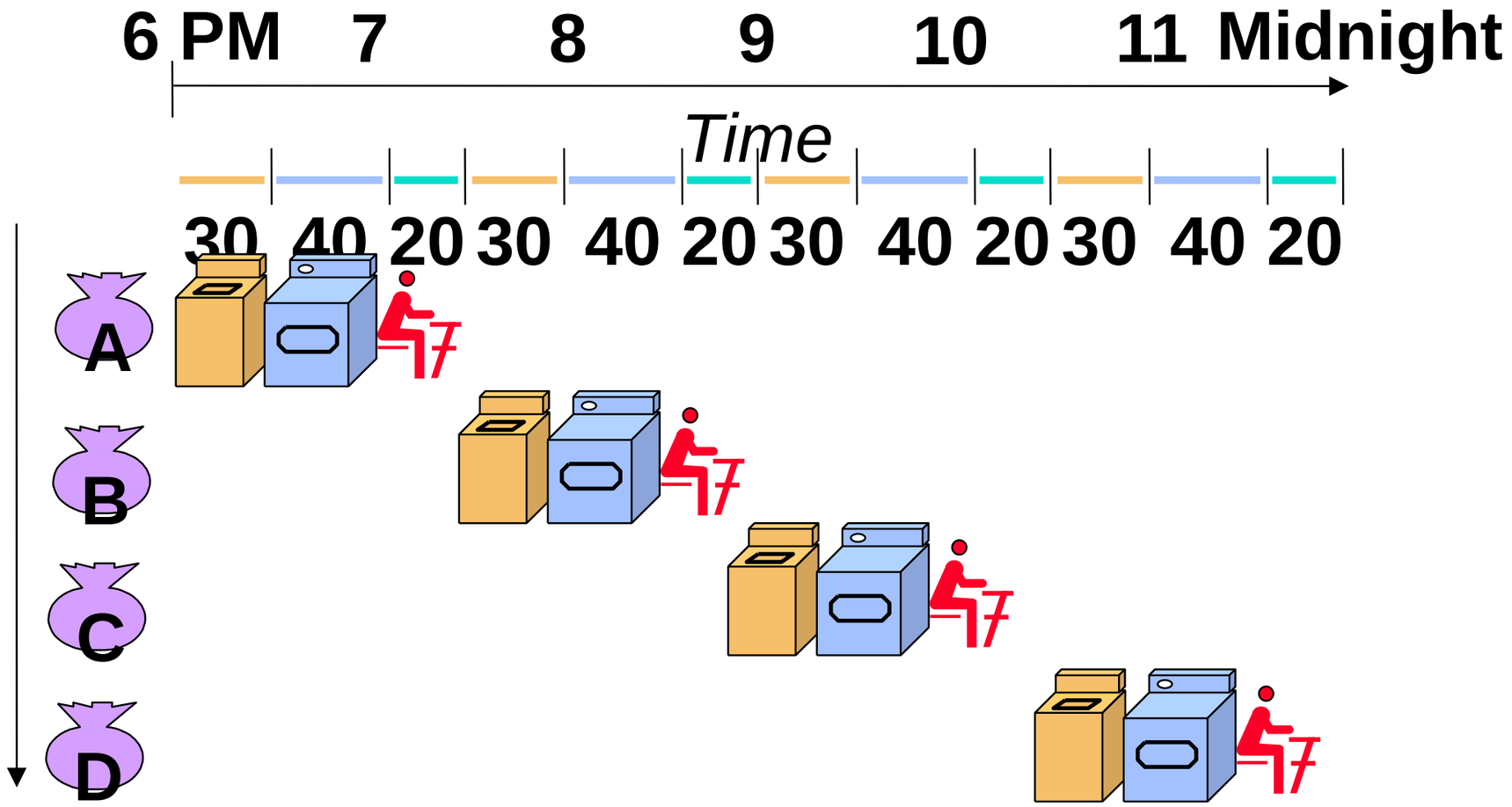
Traditional Pipeline Concept

- Laundry Example
- Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, and fold
- Washer takes 30 minutes
- Dryer takes 40 minutes
- “Folder” takes 20 minutes

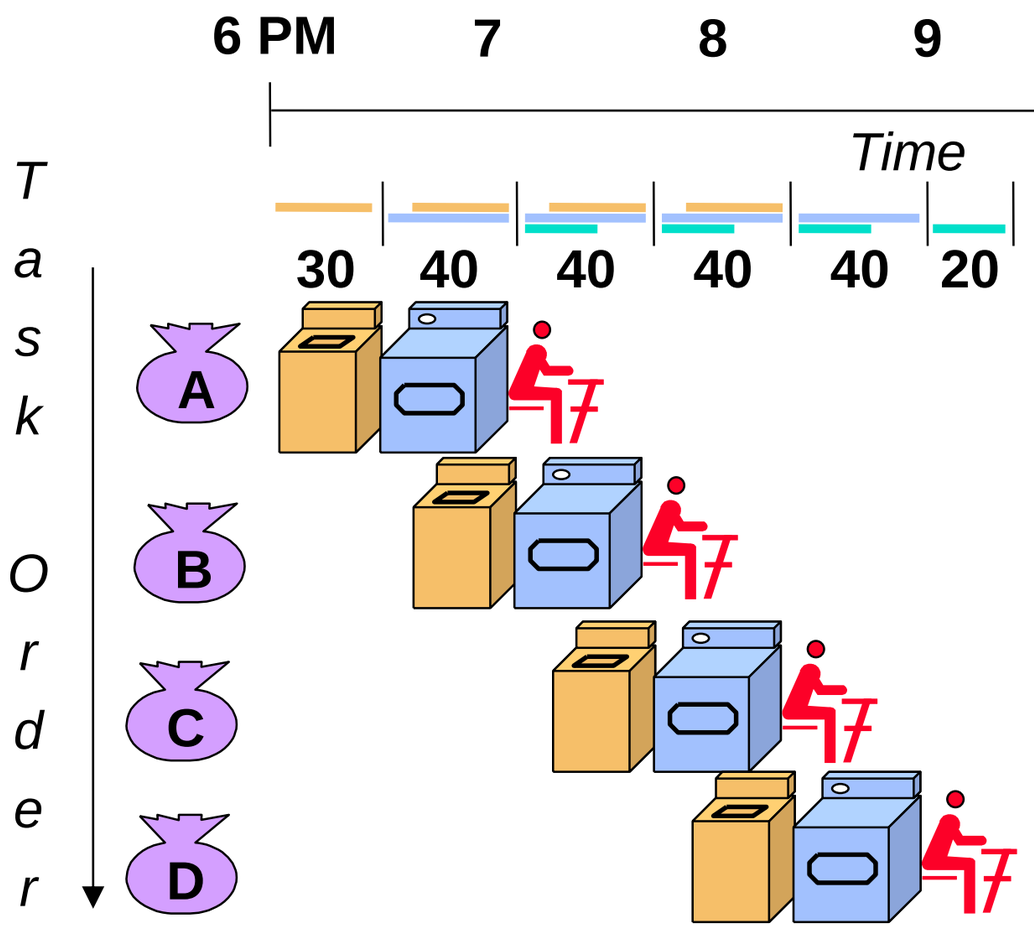


Traditional Pipeline Concept

- Sequential laundry takes 6 hours for 4 loads
- If they learned pipelining, how long would laundry take?

