Örnek Bir 5 Katmanlı RISC İş hattı

Bu derste, iş hattı ile ilgili kavramları açıklamak için örnek bir 5 katmanlı RISC iş hattı kullanılacaktır.

1. *Instruction fetch* (IF):

   Sıradaki komutu bellekten al, program sayacını (PC) arttır (komut uzunluğu kadar). Eğer bir komut 4 sekizli ise, PC ← PC + 4.

2. *Instruction Decode, Read registers* (DR)

   Komutu çözerek tüm birimlere gidecek olan denetim işaretlerini oluştur ve saklayıcı dosyasında ilgili saklayıcıları (operandları) oku.

3. *Execute* (EX)

   ALU işlemlerini yap, dallanma komutlarının hedef adreslerini hesapla.

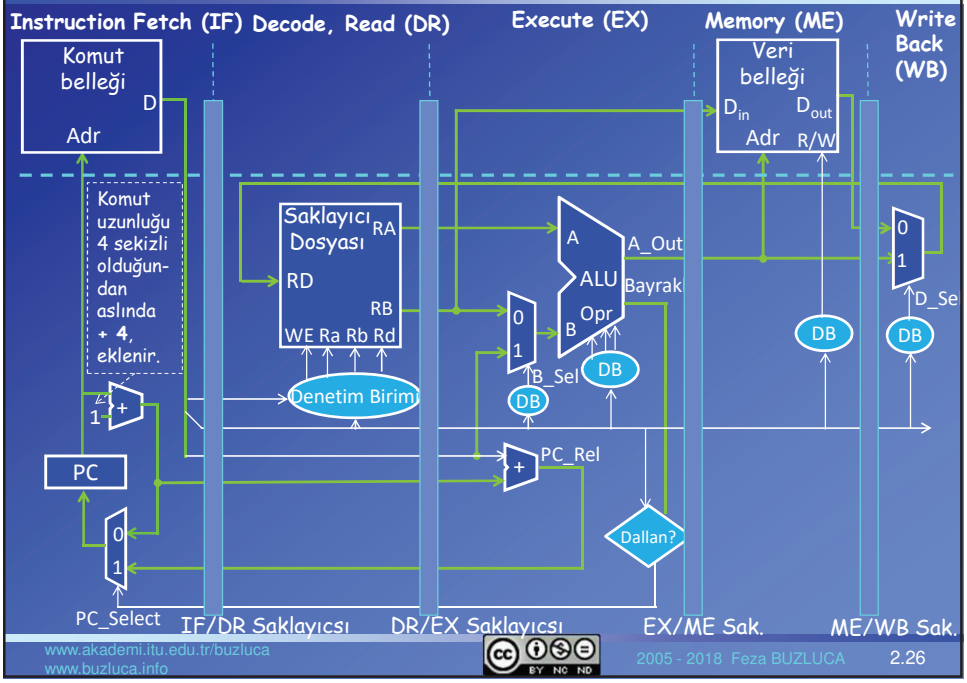
4. *Memory* (ME)

   Eğer gerekli ise veri belleğine eriş (sadece load/store komutları)

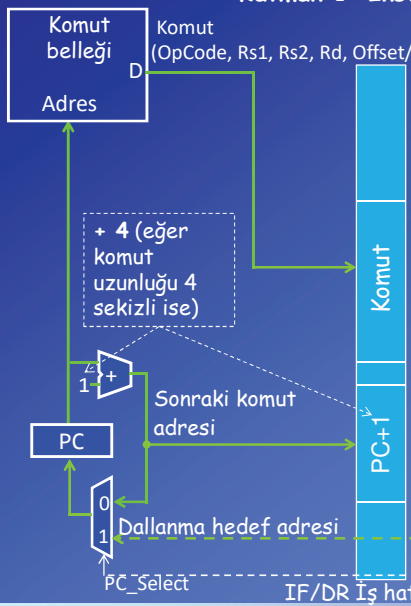
5. *Write back* (WB)

   Sonuçları saklayıcı dosyasına yaz.

