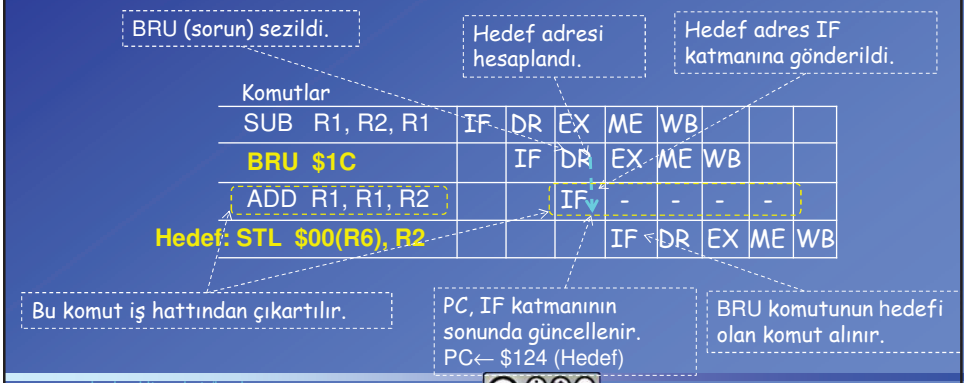
Denetim Sorunlarının (*Control Hazards*) Çözümleri:

A) Durdurma/boşaltma (*Stalling/flushing*) (donanım tabanlı):

Ek bir donanım birimi, sorunu sezer ve dallanmanın hedefi olan komut alınıncaya kadar iş hattını durdurur.

Hem koşulsuz hem de koşullu dallanmalarda uygulanabilir.

Örnek: Koşulsuz dallanma, hedef adres hesabı DR'de yapılıyor.



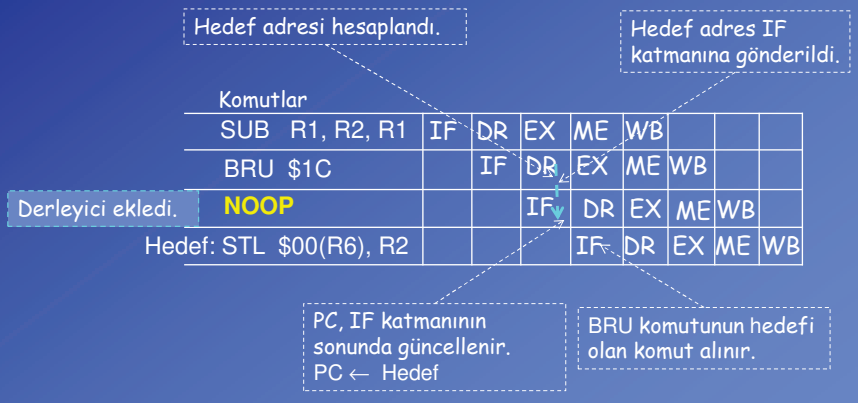
Denetim Sorunlarının (*Control Hazards*) Çözümleri (devamı):

B) NOOP (*No Operation*) komutlarının eklenmesi (Yazılım tabanlı):

Derleyici, dallanma komutundan sonra gerektiği kadar NOOP komutu ekler.

Bu çözümün etkisi iş hattını durdurmakla aynıdır.

Örnek: Koşulsuz dallanma, hedef adres hesabı DR'de yapılıyor.

