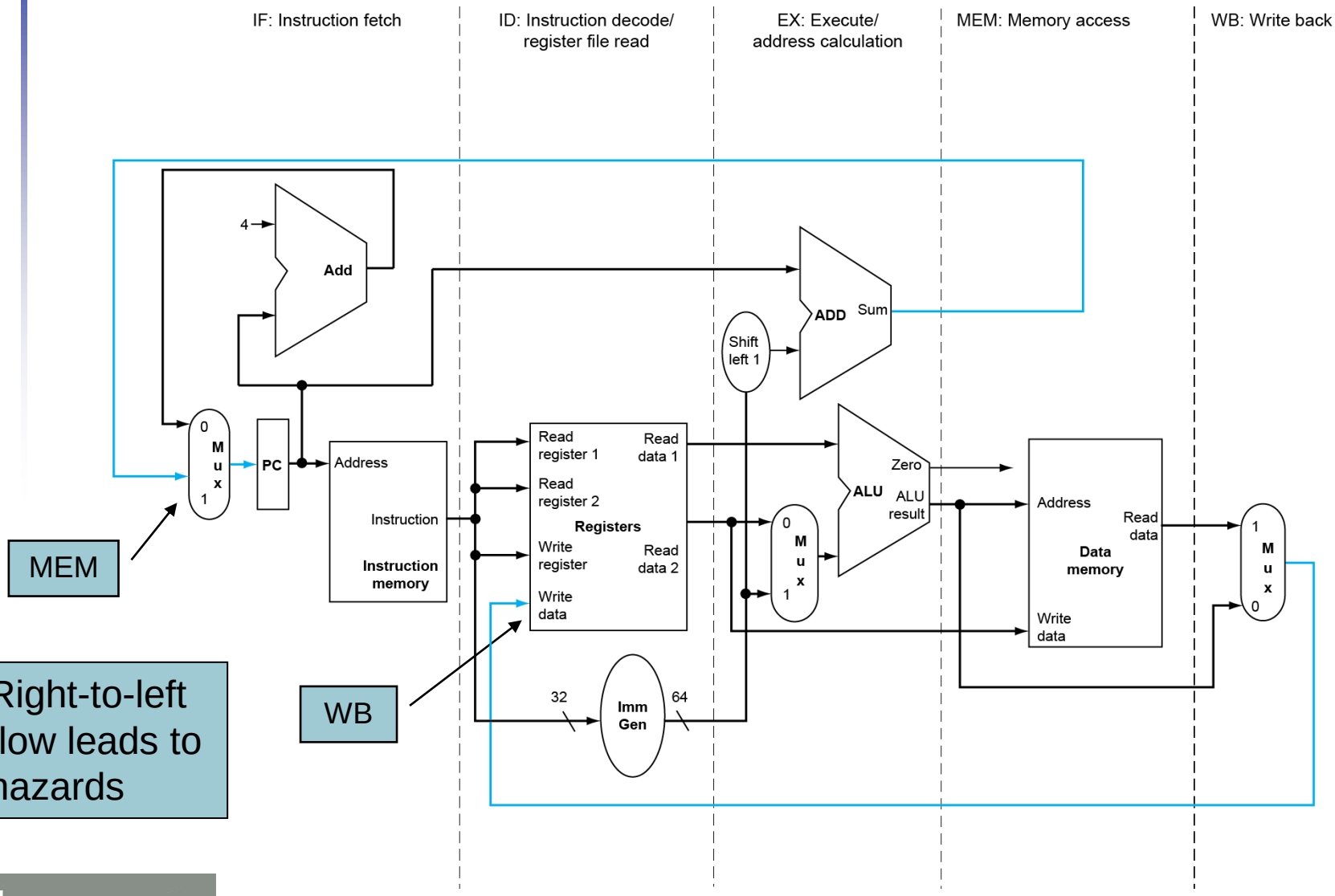
Chapter 4

The Processor

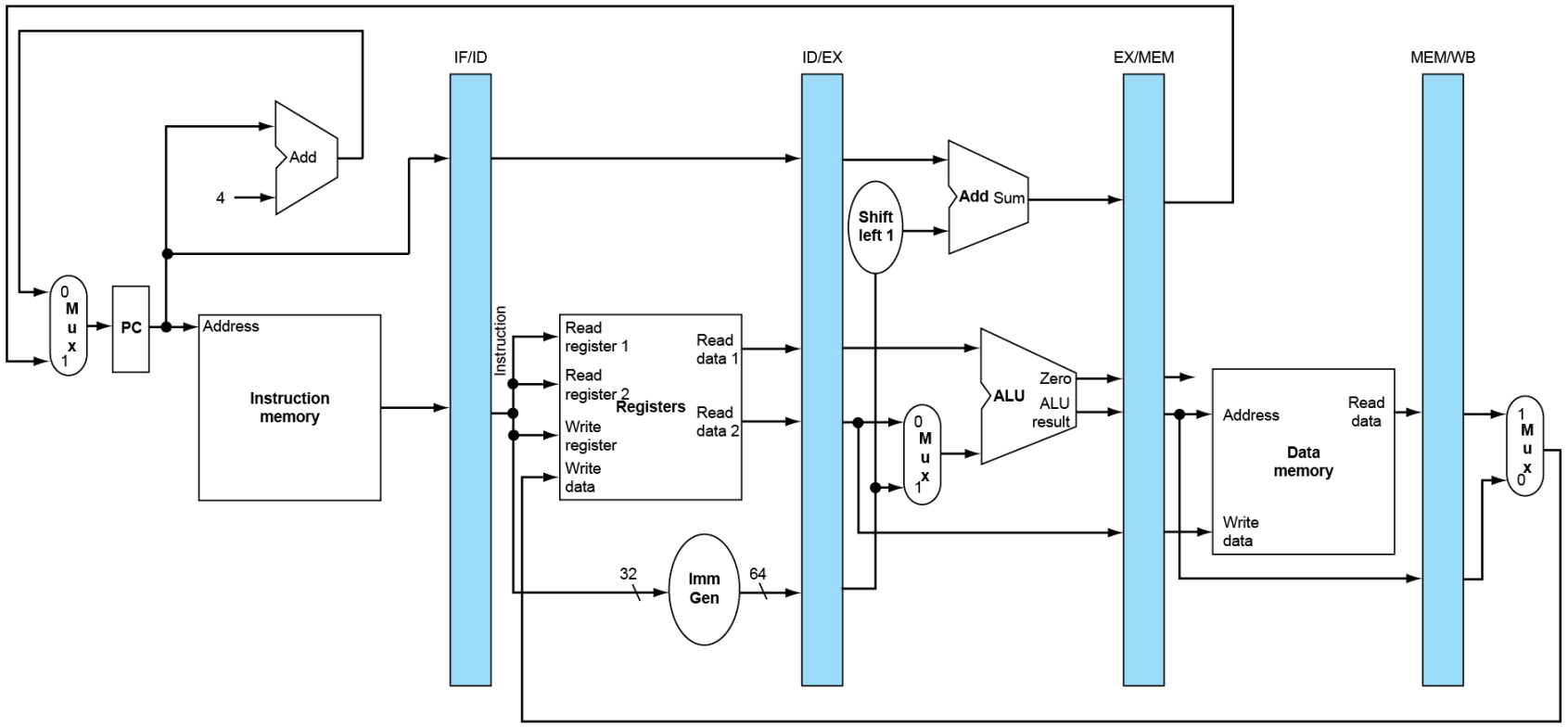
Bölüm 4.6- 4.16

RISC-V Boru Hattı Veri Yolu



Boru Hattı Kaydediler

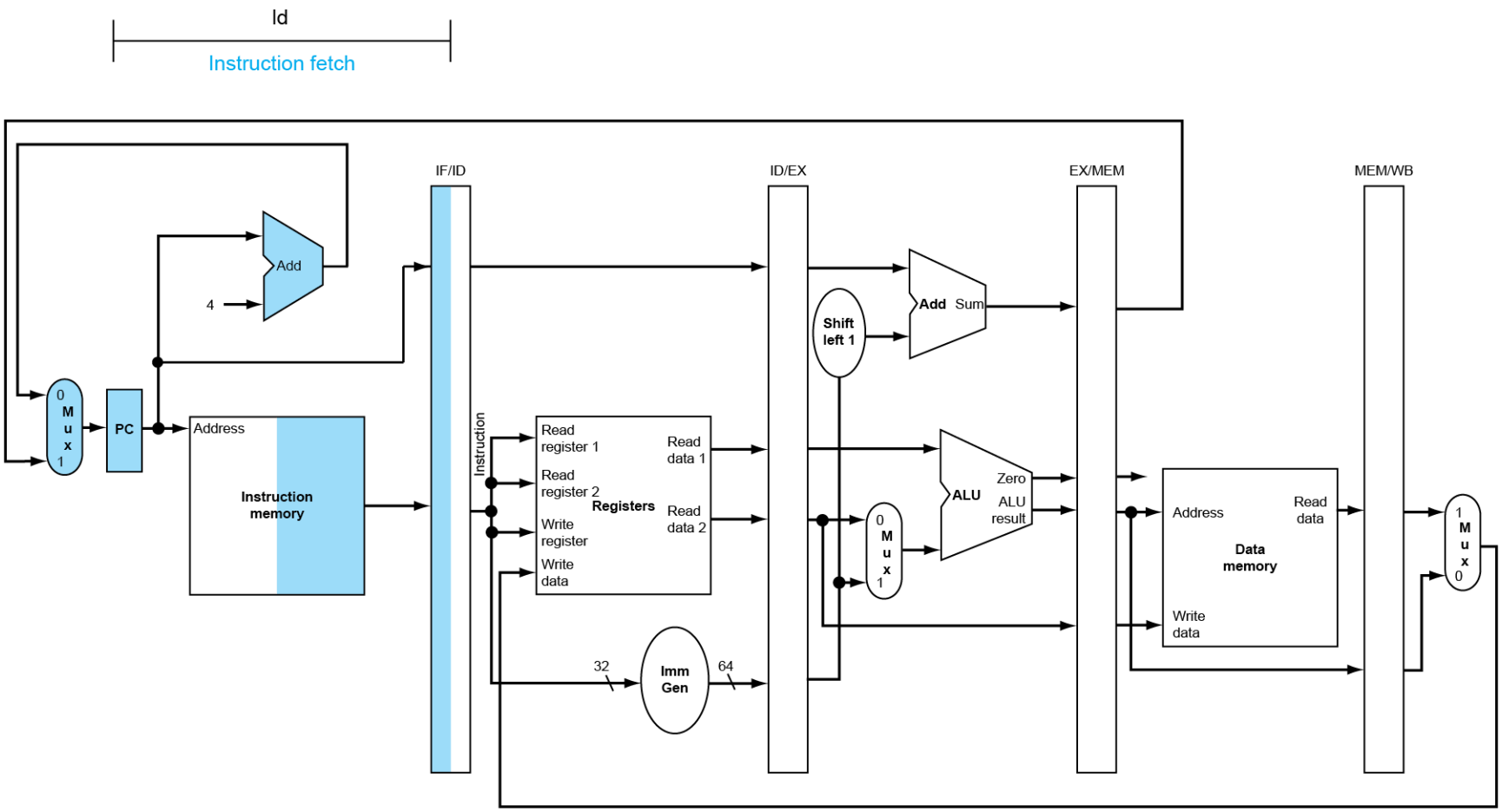
- Aşamalar arasında kaydedicilere ihtiyaç vardır
  - Önceki aşamada üretilen bilgileri tutmak için.



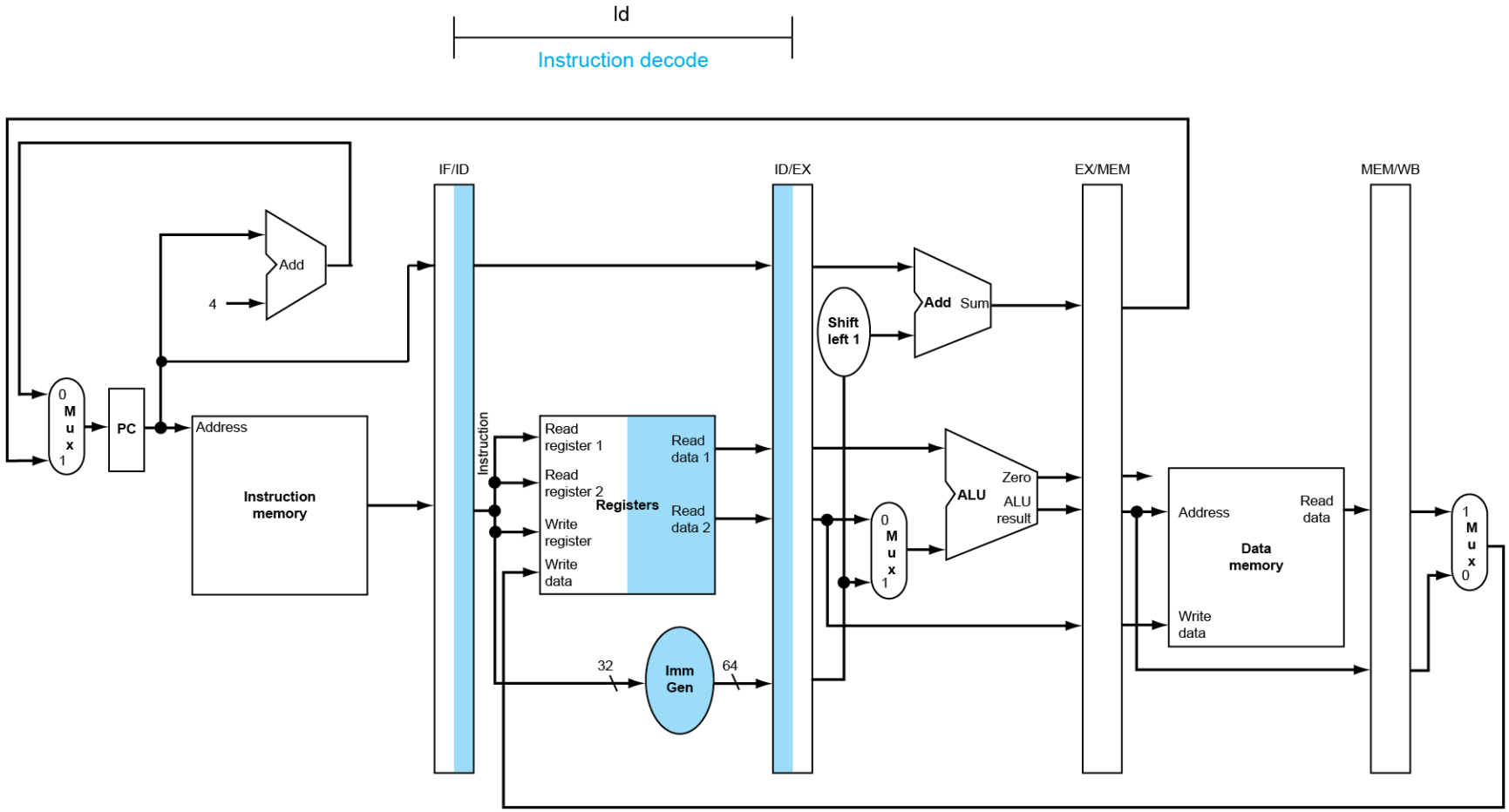
Boru Hattı İşlemleri

- Ardışık düzenlenmiş veriyolu aracılığıyla buyrukların aşamalr arası akışı
  - “Tek saat vuruşlu” boru hattı
    - Ardışık botu hattı kullanımını tek bir döngüde gösterilir
    - Kaynak kullanımı vurgulanır
  - “Çok saat vuruşlu”
    - Zaman içinde çalışma grafiği
- Load ve Store işlemleri için "tek saat vuruşlu" diyagramlara bakacağız.

