BM-311 Bilgisayar Mimarisi

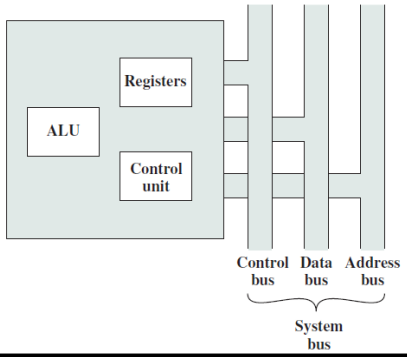
Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Konular

- Processor organization
- Register organization
- Instruction cycle
- Instruction pipelining

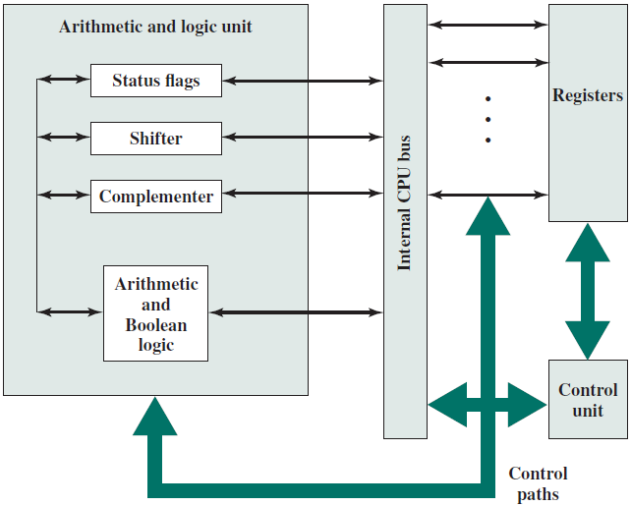
Processor organization

- İşlemci tarafından yapılan işler **fetch instruction**, **interpret instruction, fetch data, process data** ve **write data.**
- CPU, bu işlemleri yapabilmek için internal memory'ye ve internal bus'a ihtiyaç duyar.
- CPU, işlemleri **ALU** ve **kontrol birimi** ile gerçekleştirir.



Processor organization

- CPU internal bus, birimler arasındaki iletişimi sağlar.
- CPU'nun iç yapısı bilgisayarın yapısına benzer şekildedir.



Konular

- Processor organization
- Register organization
- Instruction cycle
- Instruction pipelining

Register organization

- **User-visible register'lar:** Programcı tarafından doğrudan kullanılan register'lardır.
- **Control ve status register'ları:** Kontrol birimi ve işletim sistemi tarafından kullanılan register'lardır (PC, IR, …).

User-visible register'lar

- **General purpose:** Programcı tarafından farklı işlevler için kullanılırlar.
- **Data register'ları:** Sadece veri saklamak için kullanılırlar.
- **Adres register'ları:** Adresleme modlarında kullanılırlar (segment pointer'ları, index register'ları, stack pointer).
- **Condition codes:** Flag olarak programın ve/veya işlemcinin durumunu saklar (carry, overflow, sign, zero, ...).