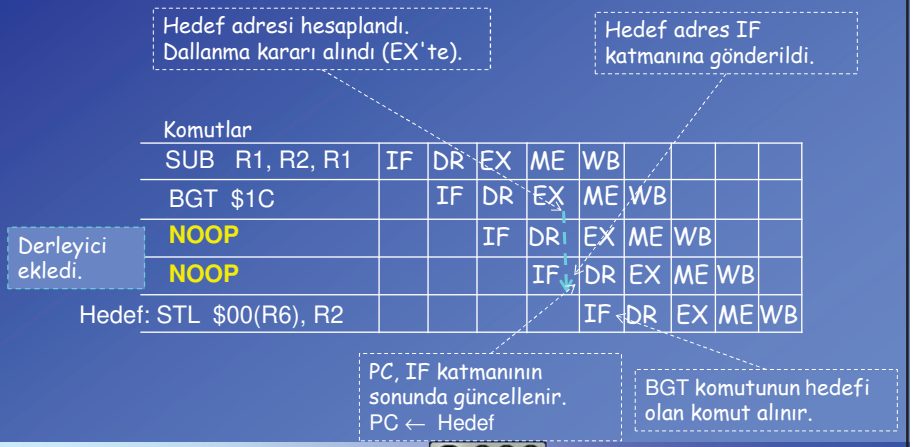


B) NOOP (*No Operation*) komutlarının eklenmesi (devamı):

Gerekli olan NOOP komutlarını sayısı iş hattını ne kadar durdurmak gerektiğine bağlıdır.

Örnek: Koşullu dallanma; adres hesabı ve dallanma kararı işlemleri EX'te.

Bu durumda 2 çevrim gecikmeye gerek olduğundan 2 tane NOOP eklenir.



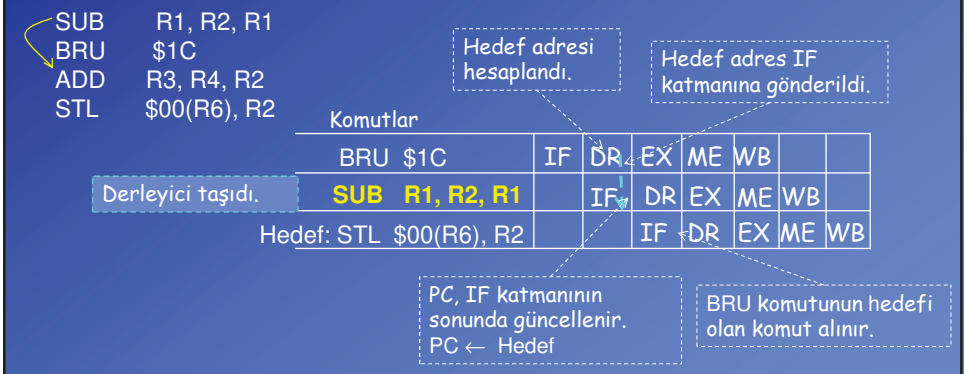
Denetim Sorunlarının (*Control Hazards*) Çözümleri (devamı):

C) Optimize Çözüm (Software-based):

Derleyici, eğer mümkünse programda uygun komutların yerini değiştirerek bu komutları dallanma komutunun peşine yerleştirir.

Bu değişiklik algoritmayı değiştirmemeli ve başka çatışmalara neden olmamalı.

Örnek: Koşulsuz dallanma, hedef adres hesabı DR'de yapılıyor.



Eğer optimize çözüm mümkünse **dallanma cezası oluşmaz**.

C) Optimize Çözüm (devamı):

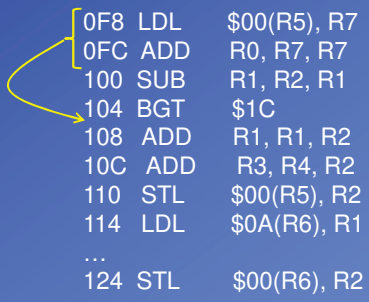
Taşınması gerekli olan komutlarını sayısı iş hattını ne kadar durdurmak gerektiğine bağlıdır.

Bu değişiklik algoritmayı değiştirmemeli ve başka çatışmalara neden olmamalı.

Örnek: Koşullu dallanma; adres hesabı ve dallanma kararı işlemleri EX'te.

Bu durumda 2 çevrim gecikmeye gerek olduğundan 2 tane komut, dallanma komutunun peşine taşınır.