Beş Katmanlı Bir RISC İş Hattı



Katman 1: *Instruction Fetch* (IF), Komut Alma

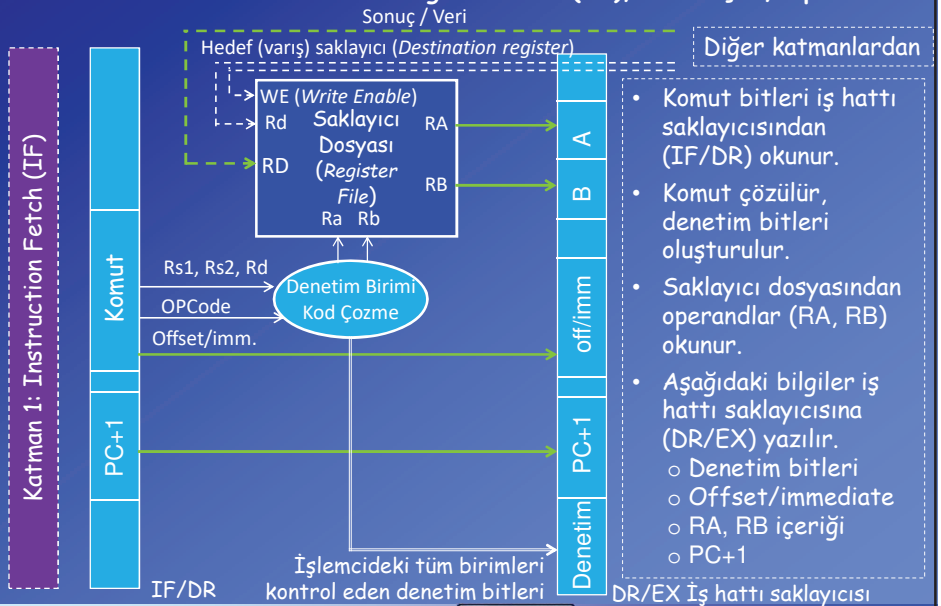


PC'nin o andaki değeri sıradaki komutun komut belleğindeki adresini gösterir

- Komut, komut belleğinden okunur.
- PC arttırılır (PC_Select=0; şimdilik dallanma olmadığı varsayılıyor).
- Komutu oluşturan bitler (op code, Rs1, Rs2, Rd, S2, offset) iş hattı saklayıcısına (IF/DR) yazılır.
- PC+1 değeri iş hattı saklayıcısına (IF/DR) yazılır. Daha sonraki katmanlarda dallanma adresi hesabında kullanılır.
- Eğer dallanma varsa, PC_Select=1 olur, dallanma hedef adresi PC'ye yazılır.

Diğer katmanlardan

Katman 2: *Instruction Decode and Register Read* (DR), Komut Çöz, Operand al

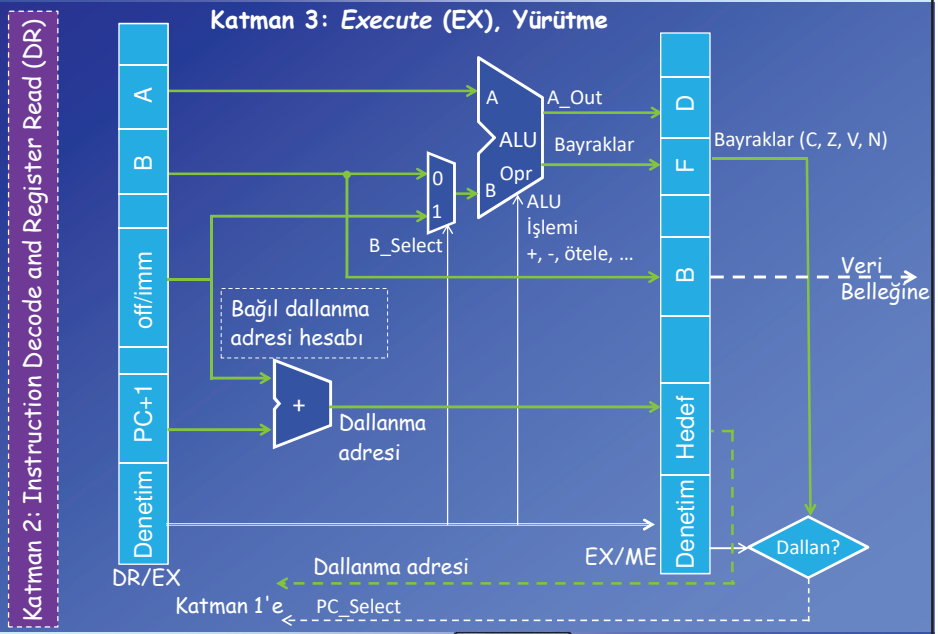


- Komut bitleri iş hattı saklayıcısından (IF/DR) okunur.
- Komut çözülür, denetim bitleri oluşturulur.
- Saklayıcı dosyasından operandlar (RA, RB) okunur.
- Aşağıdaki bilgiler iş hattı saklayıcısına (DR/EX) yazılır.
  - Denetim bitleri
  - Offset/immediate
  - RA, RB içeriği
  - PC+1