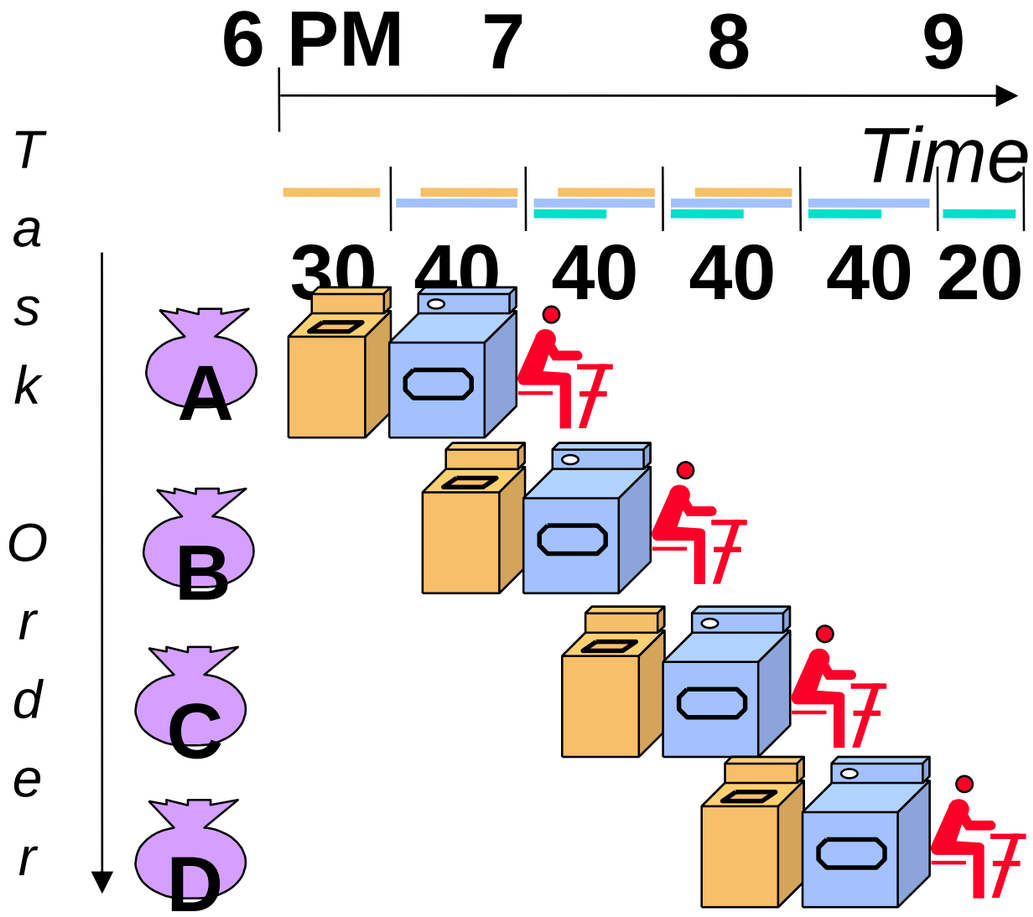


Traditional Pipeline Concept



- Pipelined laundry takes 3.5 hours for 4 loads

Traditional Pipeline Concept



- Pipelining doesn't help latency of single task, it helps throughput of entire workload
- Pipeline rate limited by slowest pipeline stage
- Multiple tasks operating simultaneously using different resources
- Potential speedup = Number pipe stages
- Unbalanced lengths of pipe stages reduces speedup
- Time to "fill" pipeline and time to "drain" it reduces speedup
- Stall for Dependences

Pipelining

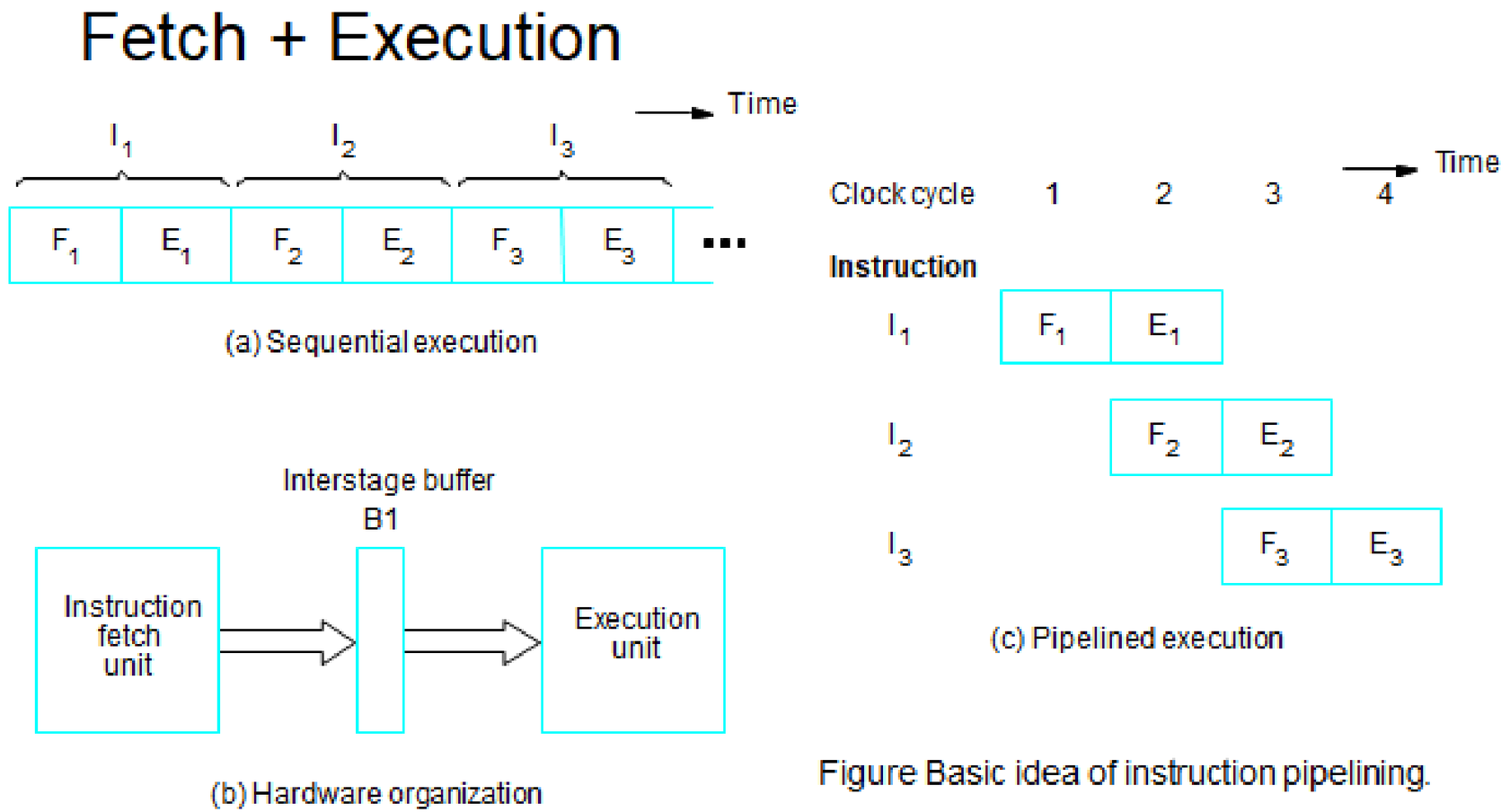
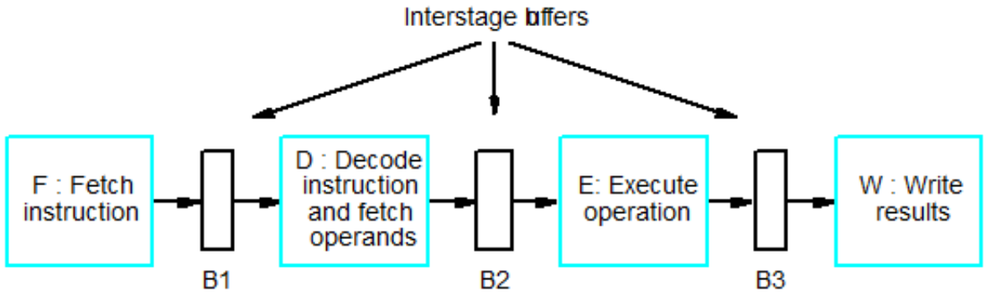


Figure Basic idea of instruction pipelining.

Use the Idea of Pipelining in a Computer

Mikroişlemci İşlevsel Döngü: Fetch + Decode+ Execution + Write/Read

(a) Instruction execution divided into four steps



(b) Hardware organization

Mikroişlemci işlevsel göngüde, komutları ardışık ve eş zamanlı yürüterek performansı artırmak hedeflenir.
Fetch: Bellekten komut ya da veri alır getirir. Veya veriyi belleğe götürür. (Adres Bus, Data Bus)
Decode: Belleği seçer. (Adres Bus)
Execution: Komut işler (CPU içi).
Write: Komut ya da veri yazar. (Control ve Adres Bus)
Read: Komut ya da veri okur. (Control ve Adres Bus)

Örnek: Pipelining

Without pipelining = 9/3 minutes = 3m

```
I F S | | | | | |
| | | I F S | | |
| | | | | | I F S (9 minutes)
```

With pipelining = 5/3 minutes = 1.67m

```
I F S | |
| I F S |
| | I F S (5 minutes)
```

Böylelikle boru hatlı çalışma, bir sistemin performansını arttırır.

Örnek: Pipelining

- Mikroişlemci işlevsen döngülerin işlem süreleri clock peryodu (T) süresi olarak verilmiştir. 4 komutluk işlevin süresini Pipelining olmadan, pipeling olarak işem sürelerini hesaplayınız.
- Fetch: 1T
- Decode: 1T
- Execution: 2T
- Write: 1T

Pipelining olmadan 4 komut işlev süresi=4* (1+1+2+1)=20T

Pipeling olduğunda: 4 komut işlev süresi=11T

1	2	3	4	5	6	7	8	9	10	11
0	F	0	D	E	E	0	W			
	0	F	0	D	E	E	0	W		
		0	F	0	D	E	E	0	W	
			0	F	0	D	E	E	0	W

Örnek: Pipelining

- Mikroişlemci işlevsen döngülerin işlem süreleri clock peryodu (T) süresi olarak verilmiştir. Bellek gözü seçilecek, Veri bellekten okunacak, Saklayıcılarda toplama işlemi yapılacak, bellek gözü seçilecek, sonuç seçilen bellek gözüne yazılacaktır. Komutların 4 defa döngüsel işlevin süresini Pipelining olmadan, pipeling olarak hesaplayınız.
- Bellek gözü seçilmesi, D1=1T
- Read, F1= 1T
- Toplama, Execution, E: 2T
- Bellek gözü seçilmesi, D2=1T
- Write, F2=1T

Bir döngüsel işlev süresi=D1+F1+E+D2+F2=1T+1T+2T+1T+1T=6T

Pipelining olmadan: Dört döngüsel işlev süresi=4*6T=24T

Pipelining olduğunda: Dört döngüsel işlev süresi=12T

Yorumlama: %50 perormans artışı olduğu görülmektedir.