Bu 2 komut, dallanma komutunun peşine taşınabilir.



```
0F8  LDL    $00(R5), R7
0FC  ADD    R0, R7, R7
100  SUB    R1, R2, R1
104  BGT    $1C
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
```


Komutların sırasını değiştirmek ile ilgili önemli noktalar:

Dallanmadan **önce gelen** gelen bir komut dallanmadan sonraya kaydırılabilir.

Dallanmanın koşulu veya hedef adresi kaydırılan komuta bağlı olmamalı.

Bu yöntem (eğer mümkünse) her zaman performansı arttırır (NOOP'a göre).

Özellikle **koşullu dallanmalarda** bu yöntem dikkatli uygulanmalı.

Dallanmanın bağlı olduğu koşulu belirleyen komut dallanmadan sonraya taşınamaz.

Bu durumda NOOP eklenir.

Diğer seçenekler:

Derleyici taşımak üzere şu komutları seçebilir:

- **Dallanmanın hedefinden** (gidilecek yerden)

- Taşınan komut dallanma gerçekleşmese de çalışacaktır. Bu programı etkilememeli.
- Dallanma gerçekleşirse performans artar.

- **Dallanma komutunun peşinden** (dallanma olmazsa devam edilen kol)

- Taşınan komut dallanma gerçekleşse de çalışacaktır. Bu programı etkilememeli.
- Dallanma gerçekleşmezse performans artar.

Denetim Sorunlarının (*Control Hazards*) Çözümleri (devamı):

D) Dallanma Öngörüsü (*Branch Prediction*):

Hatırlatma: Dallanma komutları nedeniyle iş hattında iki temel problem oluşur.

1. Dallanma komutunu hedef adresi iş hattının IF'den sonraki katmanlarında hesaplanır.

   Bu nedenle, dallanma sonucu **hangi hedef komutun iş hattına alınacağı**, işlemci hedef adresi hesaplayana kadar **belli değildir**.

   PC ← PC + offset

   a) Eğer adres hesabı EX katmanında yapılır ve sonuç EX/ME saklayıcılarından IF katmanına gönderilirse (yansı 2.30), dallanma cezası: 3 çevrim.

   b) Eğer adres hesabı EX katmanında yapılır ve sonuç doğrudan IF katmanına gönderilirse (yansı 2.51), dallanma cezası: 2 çevrim.

   c) Eğer adres hesabı DR katmanında yapılır ve sonuç doğrudan IF katmanına gönderilirse (yansı 2.53), dallanma cezası: 1 çevrim (sadece koşulsuz dallanma komutları için geçerlidir).

   Hedef adresi önceden belirleyip bu sorunu çözmek için **dallanma hedef tablosu** (*branch target table*) kullanılır (yansı 2.64).

   Dallanma hedef tablosu IF katmanında yer alan, dallanma komutlarının ve bu komutların dallanacağı hedeflerin adreslerini tutan bir cep bellektir (*cache*).

Dallanma komutları nedeniyle iş hattında iki temel problem oluşur (devamı):

2. **Koşullu dallanma** sorunu: Dallanmadan önceki komut yürütülünceye kadar bayrakların değeri belli olmadığından dallanmanın gerçekten olup olmayacağı belli değildir.

   Dallanma olmazsa PC ← PC + 4 (örnek RISC işlemcisi için)

   Dallanma olursa PC ← PC + offset

   a) Eğer dallanma karar lojiği ME katmanındaysa (EX'ten sonra) (yansı 2.30), dallanma cezası: 3 çevrim.

   b) Eğer dallanma karar lojiği EX katmanındaysa (yansı 2.51), dallanma cezası : 2 çevrim.

   Bu problemi çözmek için **dallanma öngörü** (*branch prediction*) yöntemleri kullanılır.

   Koşullu dallanma komutu ile karşılaşıldığında dallanma öngörüsü yöntemleri dallanmanın olup olmayacağını öngörmeye çalışırlar.