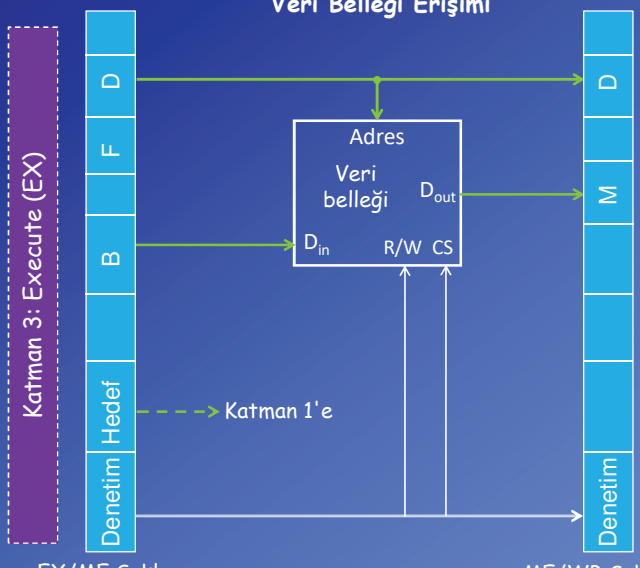
Katman 3: *Execute* (EX), Yürütme

- Denetim bitleri ve veriler (offset/immediate, RA, RB) iş hattı saklayıcısından (DR/EX) okunur.
- ALU işlemleri yapılır.
  ALU, LOAD/STORE komutları için gerekli olan adres hesaplarını da yapar.
  Örneğin; LDL $500(R4), R5 *R5 ← M[R4 + $500]*
  İvedi veri $500 ile R4 saklayıcısını içeriği ALU tarafından toplanır.
- Dallanma komutları için dallanma hedef adresi hesaplanır.
  Örneğin; BGT $0A *Büyükse, PC ← PC + $0A*
  Bu örnek işlemcide, dallanma hedef adresi hesabı için ALU'dan ayrı bir toplayıcı kullanılmaktadır.
- Dallanma olup olmayacağına karar verilir (denetim bitleri ve ALU'dan gelen bayrak değerleri kullanılır).
- Aşağıdaki bilgiler iş hattı saklayıcısına (EX/ME) yazılır.
  - Denetim bitleri
  - ALU'da oluşan sonuç (D) ve bayraklar (F)
  - Belleğe yazma işlemi için RB değeri (B)
  - Dallanma hedef adresi (Hedef)

Katman 3: *Execute* (EX), Yürütme

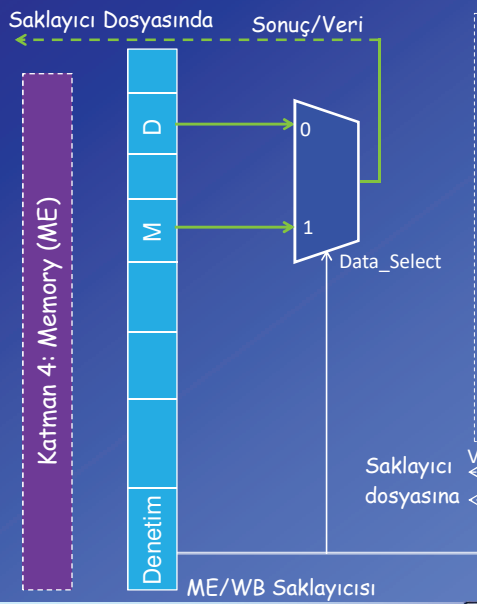


Katman 4: Memory (ME), Veri Belleği Erişimi



- Veri belleği adresi D (ALU'dan gelen sonuç) iş hattı saklayıcısından (EX/ME) okunur.
- Veri B (STORE komutu için) iş hattı saklayıcısından (EX/ME) okunur.
- Eğer gerekli ise bellek okuma/yazma işlemi yapılır.
- Aşağıdaki bilgiler iş hattı saklayıcısına (ME/WB) yazılır.
  - Denetim bitleri
  - Bellek erişimi sonucu (M)
  - ALU'dan gelen sonuç (aktarma) (D)

Katman 5: Write Back (WB), Saklayıcılara Yazma



- ALU'dan gelen sonuç (D) iş hattı saklayıcısından (ME/WB) okunur.
- Bellek erişiminin sonucu (M) iş hattı saklayıcısından (ME/WB) okunur.
- Veri seçilerek (M veya D) saklayıcı dosyasındaki hedef saklayıcıya yazılır.
- Denetim bilgileri (Rd, WE) saklayıcı dosyasına gönderilir.
- Saklayıcı dosyasına yazılır.
- Saklayıcı dosyasına 2. katman okumak, 5. katman ise yazmak için erişir.

Örnek RISC iş hattının zaman diyagramı (ideal durum):

İdeal Durum: Programda dallanma ve veri bağımlılığı yoktur.

Saat çevrimi / Komutlar	1	2	3	4	5	6	7	8
1	IF	DR	EX	ME	WB			
2		IF	DR	EX	ME	WB		
3			IF	DR	EX	ME	WB	
4				IF	DR	EX	ME	WB

İlk komut tamamlandı.
5 çevrim
İş hattı doldu.

Bir saat çevrimi sonra
ikinci komut tamamlandı.

İlk komut 5 çevrim sonunda tamamlandı (k = 5).

5nci çevrimden sonra her çevrimde yeni bir komut tamamlanır.

Komut sayısı sonsuza yaklaştığında bir komutun tamamlanma süresi de 1 saat çevrimine yaklaşır (yansı 2.9 "Hızlanma").

IF ve ME katmanları aynı anda belleğe erişmek isterler.

Bu bellek çatışması sorununu çözmek için komut ve veri bellekleri ayrılmıştır (Harvard mimarisi).

2.5 İş Hattı Sorunları (*Pipeline Hazards (Conflicts)*) ve Çözümleri

İş hattında 3 tür sorunla karşılaşılır.

1. Kaynak Çatışması (*Resource Conflict*), Yapısal Sorun (*Structural Hazard*):

İş hattında aynı anda işlenen iki komut aynı kaynağa (bellek, ALU) gerek duyarsa kaynak çatışması oluşur.

2. Veri Çatışması (*Data Conflict*), Veri Bağımlılığı (*Data Dependency*):

Henüz güncellenmemiş olan veri erkenden kullanılmak istenirse sorun ortaya çıkar.