1	1	1	1	1	1	1	1	1	1	1	1	12
D1	F1	E1	E1	D2	F2							
	D1	F1	x	E1	E1	D2	F2					
		D1	F1	x	x	E1	E1	D2	F2			
			D1	F1	x	x	x	E1	E1	D2	F2	

Sources of Pipeline Stalls

Three types of hazards may prevent an instruction from executing during its designated clock cycle.

- Structural hazard: Two instructions attempt to use the same functional unit at the same time.
- Data hazard: An instruction depends on the result of a prior instruction in the pipeline.
- Control hazard: Fetching and decoding the next instruction to execute is delayed by a decision about control flow (i.e., a conditional jump).

Pipelining: Ardışık düzende komut işleme

- Pipelining, CPU'nun donanım öğelerinin genel performansı artırılacak şekilde düzenlenmesi sürecidir. Birden fazla komutun eşzamanlı olarak yürütülmesi, ardışık düzenlenmiş bir işlemcide gerçekleşir.

- Boru hatlı operasyon kavramı üzerinde çalışan gerçek hayattan bir örnek görelim. Bir su şişesi paketleme fabrikasını düşünün. Bir şişenin geçmesi gereken 3 aşama olmasına izin verin, Şişeyi yerleştirme (I), Şişeye su doldurma (F) ve Şişeyi kapatma (S). Bu aşamaları sırasıyla 1. aşama, 2. aşama ve 3. aşama olarak ele alalım. Her aşamanın çalışmasını tamamlamak için 1 dakika ayırmasına izin verin.
- Şimdi, boru hatsız bir işlemde, tesise önce bir şişe yerleştirilir, 1 dakika sonra suyun dolu olduğu 2. aşamaya taşınır. Şimdi, 1. aşamada hiçbir şey olmuyor. Benzer şekilde, şişe 3. aşamaya geçtiğinde, hem aşama 1 hem de aşama 2 boşta kalır. Ancak boru hatlı çalışmada, şişe 2. aşamadayken, 1. aşamada başka bir şişe yüklenebilir. Benzer şekilde, şişe 3. aşamadayken 1. ve 2. aşamada birer şişe olabilir. Yani her dakika sonra 3. aşamanın sonunda yeni bir şişe alıyoruz.

Pipelining

Without pipelining = 9/3 minutes = 3m

```
I F S | | | | | |
| | | I F S | | |
| | | | | | I F S (9 minutes)
```

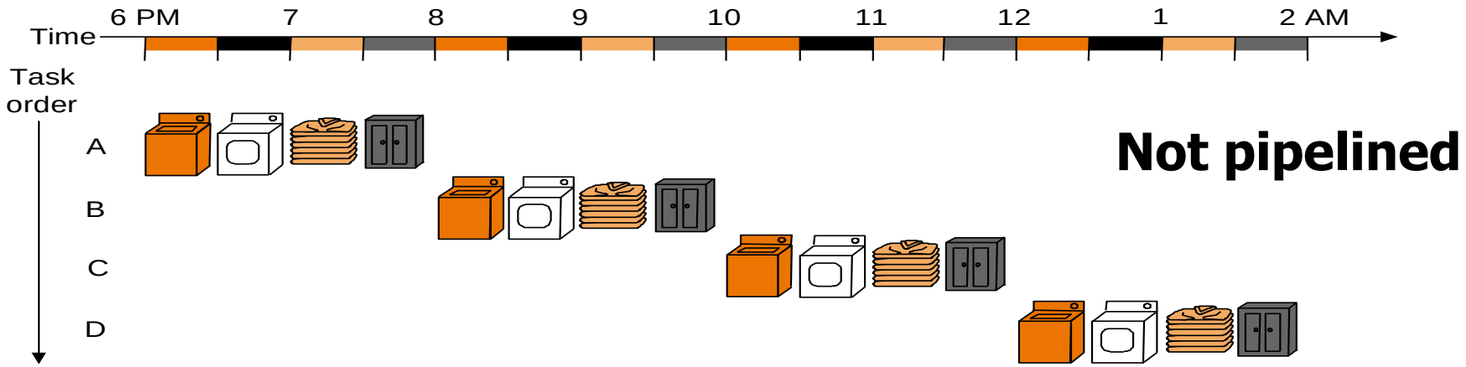
With pipelining = 5/3 minutes = 1.67m

```
I F S | |
| I F S |
| | I F S (5 minutes)
```

Böylelikle boru hatlı çalışma, bir sistemin verimini arttırır.

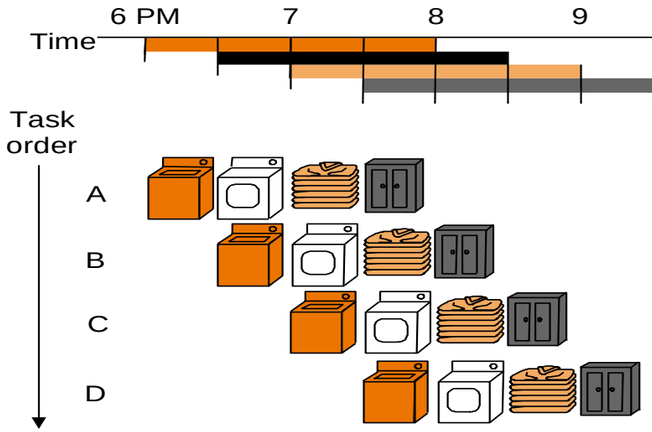
Pipelining

- Bir işe başladıktan sonra en kısa sürede bitir.!! Zaman kaybetme!



Not pipelined

Ayrı görevler için ayrı donanımlar kullanılıyorsa, farklı görevler aynı zamanda üstüste çakışabilir.

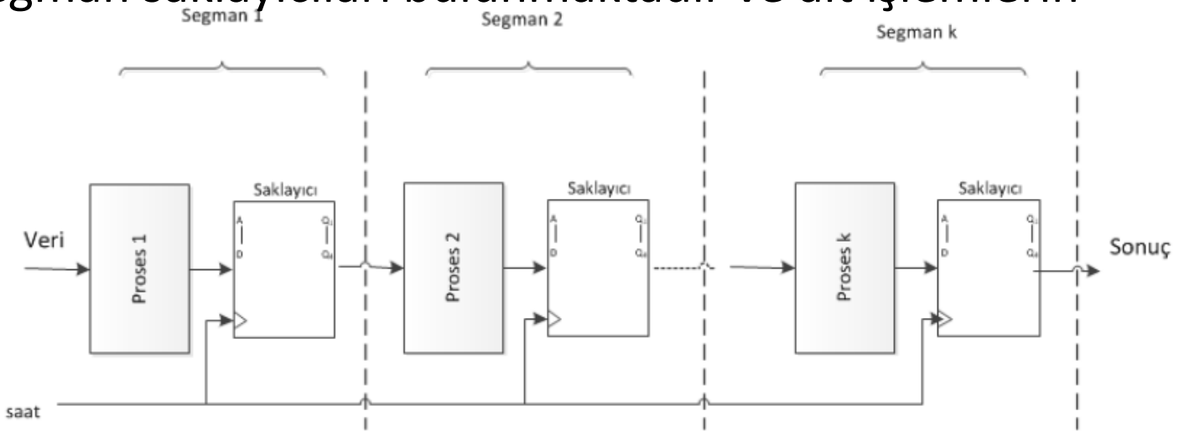


Pipelined

Bilgisayar organizasyonu açısından bir kontrol ünitesini gerçeklemenin 2 farklı yolu pipelined, not pipelined. Avantaj/ dezavantajları nelerdir?

İş Hattı (Pipelineardışık düzende komut işleme)

- İş hattı, büyük işlerin her biri paralel olarak çalışabilen küçük alt parçalara bölünmesi ile oluşturulan bir veri işleme yöntemidir.
- İş hattına örnek olarak bir otomobil fabrikasının işleyişini gösterebiliriz. Bir otomobilin bütünüyle baştan sona üretilmesi uzun bir süreç olarak gözükmesine rağmen yürüyen üretim hattı ile bu süreç hızlandırılmaktadır. Çünkü otomobilin her bir parçasının montajı ayrı robotlar tarafından paralel olarak yapılmaktadır. Örneğin bir otomobilin kapılarının montajı yapılırken bir yandan da başka bir otomobilin lastikleri monte edilebilmektedir.
- İşlemcilerde de aynı prosedür uygulanarak işlemcinin hızının artırılması amaçlanmaktadır.
- İş hattının her bir bölümüne segman adı verilmektedir ve her segmanda alt parçalara bölünmüş olan işlemler yürütülmektedir. Her segmanın sonunda segman saklayıcıları bulunmaktadır ve alt işlemlerin sonuçları bu saklayıcılara yazılmaktadır.
- Genel bir iş hattı yapısı Şekil de verilmiştir.