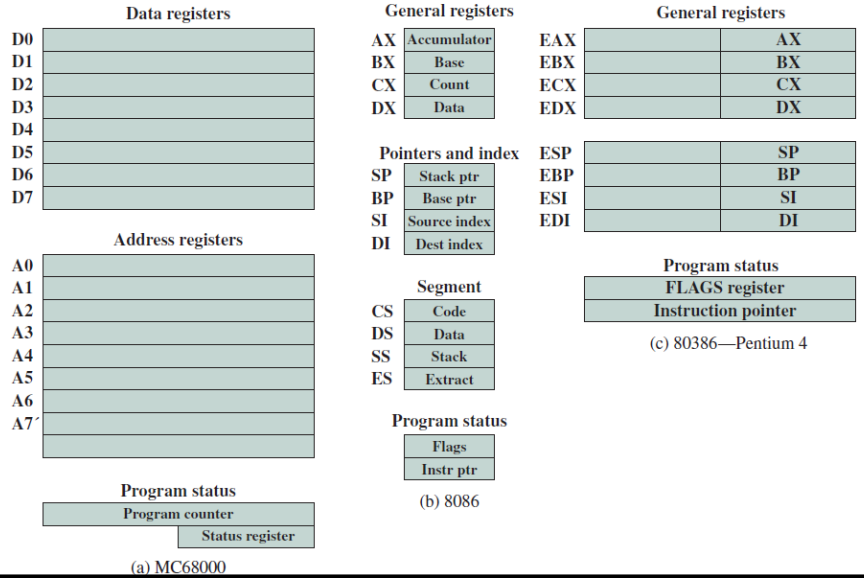
Register organization

Control ve status register'ları

- **Program counter:** Fetch edilecek komutun adresini saklar.
- **Instruction register:** En son fetch edilen komutu saklar.
- **Memory address register:** Hafızada bir yerin adresini saklar.
- **Memory buffer register:** Hafızaya yazılacak veriyi veya hafızadan son okunan veriyi saklar.
- Bir çok işlemci **program status word (PSW)** olarak durum bitlerini saklar (sign, zero, carry, equal, overflow, interrupt disable/enable, supervisor).
- **Supervisor mod:** Bazı CPU'lar belirli hafıza alanına bu mod ile erişim yapar.
- Supervisor mod, **genellikle işletim sistemi tarafından kullanılır.**

Register organization

Örnek

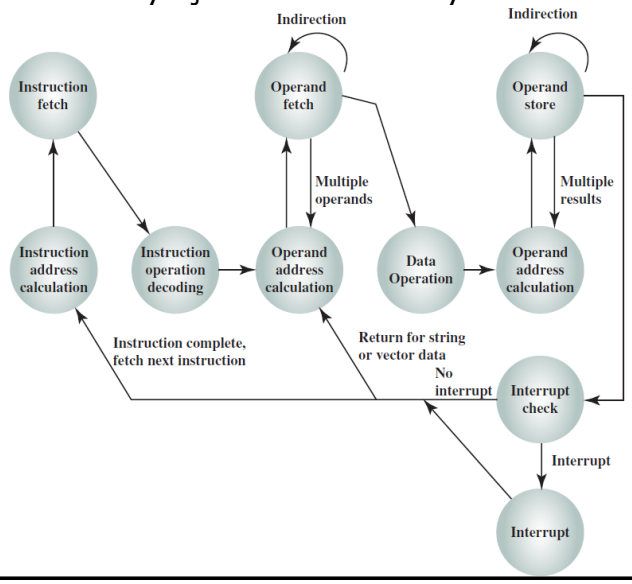


Konular

- Processor organization
- Register organization
- Instruction cycle
- Instruction pipelining

Instruction cycle

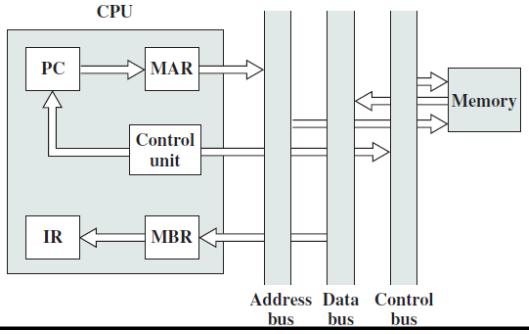
- Indirect adreslemeyi içeren instruction cycle



Instruction cycle

Data flow

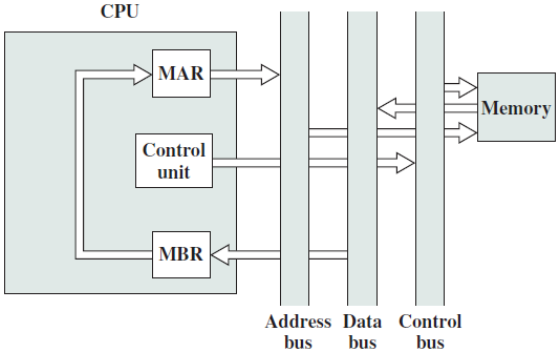
- **Instruction fetch aşamasında veri akışı**:
  - PC sonraki komutun adresini tutar.
  - PC'daki adres değeri MAR'a aktarılır.
  - MAR'a aktarılan adres bilgisi adres bus'a yerleştirilir.
  - Kontrol birimi memory read işareti üretir.
  - Sonuç data bus'a yerleştirilir, MBR'a kopyalanır, sonra IR'a aktarılır.
  - PC değeri 1 artırılır.