3. Denetim Sorunları (*Control Hazards*) Dallanma (*Branch*), Kesme (*Interrupt*):

İş hattında bir dallanma komutu işlenirken bellekte ondan sonra gelen, ancak dallanma nedeniyle yürütülmeyecek olan komut (veya komutlar) da iş hattına alınmış olur.

Dallanma sonrası iş hattına alınması gereken **hedef komut adresi**, MİB dallanma komutunu yürütene kadar (PC güncellenmeli) belli değildir.

Koşullu dallanma problemi: Bayrakları değiştiren son komut yürütülünceye kadar bayrak değerleri belli olmadığından dallanmanın olup olmayacağı (büyük?, eşit?) belli değildir.

İş hattını durdurmak bu sorunları çözer, ancak performansı düşürür.

Daha verimli çözüm yöntemleri bulunmaktadır.

2.5.1. Kaynak Çatışması (*Resource Conflict*), Yapısal Sorun (*Structural Hazard*):

İş hattında aynı anda işlenen iki (veya daha fazla) komut aynı kaynağa (bellek, ALU) gerek duyarsa kaynak çatışması oluşur.

a) Bellek çatışması: İki farklı segmanda aynı bellek modülüne erişilmek istenirse

Örneğin komut alma ile operand okuma/yazma aynı anda olamaz.

Çözümler:

- Komutların belli bölümleri paralel değil, peş peşe (seri) işlenir. İş hattının belli segmanları durdurulur. Bu çözüm performansı düşürür.
- Harvard mimarisi: Komutlar ve veriler için ayrı bellek
- Komut kuyruğu veya cep bellek: Bir komut işlenirken belleğe erişilmediği anlarda sıradaki komutlar bellekten okunarak bir kuyruğa yazılır.

b) İşlem birimi (ALU, FPU) çatışması: İki farklı segmanda aynı işlem birimine (*Arithmetic Logic Unit*- ALU, *Floating Point Unit*- FPU) gerek duyulursa.

Çözümler:

- İşlem birimlerinin sayısı arttırılır. Örneğin adres hesabı ve veri işleme için iki ayrı ALU kullanılır.
- İşlem birimleri de iş hattı olarak tasarlanarak paralellik sağlanır. Örnek FPU

2.5.2. Veri Çatışması (*Data Conflict*), Veri Bağımlılığı (*Data Dependency*):

Bir veri hazır (güncel) olmadan önce kullanılmaya çalışırsa veri çatışması olur.

Bu sorun çözülmezse iş hattında çalışan program yanlış sonuç üretebilir.

Örnek:

ADD R1, R2, R3 $R3 \leftarrow R1 + R2$

SUB R3, R4, R5 $R5 \leftarrow R3 - R4$

İş hattında veri bağımlılığı

Saat çevrimi / Komutlar	1	2	3	4	5	6
ADD R1,R2,**R3**	IF	DR	EX	ME	WB	
SUB **R3**,R4,R5		IF	DR	EX	ME	WB

ADD komutunun sonucu saklayıcı dosyasına (R3) yazıldı.

SUB komutu, R3 saklayıcısını güncellenmeden önce okur.
R3 henüz bir önceki ADD komutunun sonucunu içermiyor, çünkü henüz WB katmanında işlenmedi.

2.5.2. Veri Çatışması (*Data Conflict*), devamı

Üç farklı tipte veri çatışması (*data hazard*) oluşabilir:

- **Yazmadan sonra okuma (*Read after write*) (RAW):** Bu türe gerçek bağımlılık (*true dependency*) da denir.

  Bir komut, bir saklayıcıyı veya bellek gözünü değiştirmektedir. Daha sonra gelen bir komut da aynı saklayıcı veya bellek gözünü okumaktadır.

  Eğer iş hattı nedeniyle okuma işlemi yazmadan önce yapılırsa veri çatışması sorunu oluşur.

- **Okumadan sonra yazma (*Write after read*) (WAR):** Anti bağımlılık da denir.

  Bir komut, bir saklayıcıyı veya bellek gözünü okumaktadır. Daha sonra gelen bir komut da aynı saklayıcı veya bellek gözüne yazmaktadır.

  Eğer yazma işlemi okumadan önce yapılırsa veri çatışması sorunu oluşur.

- **Yazmadan sonra yazma (*Write after write*) (WAW):** Çıkış bağımlılığı da denir.

  İki komut aynı saklayıcıyı veya bellek gözüne yazmaktadır.

  Eğer yazma işlemleri programda belirtilenden farklı sırada olursa veri çatışması sorunu oluşur.

Veri çatışması sorununun çözümleri:

A) Durdurma, Donanım Kilidi (*Hardware interlock*) (donanım tabanlı çözümler):

Bir donanım, iş hattındaki tüm komutları (denetim bitlerini) izler. Veri bağımlılığı olan komutların iş hattına girmesi geciktirilir.

İş hattının komut alma segmanı (IF) gerekli saat çevrimi kadar durdurulur (*stall*).

Örnek:

Saat çevrimi / Komutlar	1	2	3	4	5	6	7	8	9
ADD R1,R2,**R3**	IF	DR	EX	ME	WB				
SUB **R3**,R4,R5		IF	-	-	-	DR	EX	ME	WB

Önce R3'e yazılır, sonra okunur. Yazma ve okuma farklı saat çevrimlerinde.