   Öngörü sonucuna göre bellekteki bir sonraki komut veya dallanmanın hedefi olan komut iş hattına alınır.

D) Dallanma Öngörüsü (*Branch Prediction*) (devamı):

Koşullu dallanma komutu ile karşılaşıldığında dallanma öngörüsü yöntemleri dallanmanın olup olmayacağını öngörmeye çalışırlar.

Öngörü sonucuna göre bellekteki bir sonraki komut veya dallanmanın hedefi olan komut iş hattına alınır.

Eğer öngörü doğru çıkarsa dallanma cezası olmaz.

Öngörü yanlış olursa iş hattı durdurulur ve boşaltılır.

İki tür dallanma öngörüsü yöntemi vardır; **statik** ve **dinamik**.

Statik dallanma öngörüsü stratejileri:

a) "Her zaman dallanma yok" öngörüsü: Her zaman dallanma olmayacağı öngörülür ve bellekte dallanmadan sonra gelen komut iş hattına alınır.

b) "Her zaman dallanma var" öngörüsü : Her zaman dallanma olacağı öngörülür ve dallanmanın hedefi olan komut iş hattına alınır dallanma hedef tablosu gereklidir.

Programların davranışını inceleyen çalışmalar, koşullu dallanmaların %50'sinden fazlasında dallanmanın gerçekleştiğini göstermişlerdir.

Bu nedenle "her zaman dallanma var" öngörüsü performans açısından daha iyi sonuç vermektedir.

D) Dallanma Öngörüsü (*Branch Prediction*) (devamı):

Önceden komut alma (*Target Instruction prefetch*): **Dallanma hedef tablosu**

"Her zaman dallanma var" stratejisi: Her zaman dallanmanın hedef komutu alınır.

Ancak dallanma adresi hesaplanmadan önce hedef komutun adresi **belli değildir**.

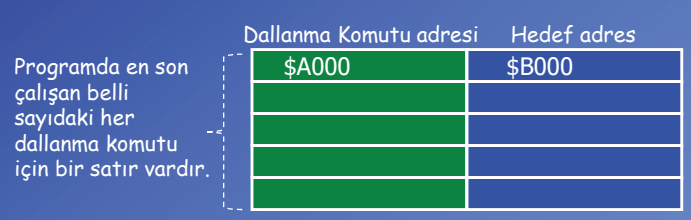
Dallanmanın hedef adresini daha önceden belirleyebilmek **dallanma hedef tablosu (*branch target table*)** kullanılır.

Dallanma hedef tablosu: Son çalışan belli sayıdaki dallanma komutunun adresleri ve son çalıştıklarında nereye gidildiği cep bellekte (*cache memory*) (bkz 6) tutulur.

Son zamanlarda yürütülen her dallanma komutu için ayrı bir satır bulunmaktadır.

Tutulan komut sayısı tablo boyutu ile sınırlıdır.

Tablo sayesinde hedef adres hesaplanmadan önce dallanma komutunun dallanacağı adresteki komutlara erişilebilir.



Örnek:

```
....
$A000  BGT Hedef
....          .....
....          .....
$B000 Hedef ADD ...
```

D) Dallanma Öngörüsü (*Branch Prediction*) (devamı):

Dinamik dallanma öngörüsü stratejileri:

Dinamik dallanma öngörüsü stratejileri o anda çalışan programdaki tüm koşullu dallanma komutlarının geçmişi ile ilgili istatistik tutarak dallanmanın olup olmayacağını öngörmeye çalışırlar.

Programdaki her koşullu dallanma komutu ile bir veya daha fazla sayıda **öngörü biti** (veya sayaç) (*prediction bits*) ilişkilendirilir.

Komutların geçmişi ile ilgili bilgi (daha önceki çalışmalarda dallanma olup olmadığı) sağlayan bu bitler bir **dallanma geçmişi tablosunda** (*branch history table*) tutulur (yansı 2.67).