Instruction cycle

Data flow

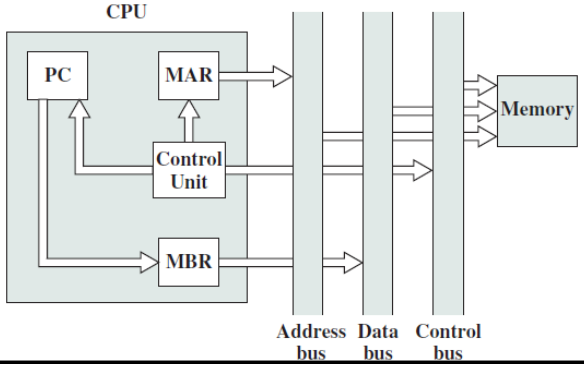
- **Indirect cycle aşamasında veri akışı:**
  - MBR'nin sağdaki N biti MAR'a aktarılır.
  - Kontrol birimi memory read işareti üretir.
  - Sonuç data bus'a yerleştirilir ve MBR'a kopyalanır.



Instruction cycle

Data flow

- **Interrupt cycle aşamasında veri akışı:**
  - PC değeri MBR aracılığıyla saklanır.
  - MBR hafızaya kopyalanır.
  - MAR'a özel bir adres değeri yüklenir (stack pointer veya ISR adresi).
  - PC'a ISR adresi aktarılır.
  - Sonraki instruction fetch edilir.



Konular

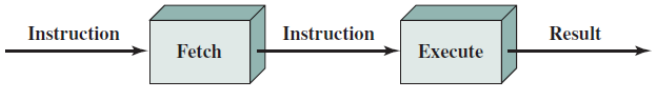
- Processor organization
- Register organization
- Instruction cycle
- Instruction pipelining

Instruction pipelining

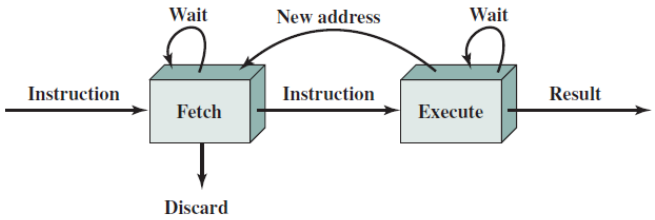
- Instruction pipelining bir üretim hattının çalışmasına benzer.
- Üretim hattında **farklı işler eş zamanlı yapılır.**
- Fetch aşamasında hafızaya erişilir, ancak execute aşamasında genellikle hafızaya erişilmez.
- Execute aşamasında **sonraki komutun fetch edilmesi (prefetch)** performansı artırır.
- **Fetch aşaması** execute aşamasından **çok kısa sürerse,** birden fazla komut prefetch yapılabilir.
- **Branch** ve **jump** komutlarında **gerekli olmayan komut prefetch yapılabilir.**

Instruction pipelining

- İki aşamalı instruction pipelining aşağıdaki gibidir.
- Fetch aşaması, **branch işlemleri için atlanacak adresi execute aşamasından alır.**
- Aşama sayısı arttıkça performans artar.



(a) Simplified view



(b) Expanded view

Instruction pipelining

- İki aşamalı instruction pipelining **6 aşama** olarak oluşturulabilir.
  - **Fetch instruction (FI):** Sonraki komut alınır.
  - **Decode instruction (DI):** Komut decode edilir.
  - **Calculate operands (CO):** Giriş operand'larının adresleri hesaplanır.
  - **Fetch operands (FO):** Giriş operand'ları alınır.
  - **Execute instruction (EI):** Komutun gerektirdiği işlem yapılır.
  - **Write operand (WO):** Sonuç operand varsa saklanır.
- Bazı aşamalar her komutta kullanılmaz (FO, WO).