Veri çatışması sezildi. IF/DR.Rs1 = DR/EX.Rd

İş hattında komutun ilerlemesi durduruldu. 3 saat çevrimi gecikme oluştu.

İş hattının durdurulması:

IF/DR saklayıcısına yükleme izni verilmez.

DR katmanına NOOP (*No Operation*) komutunun denetim bitleri yazılır.

PC'nin güncellenmesine izin verilmez.

Veri çatışması sorununun çözümleri (devamı):

Saklayıcı dosyasına (*register file*) erişim sorunun çözümü:

Saklayıcı dosyasına, aynı saat çevriminde hem okuma hem de yazma için erişilebilir.

Saat çevriminin ilk yarısında (saat işaretinin çıkan kenarında) veri yazılır, ikinci yarısında (inen kenar) ise okunur.

Bu yöntem, gecikme (durdurma) süresini 3 çevrimden 2 çevrime düşürür.

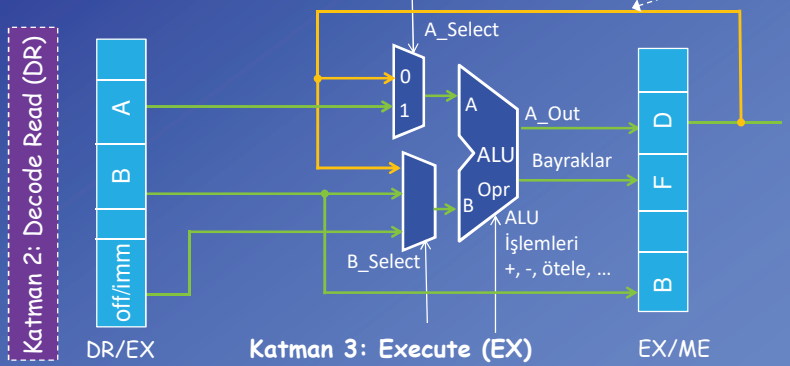
Veri çatışması sorununun çözümleri (devamı):

B) Operand yönlendirme (*Operand forwarding or Bypassing*) (Donanım):

EX katmanının çıkışı (EX/ME saklayıcısı) ile ALU girişleri arasında doğrudan bir bağlantı (*bypass*) oluşturulur.

A_Select ve B_Select seçme girişleri iş hattındaki çatışma sezme (*hazard detection*) birimi tarafından belirlenirler. Bu birim, ya saklayıcı dosyasından okunan verinin ya da bir önceki ALU işleminin sonucunun ALU girişlerine aktarılmasını sağlar.

Forwarding (Bypass)



Operand yönlendirme EX/ME saklayıcısından ALU'ya (devamı):

Eğer çatışma sezme birimi bir önceki ALU işleminin hedef saklayıcısının şimdiki ALU işleminin kaynağı olduğunu sezerse, denetim birimi ALU'nun girişine saklayıcıdan gelen değeri değil, ALU'nun çıkışından doğrudan gelen değeri (*bypass*) yönlendirir.

Örnek:

Komutlar / Saat çevrimi		1	2	3	4	5
ADD R1, R2, **R3**;	R3←R1 + R2	IF	DR	EX	ME	WB
SUB **R3**, R4, R5;	R5←R3 - R4		IF	DR	EX	ME

R3'ün geçerli olmayan önceki değeri okunuyor.
Bu geçersiz değer EX segmanında kullanılmayacak.

İş hattı denetim birimi ALU'nun girişine saklayıcıdan DR'de alınan geçersiz değeri değil, ALU'nun çıkışından doğrudan gelen değeri (*bypass*) yönlendirir (A_Select = 0).

Eğer sorunu operand yönlendirme ile çözmek mümkün olursa iş hattını durdurmaya gerek olmaz ve performans düşmez.

Bellekten okuma çatışmasının (*Load-use data hazard*) operand yönlendirme ile çözülmesi:

Bellekten okuma çatışması (*Load-use data hazard*):

Load komutları da veri çatışmasına neden olur.

Örnek:

Bellekten okunan veri saklayıcı dosyasına (R1) yazıldı.

Bellekten okuma çatışması

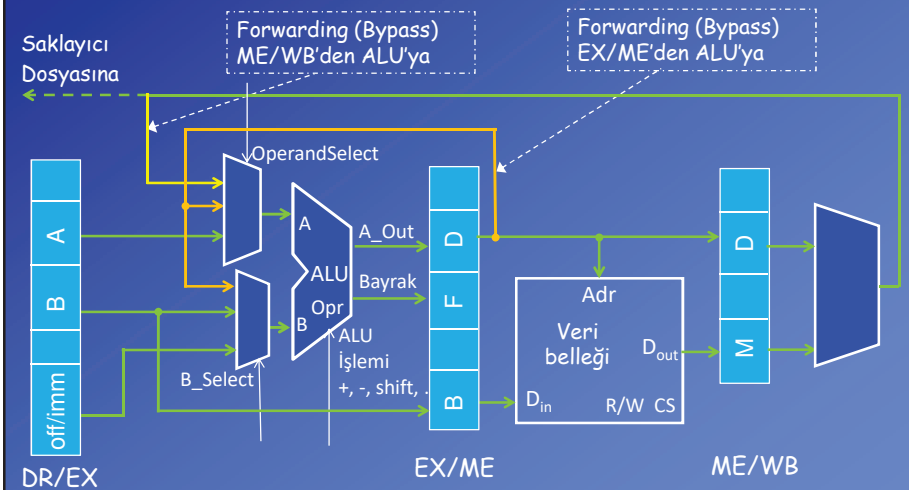
Komutlar / Saat çevrimi		1	2	3	4	5	6
LDL $500(R4), **R1**	*R1 ← M[R4 + $500]*	IF	DR	EX	ME	WB	
ADD **R1**, R2, R3	*R3 ← R1 + R2*		IF	DR	EX	ME	WB