1 bit dinamik dallanma öngörü yöntemi :

Her koşullu dallanma komutu için dallanma geçmişi tablosunda bir **öngörü biti (p_i)** tutulur.

p_i , i. koşullu dallanma komutunun öngörü bitidir.

Öngörü biti, ilgili komutun son çalışmasında dallanma olup olmadığını gösterir.

Eğer komutun son çalışmasında dallanma olduysa bir sonraki çalışmasında da dallanma olacağı varsayılır.

Algoritma:

i. Koşullu dallanma komutunu al

Eğer ($p_i = 0$) ise öngörü: "dallanma YOK", bellekte sıradaki komutu al

Eğer ($p_i = 1$) ise öngörü: "dallanma VAR", dallanmanın hedefi olan komutu al

Eğer dallanma gerçekten olursa $p_i \leftarrow 1$

Eğer dallanma gerçekten olmazsa $p_i \leftarrow 0$

Dallanma hedef tablosu ve dallanma geçmişi tablosu (*branch history table*):

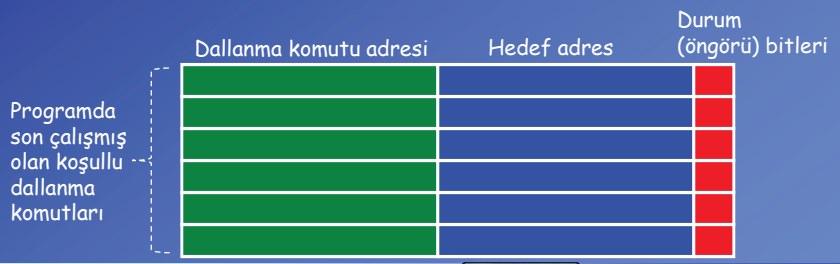
Öngörü bitleri, hızlı erişilebilen bir bellekte oluşturulan dallanma geçmişi tablosunda (*branch history table* - BHT) tutulur.

Dallanma geçmişi tablosunda, en son çalışan belli sayıdaki her koşullu dallanma komutu için komutun bellek adresi, hedef adresi ve durum (öngörü) bitleri tutulur.

Öngörü bitleri dallanma komutunun her çalışmasında dallanma olup olmamasına göre değer alırlar.

Koşullu dallanma komutu tekrar çalıştığında bu bitler iş hattı denetim birimi tarafından karar vermek için kullanılır.

Eğer "dallanma VAR" öngörüsü yapılırsa dallanma komutu yürütülmeden önce tablodaki hedef adresi kullanılarak gidilecek olan komut iş hattına alınabilir.



Örnek: 1 bitlik öngörü yöntemi ve döngüler

Öngörü yöntemleri özellikle döngülerde yararlı olur.

Örnek:

```
        counter ← 100       ; saklayıcı veya bellek gözü
LOOP    ----                ; döngüdeki komutlar
        ----
        Decrement counter
        BNZ  LOOP           ; Branch if Not Zero (koşullu dallanma, p biti vardır)
        ----                ; döngüden sonraki komut
```

Program çalışmaya başladığında BNZ komutunun p biti 1'dir (dallanma VAR öngörüsü).

Döngünün ilk çalışmasında BNZ'de doğru öngörü yapılacak ve döngünün başındaki komut iş hattına alınacak.

p bitinin değeri (p=1) döngünün son çalışmasına kadar değişmeyecek.

Döngünün son çalışmasında p biti hâlâ 1'dir ve "dallanma VAR" öngörüsü yapılır; ama counter sıfır olduğu için program döngünün başına dallanmaz ve döngüden sonraki komut ile devam eder (yanlış öngörü). p sıfır yapılır (p ← 0).