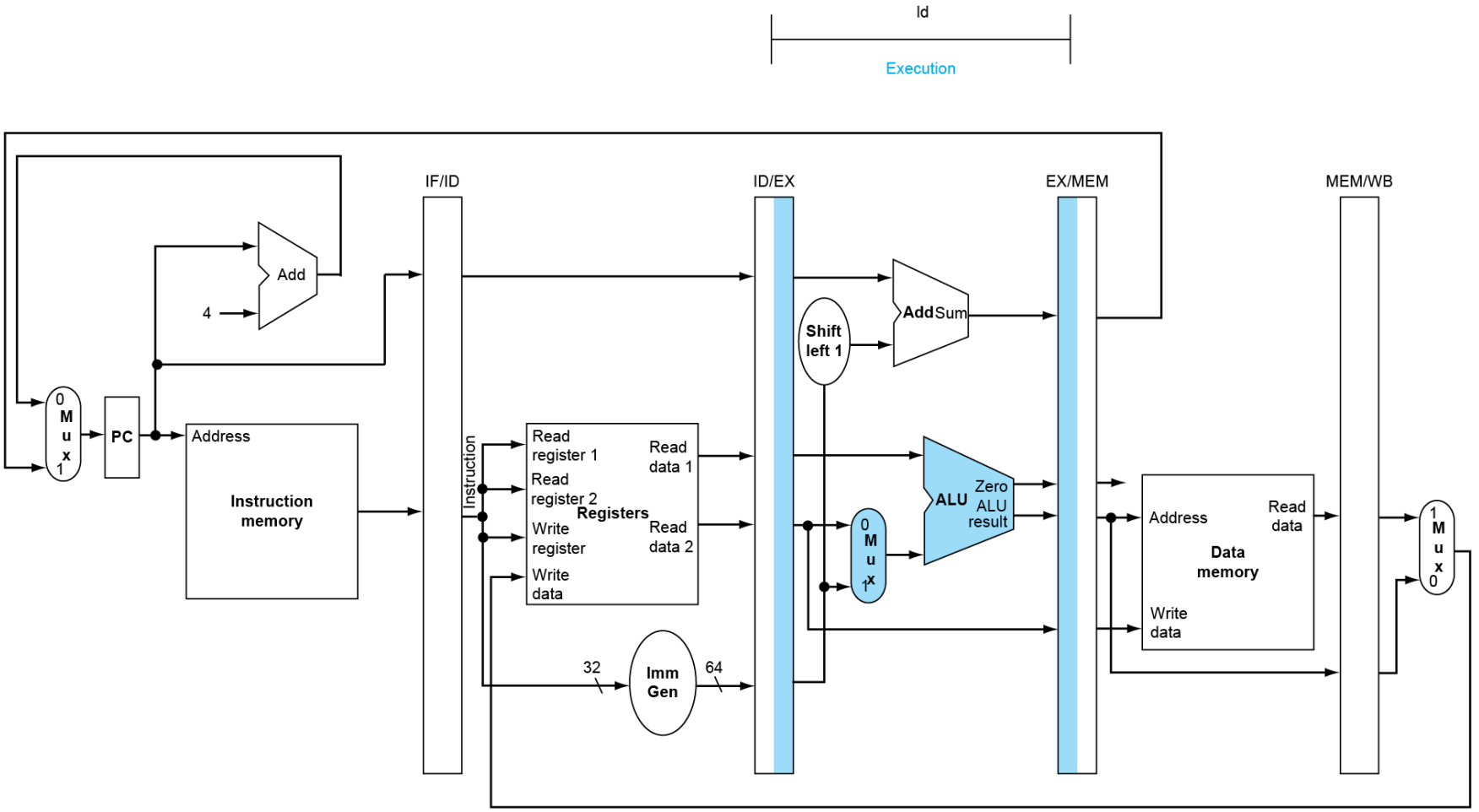
IF Aşaması Load, Store, …



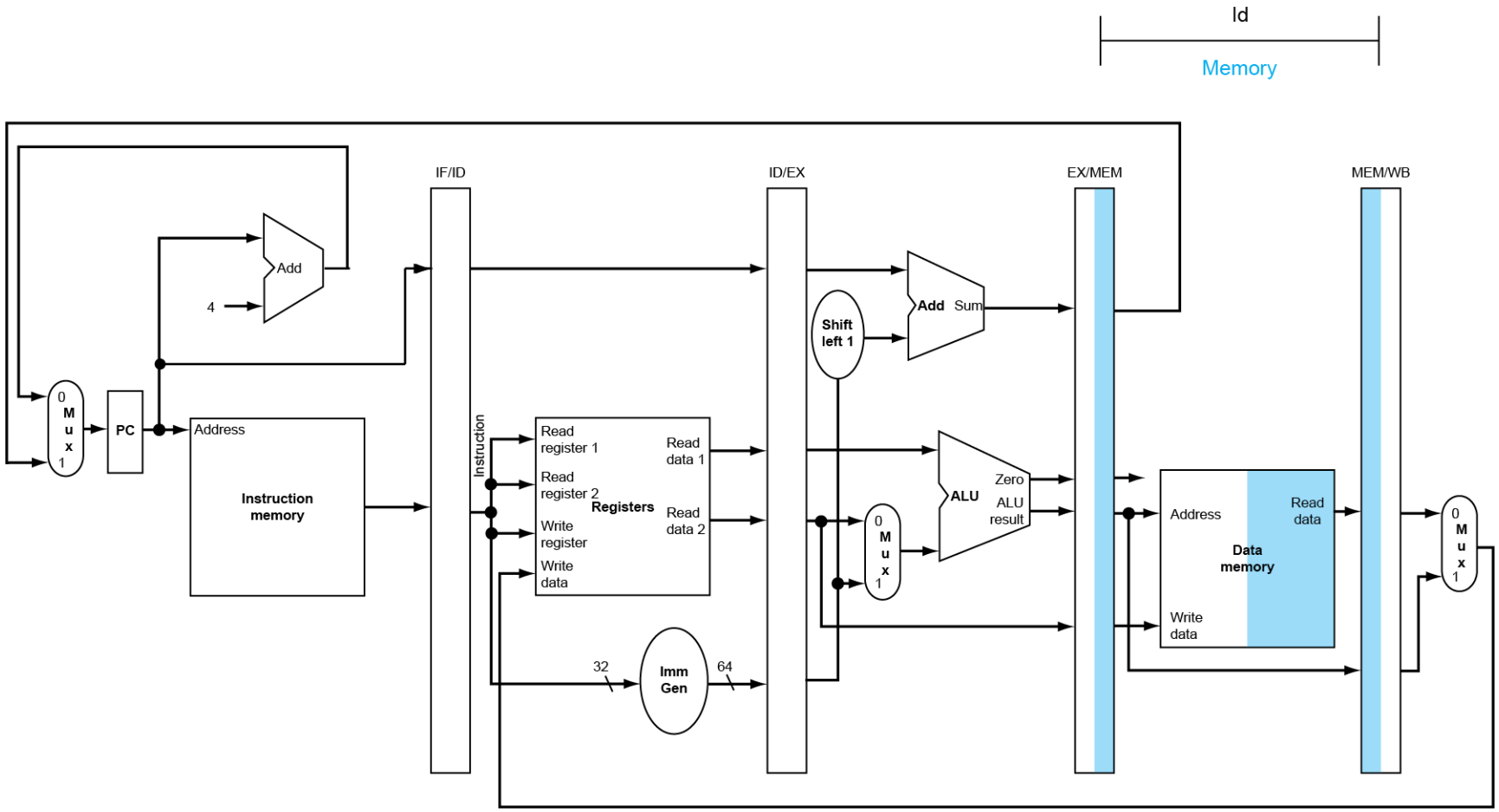
ID Aşaması Load, Store, …



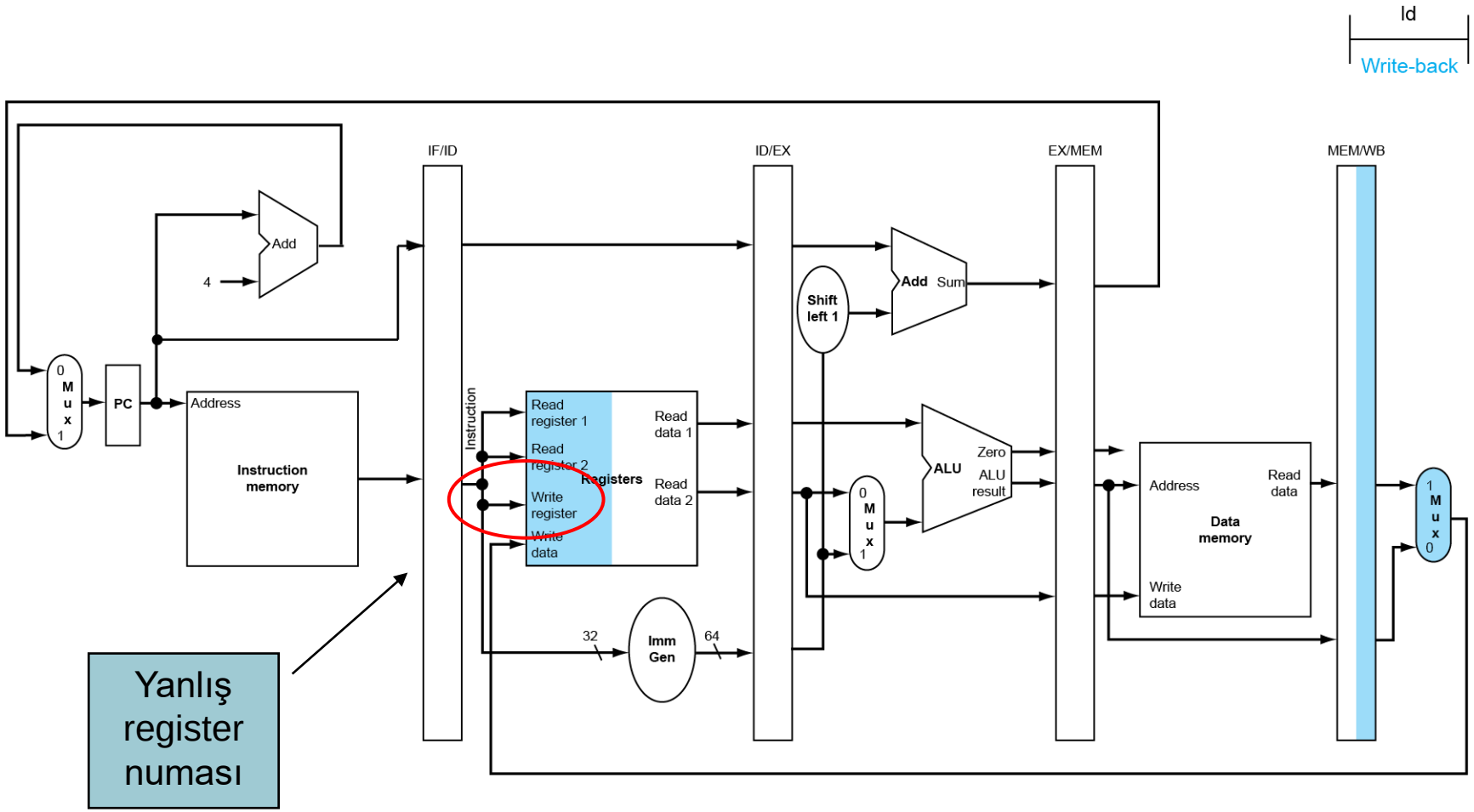
EX Aşaması Load



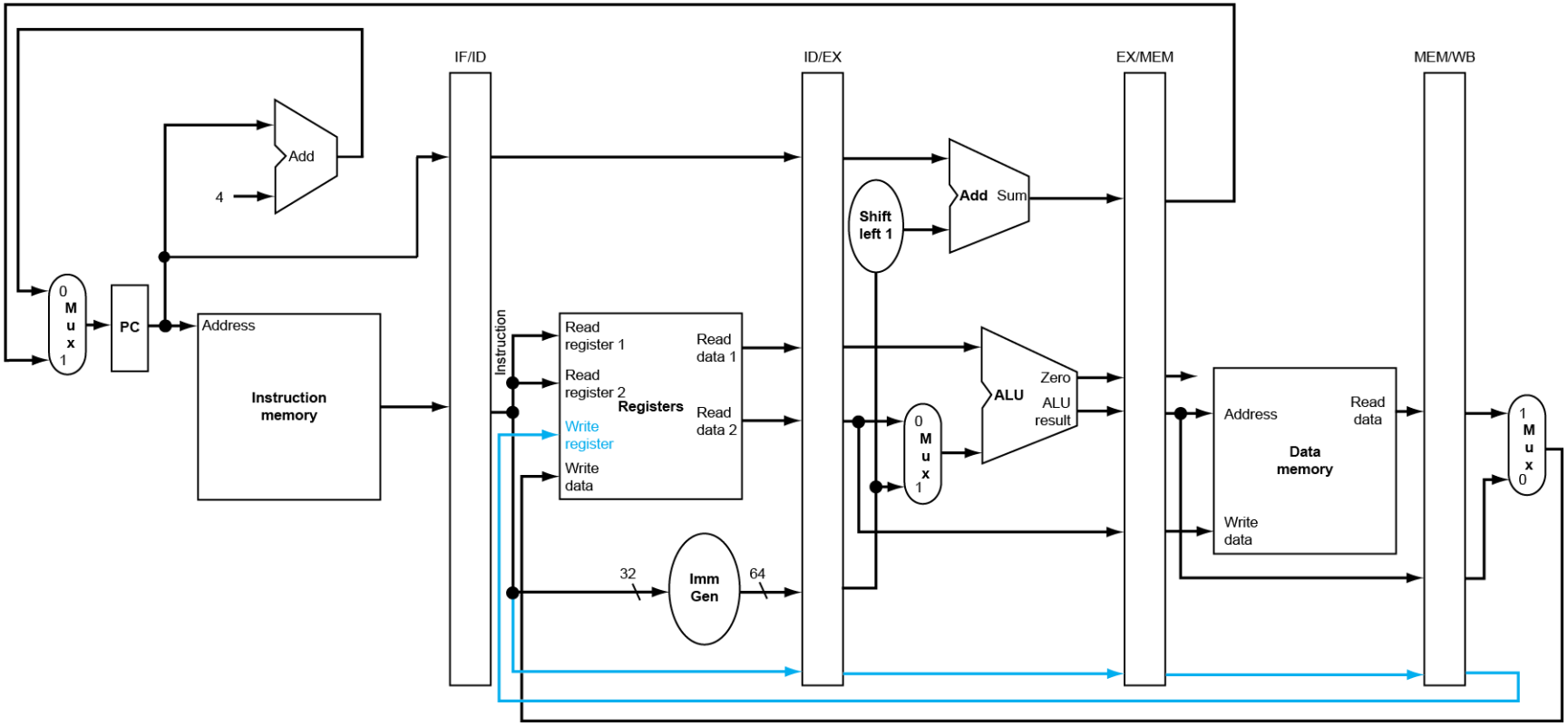
MEM Aşaması Load



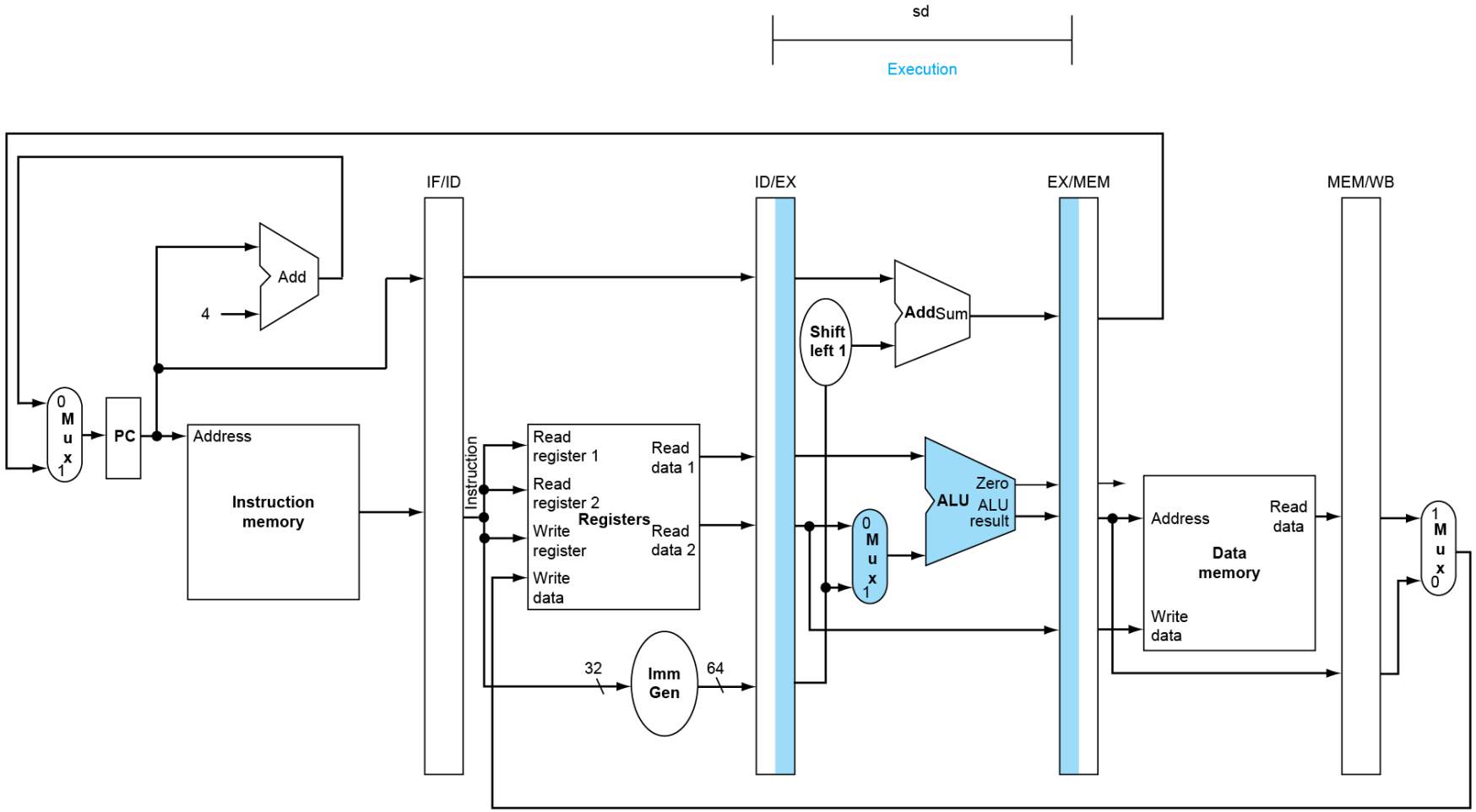
WB Aşaması Load



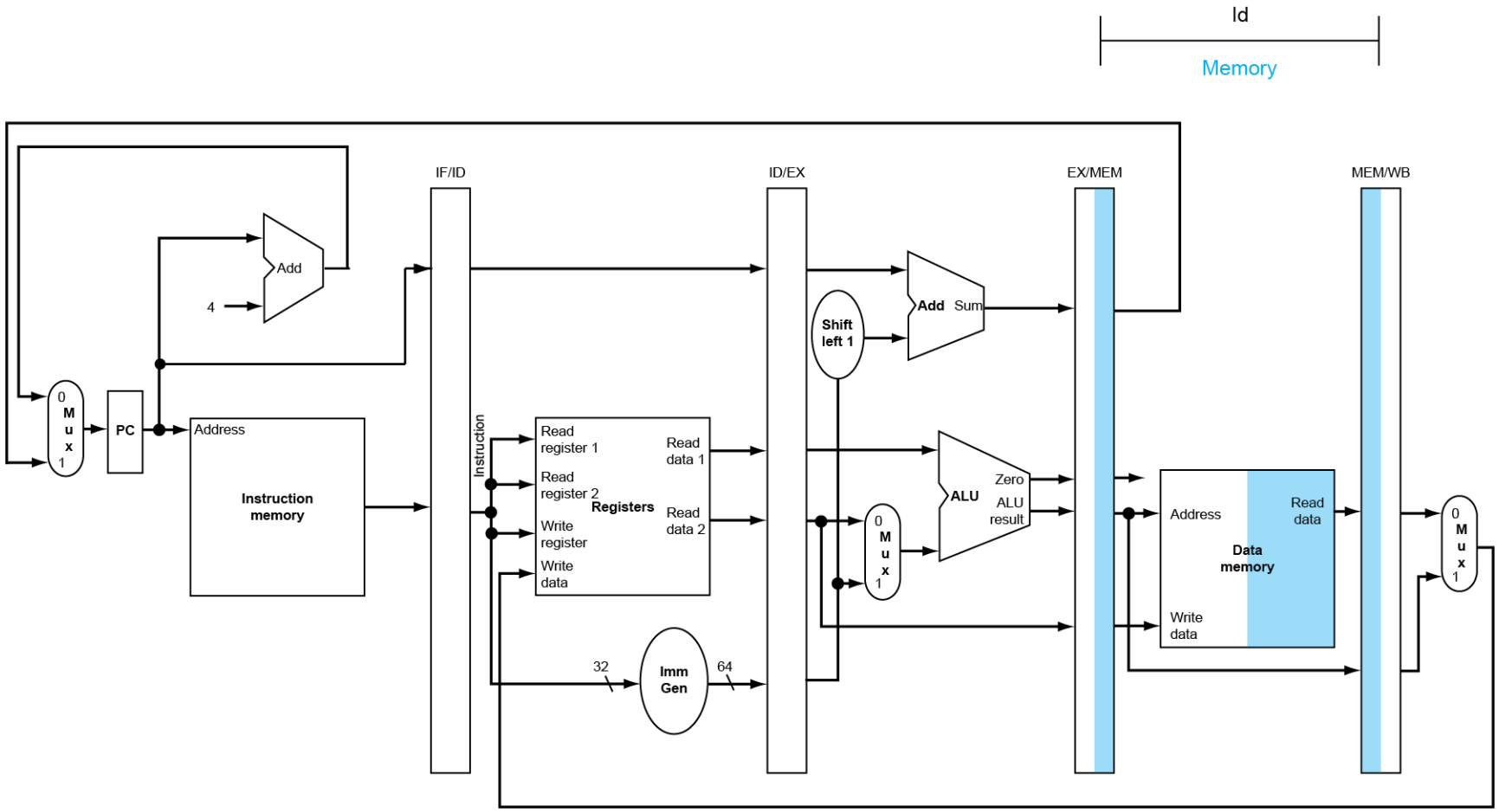
Düzeltilmiş Veriyolu Load



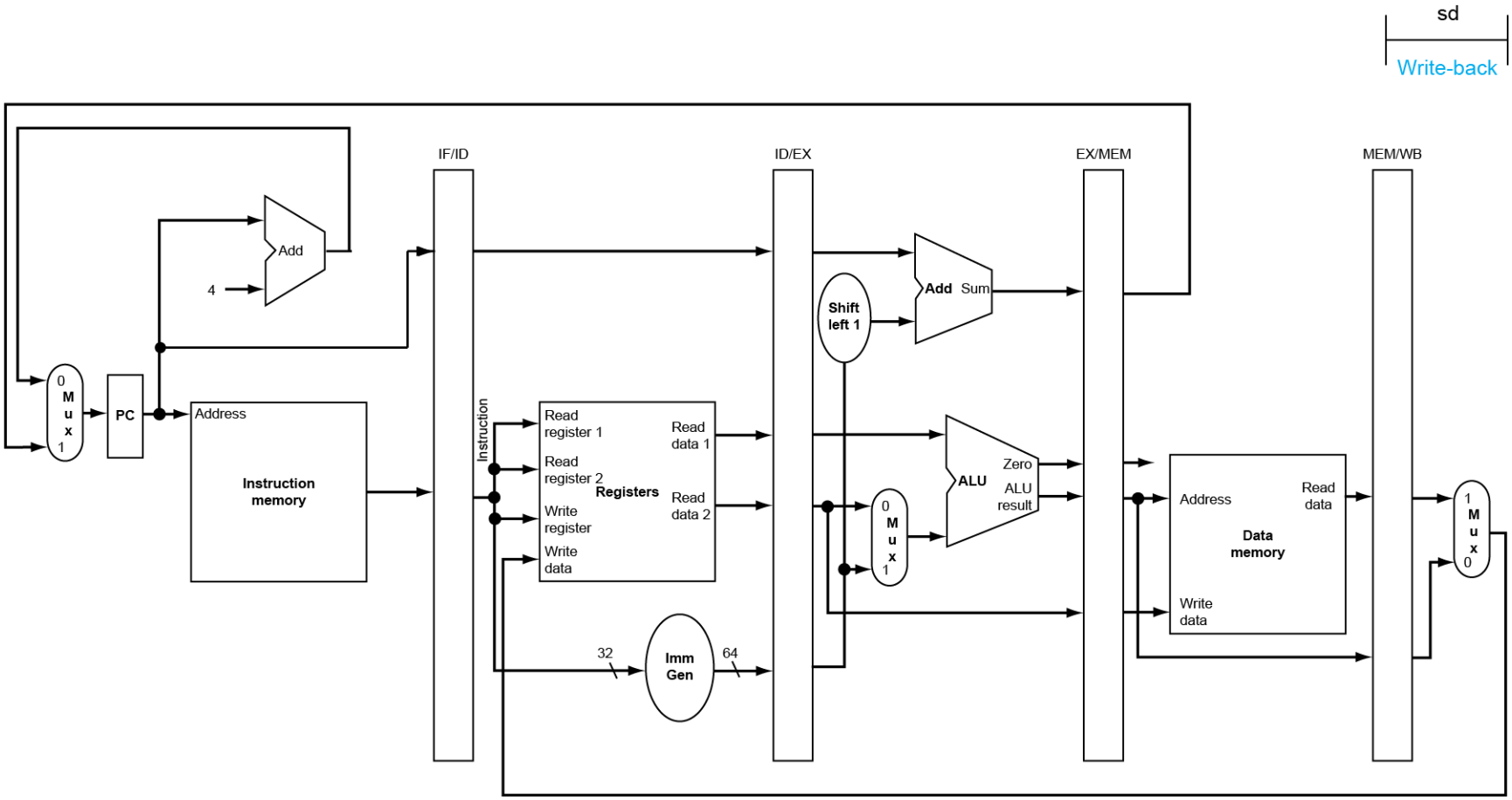