ADD komutu R1 saklayıcısını güncellenmeden önce okur.
R1'deki değer güncel (geçerli) değildir.

Operand yönlendirme ME/WB saklayıcısından ALU'ya :

Bellekten yükleme çatışması nedeniyle oluşan gecikmeyi azaltmak için ME katmanının çıkışından (ME/WB saklayıcısı) ALU'nun girişine doğrudan bir bağlantı oluşturulur.

Ancak bir saat çevrimi gecikme hala gereklidir.



Bellekten okuma çatışması (*Load-use data hazard*) (devamı):

Operand yönlendirme (*forwarding*); + 1 çevrim gecikme ile çözüm:

Örnek:

	Saat çevrimi / Komutlar	1	2	3	4	5	6	7
LDL	$500(R4), R1	IF	DR	EX	ME	WB		
ADD	R1, R2, R3		IF	-	DR	EX	ME	WB

R1'in geçerli olmayan önceki değeri okunuyor.
Bu **geçersiz değer** EX segmanında kullanılmayacak.

İş hattı denetim birimi ALU'nun girişine saklayıcıdan DR'de alınan geçersiz değeri değil, ALU'nun çıkışından doğrudan gelen değeri (*bypass*) yönlendirir.

Veri çatışması sorununun çözümleri (devamı):

C) NOOP (*No Operation*) komutları eklemek (Yazılım temelli):

Derleyici, çatışmaya neden olan komutlar arasına gerektiği kadar NOOP komutu ekler.

Bu çözümün etkisi iş hattını durdurmak ile aynıdır.

Örnek:

Saat çevrimi / Komutlar	1	2	3	4	5	6	7	8
ADD R1,R2,**R3**	IF	DR	EX	ME	WB			
NOOP		IF	DR	EX	ME	WB		
NOOP			IF	DR	EX	ME	WB	
SUB **R3**,R4,R5				IF	DR	EX	ME	WB

Derleyici ekler.

İlk yarıda R3'e yazılır, sonra ikinci yarıda okunur.

NOOP bir makine dili komutu olduğundan, iş hattında diğer komutlar gibi tüm katmanlardan geçerek aynı şekilde işlenir.

NOOP komutları nedeniyle gecikme oluştuğundan sistemin performansı düşer.

Veri çatışması sorununun çözümleri (devamı):

D) Optimize edilmiş çözüm (Yazılım temelli):

Derleyici, eğer mümkünse programda uygun komutların yerini değiştirerek bu komutları çatışmaya neden olan komutların arasına yerleştirir.

Bu değişiklik algoritmayı değiştirmemeli ve başka çatışmalara neden olmamalı.

Örnek:

```
STL  $00(R6), R1   M[R6 + $00] ← R1
STL  $04(R6), R2   M[R6 + $04] ← R2
ADD  R1, R2, R3    R3 ← R1 + R2
SUB  R3, R4, R5    R5 ← R3 – R4
```

Saat çevrimi / Komutlar	1	2	3	4	5	6	7	8
ADD R1,R2,**R3**	IF	DR	EX	ME	WB			
STL $00(R6), R1		IF	DR	EX	ME	WB		
STL $04(R6), R2			IF	DR	EX	ME	WB	
SUB **R3**,R4,R5				IF	DR	EX	ME	WB

Derleyici taşıdı.

İlk yarıda R3'e yazılır, sonra ikinci yarıda okunur.

NOOP eklemeye göre performans iyileşmiştir.

Bu çözümde NOOP komutları veya durdurma nedeniyle oluşan gecikmeler yoktur.

2.5.3. Denetim Sorunları (Dallanmalar, Kesmeler) (*Control Hazards*):

Örnek RISC işlemcide dallanma komutlarının (branch/jump) hedef adresleri Yürütme katmanında (*Execution* - EX) hesaplanır (yansı 2.30).

Hedef adres EX/ME iş hattı saklayıcısına yazılır.

Dallanma kararı yürütmeden sonra oluşan bayrak değerlerine göre Bellek katmanında (*Memory* - ME) verilir (yansı 2.30).

EX katmanından sonra dallanmaya ilişkin karar (PC_Select) ve hedef adres, 1. katman IF'e gönderilir.

IF katmanında önce PC'nin işaret ettiği komut okunur, sonra PC güncellenir.

Bu işlemler sırasında dallanma komutunun altında yer alan (dallanmanın hedefi olmayan) sıradaki komutlarda iş hattına alınmış olur.

