BİLGİSAYAR MİMARİSİ

CISC Mimarisi

- Çok sayıda buyruk varsa ⇒ CISC
- Temel hedefi : Yüksek seviye dille yazılmış programın her bir komutunu tek buyruğa indirmek
- Farklı uzunluklardaki buyrukların kullanımına izin verir
- Çok sayıda adresleme kipi bulunur
- Buyruklar verileri bellekten kullanabilir

RISC Mimarisi

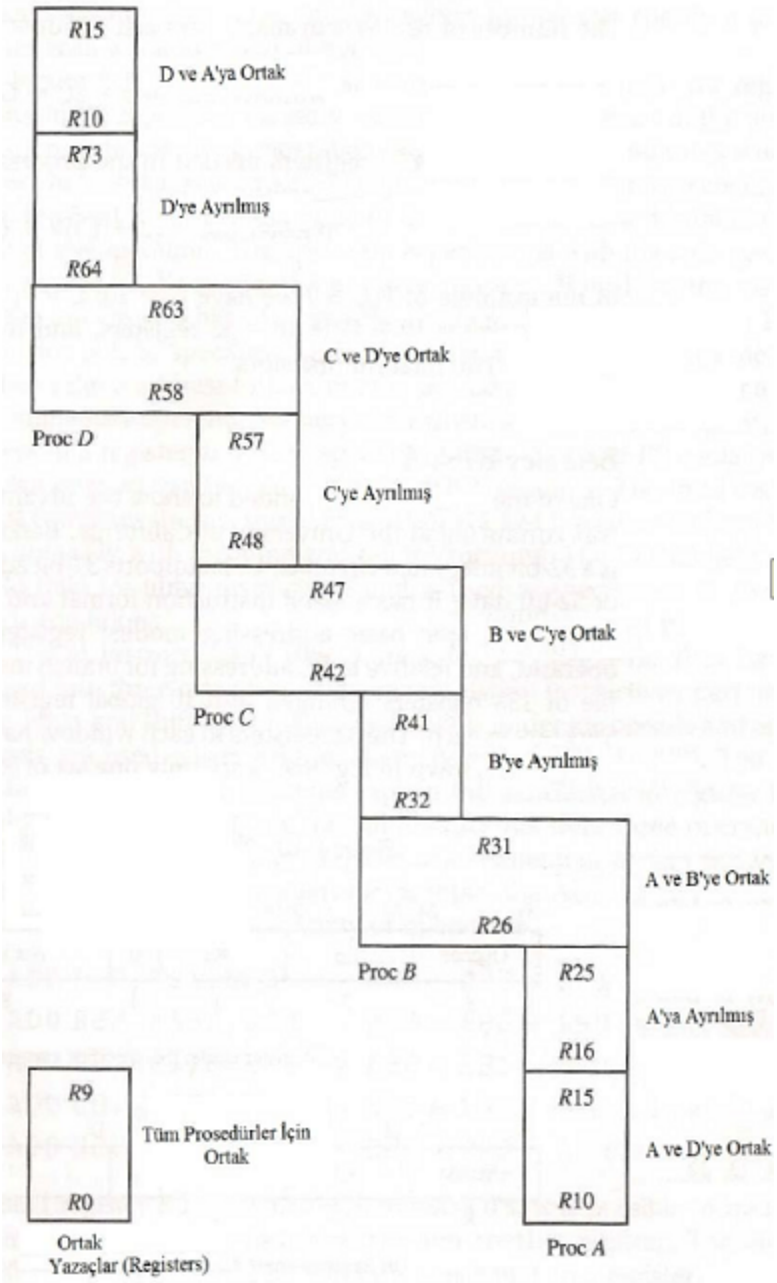
- Az sayıda buyruk
- Az bellek kullanımı (sadece LOAD ve STORE)
- CPU içinde daha hızlı (bütün işlemler CPU yazaçlarında icra edilir)
- Temel hedefi: Buyruk kümesi basitleştirilerek icra zamanı azaltılır.
- Az adresleme kipi
- Sabit uzunluklu kod
- Tek evreli buyruk icrası (3 buyruk 3 evre, en önemli özellik)
- Donanımsal denetim
- Çok sayıda ve üst üste binen yazaçlar
- Pipeline

Üst Üste Binen Yazaç Pencereleri

- Saklama ve geri yazma işleminden kurtulmayı sağlar
- Her alt program çağrısında yeni bir pencere açılır
- Her geri dönüş komutu ile bir pencere kapanır önceki pencere aktif olur
- Komşu veya sıralı alt programların pencerelerinin üst üste gelen yazaçları parametre ve sonuçların transferini sağlarlar