EX Aşaması Store



MEM Aşaması Store

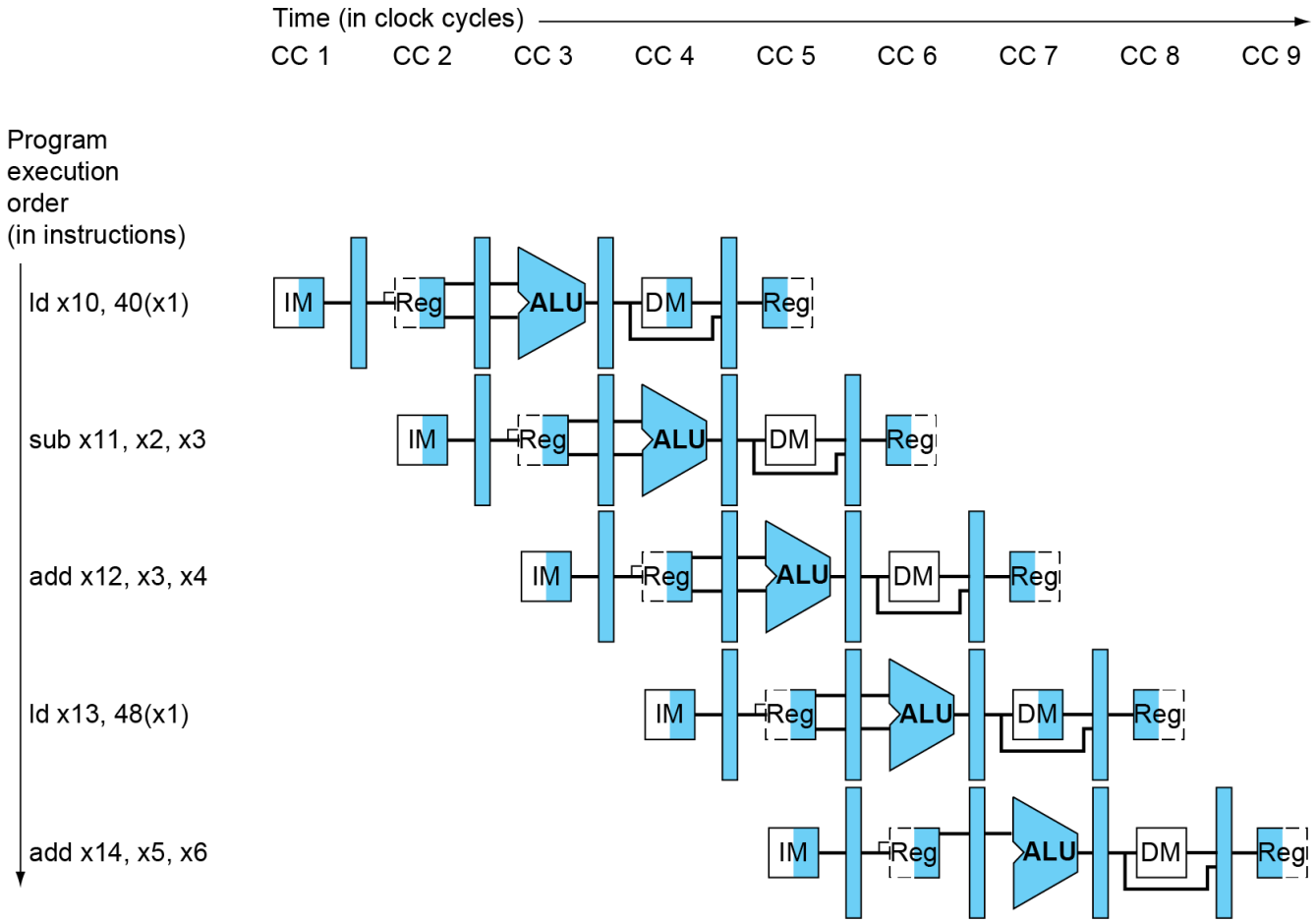


WB Aşaması Store



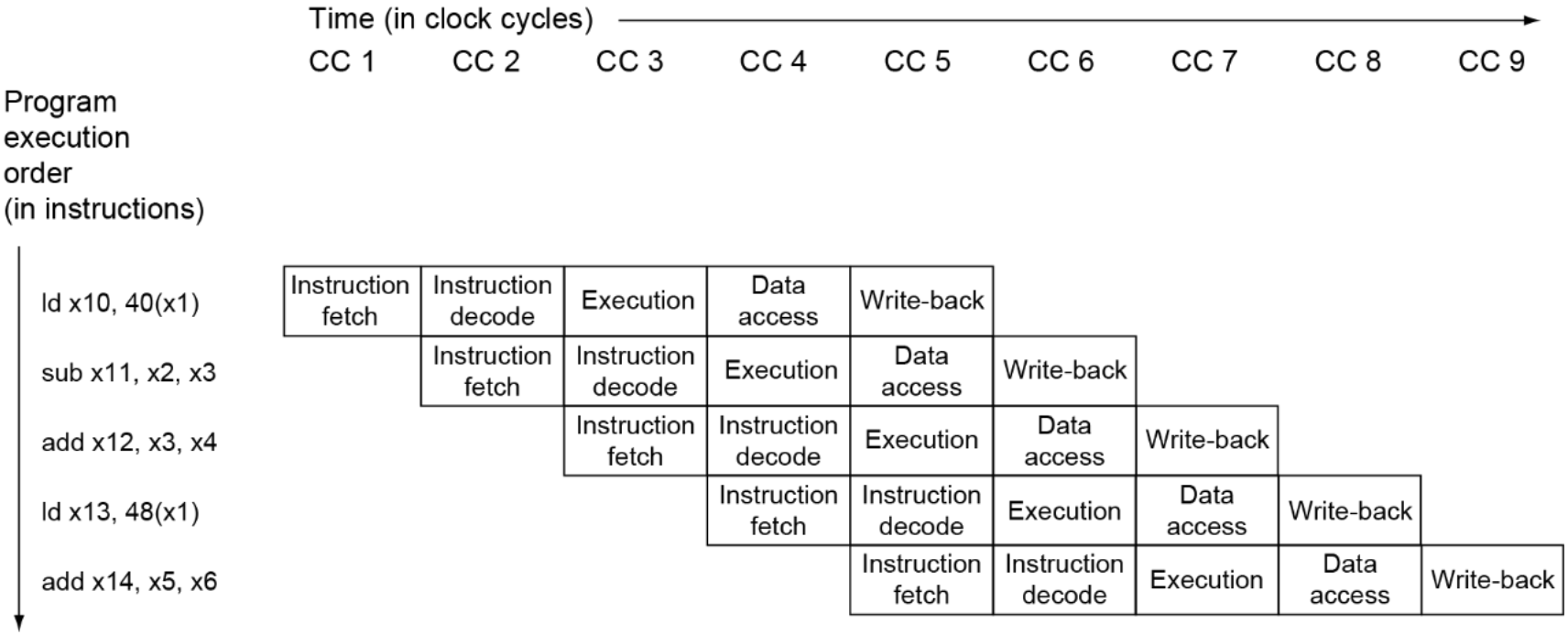
Çok Vuruşlu Boru Hattı Diyagramı

- Kaynak kullanımını gösteren form



Çok Vuruşlu Boru Hattı Diyagramı

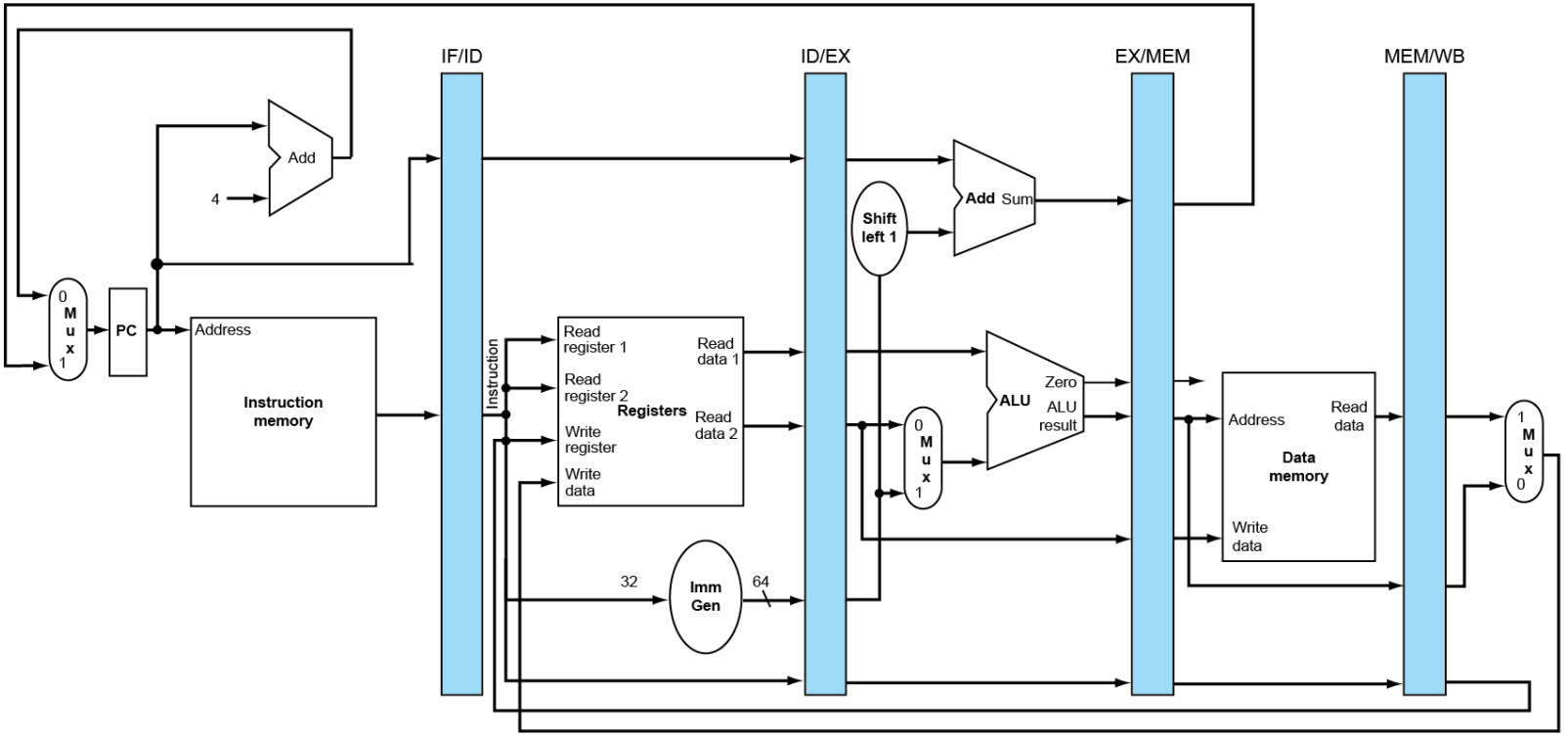
- Geleneksel form



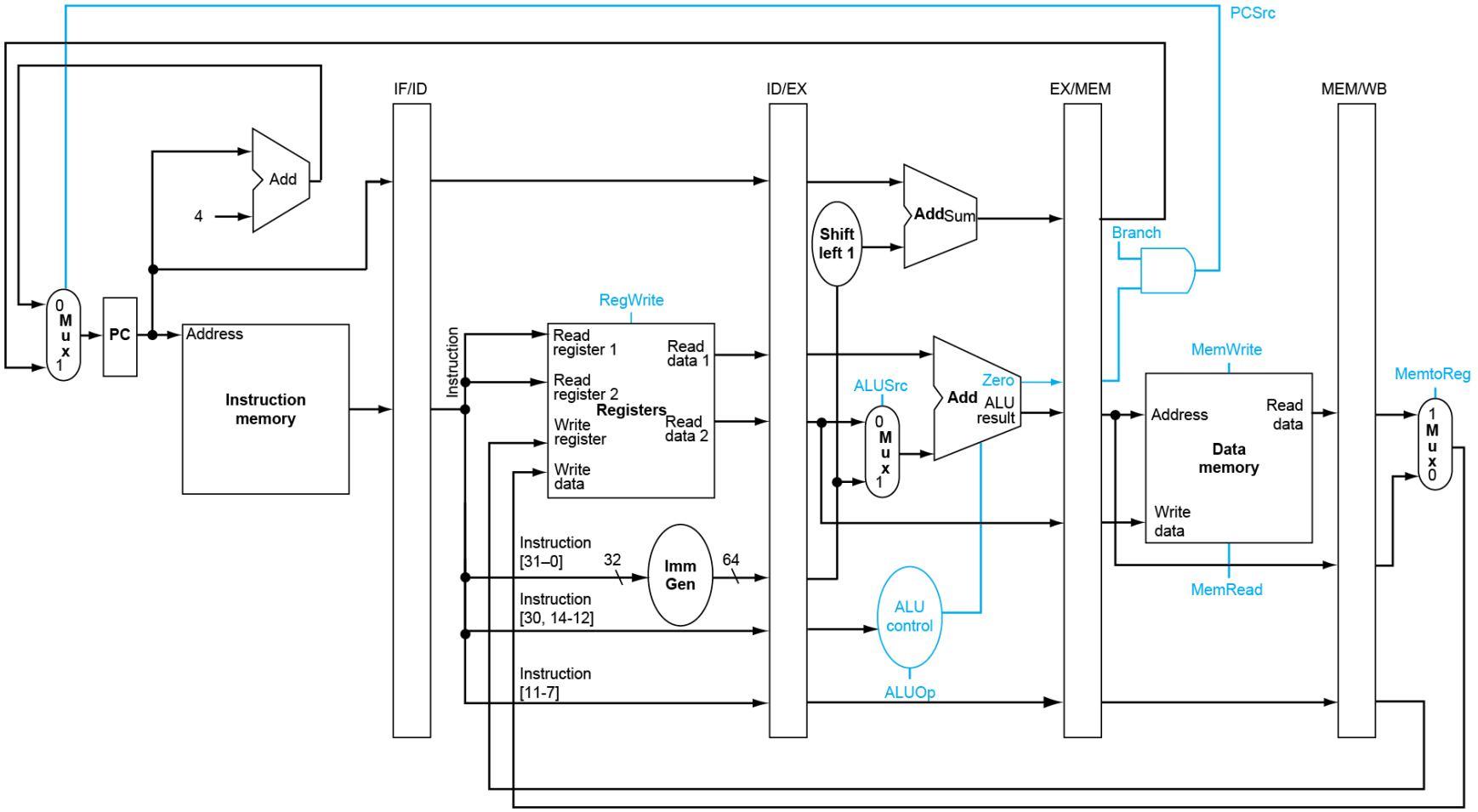
Tek Vuruş Boru Hattı Diyagramı