Halbuki dallanma olduğunda bu komutların atlanması gerekirdi.

Bu durumda, ya bir donanım birimi iş hattını durdurup boşaltmalı ya da derleyici temelli bir çözüm olan gecikmeli dallanma (*delayed branch*) uygulanmalı.

İş hattına alınan gereksiz komutlar WB katmanına ulaşmadan önce durdurulmalılar.

İşlemcideki değişiklikler WB katmanında yapılır.

Koşullu Dallanma Sorunları:

Örnek:

```
100  SUB    R1, R2, R1     R1 ← R1 - R2
104  BGT    $1C            Branch if greater ($108 + $1C = $124 Hedef adres)
108  ADD    R1, R1, R2
10C  ADD    R3, R4, R2
110  STL    $00(R5), R2
114  LDL    $0A(R6), R1
...
124  STL    $00(R6), R2    BGT'nin hedefi
```

Eğer dallanma gerçekleşirse bu komutların atlanması gerekir.

Hatırlatma: Bcc koşullu dallanma komutları son ALU işleminde oluşan bayrak değerlerine göre davranırlar.

Örneğin; BGT komutu (işaretli karşılaştırma), işaret "N" (*Negative*), taşma "V" (*Overflow*) ve sıfır "Z" (*Zero*) bayraklarını değerlendirir.

R = A – B
İşleminden sonra işaretli sayıları karşılaştırmak için sağdaki tablo kullanılır.

Taşma (V)	İşaret (N)	Sıfır (Z)	Karşılaştırma
X (önemli değil)	Pozitif (0)	EVET (1)	A=B
YOK (0)	Pozitif (0)	HAYIR (0)	A>B
YOK (0)	Negatif (1)	HAYIR (0)	A<B
VAR (1)	Pozitif (0)	HAYIR (0)	A<B
VAR (1)	Negatif (1)	HAYIR (0)	A>B

Koşullu Dallanma Sorunları (devamı):
Örnek (devamı): **Eğer dallanma olursa**

Hedef adres ($108 + $1C = $124) EX katmanında hesaplandı ve EX/ME saklayıcısına yazıldı.

Dallanma kararı verildi (EX'ten sonra). "Dallanma var"

Hedef adres EX/ME saklayıcısından IF katmanına gider.

Komutlar										
SUB R1, R2, R1	IF	DR	EX	ME	WB					
BGT $1C		IF	DR	EX	ME	WB				
ADD R1, R1, R2			IF	DR	EX	ME	WB			
ADD R3, R4, R2				IF	DR	EX	ME	WB		
STL $00(R5), R2					IF	DR	EX	ME	WB	
Hedef: STL $00(R6), R2						IF	DR	EX	ME	WB

Bu komutlar atlanmalıydı.

İş hattı durdurup boşaltılmalı veya yazılım temelli çözümler uygulanmalı.

PC, IF katmanının sonunda güncellenir.
PC← $124 (Hedef)

BGT komutunun hedefi olan komut alınır.

Eğer iş hattını durdurma çözümü uygulanırsa bu örnek işlemcide, **dallanma cezası 3 çevrimdir.**

Koşullu Dallanma Sorunları (devamı):
Örnek (devamı): **Eğer dallanma olmazsa**

Hedef adres ($108 + $1C = $124) EX katmanında hesaplandı ve EX/ME saklayıcısına yazıldı.

Dallanma kararı verildi (EX'ten sonra). "Dallanma yok"

Hedef adres EX/ME saklayıcısından IF katmanına gider.

Komutlar										
SUB R1, R2, R1	IF	DR	EX	ME	WB					
BGT $1C		IF	DR	EX	ME	WB				
ADD R1, R1, R2			IF	DR	EX	ME	WB			
ADD R3, R4, R2				IF	DR	EX	ME	WB		
STL $00(R5), R2					IF	DR	EX	ME	WB	
LDL $0A(R6), R1						IF	DR	EX	ME	WB

PC, IF katmanının sonunda güncellenir.
PC← PC+1 (Sıradaki komut)
Dallanmanın hedef adresi değil.

Sıradaki komut

Eğer dallanma olmazsa dallanma cezası oluşmaz.

Dallanma cezasının azaltılması:

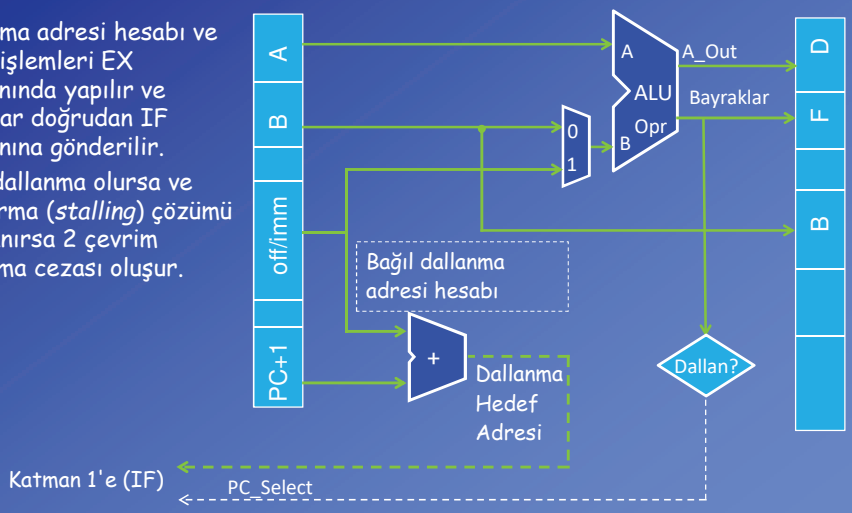
Koşullu dallanma:

Execute (EX) katmanı değiştirilir.