Örnek gösterim:

1. Sistemde toplam 74 yazaç var
2. R0 - R9 arası ⇒ ORTAK yazaçlar (bütün alt programlar kullanır)
3. Geriye kalan 64 yazaç 4 pencereye bölünüp A,B,C,D alt programlarını tutar
    a. Her pencerede 10 yerel yazaç var (Yerel değişkenlerin kullanması için)
    b. Sıralı alt programlarda ortak kullanım için 6 yazaçlık 2 küme bulunur
    c. Ortak yazaçların varlığı verinin gerçekten hareket etmeden kullanılmasını sağlar
    d. Yeni bir pencere açıldığında (bir alt program çalıştırıldığında) çağıran programın yüksek yazaçları ile çağrılan programın düşük yazaçları kesişir. Böylelikle parametreler otomatik olarak çağırandan çağrılana aktarılmış olur.



Örneğin A ‘nın B’yi çağırdığı varsayılsın:

1. R26 - R31 arası yazaçlar iki programda ortak
2. A, B programı için gerekli parametreleri bu ortak yazaçlarda saklar
3. B programı R32-R41 arası yazaçları yerel olarak kullanır

4. Daha sonra B, C programını çağırırsa gerekli parametreleri R42-R47 arasındaki yazaçlara aktarır
5. İşlem tamamlandıktan sonra B, geriye dönen değeri R26-R31 yazaçlarına yazar ve A penceresine geri döner
6. R10-R15 yazaçları A ve D arasında ortaktır (ardışıklık dairesel kabul edilir)
7. R0-R10 yazaçları tüm programlara açıktır
8. Şekildeki her bir program aktif olduğu anda ; 10 ortak, 10 yerel, 6 yüksek ve 6 düşük üst üste binen yazaç ile toplam 32 yazaca sahip olur

Genel bağıntı şu şekilde gösterilir:

- Ortak yazaç sayısı : G
- Her penceredeki yerel yazaç sayısı : L
- İki pencere arasındaki ortak yazaç sayısı : C
- Pencere sayısı : W

Buna göre:

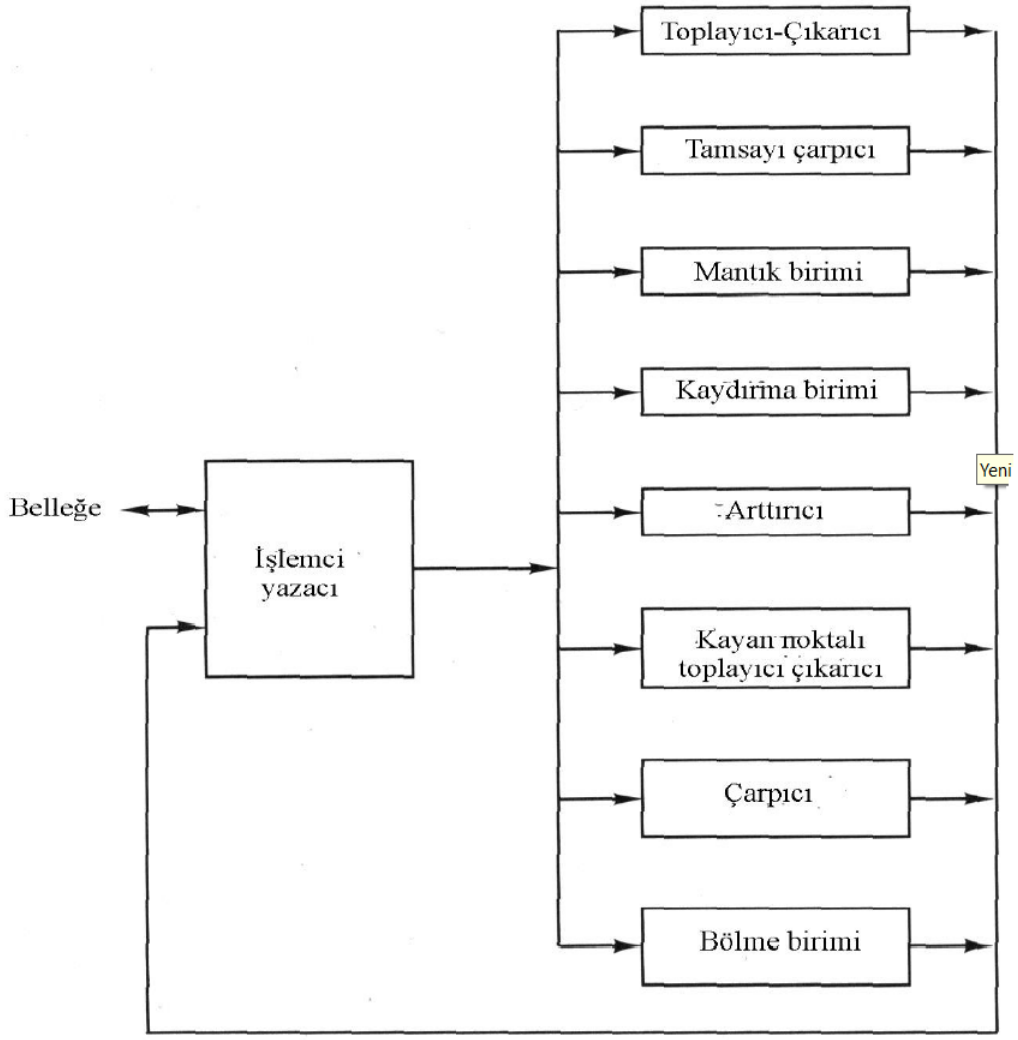
- Bir penceredeki yazaç sayısı = L + 2C + G
- İşlemcinin ihtiyacı olan yazaç sayısı = (L + C) W + G