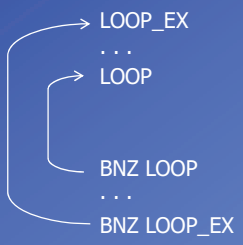
Sonuç olarak 100 defa dönen bir döngüde 99 defa doğru, sadece bir defa yanlış öngörü yapılmış oldu.

Döngüden sonra BNZ'nin p biti 0'dır, çünkü son çalışmada dallanma olmamıştır.

Aynı döngü başka bir döngünün içinde olduğu için tekrar çalıştığında ne olur?

1 bit dinamik dallanma öngörü yöntemindeki sorun:
(İç içe döngüler)

Birden fazla defa çalışan (içteki) döngülerde her defasında iki defa yanlış öngörü yapılır;
biri döngü ilk çalıştığında, diğeri de döngüden çıkarken.



Hatırlayın; önceki örnekte döngüden sonra BNZ'nin p biti 0'dır.

İçteki döngüye tekrar gelindiğinde, ilk çalışmada BNZ'deki öngörü "dallanma YOK" olacaktır (p=0) .

Ancak program dallanarak döngünün başına dönecektir (birinci hata).

Şimdi p biti 1 olur, çünkü dallanma olmuştur (p ← 1).

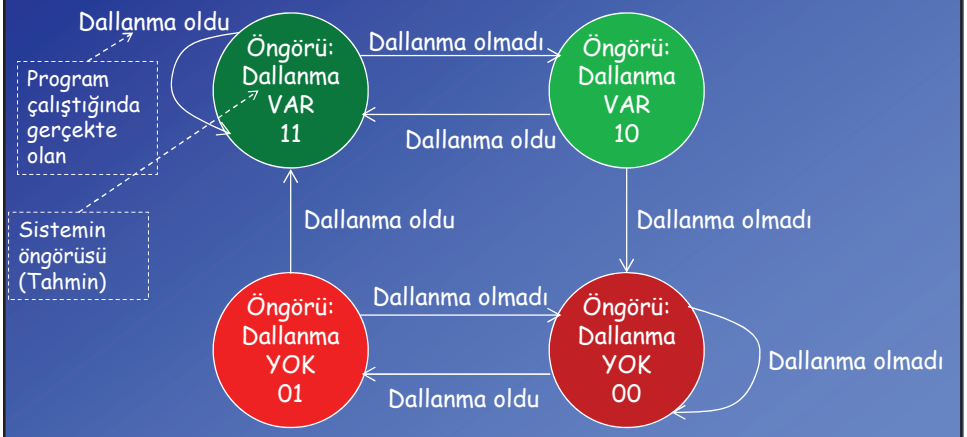
Döngünün son çalışmasına kadar öngörüler doğru olacaktır.

Döngünün son çalışmasında önceki örnekte de gösterildiği gibi yine hatalı öngörü yapılır (ikinci hata).

2 bit dinamik dallanma öngörü yöntemi:

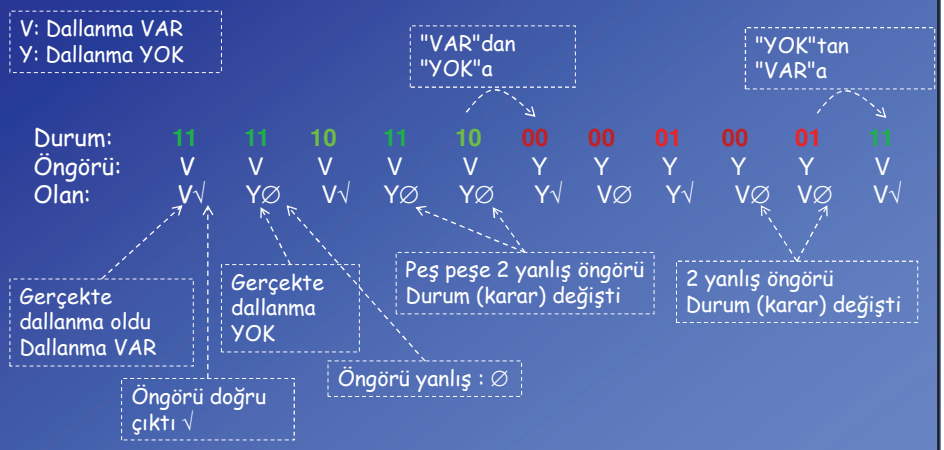
Her koşullu dallanma komutuna iki öngörü (durum) biti atanır.