- Belirli aşamalardaki boru hattının durumu

add x14, x5, x6	ld x13, 48(x1)	add x12, x3, x4	sub x11, x2, x3	ld x10, 40(x1)
Instruction fetch	Instruction decode	Execution	Memory	Write-back

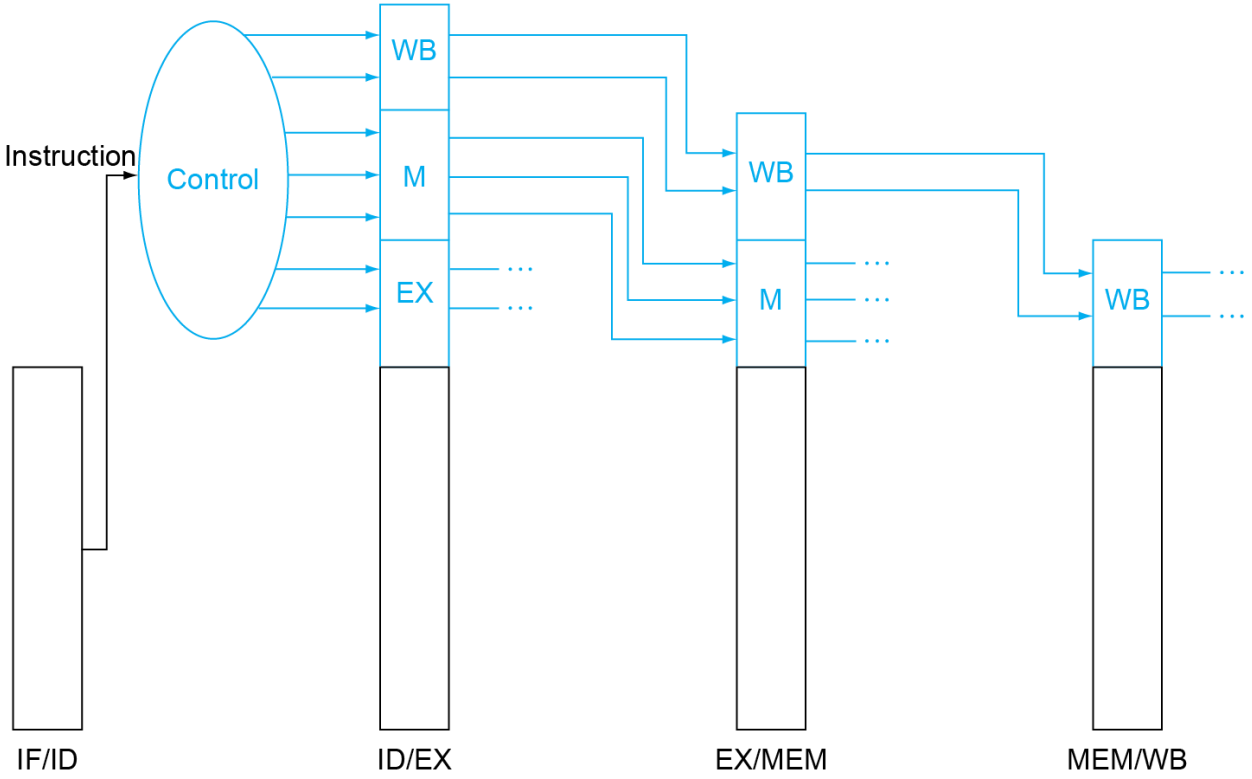


Boru Hattı Kontrol (Basitleştirilmiş)

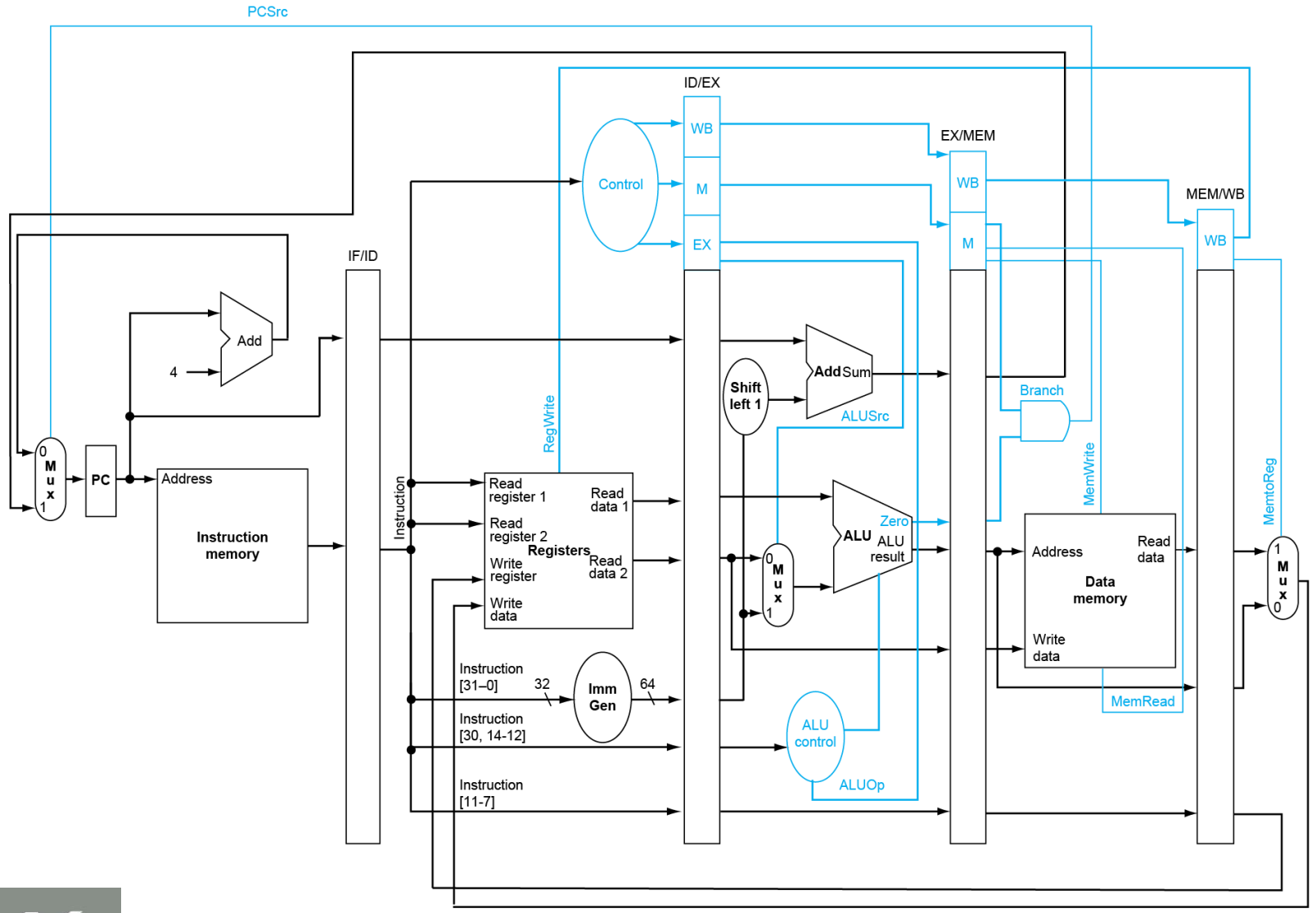


Boru Hattı Kontrol

- Buyruklar için türetilen kontrol sinyalleri
- Tek vuruşlu uygulamada olduğu gibi