Paralel İşleme

- Aynı zamanda yapılan veri işleme olayıdır
- Aynı anda birçok buyruk icra edilir
- Karmaşık düzeyde paralellik iş yapan birimlerin çoğaltılmasıyla elde edilir

Kaydırma yazaçları seri olarak çalışırlar ve 1 zaman biriminde 1 bit işlerler. Ancak bu durumu paralel işleme ile tüm bitlerin 1 zaman biriminde işlenmesi şeklinde değiştirebiliriz

Çok Fonksiyonlu Birimler



- Şekilde işlemsel birimin paralel çalışması gösterilmiştir
- Yazaçlardaki veriler buyruğun belirlediği birimlerden birine gönderilir
- Kayan noktalı işlemler paralel olarak çalışan 3 devreye ayrılmıştır ve birbirinden bağımsızdır. Dolayısıyla bir sayı kaydırılırken başka bir sayı artırılabilir.

Paralel işlemenin sınıflandırılması:

- Tek buyruk - tek veri

- Tek buyruk - çok veri
- Çok buyruk - tek veri
- Çok buyruk - çok veri

Pipeline İşlemleri

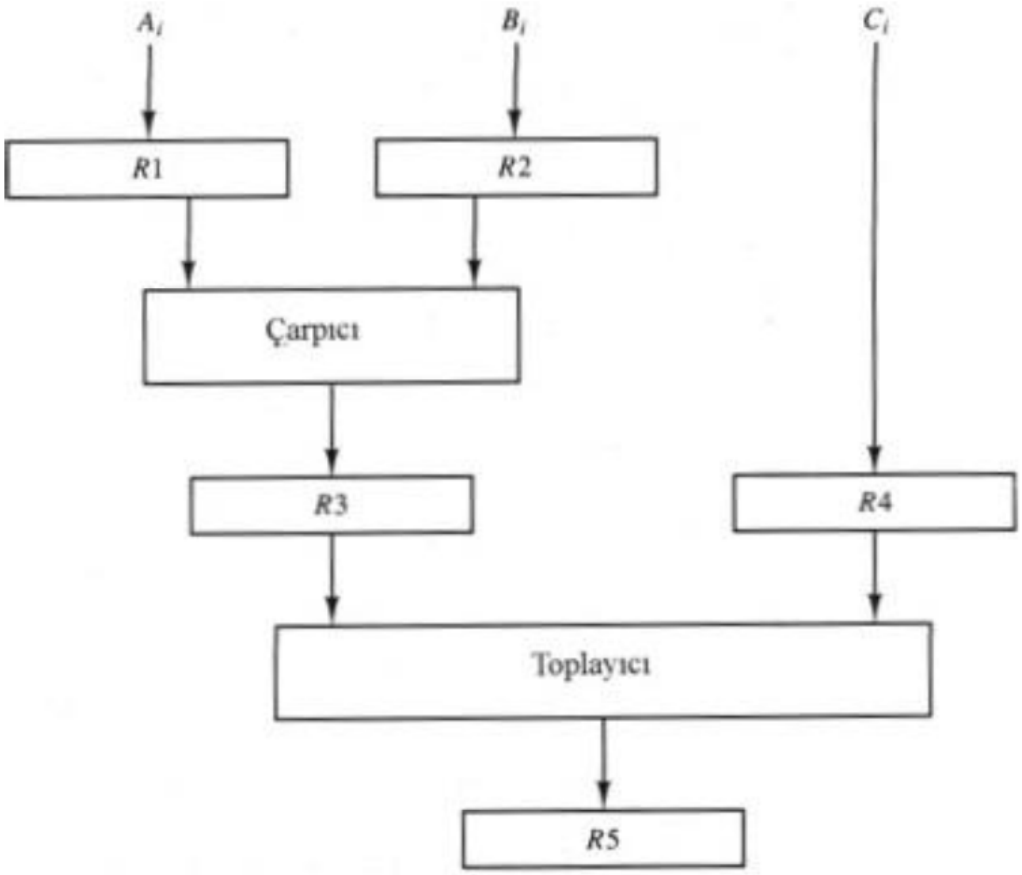
- Boru hattı işlemi, sıralı bir işlemi alt işlemlere bölmeye yarayan bir yöntemdir.
- Her bir alt işlem kendisine ayrılmış bir kesimde icra edilir
- Diğer kesimlerde de aynı anda başka işlemler yerine getirilir.
- Her bir kesimde elde edilen sonuç bir sonraki kesime aktarılır
- Veri bütün kesimlerden geçtiğinde sonuç elde edilir
- Hesaplamaların üst üste yapılabilmesi, pipeline içindeki her kesime bir yazaç bağlamakla mümkün olur.
- Yazaçlar kesimler arası izolasyonu sağlar