Eğer komut 11 veya 10 durumlarındaysa "dallanma VAR" öngörüsü yapılır.
Eğer komut 00 veya 01 durumlarındaysa "dallanma YOK" öngörüsü yapılır.



Bu yöntemde ancak **peş peşe iki defa** yanlış öngörü yapılırsa öngörü kararı değişir.

Örnek: 2 bit dinamik dallanma öngörüsü



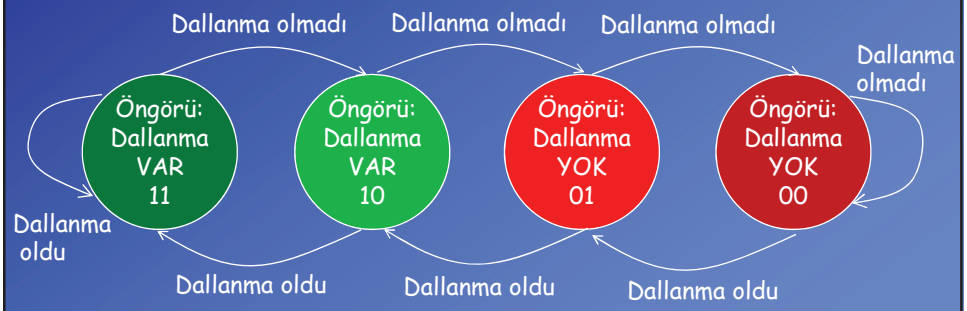
Doyan sayaç (*Saturating counter*):

Diğer bir 2 bit dinamik dallanma öngörü yöntemi:

Dallanma öngörüsü yöntemlerinin sonlu durumlu makineleri (*finite state machine*) farklı şekillerde tasarlanabilir.

Doyan sayaç alternatif bir öngörü yöntemidir. Durum geçişleri farklıdır.

Eğer komut 11 veya 10 durumlarındaysa "dallanma VAR" öngörüsü yapılır.
Eğer komut 00 veya 01 durumlarındaysa "dallanma YOK" öngörüsü yapılır.



Örnek:

Problem:

Bir MİB'te dallanma sorunlarını çözümünde donanım tabanlı yöntemlerin kullanıldığı bir iş hattı (*pipeline*) bulunmaktadır.

Bu MİB'te aşağıda verilen ve iç içe iki döngü içeren kod parçası çalıştırılmaktadır.

```
                    Counter1 ← 10
----> LOOP1         ------                      ; Herhangi bir komut
                    Counter2 ← 10
  --> LOOP2         ------                      ; Herhangi bir komut
                    ------                      ; Herhangi bir komut
                    Counter2 ← Counter2 - 1
  ------------------BNZ   LOOP2                  ; Sıfır değilse dallan
                                                 (Branch if not zero)
                    ------                      ; Döngüden sonraki komut
  ------------------Counter1 ← Counter1 - 1
                    BNZ   LOOP1                 ; Sıfır değilse dallan
                    ------                      ; Döngüden sonraki komut
```

Farklı dallanma öngörüsü yöntemlerinin kullanılması durumunda, yukarıda verilen kod parçasındaki iki dallanma komutunun (BNZ) yürütülmesinde oluşan doğru ve hatalı dallanma öngörülerinin sayılarını veriniz.
Yanıtlarınızı kısaca açıklayınız.