İş hattı Yapısı

Pipeline Boru Hattı Mimarisi

- Zamanı kısaltarak toplam işi arttıran, toplu komutların daha hızlı çalışmasını sağlayan günümüzde daha hızlı işlemci tasarımında kullanılan önemli bir yöntemdir.
- Komutlar saat vuruşlarıyla yürütülür.
- Her saat vuruşunda bir komut girer, bir komut çıkar.
- Boru hattı, işlemcinin komut döngü süresini azaltır bundan dolayı da birim döngü zamanına düşen komut sayısı artar. Tek bir komutun işini değil toplu komutların işlerini hızlandırır. Asıl amacı saat sıklığını artırarak başarımı artırmaktır.
- İşlemciler, mantık ve flip-floplardan oluşmaktadır. Saat vuruşu geldiğinde, flip-floplar yeni değerlerini alırlar sonra kodları çözüp yapmaları gereken şeyi yapmaları için belli bir süre gerekir. Bir sonraki saat vuruşu geldiğinde, flip-floplar tekrar kendi değerlerini alırlar ve bu böyle devam eder.
- Mantık yürütülmesini ufak bölümlere bölüp aralarına flip-floplar koyarsak, veri çıkışımızdaki gecikme azalmış olur. Saat vuruş zamanını azalma nedeni de budur.

4 segmanlı bir iş hattı

Bir iş hattında, belirli bir süre zarfında hangi segmanda hangi işlerin yürütüldüğünü göstermek için uzay-zaman diyagramları kullanılır. 4 segmanlı bir iş hattının uzay zaman diyagramı Şekilde gösterilmiştir. Herbir iş 4 parçaya bölünmüştür. T1 işi 4 saat darbesinde tamamlanmaktadır.

Segman \ Saat Darbesi	1	2	3	4	5	6	7
1	T1	T2	T3	T4	T5	T6	
2		T1	T2	T3	T4	T5	T6
3			T1	T2	T3	T4	T5
4				T1	T2	T3	T4

4 Segmanlı Bir İş Hattının Uzay-Zaman Diyagramı

Segmanlı Bir İş Hattının Uzay-Zaman Diyagramı

- Verilen uzay-zaman diyagramında, yapılacak işler T ile gösterilmiştir. Görüldüğü üzere T1 işi ikinci saat darbesinde ikinci segmana geçtiğinde, birinci segmanda T2 işinin yürütülmesine başlanmaktadır.
- Her bir iş 4 saat darbesinde tamamlanmaktadır fakat iş hattı sayesinde dördüncü saat darbesinden sonra her saat darbesinde bir iş tamamlanabilmektedir. Ancak iş hattının bu paralel veri işleyen yapısı bazı sıkıntıları da beraberinde getirmektedir. Bir iş hattı yapısında dallanma buyruğu icra edilirken, iş hattına bu komuttan sonra gelen komutlar alınmış olacaktır.
- Dallanma gerçekleşmesi beklendiğinden iş hattına giren bu komutların gerçekleştirilmesi istenmez ve iş hattının boşaltılması gerekir. Bu duruma **dallanma cezası (branch penalty)** adı verilir.
- Bir diğer problem ise aynı veri üzerinde işlem yapacak olan komutların aynı anda iş hattı üzerinde bulunmasıdır. Farklı segmanlarda bulunan komutlar veriyi değiştirmesi hatalı işlem yapılmasına yol açabilmektedir. Bu probleme de **veri bağımlılığı (data dependency)** adı verilmektedir.

Boru hattı mimarisi işlem aşamaları

Boru hattı kullanımında komutlar arası aşama farklılıkları görülmektedir.

- Getir: Komutlar, komut belleklerinden getirilir.
- Oku/Çöz: İşlenenler okunur ve komut belleklerinden getirilen komutlar çözülür.
- Yürüt: Bellek adresleri hesaplanır.
- Bellek: Veriler, veri belleklerinden okunur.
- Yaz: Elde edilen sonuçlar yazmaç (register) öbeğine yazılır.

Recall the 5 steps in instruction execution:
1. Instruction Fetch (IF)
2. Instruction Decode and Register Read (ID)
3.Execution operation or calculate address (EX)
4.Memory access (MEM)
5.Write result into register (WB)

• Review: Single-Cycle Processor
– All 5 steps done in a single clock cycle
– Dedicated hardware required for each step

Boru Hattı Mimarisinin Hızını Etkileyen Faktörler

- Boru hattında aşama sayısının fazlalılığı hızı azaltır.
- Boru hattında aşama sayısı uzunluğunun sabit veya değişken olması hızı etkiler.
- Komut bağımlılıkları hızı etkiler.
- Hattın dolma/boşalma süreleri hızı etkiler.

Boru Hattı Mimarisinin Avantajları ve Dezavantajları

Boru hattı mimarisinin avantajları:

- İşlemcinin döngü süresi azaltılır, böylece komut genişliğini birçok durumda arttırır.
- Toplu komut işlemlerinde çevrim zamanı azalır. Bu durumda yürütme zamanı azaltılmış olur. Bu da başarımı artırır.
- Kullanılmayan kaynakların boş kalmasını engeller. Kaynakların verimli kullanılmasını sağlar. Sistemin verimini artırır.
- Sistemi güvenilir kılar.

Boru hattı mimarisinin dezavantajları:

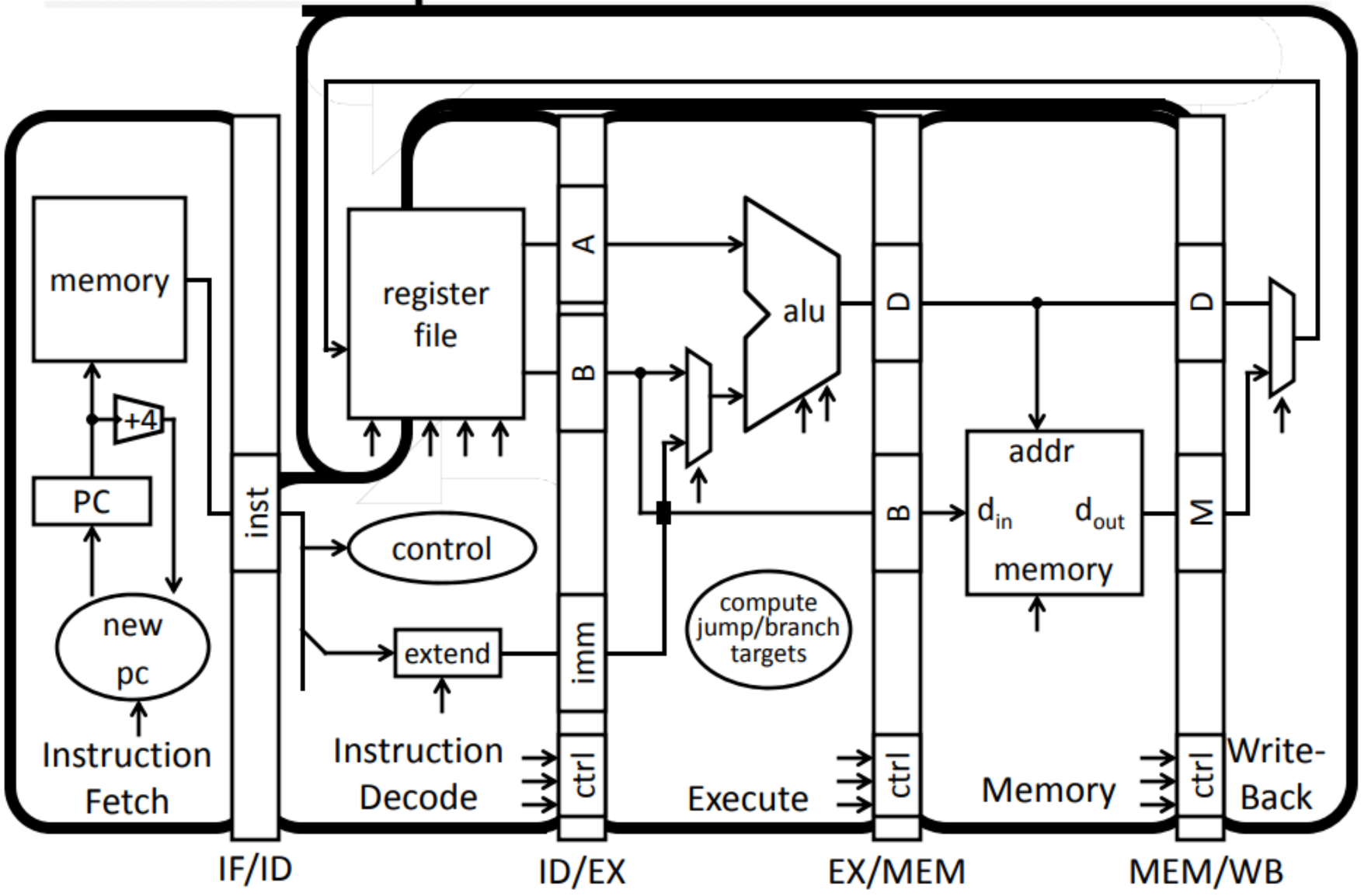
- Boru hatlı işlemcinin tasarımı karmaşıktır ve üretimi maliyetlidir. komut gecikmesi daha fazladır.
- **Yapı Sorunları :** Aynı kaynak birden fazla kez kullanılmak istenirse bu sorun yaşanır. Sorun bekleyerek çözülebilir.
- **Denetim Sorunları :** Dallanma komutlarıyla ilgilidir. Koşullar belli olmadan karar verilirse gerçekleşir. Çözüm olarak dallanma kararı verilmeden önce yürütüm durdurulur ve karar belli olana kadar beklenir. Bir başka çözüm yolu da; kararı tahmin edip ona göre işlemi devam ettirmek, eğer karar hatalıysa geri dönüp tekrar denemektir.
- **Veri Sorunları :** İşi bitmeyen verinin kullanılmak istenmesiyle bu sorun oluşur. İşlemin durdurulup beklenmesi ile çözümlenebilir.