Örnek;

Ai * Bi + Ci i = 1,2,…,7

1. R1 ← Ai, R2 ← Bi Ai ve Bi yi gir
2. R3 ← R1 * R2, R4 ← Ci Çarp ve Ci yi gir
3. R5 ← R3 + R4 Sonuçla Ci yi topla

Boru hattı işlemi örneği;



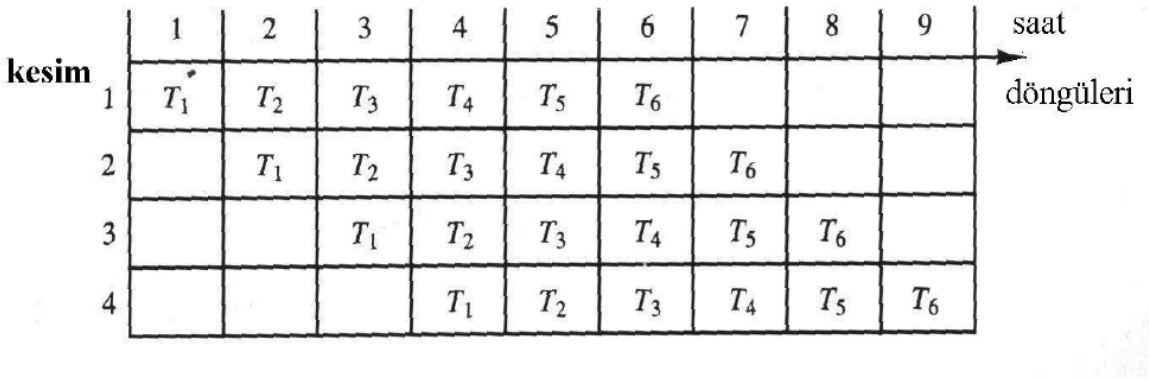
Boru hattı örneğindeki yazaçların içerikleri

Saat vuruş sayısı	kesim1		kesim2		kesim3
	R1	*R2*	*R3*	*R4*	*R5*
1	A_1	B_1	-	-	-
2	A_2	B_2	$A_1 * B_1$	C_1	
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_8	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	-	-	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9	-	-	-	-	$A_7 * B_7 + C_7$

Uzay - Zaman Diyagramı

- Pipeline davranışını göstermenin en iyi yoludur
- Diyagram kesimlerin kullanılışını zamanın fonksiyonu olarak gösterir.

4 kesimli pipeline için uzay - zaman diyagramı;



Şekil. Boru hattı için uzay-zaman diyagramı

- Yatay eksen saat vuruşlarında zaman ekseni, dikey eksen ise kesim numarasıdır.
- T1 görevi 4 saat vuruşu sonra tamamlanır. Bu andan itibaren her saat vuruşu sonrası 1 görev tamamlanır.

Teorik hesaplar;

1. n adet görevin, k kesimli boru hattı içinde ve Tp saat vuruşu ile işlendiğini düşünelim
2. Birinci görevin gerçekleşmesi k.Tp kadar zaman alır. k kesimin her biri ⇒ Tp zamanında geçer
3. Kalan n-1 görev için gerekli zaman (n-1).Tp olur
4. n görevin, k kesimde tamamlanması için k + (n-1) zaman gereklidir