Çözüm:

a. Statik öngörü

i) Her zaman "dallanma var"

BNZ LOOP1: Sadece son yinelemede döngüden çıkarken yanlış öngörü olur; diğer öngörüler doğrudur.

Doğru: 9 Yanlış: 1

BNZ LOOP2: Sadece son yinelemede döngüden çıkarken yanlış öngörü olur; diğer öngörüler doğrudur.

Doğru: 10x9 = 90 Yanlış: 10x1 = 10

Toplam: Doğru: 99 Yanlış: 11

ii) Her zaman "dallanma yok"

BNZ LOOP1: Sadece son yinelemede döngüden çıkarken doğru öngörü olur; diğer öngörüler yanlıştır.

Doğru: 1 Yanlış: 9

BNZ LOOP2: Sadece son yinelemede döngüden çıkarken doğru öngörü olur; diğer öngörüler yanlıştır.

Doğru: 10x1 = 10 Yanlış: 10x9 = 90

Toplam: Doğru: 11 Yanlış : 99

Çözüm (devamı):

b. Bir bitlik dinamik öngörü yöntemi

Dikkat: Her dallanma komutu için ayrı bir öngörü biti kullanılır (Yansılar 2.66, 2.67).

i) Başlangıç kararı "dallanma var"

BNZ LOOP1:
Sadece son yinelemede döngüden çıkarken yanlış öngörü olur; diğer öngörüler doğrudur.

Doğru: 9 Yanlış: 1

BNZ LOOP2:

Döngünün ilk çalışmasında sadece son yinelemede döngüden çıkarken yanlış öngörü olur; diğer öngörüler doğrudur.

Döngüden çıkıldığında öngörü biti p "dallanma yok" olarak değişir. Bu nedenle döngünün 2.-10. çalışmalarında hem ilk hem de son yineleme de hatalı öngörü olur (Yansı 2.69).

Doğru: 9 + 9x8 = 81 Yanlış: 1+ 9x2 =19

Toplam: **Doğru: 90** **Yanlış: 20**

b. Bir bitlik dinamik öngörü yöntemi **(devamı):**

ii) Başlangıç kararı "dallanma yok"

BNZ LOOP1:
İlk ve son yinelemelerde yanlış öngörü olur; diğer öngörüler doğrudur.

Doğru: 8 Yanlış: 2

BNZ LOOP2:
İlk ve son yinelemelerde yanlış öngörü olur; diğer öngörüler doğrudur.

Doğru: 10x8 = 80 Yanlış: 10x2 =20

Toplam: **Doğru: 88** **Yanlış: 22**

c. İki bitlik dinamik öngörü yöntemi:

i) Başlangıç kararı "dallanma var"

BNZ LOOP1: Sadece son yinelemede döngüden çıkarken yanlış öngörü olur; diğer öngörüler doğrudur.

Doğru: 9 Yanlış: 1

BNZ LOOP2: Sadece son yinelemede döngüden çıkarken yanlış öngörü olur; diğer öngörüler doğrudur.

Doğru: 10x9 = 90 Yanlış: 10x1 = 10

Toplam: Doğru: 99 Yanlış: 11

ii) Başlangıç kararı "dallanma yok"

BNZ LOOP1: Birinci, ikinci ve son yinelemelerde yanlış öngörü olur.
Hatırlatma, bu yöntemde karar iki yanlış öngörü-den sonra değişir.

Doğru: 7 Yanlış: 3

BNZ LOOP2: Döngünün ilk çalışmasında; birinci, ikinci ve son yinelemelerde yanlış öngörü olur. Döngünün ilk çalışmasından sonra karar hala "dallanma var" şeklindedir. Bu nedenle, döngünün 2.-10. çalışmasında sadece son yinelemede hatalı öngörü olur.

Doğru: 7+ 9x9 = 88 Yanlış: 3 + 9x1 = 12

Toplam: Doğru: 95 Yanlış: 15