Instruction pipelining

- Pipelining yaparak çalışma

Time

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	FI	DI	CO	FO	EI	WO								
Instruction 2		FI	DI	CO	FO	EI	WO							
Instruction 3			FI	DI	CO	FO	EI	WO						
Instruction 4				FI	DI	CO	FO	EI	WO					
Instruction 5					FI	DI	CO	FO	EI	WO				
Instruction 6						FI	DI	CO	FO	EI	WO			
Instruction 7							FI	DI	CO	FO	EI	WO		
Instruction 8								FI	DI	CO	FO	EI	WO	
Instruction 9									FI	DI	CO	FO	EI	WO

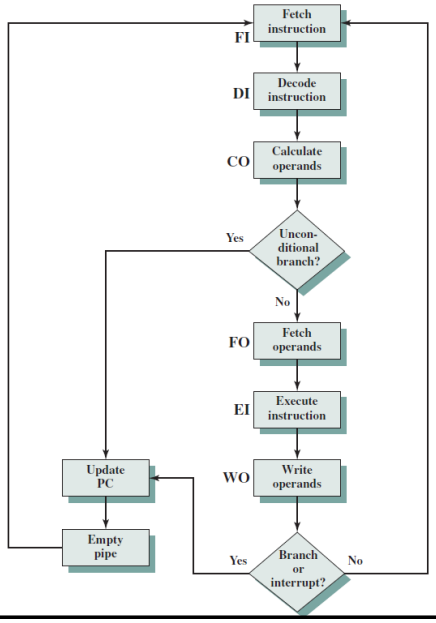
Instruction pipelining

- **Instruction 3 ile instruction 15'e atlama yapılıyor** (branch taken).

	Time →							Branch penalty ↔						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	FI	DI	CO	FO	EI	WO								
Instruction 2		FI	DI	CO	FO	EI	WO							
Instruction 3			FI	DI	CO	FO	EI	WO						
Instruction 4				FI	DI	CO	FO							
Instruction 5					FI	DI	CO							
Instruction 6						FI	DI							
Instruction 7							FI							
Instruction 15								FI	DI	CO	FO	EI	WO	
Instruction 16									FI	DI	CO	FO	EI	WO

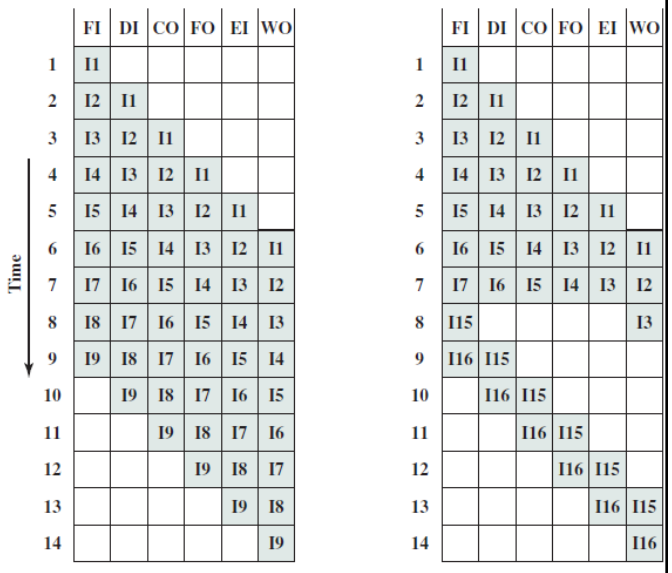
Instruction pipelining

- Instruction pipeline akış şeması



Instruction pipelining

- Zamana göre pipeline gösterimi
- **I3 ile I15'e atlama yapılıyor**



Instruction pipelining

- Pipeline ile performans artışı aşağıdaki gibi ifade edilir:

$$\tau = \max_i [\tau_i] + d = \tau_m + d \qquad 1 \le i \le k$$

burada,

τ_i = i. aşamadaki gecikme

τ_m = tüm aşamalardaki gecikmelerden maksimum olan

k = pipeline'daki aşama sayısı

d = verinin aşamalar arasındaki aktarımı için geçen süre

$\tau_m >> d$ olduğu için d ihmal edilir.

$T_{k,n}$, n satır programın k aşamalı pipeline ile çalışma süresidir.