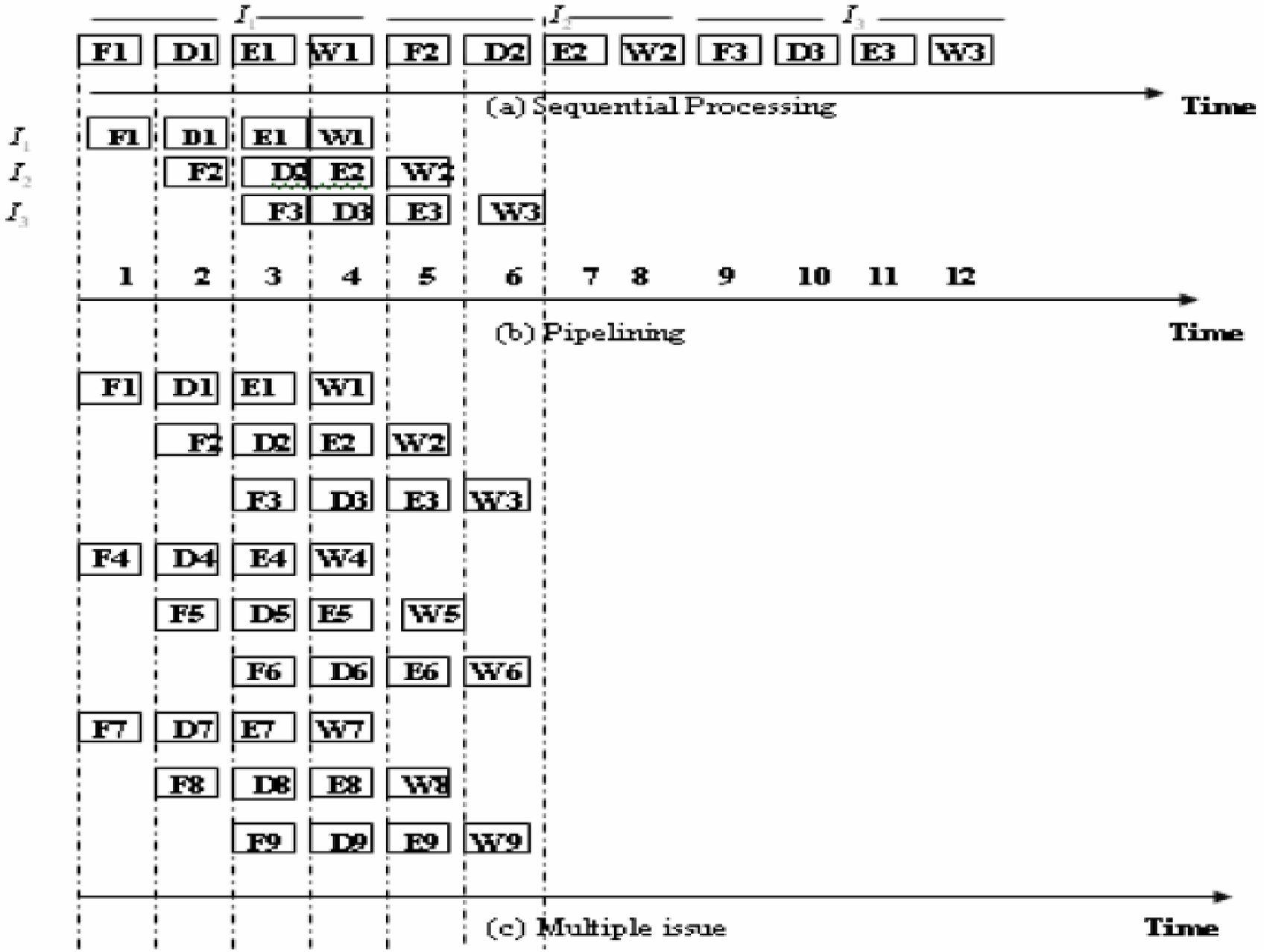
Paralel İşleme ve Pipeline

- Aynı zamanda, birden fazla veya çok sayıda işlemin yapılabilmesi bilgisayar sisteminin hızını (Throuhput) arttırır.
- Paralel işlemede ise aynı anda birden fazla komut icra edilir. Örneğin, bir zaman diliminde bir komut ALU'da yürütülürken (execute), aynı zaman diliminde daha sonraki komut ise bellekten okunabilir. Bu işlem tek işlemcili paralel çalışmaya örnektir.
- Paralel çalışmaya diğer örnek , birden fazla işlemcisi bulunan sistemlerdir.
- Paralel işleme aşağıdaki başlıklarda incelenebilir.
- 1- Pipeline (Boru Hattı – İş Hattı) İşlemleri
- 2-Vektör İşlemleri
- 3-Dizi İşlemcileri

  Pipeline bir yürütme tekniği olup, artmetik alt işlemlerde veya bir komutun fazlarının (evrelerinin) işlenmesi sırasında üstüste gelmesidir. Vektör işlemleri büyük boyutlu vektörel veya matris işlemlerinin yapılması içindir. Dizi İşlemcileri ise büyük çaplı diziler üzerinde işlem yaparlar.

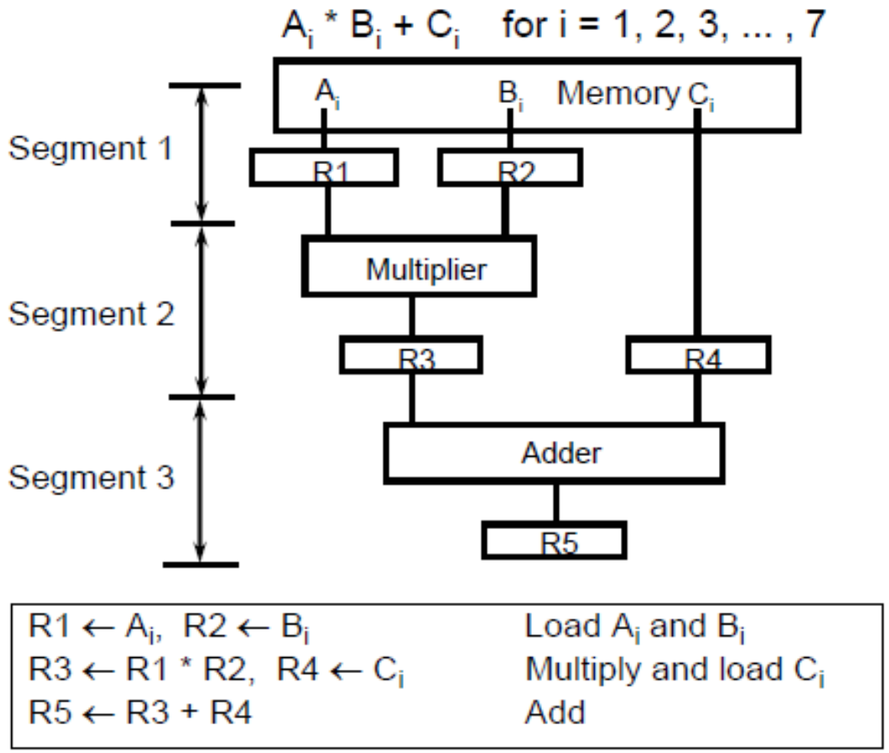
Pipelined Processor




I_1
I_2
I_3
F1 D1 E1 W1 F2 D2 E2 W2 F3 D3 E3 W3
(a) Sequential Processing
Time
1 2 3 4 5 6 7 8 9 10 11 12
(b) Pipelining
Time
F4 D4 E4 W4 F5 D5 E5 W5 F6 D6 E6 W6 F7 D7 E7 W7 F8 D8 E8 W8 F9 D9 E9 W9
(c) Multiple issue
Time

Pipeline İşlemlerinin Anlaşılması

- Pipeline İşlemenin 2 temel uygulaması vardır: 1- Aritmetik işlemlerde pipeline, 2-Komut Yürütmede pipeline
- Pipeline işlemi birtakım işlemlerin topluluğudur ve boru hattını oluşturan her bir kesimde (segmentte) herbir farklı işin farklı alt işlemleri aynı zamanda gerçekleşir. Bir kesimde elde edilen sonuç aynı iş için sonraki kesimde kullanılmak üzere saklanır.



Her bir kesimde icra edilecek alt işlemler şekilde verilmiştir. 5 Registerden herbiri, her clock cyce'lında yeni veriler ile yüklenir.

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A1	B1			
2	A2	B2	A1 * B1	C1	
3	A3	B3	A2 * B2	C2	A1 * B1 + C1
4	A4	B4	A3 * B3	C3	A2 * B2 + C2
5	A5	B5	A4 * B4	C4	A3 * B3 + C3
6	A6	B6	A5 * B5	C5	A4 * B4 + C4
7	A7	B7	A6 * B6	C6	A5 * B5 + C5
8			A7 * B7	C7	A6 * B6 + C6
9					A7 * B7 + C7

Reg'lerin içeriklerinin değişimi

PİPELİNE işlemenin hızı (Kazancımız ne?)

n: Görev Sayısı (Bir assemler programdaki işlenecek komut sayısı diyebiliriz.)