Boru Hattı Kontrol



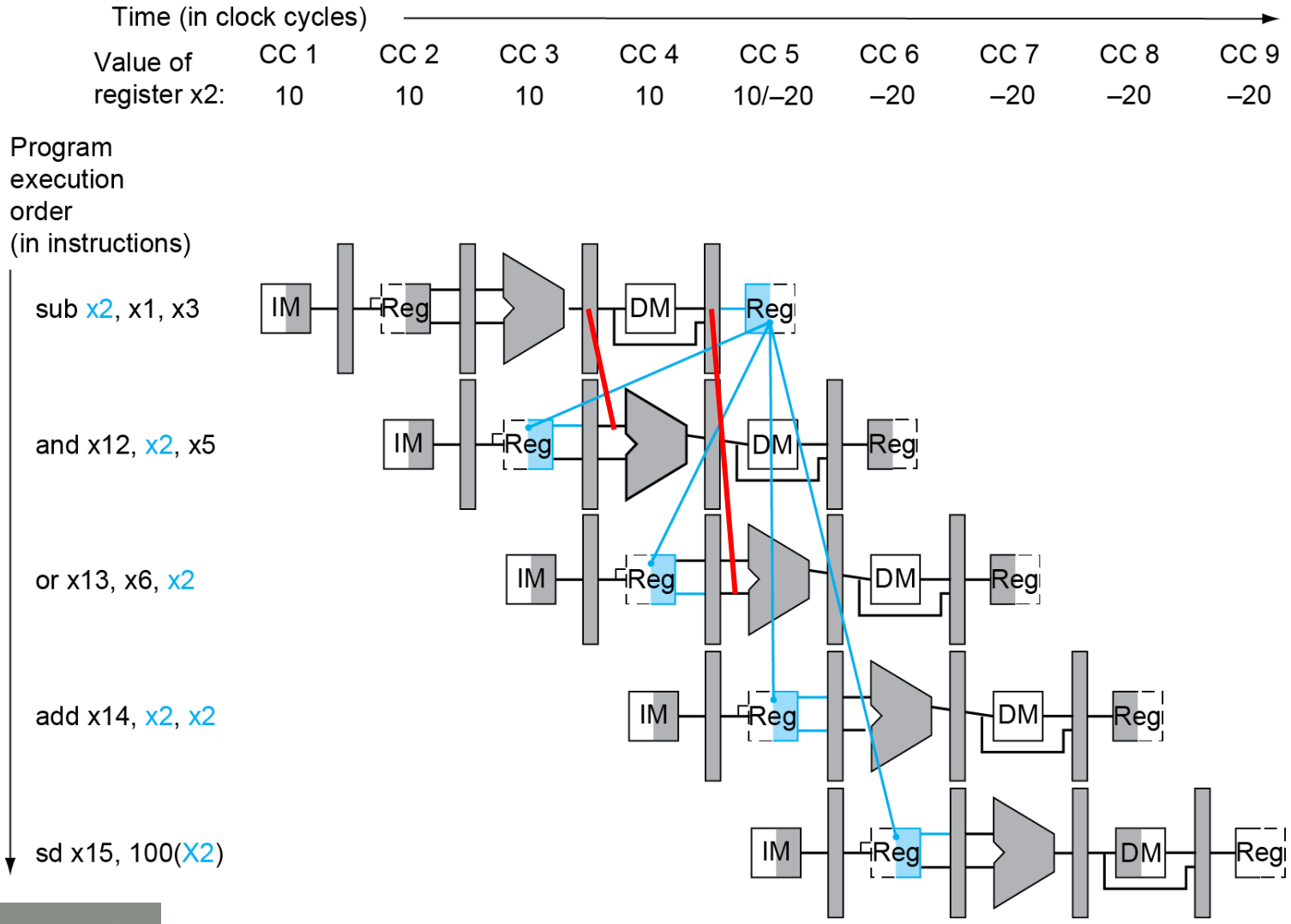
Veri Tehlikesi ALU Buyrukları

- Bu sırayı göz önünde bulundurun:

```
sub  x2, x1,x3
and  x12,x2,x5
or   x13,x6,x2
add  x14,x2,x2
sd   x15,100(x2)
```

- Tehlikeleri yönlendirme ile çözebiliriz
  - Ne zaman iletebileceğimizi nasıl tespit edeceğiz?

Bağımlılık & İletme



Detecting the Need to Forward

- Kaydedici numaralarını boru hattı boyunca geçir
  - e.g., ID/EX.RegisterRs1 = kaydedici numarası Rs1 ID/EX boru hattı kaydedicisine yerleşir.
- ALU aritmetik işlem kaydedicileri EX aşamasında
  - ID/EX.RegisterRs1, ID/EX.RegisterRs2
- Veri tehlikelerinde
  1a. EX/MEM.RegisterRd = ID/EX.RegisterRs1
  1b. EX/MEM.RegisterRd = ID/EX.RegisterRs2
  2a. MEM/WB.RegisterRd = ID/EX.RegisterRs1
  2b. MEM/WB.RegisterRd = ID/EX.RegisterRs2

Fwd from EX/MEM pipeline reg

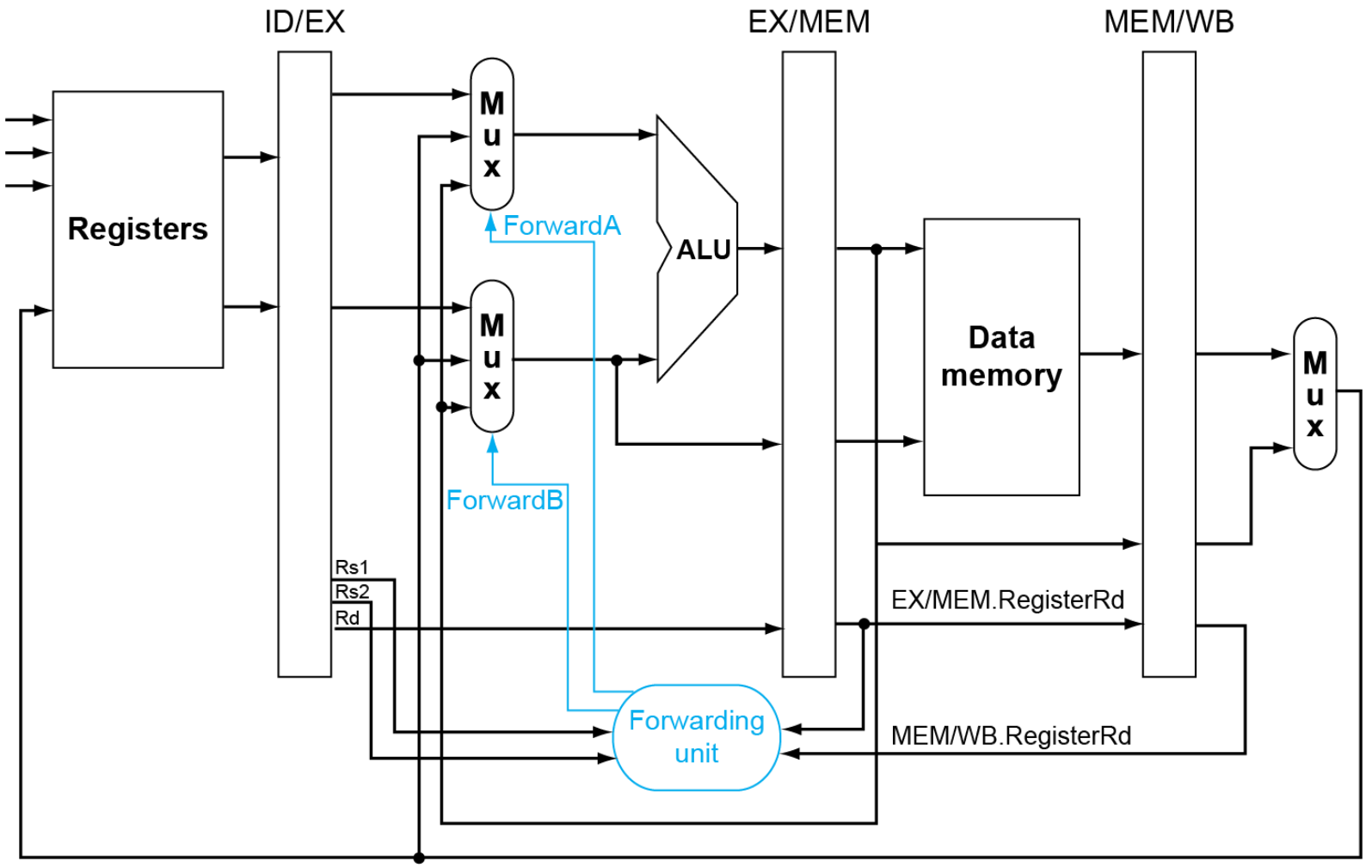
Fwd from MEM/WB pipeline reg

Detecting the Need to Forward

- Ancak, yalnızca iletme buyruğu bir kaydediciye yazacaksa!
  - EX/MEM.RegWrite, MEM/WB.RegWrite

- Buyruklarda Rd-x0 değilse
  - EX/MEM.RegisterRd ≠ 0, MEM/WB.RegisterRd ≠ 0

Forwarding Paths



İletme Koşulları

Mux kontrolü	Kaynak	Açıklama
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

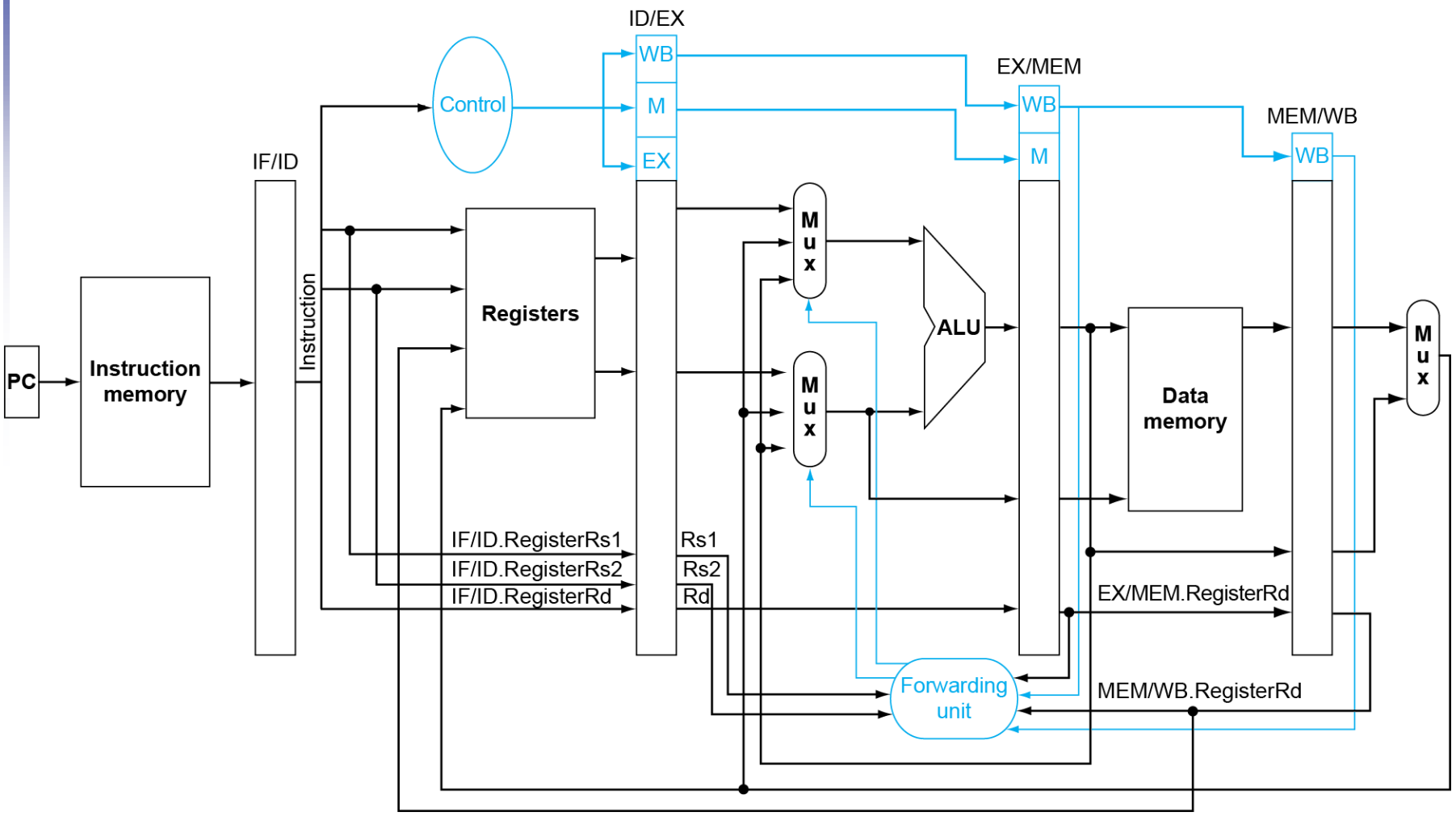
Çift Veri Tehlikesi

- Sırayı göz önünde bulundurun:

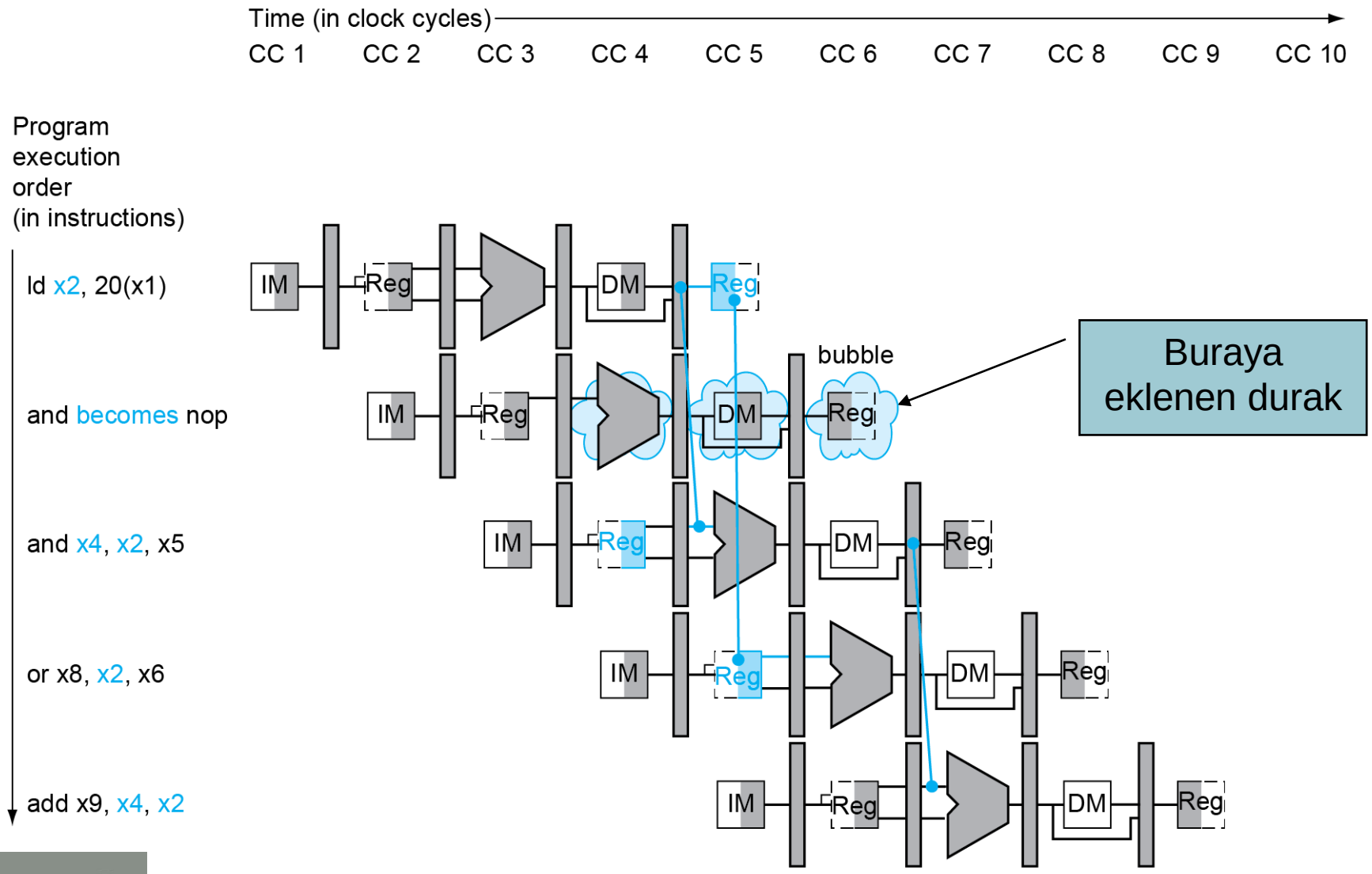
```
add x1,x1,x2
add x1,x1,x3
add x1,x1,x4
```