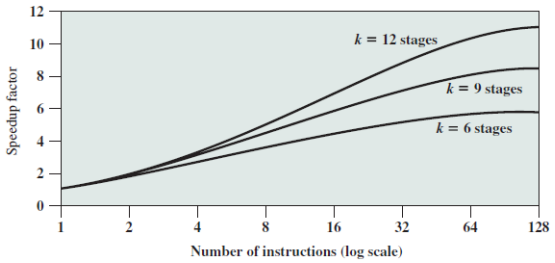
$T_{k,n} = [k + (n - 1)].\tau$ şeklinde gösterilir. Performans artışı ise;

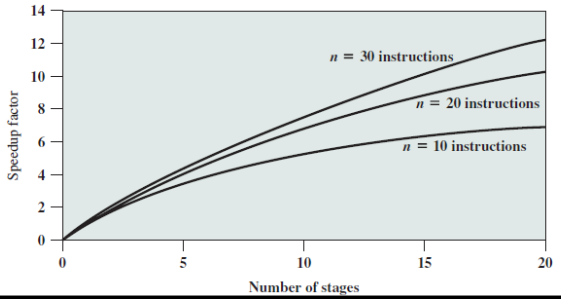
$$S_k = \frac{T_{1,n}}{T_{k,n}} = \frac{nk\tau}{[k + (n - 1)]\tau} = \frac{nk}{k + (n - 1)}$$

*$n \to \infty$ giderken performans artışı **k** olur.*

Instruction pipelining

- Pipelining ile komut sayısı ve aşama sayısına göre performans artışı





Instruction pipelining

- Branch komutlarında **atlanacak yerin önceden bilinememesi performansı düşürür.**
- **Gereksiz alınmış komutların** pipeline'dan **atılması gerekir.**
- Atlama komutu çalışmadan **sonraki komutun hangisi olacağı bilinemez.**
- **Şartlı atlama işlemlerinde** sonraki komutun tahminine yönelik kullanılan yaklaşımlar:
  - **Multiple streams**
  - **Prefetch branch target**
  - **Loop buffer**
  - **Delayed branch**
  - **Branch prediction**

Instruction pipelining

Multiple streams

- Pipeline'da **iki yöndeki komutlar fetch** edilerek çalışma devam eder.
- Birden fazla atlama komutu pipeline'a girerse **her atlama komutu için bir stream gerekir.**

Prefetch branch target

- Şartlı atlama komutu geldiğinde **sonraki komut** ve **hedef komut birlikte prefetch edilir.**
- **Target** saklanır ve **branch taken olursa kullanılır.**

Instruction pipelining

Loop buffer

- En son çalıştırılan **döngüye ait fetch edilen n adet komutu saklar.**
- Komutlar sıralı saklanır ve hafıza erişimi olmadan alınır.
- Atlama aralığı az olursa veya **döngü kısa olursa sürekli buffer üzerinde çalışılır.**

Delayed branch

- **Komutların sırası yeniden düzenlenir.**
- Bir atlama komutu ile başka komut yer değiştirilir.
- **Bağımsız bir veya birkaç komut,** atlama komutu ile ardındaki komutlar arasına alınır.

Instruction pipelining

Branch prediction

- Bir atlama komutunun **taken olup olmayacağı tahmin edilir.** Atlama tahmini için,
  - **Predict never taken**
  - **Predict always taken**
  - **Predict by opcode**
  - **Taken/not taken switch**
  - **Branch history table**

  kullanılır.
- **İlk üç yöntem statiktir** ve programın çalışmasına bağlı değildir.
- **Son iki yöntem dinamiktir** ve programın çalışması sırasında karar verilir.

Instruction pipelining

Branch prediction – predict never taken

- Atlamaların **hiçbir zaman taken olmayacağı** varsayılır.
- Yapılan deneysel çalışmalarda **%50'den fazla** atlamanın **taken** olduğu görülmüştür.

Branch prediction – predict always taken

- Atlamaların **her zaman taken olacağı** varsayılır.
- Never taken yöntemine göre daha başarılı sonuç alınır.