PİPELİNE çalışmayan bir makine için;

tn: Clock cycle (her bir görevi bitirmek için gerekli zaman)

t1: n tane görevi tamamlamak için gerekli harcanacak zaman.

→ **t1 = n * tn**

PİPELİNE bir makine için. (k kesimli (durumlu))

tp: Clock cycle (Herbir alt işlemin tamamlanması için gerekli zaman)

tk: n adet görevin başarılması için gerekli zaman

→ **tk = (k + n - 1) * tp**

- **Hızlanma (Speedup)**
- **Sk:** Speedup

→ **Sk = t1 / tk = n*tn / (k + n - 1)*tp**

n görev sayısı arttıkça (k + n -1) terimi yaklaşık n olur. Buna göre aşağıdaki bağıntı ortaya çıkar.

Sk= tn / tp

Eğer; pipeline ve pipeline olmayan bilgisayarlarda bir görevin tamamlanması için geçen zaman eşit ise, Bu durumda; pipeline toplam hızlandırması k (kesim sayısı) kadar olabilir.

$$\lim_{n \to \infty} S_k = \frac{t_n}{t_p} \quad (= k, \text{ if } t_n = k * t_p)$$

Örnek:

4-kesimli pipeline'da, bir alt işlemin her bir kesimde işlenebilmesi için gerekli zaman;

tp = 20nS olsun.

İcra edilecek görev sayısı 100 olsun. Sk hızlanma değerini hesaplayınız.

Çözüm:

- Pipeline olmayan bir sistemde, tekbir görev için harcanacak zaman 20*4 = 80nS
  Tüm görevin icra edilmesi için harcanacak zaman: 100*20*4 =8000 ns.

- Pipeline makine için toplam süre ; (k + n - 1)*tp =(4+100 -1)*20=2060 ns

- Buna göre;

$$Sk = 8000 / 2060 = 3.88$$

Elde edilir.

Önemli not: Bütün alt işlemlerin icra sürelerinin aynı olduğunu kabul ediyoruz.

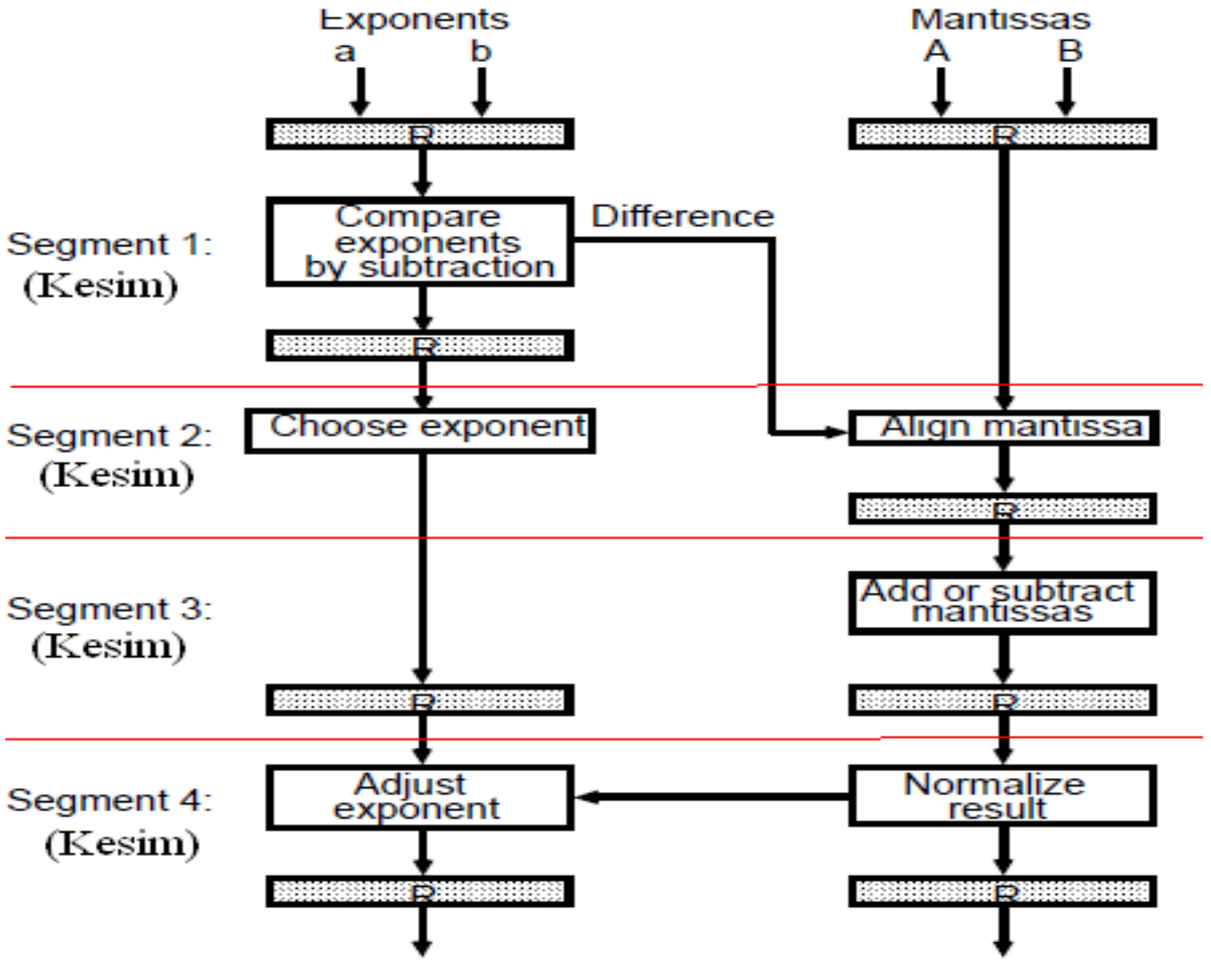
Aritmetik İşlemlerde PİPELİNE

Pipeline aritmetik birimleri, FP işlemler için ve çarpma işlemleri için önemli bir hızlandırıcı yapıdır.

Floating-point Toplama

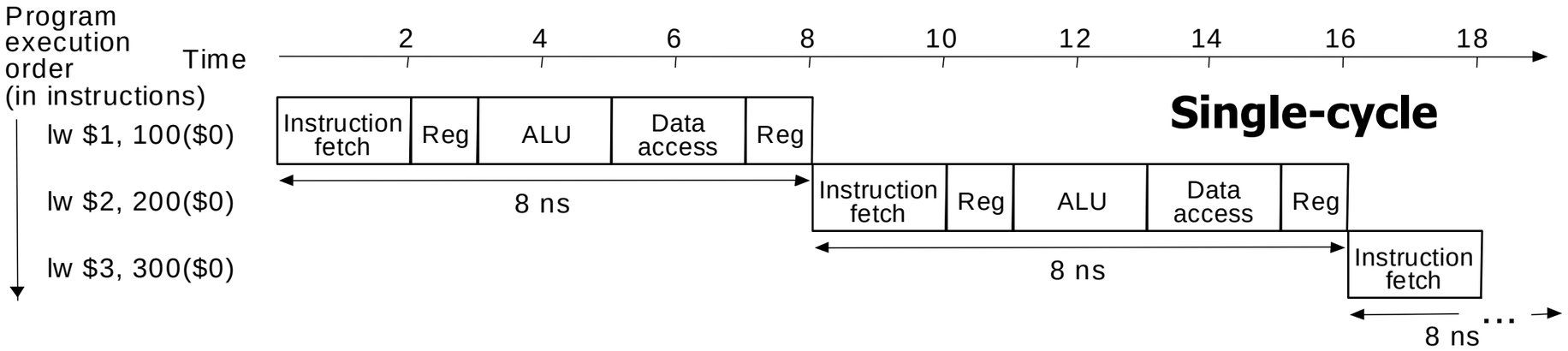
$X = A \times 2^a$
$Y = B \times 2^b$

[1] Exponentlerin karşılaştırılması
[2] Mantisanın hizalanması
[3] Mantisaların toplanması/Çıkarılması
[4] Sonuç Normalizasyonu

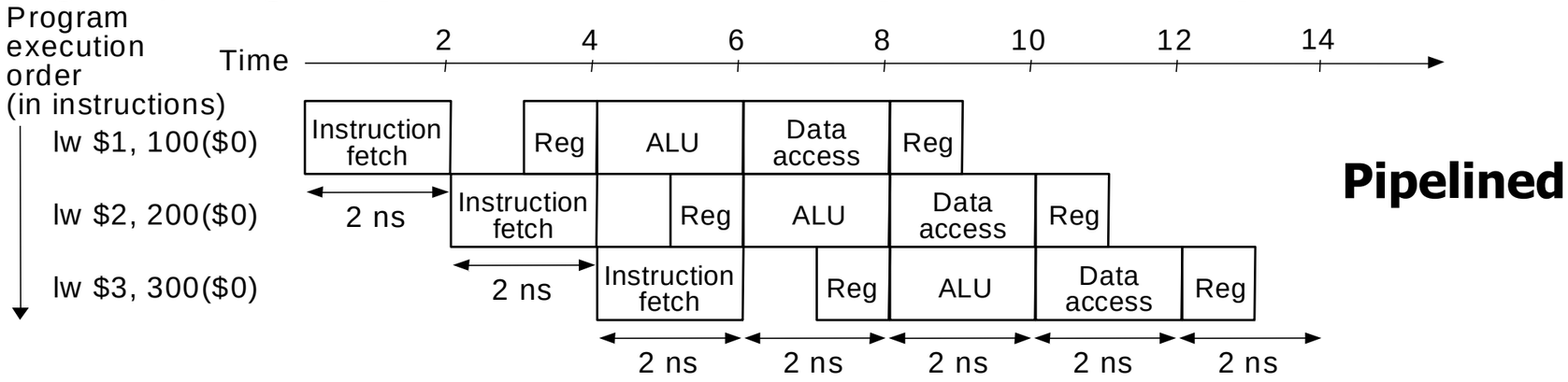


Komut Yürütme için Pipeline

Single-Cycle Komut Yürütme: Planlama



Farzedin ki memorye erişim ve ALU işlemleri 2 ns ; register erişimi 1 ns. Buna göre, single cycle clock 8 ns'dir. Pipelined clock cycle ise 2 ns.