- Her iki tehlike de meydana gelir

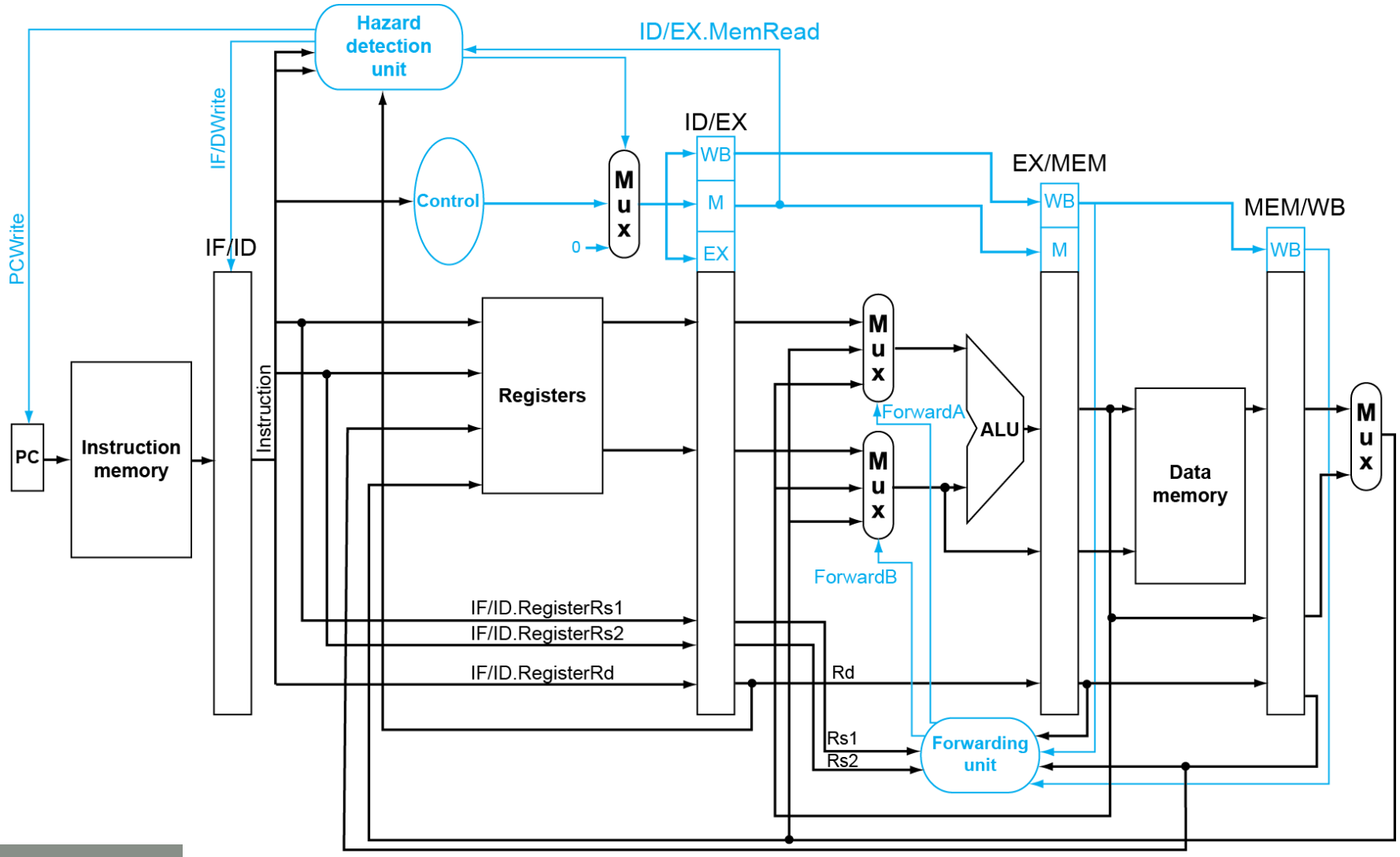
İletim ile Veri Yolu



Load Veri Tehlikesi



Veri Yolu ve Tehlike Tespiti



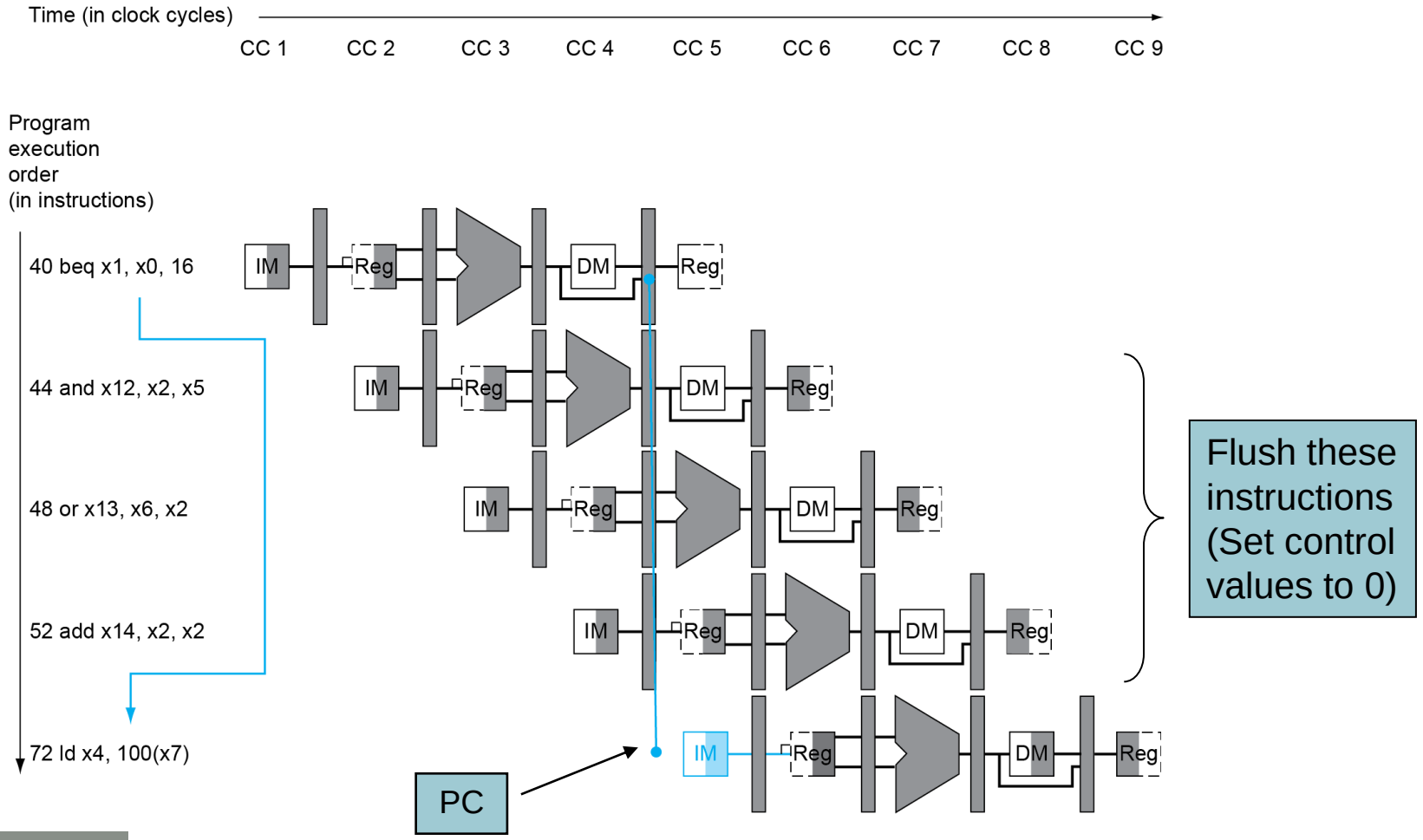
Duraklama ve Performans

Büyük Rsim

- Duraklama performansı düşürür
  - Ancak doğru sonuçlar almak için gereklidir
- Derleyici, tehlikeleri ve duraklamaları önlemek için kod düzenleyebilir
  - Ardışık düzen yapısı hakkında bilgi gerektirir

Branch Hazards

- If branch outcome determined in MEM



Durağan Öngörü

Dallanmanın koşulundan bağımsız olarak sonucu hep **atlar** ya da **atlamaz** tahmin eden öngörücüler durağan öngörücülerdir.

Örnek: Aşağıdaki kod parçası için durağan **atlamaz** öngörücünün başarımı nedir?

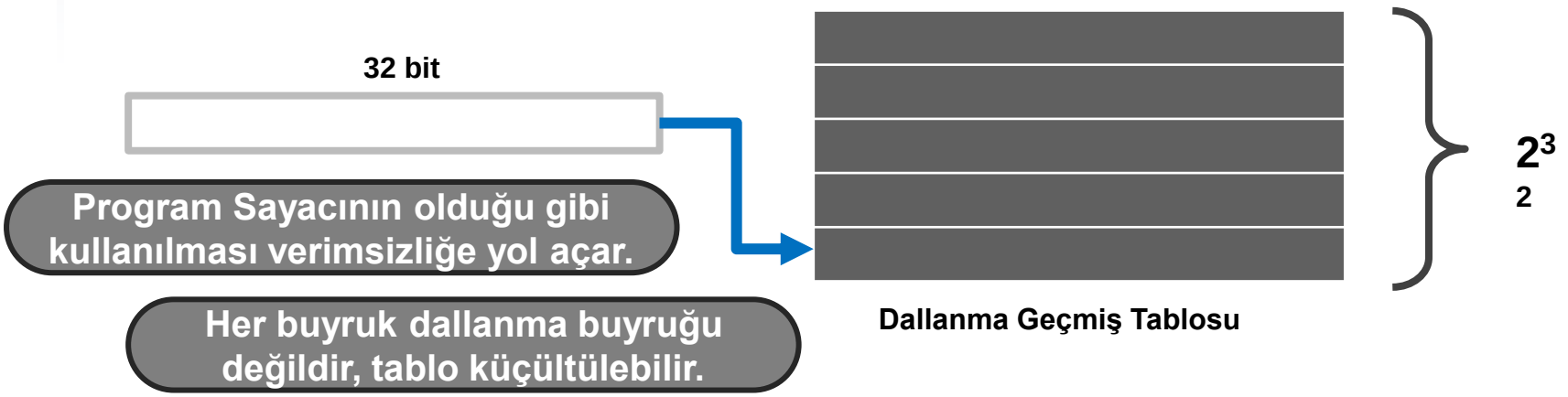
```
int i = 0;
while(i<10) {
        …
        i++;
}
```

Peki öngörücü durağan atlar öngörü yapsaydı başarım ne olurdu?

Devingen Öngörü

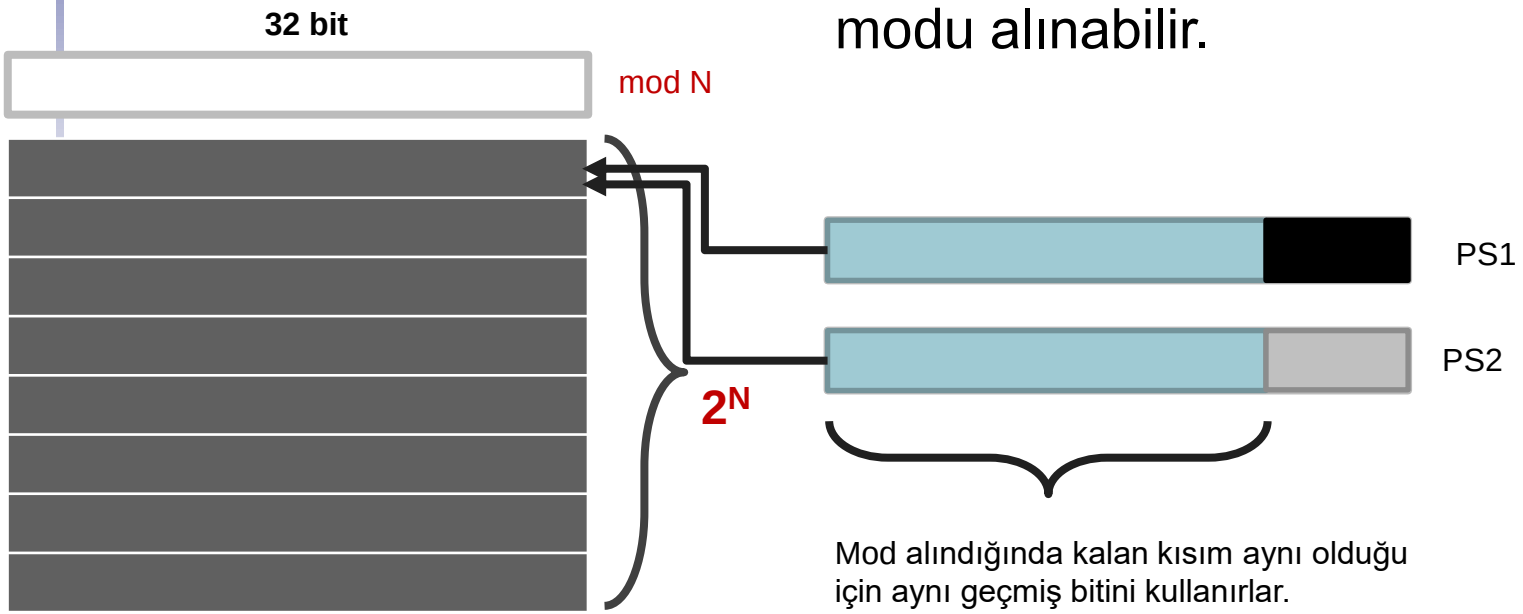
Devingen bir öngörü yapmak için program akışı ile ilgili bilgileri kullanmak gerekmektedir.

En kolay yöntemlerden biri bir önceki dallanmanın atlayıp atlamadığına bakılmasıdır (*ing.* Last value prediction).



Geçmişe Bağlı Öngörücü

Dallanma geçmiş tablosunu küçültmek için program sayacının modu alınabilir.



Dallanma Geçmiş Tablosu

N artırılarak bu sorun çözülebilir

Geçmişe Bağlı Öngörücü

```
for (int i =0;i<100;i++){
    if(i%2==0) {
        …
    }
}
```

1. Atlar durumunda for döngüsünden çıkar.
2. Atlamaz durumunda if'in içine girer.

1 ve 2 dallanma buyrukları için bir önceki dallanmaya bakarak tahmin yapan devingen bir öngörücünün doğru tahmin yapma oranı nedir?

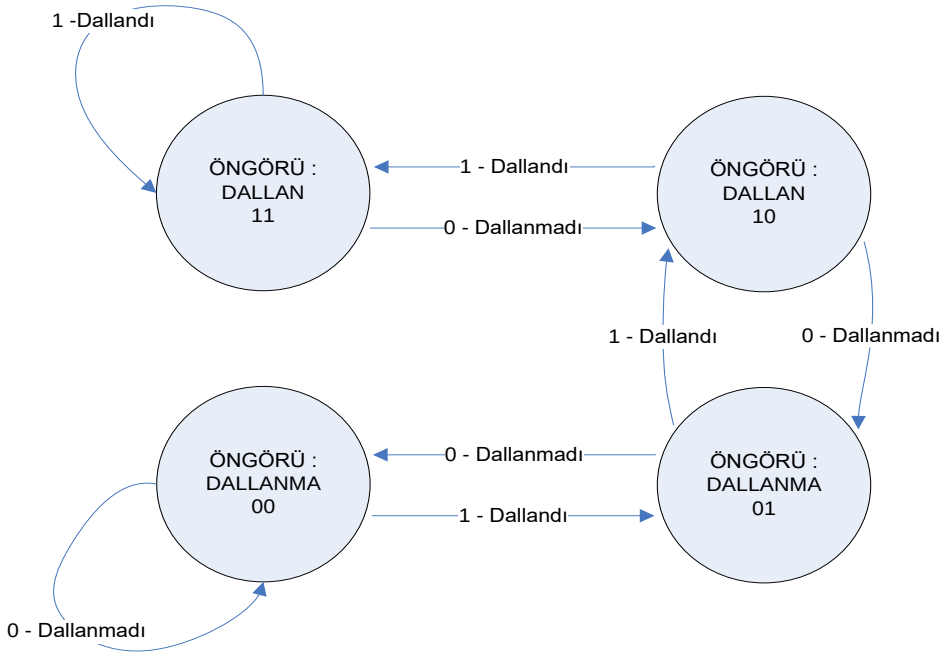
(İlk başta öngörücünün atlamaz tahmini yaptığını kabul edebilirsiniz.)