Branch prediction – predict by opcode

- Atlamalar **opcode'lara göre taken** ve **not taken** şeklinde gruplandırılır.
- **Başarı oranı %75'den büyüktür.**

Instruction pipelining

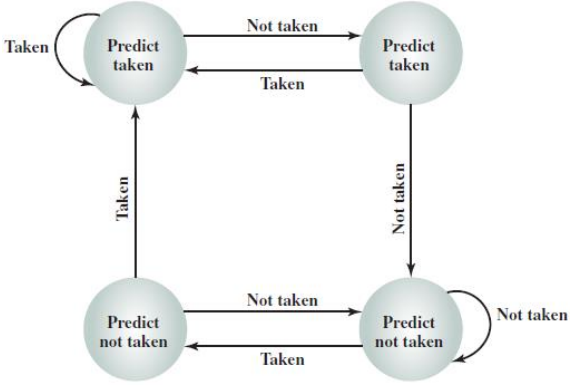
Branch prediction – taken/not taken switch

- Atlama tahmini önceki çalışma durumuna bağlıdır.
- Bir veya birden fazla bit ile **her komutun önceki durumu saklanır.**
- **Bir bitle** taken ve not taken arasında **her hatalı atlamada sürekli switch yapılır.**
- **Bir bitle** saklandığı zaman **her döngünün başında** ve **sonunda hata yapılır.**
- **İki bitle iki hata üst üste yapıldığında switch yapılır.**
- Art arda gelen **döngülerin ilk iterasyonundaki hatalar ortadan kaldırılır.**

Instruction pipelining

Branch prediction – taken/not taken switch

- İki bitle yapılan taken ve not taken arasındaki switch aşağıdaki gibidir.

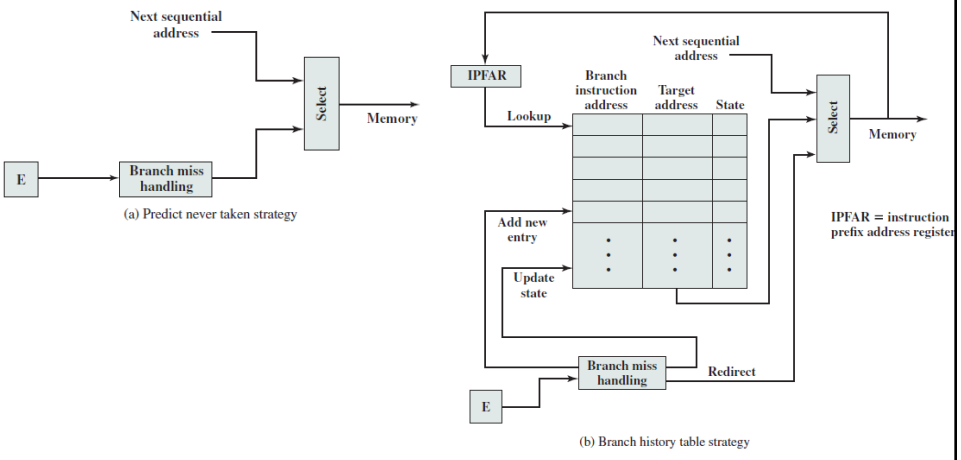



```
for (j=1; j<=4;j++)
{
      a[j]=3;
      a[j]=a[j]*2*h;
};
:
for (k=1; k<=4; k++)
{
      b[k]=6;
      b[k]=b[k]+3*k*k;
};
```

Instruction pipelining

Branch prediction – branch history table

- **E:** Execute aşaması, **IPFAR:** Instruction Prefix Address Register



Instruction pipelining

Branch prediction – branch history table

- Branch history table (BHT), **fetch aşamasıyla ilişkilendirilmiş küçük bir önbellektir.**
- **Her prefetch BHT'de bir arama tetikler.**
- Kayıt bulunamazsa, **next sequential address fetch edilir.**
- Kayıt bulunursa, **BHT'deki tahmine göre işlem yapılır** (sonraki sıralı adres veya hedef atlama adresi alınır.).
- Branch komutu çalıştırıldığında, **execute aşaması BHT'yi bilgilendirir.**
- Komutun durumu, **tahminin doğruluğuna göre güncellenir.**
- **Tahmin yanlış ise,** select birimi **doğru komuta redirect yapar.**
- **Şartlı atlama komutu BHT'de yoksa** mevcut bir komut atılarak **eklenir** (replacement algoritmaları).