Pipelining: Unutma

- Pipeline işlemi tek bir görev (veya komut işlemesi) için harcanacak zamanı (latency) azaltmaz fakat görevin bütünü (bir programın sıralı komutları v.b) için harcanan zamanı azaltır (Verilen sabit bir zamanda yapılan iş miktarını –throughput- arttırır).

- Pipeline hızı en uzun durum ile sınırlıdır.
  - *potential* hızlandırıcı= Boru durumları sayısıdır (kesim sayısı).
  - *Dengesiz boru durumları hızı azaltır (her kesim için farklı süreler).*
- Pipeline sürecinde herhangi bir fazda boş kalmak pipeline'ı yavaşlatır.
- En verimli pipeline işlemi, komutların parçalanmasıyla oluşan alt işlem sürelerinin (fetch, decode v.b) eşit sürelerde işlenmesi ile olur.

Pipeline Hazards

- Denetim Sorunları (Control Hazards)
  - Dallanma ve Kesmeler (Branches, Interrupts)
    - Koşulsuz Dallanma (Unconditional Branch)
    - Koşullu Dallanma (Conditional Branch); Doğru-Dallanma olur, Yanlış – Dallanma Olmaz
- Kaynak Çatışması (Resource Conflict), Yapısal Sorun (Structural Hazard)
  - Bellek Çatışması
  - İşlem Birimi (ALU, FPU) Çatışması
- Veri Çatışması (Data Conflict), Veri Bağımlılığı (Data Dependency)
  - Operand Bağımlılığı
  - Adres Bağımlılığı

Pipeline Hazards

- komut akışındaki bir sonraki komutun belirlenen saat döngüsü sırasında yürütülmesini engelleyen tehlikeler adı verilen durumlar vardır. Tehlikeler, boru hattının sağladığı ideal hız artışından performansı düşürür.

Üç tehlike sınıfı vardır:

- **Yapısal Tehlikeler:** Donanım eşzamanlı örtüşen yürütmede olası tüm komut kombinasyonlarını destekleyemediğinde kaynak çakışmalarından kaynaklanırlar.
- **Veri Tehlikeleri:** Bir komut, boru hattındaki komutların üst üste binmesiyle ortaya çıkacak şekilde önceki bir komutun sonucuna bağlı olduğunda ortaya çıkarlar.
- **Kontrol Tehlikeleri:** Dalların boru hattından ve bilgisayarı değiştiren diğer komutlardan kaynaklanır.

Boru hatlarındaki tehlikeler, boru hattının durmasını gerekli kılabilir. İşlemci farklı olaylarda durabilir:

- Bir önbellek eksik. Bir önbellek, yanlışa neden olan komuttan önce ve sonra ardışık düzendeki tüm komutları durdurur.
- Boru hattında bir tehlike. Bir tehlikenin ortadan kaldırılması, genellikle boru hattındaki bazı komutların devam etmesine izin verilmesini gerektirirken diğerleri geciktirilir. Komut durdurulduğunda, durdurulan komuttan sonra verilen tüm komutlar da durur. Durdurulan komuttan daha önce verilen komutla devam etmelidir, aksi takdirde tehlike asla ortadan kalkmayacaktır.

Yapısal tehlikelerin çözülmesi

1.Çözüm: Bekleyin

- Tehlikeyi tespit etmeli
- Durdurma mekanizması olmalı
- Düşük maliyetli ve basit
- TÜFE'yi artırır
- Nadir durumlarda kullanılır

2.Çözüm: Soruna daha fazla donanım atın

- Boru hattı donanım kaynağı, çok döngülü kaynaklar için yararlıdır. iyi performans, Bazen karmaşık, örneğin RAM
- Kaynağı çoğalt: iyi performans. maliyeti artırır (+ belki ara bağlantı gecikmesi). Ucuz veya bölünebilir kaynaklar için kullanışlıdır

Structural dependency

- Bu bağımlılık, boru hattındaki kaynak çakışması nedeniyle ortaya çıkar. Kaynak çakışması, aynı döngüde birden fazla komutun aynı kaynağa erişmeye çalıştığı bir durumdur. Bir kaynak bir kayıt, bellek veya ALU olabilir.
- Yukarıdaki senaryoda, döngü 4'te, komutlar I1 ve I4, bir kaynak çakışmasına neden olan aynı kaynağa (Hafıza) erişmeye çalışmaktadır.
- Bu sorunu önlemek için, gerekli kaynak (bizim durumumuzda bellek) kullanılabilir olana kadar komutu beklemeye almalıyız.
- Bu bekleme, aşağıda gösterildiği gibi boru hattında duraklamalara neden olacaktır.

Instruction / Cycle	1	2	3	4	5
I_1	IF(Mem)	ID	EX	Mem	
I_2		IF(Mem)	ID	EX	
I_3			IF(Mem)	ID	EX
I_4				IF(Mem)	ID

Cycle	1	2	3	4	5	6	7	8
I_1	IF(Mem)	ID	EX	Mem	WB			
I_2		IF(Mem)	ID	EX	Mem	WB		
I_3			IF(Mem)	ID	EX	Mem	WB	
I_4				–	–	–	IF(Mem)	

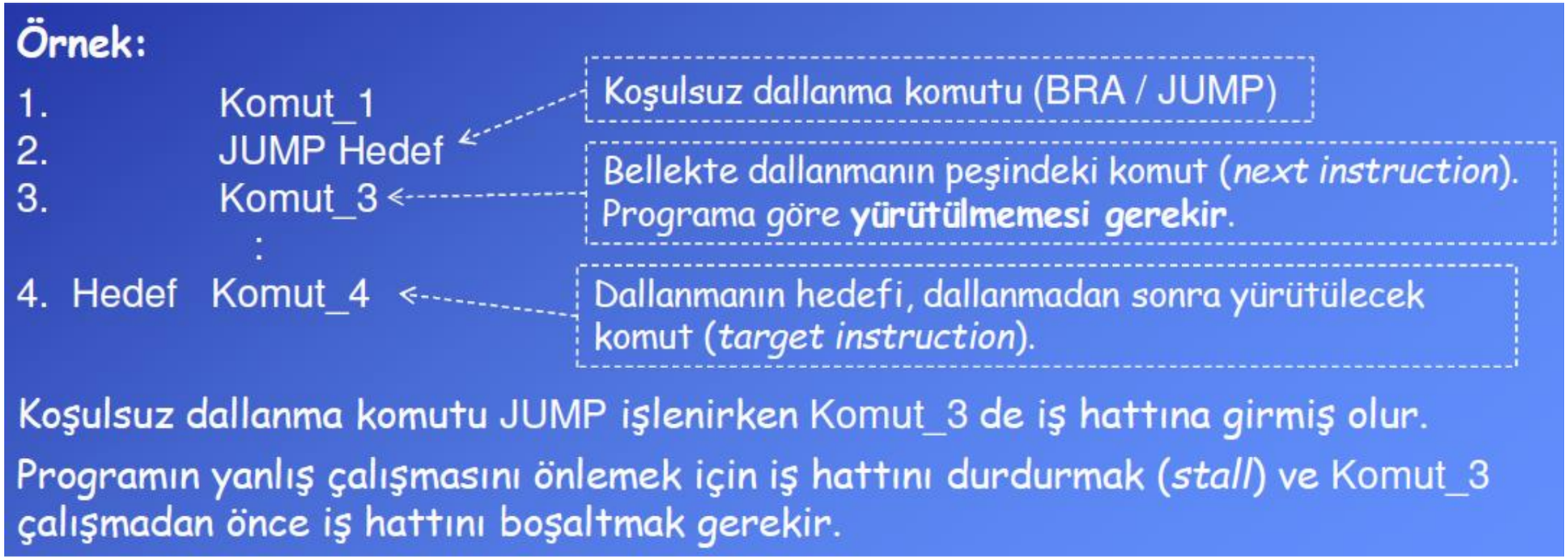
Solution for structural dependency

- İşlem hattındaki yapısal bağımlılık duraklamalarını en aza indirmek için, Yeniden Adlandırma adlı bir donanım mekanizması kullanıyoruz.
- Yeniden adlandırma: Yeniden adlandırmaya göre, belleği, sırasıyla Kod belleği (CM) ve Veri belleği (DM) olarak adlandırılan komutları ve verileri ayrı ayrı depolamak için kullanılan iki bağımsız modüle böleriz. CM tüm komutları içerecek ve DM komutlar için gerekli tüm işlenenleri içerecektir.