1. dallanma buyruğu için öngörücü ilk 101 kere atlamaz tahmini yapar, 100 kez doğru tahmin etmiş olur. **100 / 101**

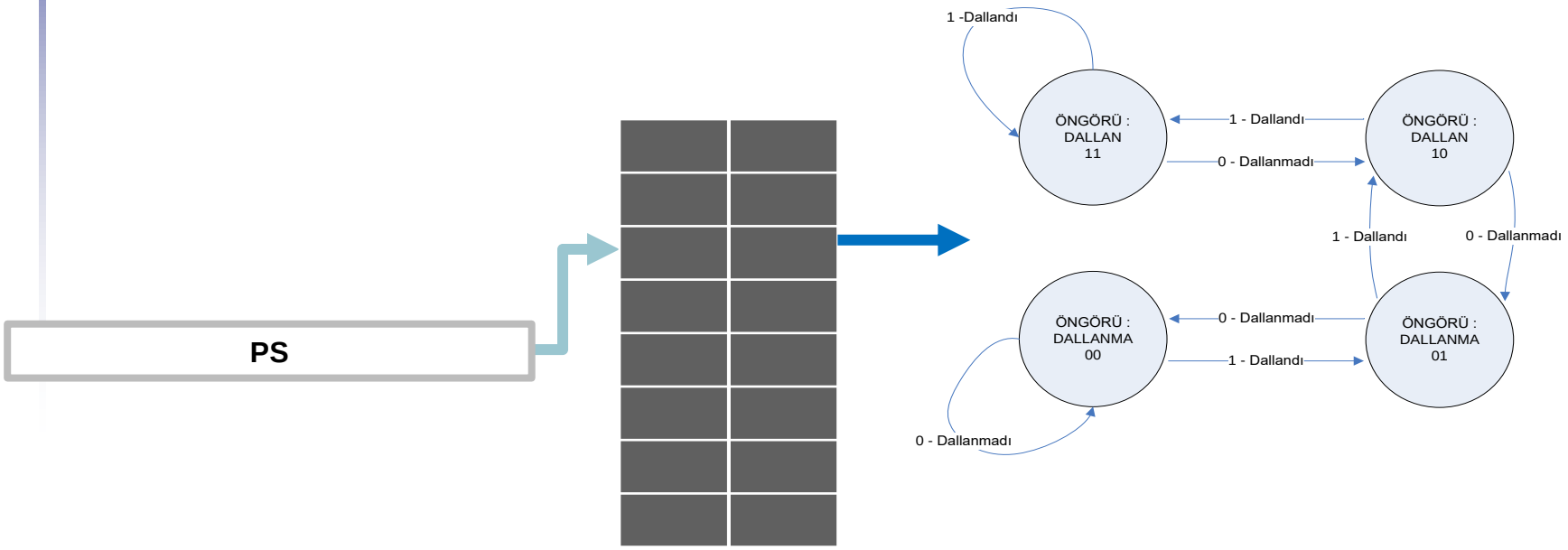
2. dallanma buyruğu için öngörücü ilk olarak atlamaz tahmini yapar ve yalnızca bu tahmini doğrudur. **1 / 100**

Çift Doruklu Öngörücü

Tek bit geçmiş ile yapılan öngörüler istenen hassaslığı her durumda karşılayamayabilir. Bu durumda 2 bitlik sayaçlar kullanılır. Bu öngörücülere **çift doruklu öngörücü** denir.



Çift Doruklu Öngörücü



Dallanma Geçmiş Tablosu

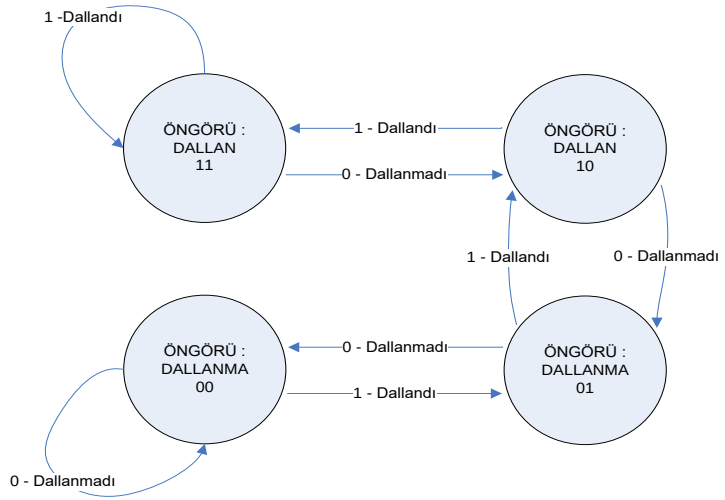
Çift Doruklu Öngörücü

Bu öngörücü kullanıldığında **ikinci dallanmanın** doğru öngörülme oranı ne olur?

```
for (int i =0;i<100;i++){
    if(i%2==0) {
        …
    }
}
```

1 Atlar durumunda for döngüsünden çıkar.

2 Atlar durumunda if'in içine girmez.

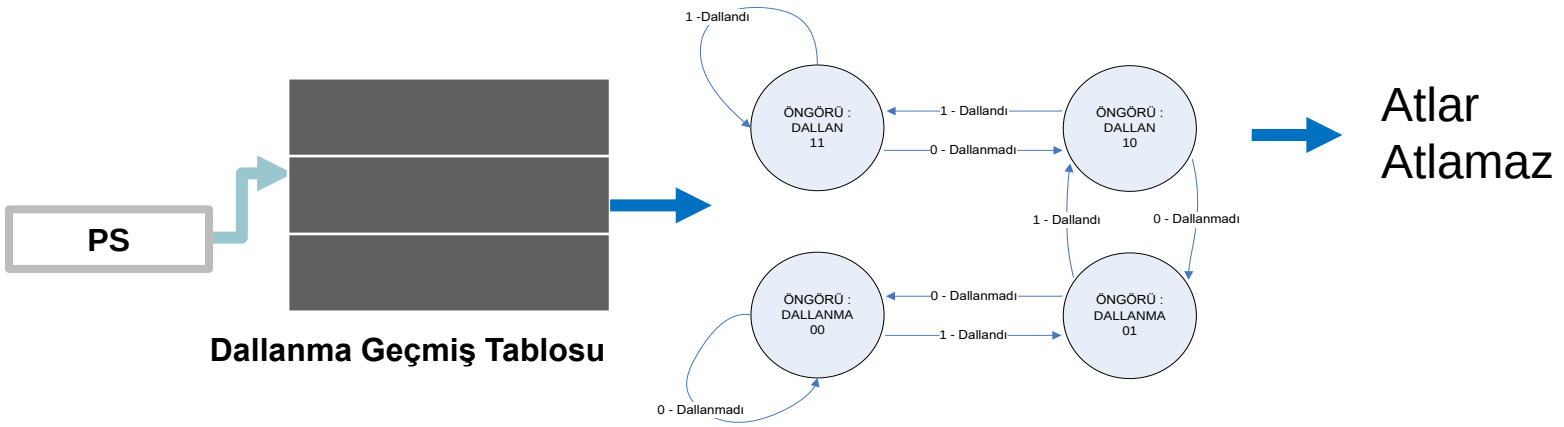


Tahmin	Atlamaz	Atlamaz	Atlamaz	Atlamaz	Atlamaz	…	Atlamaz
Dallanma	Atlamadı	Atladı	Atlamadı	Atladı	Atlamadı	…	Atladı

%50 doğru tahmin

Dallanma Hedef Ara Belleği

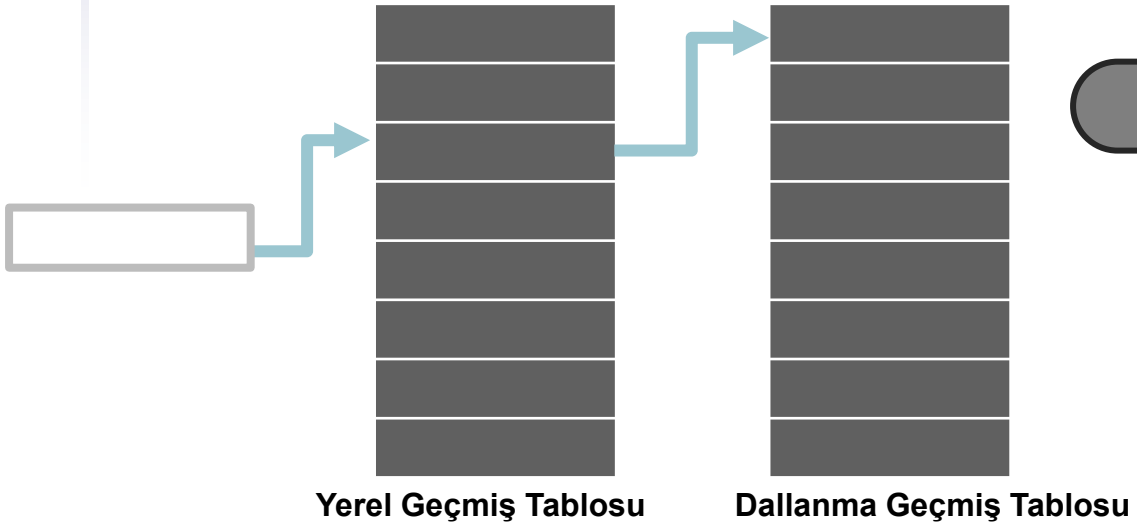
Dallanma öngörücü sadece her satırı iki bit sayaçlardan oluşan bir tablo olarak tasarlanabilir. Bu şekilde öngörücü bize yalnızca dallanmanın atlayacağı ya da atlamayacağı tahminini verebilir.



Yerel Geçmişe Dayalı Öngörücüler

Yerel Geçmiş (*ing.* local history**):** Her dallanmanın kendisine ait geçmişidir.

Yerel geçmişe dayalı öngörücüler yalnızca dallanmanın kendi geçmişi bilgisi ile öngörü yaparlar.



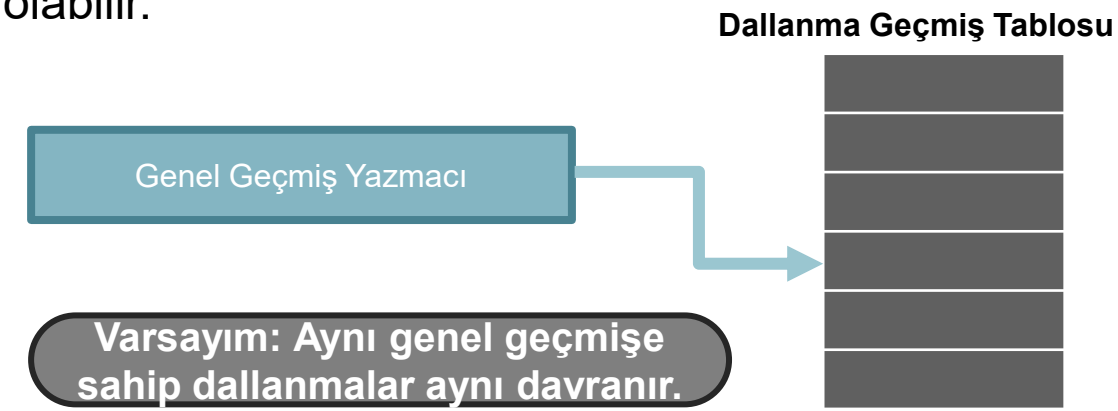
Varsayım: Geçmişi aynı olan dallanmalar aynı davranır.

Genel Geçmişe Dayalı Öngörücüler

Genel Geçmiş (*ing.* global history): Programın içindeki dallanmaların beraber geçmişidir.

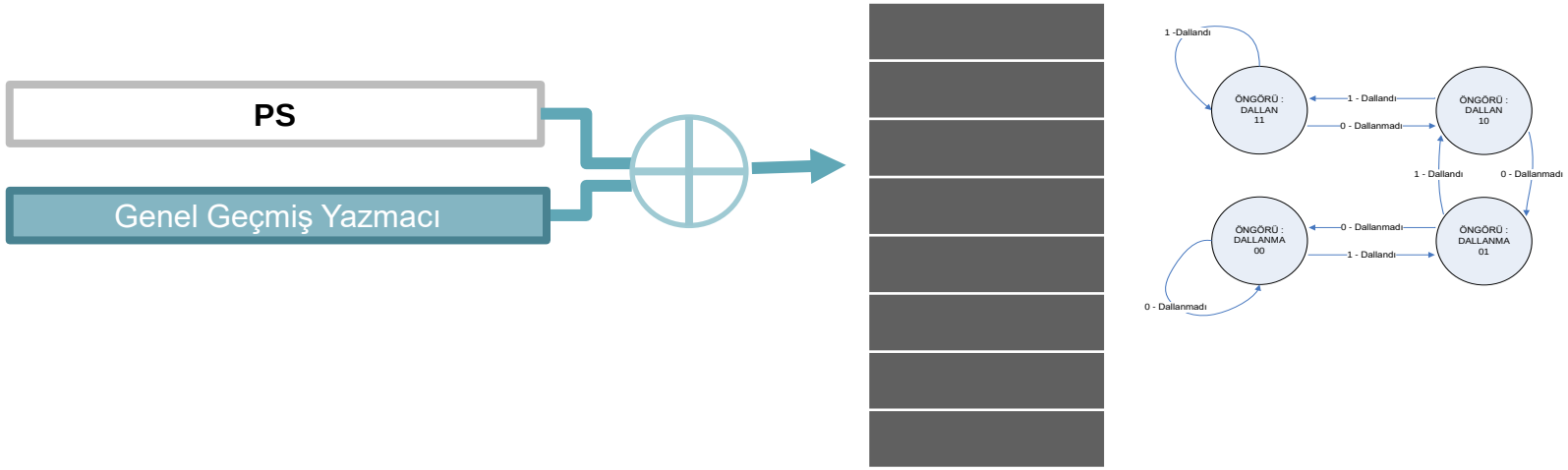

```
...
if(a>5) {
            ...
}
if(a<5) {
            ...
}
if(a==5) {
            ...
}
...
```

Bir dallanmanın öngörüsü kendinden önceki dallanmalara bağlı olabilir.



Yerel ve Genel Geçmişi Beraber Kullanmak

Hem yerel hem de genel geçmiş bilgilerini beraber kullanan öngörücüler vardır.

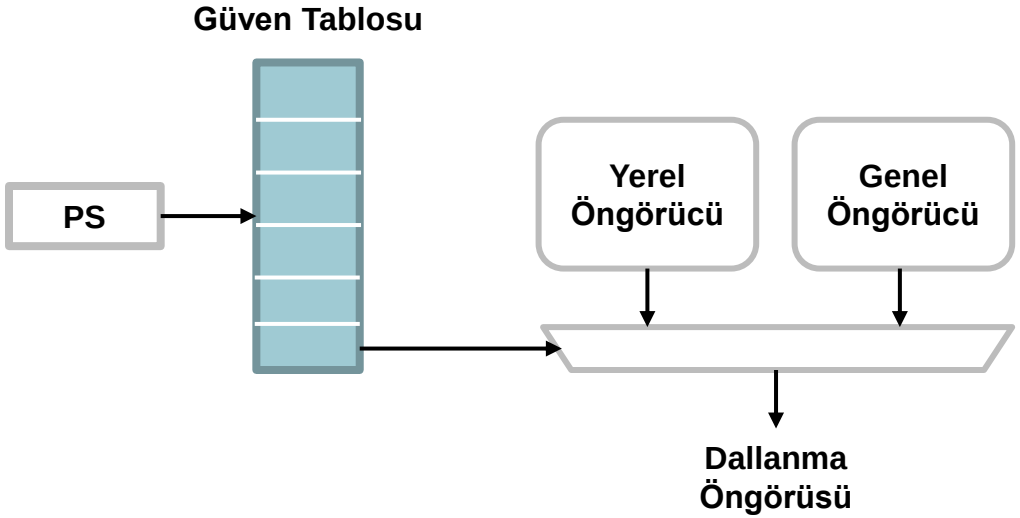


Dallanma Geçmiş Tablosu

Birden Çok Öngörücüyü Beraber Kullanmak

Birden çok öngörücü kullanıp her dallanma için en iyi öngörücüyü seçen dallanma öngörücülerine **turnuva öngörücüleri** (-ing. tournament predictors) denir.

- Her dallanma ve öngörücü kombinasyonu için bir **güven** değeri tutar.