Yürütme (*Execute* EX) katmanı

Dallanma adresi hesabı ve karar işlemleri EX katmanında yapılır ve sonuçlar doğrudan IF katmanına gönderilir.

Eğer dallanma olursa ve durdurma (*stalling*) çözümü uygulanırsa 2 çevrim dallanma cezası oluşur.



Dallanma cezasının azaltılması (devamı):

Koşullu dallanma (devamı) : Dallanma olursa

Örnek:

Hedef adres ($108 + $1C = $124) hesaplandı. Dallanma kararı alındı (EX'te).

Hedef adres IF katmanına gönderildi.

Komutlar

SUB R1, R2, R1	IF	DR	EX	ME	WB				
BGT $1C		IF	DR	EX	ME	WB			
ADD R1, R1, R2			IF	DR	~~EX~~	~~ME~~	~~WB~~		
ADD R3, R4, R2				IF	~~DR~~	~~EX~~	~~ME~~	~~WB~~	
Hedef: STL $00(R6), R2					IF	DR	EX	ME	WB

Bu komutlar atlanmalıydı.

İş hattı durdurup boşaltılmalı veya yazılım temelli çözümler uygulanmalı.

PC, IF katmanının sonunda güncellenir. PC← $124 (Hedef)

BGT komutunun hedefi olan komut alınır.

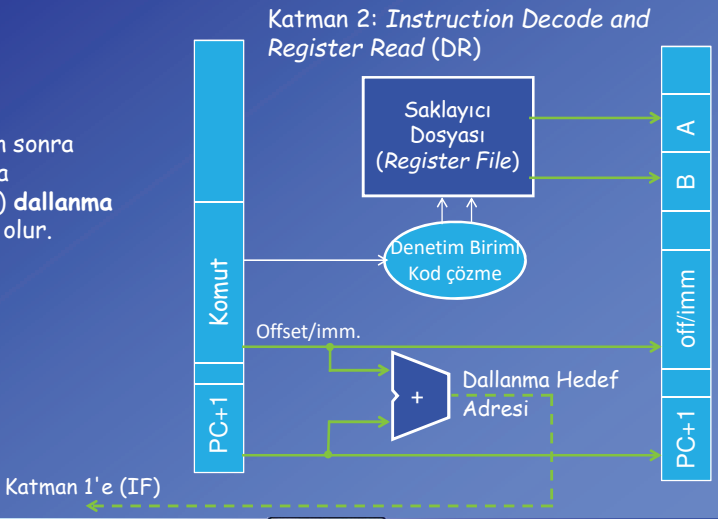
Eğer iş hattını durdurma çözümü uygulanırsa bu örnek iş hattında, **dallanma cezası 2 çevrim** olur.

Dallanma cezasının azaltılması (devamı):

Koşulsuz dallanma:

Bayrak değerlerine gerek olmadığından, dallanma hedef adresi hesabı DR katmanına aktarılabilir.

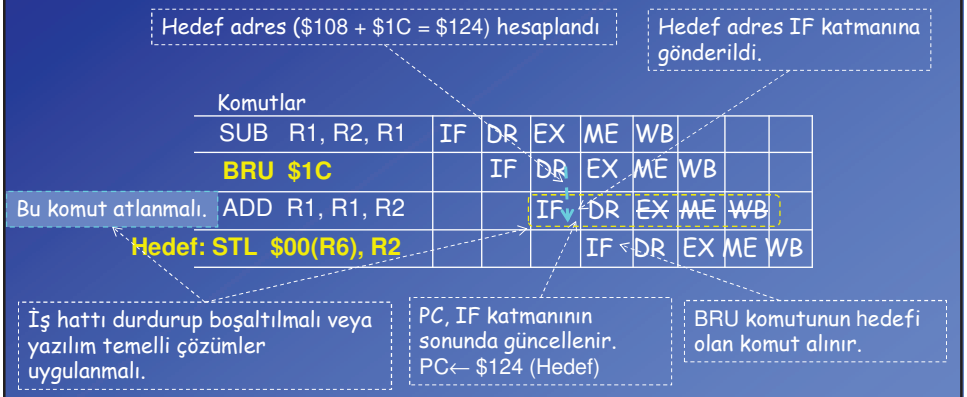
Bu iyileştirmeden sonra koşulsuz dallanma komutunun (BRU) **dallanma cezası 1 çevrim** olur.



Dallanma cezasının azaltılması (devamı):

Koşulsuz dallanma (devamı) :

Örnek:



Dallanma hedef adresi hesabının DR katmanına taşınmasından sonra, örnek iş hattında, **koşulsuz dallanmanın cezası 1 çevrim** olur.