Instruction/ Cycle	1	2	3	4	5	6	7
I_1	IF(CM)	ID	EX	DM	WB		
I_2		IF(CM)	ID	EX	DM	WB	
I_3			IF(CM)	ID	EX	DM	WB
I_4				IF(CM)	ID	EX	DM
I_5					IF(CM)	ID	EX
I_6						IF(CM)	ID
I_7							IF(CM)

Dallanma ve Kesmeler (Branches, Interrupts)

- Pipeline'da komutlar paralel bir dallanma komutu işlenirken bellekte komuttan sonra gelen ancak dallanma nedeniyle yürütülmeyecek olan komutlarında pipeline'a alınmış olur. Önlem alınmaz ise programın mantığı gereği yürütülmemesi gereken komutlar da yürütülmüş olur.



Dört kesimli (Segmentli) Komut İçin Pipeline

Dört aşamaya (faz – Kesim-segment) bölünmüş komutlar için kesimler ;

FI: Komutu getir
DA: Komutu çöz, etkin adresi hesapla
FO: Veriyi getir
EX: Komutu yürüt

Not: komut ve data hafızalarına ayrı ayrı erişildiğini düşününüz.

Dallanma komutu olmadığı sürece, hiçbir kesim boş geçilmez. Farklı işlemler yapar. Ancak 3. komutun dallanma olduğu durumu inceleyelim. Neler olur ve neden olur? Niye 5.ve 6. adımlarda işlem yapılmaz?

Dallanma komutunun işleyip dallanılacak adres ortaya çıkıncaya kadar bir sonraki komut ile ilgili sadece Fetch işlemi yapılabilir. 5.6. adımlarda işlem yapılmaz.

	Step:	1	2	3	4	5	6	7	8	9	10	11	12	13
Komut	1	FI	DA	FO	EX									
	2		FI	DA	FO	EX								
(Branch) Dallanma	3			FI	DA	FO	EX							
	4				FI	–	–	FI	DA	FO	EX			
	5					–	–	–	FI	DA	FO	EX		
	6									FI	DA	FO	EX	
	7										FI	DA	FO	EX

Control Dependency (Branch Hazards)

- Bu tür bir bağımlılık, BRANCH, CALL, JMP, vb. Gibi kontrol komutlarının aktarımı sırasında meydana gelir. Birçok komut mimarisinde, işlemci, yeni komutu boru hattına eklemesi gerektiğinde bu komutların hedef adresini bilmeyecektir. Bundan dolayı, istenmeyen komutlar boru hattına beslenir. Programdaki aşağıdaki komut sırasını göz önünde bulundurun:
- NOT: Genellikle, JMP komutunun hedef adresi yalnızca Kimlik aşamasından sonra bilinir.

100: I_1

101: I_2 (JMP 250)

102: I_3

.

.

250: BI_1

Expected output: I_1 -> I_2 -> BI_1

Instruction/ Cycle	1	2	3	4	5	6
I_1	IF	ID	EX	MEM	WB	
I_2		IF	ID (PC:250)	EX	Mem	WB
I_3			IF	ID	EX	Mem
BI_1				IF	ID	EX

Output Sequence: I_1 -> I_2 -> I_3 -> BI_1

Control Dependency (Branch Hazards)

- Dolayısıyla, çıktı dizisi beklenen çıktıya eşit değildir, bu da boru hattının doğru şekilde uygulanmadığı anlamına gelir.
- Yukarıdaki sorunu düzeltmek için, dal komutunun hedef adresini alana kadar Yönerge alımını durdurmamız gerekir. Bu, hedef adresi alana kadar gecikme yuvası ekleyerek uygulanabilir.
- Gecikme yuvası hiçbir işlem yapmadığından, bu çıkış sırası beklenen çıktı sırasına eşittir. Ancak bu yuva, boru hattında durma sağlar.

Instruction/ Cycle	1	2	3	4	5	6
I_1	IF	ID	EX	MEM	WB	
I_2		IF	ID (PC:250)	EX	Mem	WB
Delay	–	–	–	–	–	–
BI_1				IF	ID	EX

Output Sequence: I_1 -> I_2 -> Delay (Stall) -> BI_1

Solution for Control dependency

- Dallanma Tahmini, kontrol bağımlılığından kaynaklanan duraklamaların ortadan kaldırılabileceği yöntemdir. Bunda 1. aşamada hangi dallanmaya alınacağına dair tahmin yapılır. Dallanma tahmini için Dal cezası sıfırdır.

- Dal cezası: Ardışık düzen işlemcideki dallanma işlemleri sırasında ortaya çıkan durak sayısı, dallanma cezası olarak bilinir.

- NOT: Hedef adresin ID aşamasından sonra mevcut olduğunu gördüğümüz için, boru hattına dahil edilen durak sayısı 1'dir. Farz edelim ki, dal hedef adresi ALU aşamasından sonra mevcut olurdu, 2 durak olurdu. Genel olarak, hedef adres k'inci aşamadan sonra mevcutsa, boru hattında (k - 1) duraklamalar olacaktır.

- Dal komutu nedeniyle boru hattına giren toplam durak sayısı = Dal sıklığı * Dal Cezası

Data Dependency (Data Hazard)

- Let us consider an ADD instruction S, such that
- S : ADD R1, R2, R3
- Addresses read by S = I(S) = {R2, R3}
- Addresses written by S = O(S) = {R1}
- Now, we say that instruction S2 depends in instruction S1, when
- $[I(S1) \cap O(S2)] \cup [O(S1) \cap I(S2)] \cup [O(S1) \cap O(S2)] \neq 0$
- Bu duruma Bernstein koşulu denir.

Üç durum vardır:

- Akış (veri) bağımlılığı: $O(S1) \cap I(S2)$, S1 → S2 ve S1, S2 tarafından okunan bir şeyden sonra yazar
- Bağımlılık karşıtı: $I(S1) \cap O(S2)$, S1 → S2 ve S1, S2 onun üzerine yazmadan önce bir şeyi okur
- Çıktı bağımlılığı: $O(S1) \cap O(S2)$, S1 → S2 ve her ikisi de aynı bellek konumunu yazar.

Örnek: I1 ve I2 olmak üzere iki komut olsun:

- I1 : ADD R1, R2, R3
- I2 : SUB R4, R1, R2
- Yukarıdaki komutlar ardışık düzenlenmiş bir işlemcide yürütüldüğünde, veri bağımlılığı koşulu ortaya çıkar, bu da I2'nin verileri I1 yazmadan önce okumaya çalıştığı anlamına gelir, bu nedenle I2, I1'den eski değeri hatalı olarak alır.
- İşlem hattındaki veri bağımlılığı duraklamalarını en aza indirmek için, Operand yönlendirme kullanılır.
- Operand yönlendirmede, ara çıktıyı tutmak için aşamalar arasında bulunan arayüz kayıtlarını kullanırız, böylece bağımlı komut, arayüz yazmacından yeni değere doğrudan erişebilir.

Instruction / Cycle	1	2	3	4
I_1	IF	ID	EX	DM
I_2		IF	ID(Old value)	EX

Instruction / Cycle	1	2	3	4
I_1	IF	ID	EX	DM
I_2		IF	ID	EX

Data Hazards

Veri bağımlılığı sergileyen komutlar, bir boru hattının farklı aşamalarındaki verileri değiştirdiğinde veri tehlikeleri ortaya çıkar. Tehlike, boru hattında gecikmelere neden olur. Temelde üç tür veri tehlikesi vardır:

1) RAW (Yazdıktan Sonra Oku) [Akış / Gerçek veri bağımlılığı]

2) WAR (Okuduktan Sonra Yaz) [Anti-Data bağımlılığı]

3) WAW (Yazdıktan Sonra Yaz) [Çıktı verisi bağımlılığı]

- J'nin I'i takip etmesi için I ve J olmak üzere iki komut olsun. Sonra,

RAW tehlikesi, J komutu, komut yazmadan önce verileri okumaya çalıştığında ortaya çıkar. Örneğin:

- I: R2 <- R1 + R3
- J: R4 <- R2 + R3

WAR tehlikesi, komut J komutunu okumadan önce verileri yazmaya çalıştığında ortaya çıkar. Örneğin:

- I: R2 <- R1 + R3
- J: R3 <- R4 + R5

WAW tehlikesi, komut J komutunu yazmadan önce çıktı yazmaya çalıştığında ortaya çıkıyor. Örneğin:

- I: R2 <- R1 + R3
- J: R2 <- R4 + R5
- komutların sıra dışı uygulanması sırasında WAR ve WAW tehlikeleri ortaya çıkar.

Kaynaklar

- Dandamudi S., 2003. Fundamentals of Computer Organization and Design, Springer-Verlag, Heidelberg
- Mano, M.M., 2007. Bilgisayar Sistemleri Mimarisi, Literatür Yayıncılık, İstanbul
- Buzluca F., “Bilgisayar Mimarisi Ders Notları”, Bilgisayar Mühendisliği Bölümü, İTÜ Bilgisayar ve Bilişim Fakültesi
- https://tr.wikipedia.org/wiki/Aritmetik_mant%C4%B1k_birimi
- CSE 675.02: Introduction to Computer Architecture, Slides by Gojko Babiç