Power Efficiency

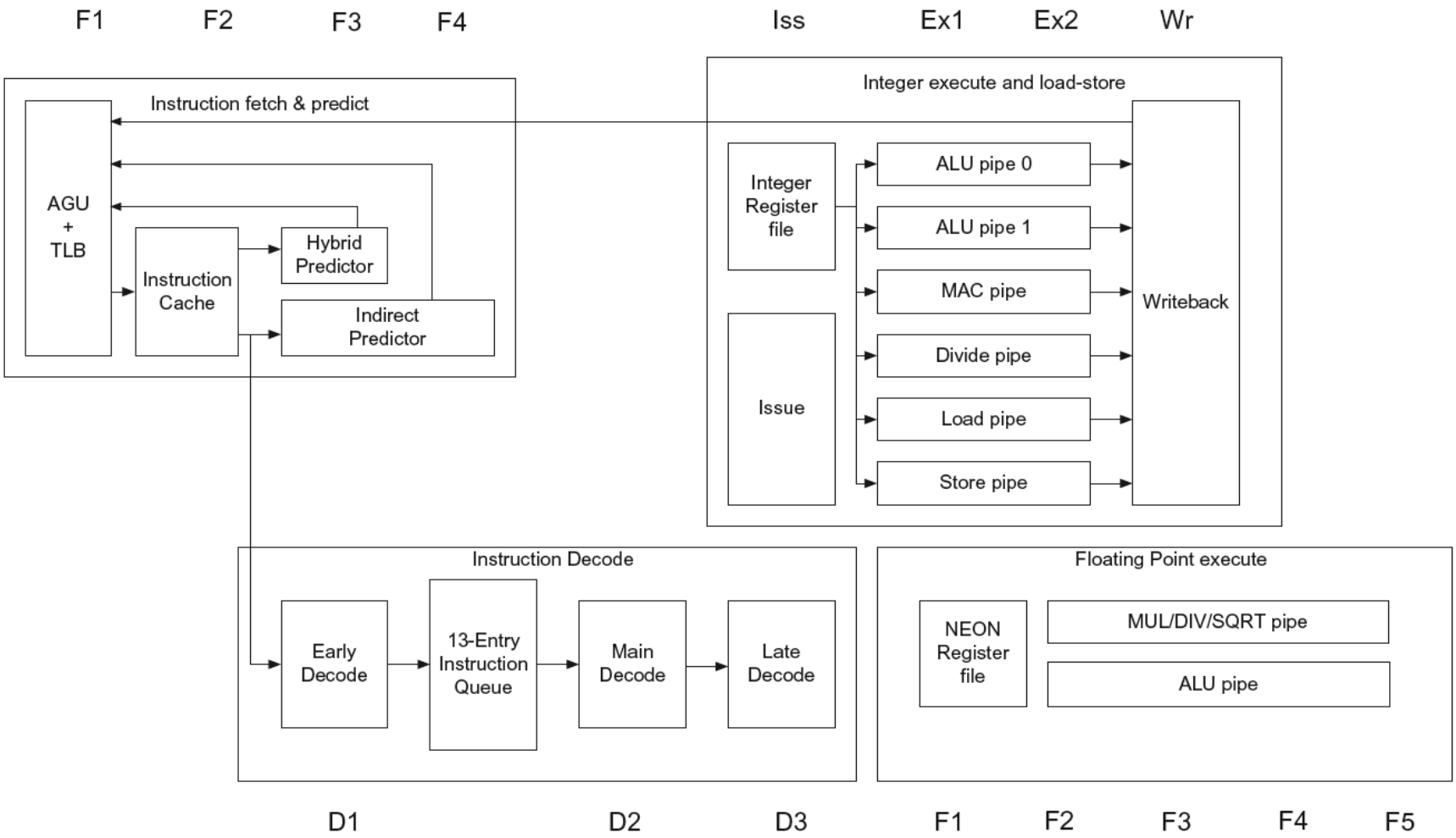
- Dinamik zamanlamanın karmaşıklığı güç gerektirir
- Birden fazla işlemci çekirdeği daha iyi olabilir

Microprocessor	Year	Clock Rate	Pipeline Stages	Issue width	Out-of-order/ Speculation	Cores	Power
i486	1989	25MHz	5	1	No	1	5W
Pentium	1993	66MHz	5	2	No	1	10W
Pentium Pro	1997	200MHz	10	3	Yes	1	29W
P4 Willamette	2001	2000MHz	22	3	Yes	1	75W
P4 Prescott	2004	3600MHz	31	3	Yes	1	103W
Core	2006	2930MHz	14	4	Yes	2	75W
UltraSparc III	2003	1950MHz	14	4	No	1	90W
UltraSparc T1	2005	1200MHz	6	1	No	8	70W

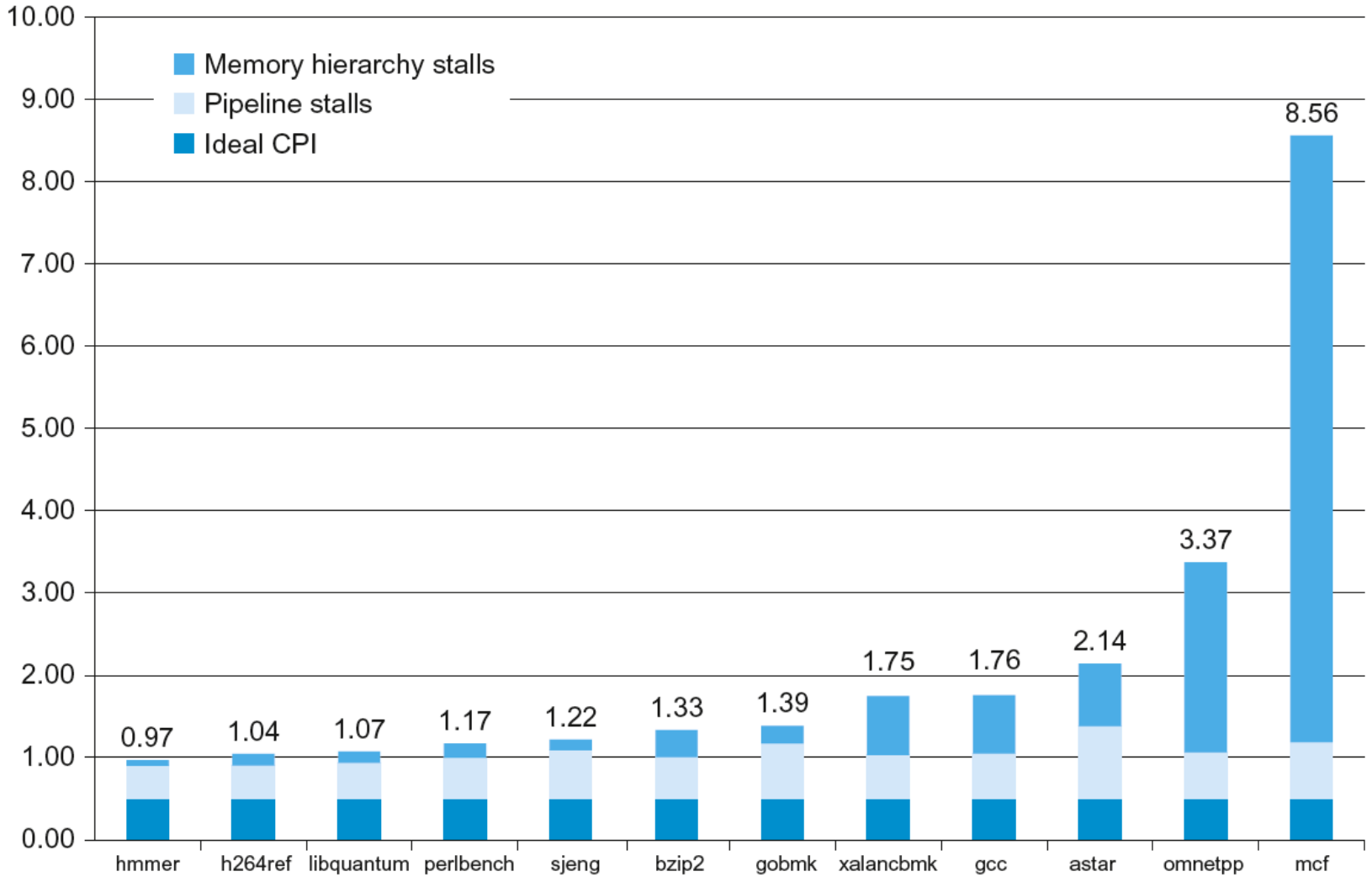
Cortex A53 and Intel i7

Processor	ARM A53	Intel Core i7 920
Market	Personal Mobile Device	Server, cloud
Thermal design power	100 milliWatts (1 core @ 1 GHz)	130 Watts
Clock rate	1.5 GHz	2.66 GHz
Cores/Chip	4 (configurable)	4
Floating point?	Yes	Yes
Multiple issue?	Dynamic	Dynamic
Peak instructions/clock cycle	2	4
Pipeline stages	8	14
Pipeline schedule	Static in-order	Dynamic out-of-order with speculation
Branch prediction	Hybrid	2-level
1st level caches/core	16-64 KiB I, 16-64 KiB D	32 KiB I, 32 KiB D
2nd level caches/core	128-2048 KiB	256 KiB (per core)
3rd level caches (shared)	(platform dependent)	2-8 MB

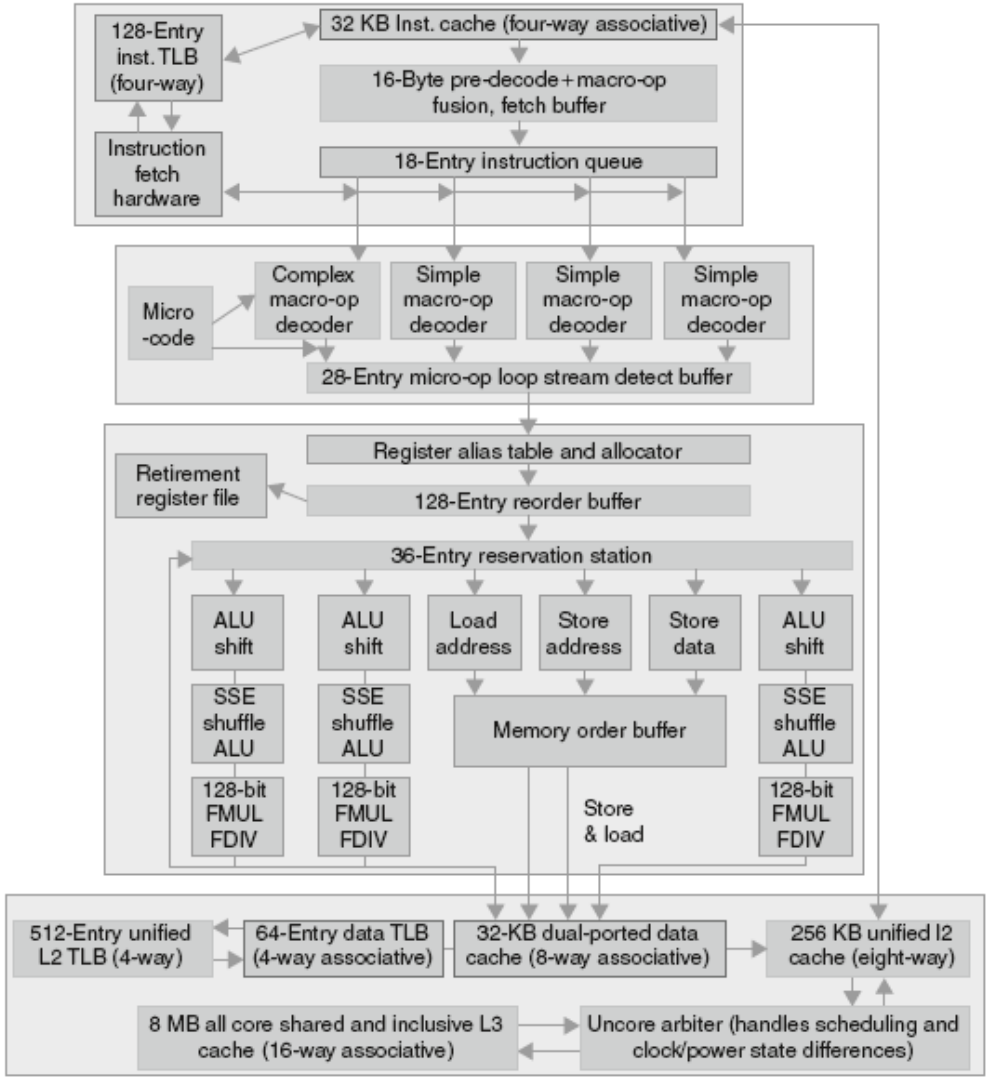
ARM Cortex-A53 Pipeline



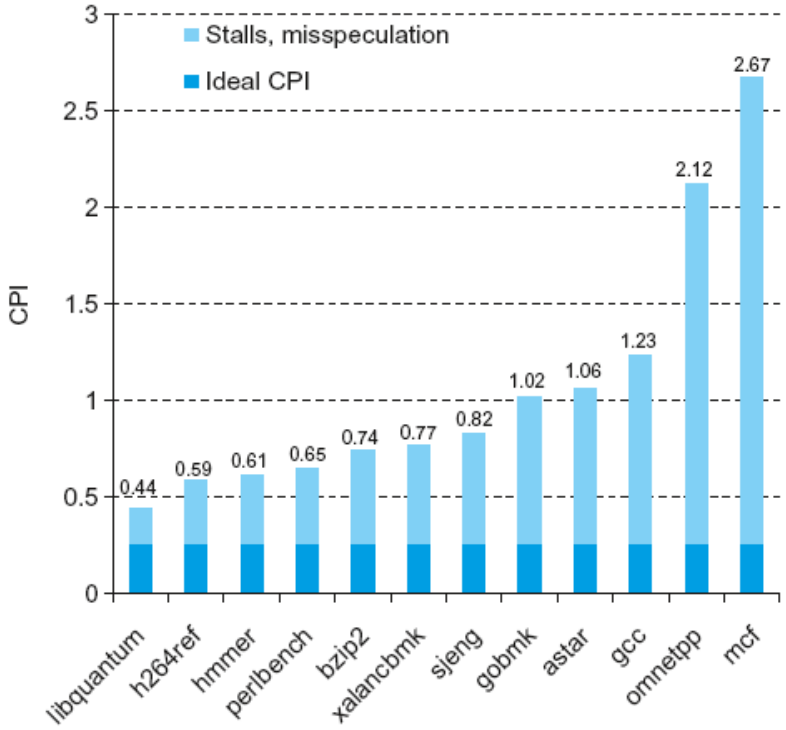
ARM Cortex-A53 Performance

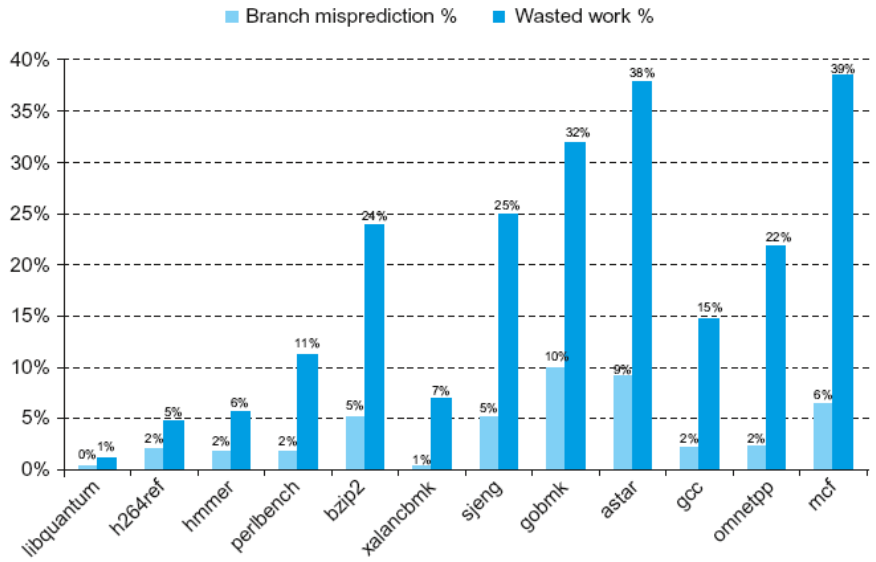


Core i7 Pipeline



Core i7 Performance





Kaynaklar

- Computer Organization and Design: The Hardware Software Interface [RISC-V Edition] David A. Patterson, John L. Hennessy
- TOBB üniversitesi, Prof Dr. Oğuz ERGİN, Bilgisayar Mimarisi ve Organizasyonu dersi ders sunumları