Tekrar: İşlemci Boru hattı
Boru hattına sahip basit işlemci tasarımı:

- tek yürütme birimi tüm buyrukların sonucunu hesaplıyor
- her aşama tek çevrim sürüyor(duraklama olmazsa)

Bugüne kadar hep yürütme aşamasını bir vuruşta yapılıyor diye varsaydık. Gerçek hayatta her işlem tek vuruşta yapılamıyor ya da tek vuruşta yapmaya kalkarsan işlemler daha uzun sürüyor. Saatin vuruş sıklığı arttırılamıyor. Örneğin; Toplama işlemi ile çarpma işlemi aynı zaman sürmüyor.

Değişken Yürütme Zamanlı Boru Hattı

Bellek değişkenleri görüldüğü gibi tamamen değişken yapıdadır. Çünkü önbellekten mi erişecek ana bellekten mi erişecek belli değil bundan dolayı erişim süresi farklı olacaktır. Bu şekilde değişken gecikmeli buyruklar olduğundan boru hattının ne şekilde yapılacağı da bir soru işareti haline gelir. Uzun süren buyruklar için boru hattını durduracak mıyız? Arkadan gelen buyruklar bekleyecek mi? Boru hattı daha derin olduğunda ve özellikle dallanma koşulları daha geç çözüldüğünde bu sefer dallanmadan kaynaklı başarım düşüşü ile karşılaşılacak. Değişken sayılı aşamalı boru hattı karmaşıklık demektir ve yönetilmesi daha da güç hale gelir.

Değişken zamanlı yürütme: farklı gecikmelere sahip çok sayıda yürütme birimi var. Bu mimarinin boru hattı diyagramı neye benzer?

fmul x0 x1 x2	G	Ç	Ü	Ü	Ü	Ü	Y			
add x3 x4 x5		G	Ç	Ü	Y					
add x3 x4 x5			G	Ç	Ü	Y				
fadd f1 f2 f3				G	Ç	Ü	Ü	Ü	Ü	Y

çarpıcı birim ile toplayıcı birimin ayrı olduğunu düşünüyoruz. Yani aynı aritmetik mantık birimi içerisinde değiller. Güncel işlemcilerde de bu şekildedir.

Burada sonuçlarının yazıldığı aşamalar farklı farklı. Program sırasının dışında yazma işlemi gerçekleşti. Yürütmenin değişken zamanlı yürütmesine izin verilmesi sonucunda bunlar oldu. Yapı sorunu oluşmadı çünkü işlem yaptıkları birimler ayrı yukarıda da söylediğim gibi farklı aritmetik mantık birimlerinde işlem görüyorlar. Bu şekilde program sırasının dışında işlem yapar hale geliyor.

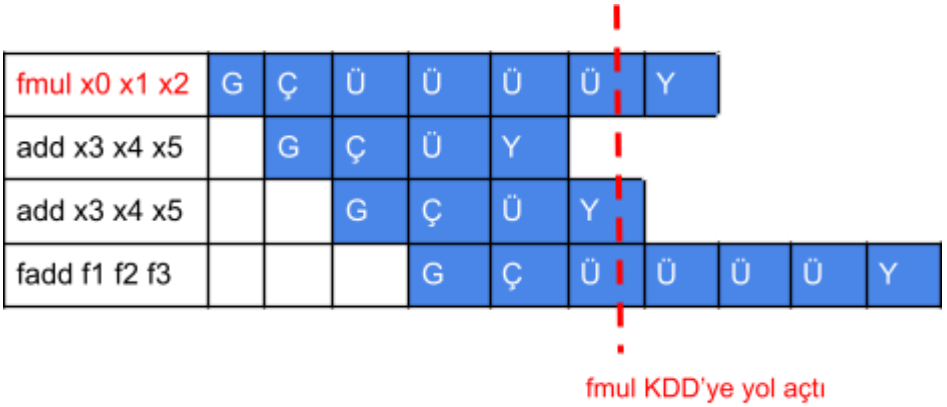
Burada sorun var mı ?

- Buyruklar program sırasına göre mimari durumunu güncelleyemiyorlar.

Bir kod yazılıp program derlendiğinde programın yazıldığı şekildeki sırayla çalışması beklenir. Mimari açıdan yani dışarıdan bakıldığında program sırası neyse o şekilde değişmesi beklenir. Program sırasının dışında bir değişiklik yapıldığında istenen durumun dışında bir duruma ulaşması söz konusu olur. Buradan program sırası kutsaldır diyebiliriz.

Tam Doğru Kural Dışı Durumlar

Program sırasının dışında yazmaçlara bir yazma işlemi gerçekleştirmeye başlandığında arada bir sorun olabilir. Diyelim ki üstteki gibi bir boru hattı var ve ilk çarpma satırdaki çarpma işlemi sırasında yani yürütmenin son aşamasında bir KDD(kural dışı durum) yarattı. Böyle bir durumda denetimin kesilmesi ve tam doğru işlenecekse o buyruktan sonraki tüm buyrukların atılması gerekir. Ama aşağıdaki buyruklar yazmaçları değiştirdi. Yazmaçlar değiştiği için işlemcinin durumu değişti. Onları ya geri alabilir olması gerekir ya da bunlara bir şekilde mimari durumunu değiştirtmememiz gerekir.



KDD anında mimari durumu program sırasında uygun değil ! x3 yazmacı güncellendi.

x3 yazmacı ya geri alınması gerekir ya da bu şekilde program istediğimiz gibi çalışmayacaktır. Bu da bir sorun demektir. Burada KDD dedik ama başka durumlar da ortaya çıkabilir. Örneğin çarpmanın arkasından bir dallanma buyruğu olabilirdi. Dallanma buyruğunun koşulu çözülmeden önce aşağıdaki toplama işleminin sonucu yazılabilirdi. Buna dallanma buyruğu tahmine dayalı işlem yaptığı için tahmine dayalı yürütüm denir(**speculative execution**). Ama hiç böyle öngörüye dayalı bir şey olmasa bile buyruklar hata yapabilir, hata çıkarabilir, kural dışı durumlar oluşabilir. Mesela sıfıra bölme hatası. Bu yüzden arkadan gelenlerin işlemci durumunun sistem durumunu değiştirmesine karşı bir önlem alınması gerekebilir.

Tam Doğru Olağan dışı Durumlar

Yazmaçları güncelleyen en yaşlı buyruğun KDD'ye yol açtığı anlaşıldığında denetim biriminin yapması gerekenler:

- mimari durumunun (*örn. yazmaç değerleri, program sayacı*) tutarlı olmasını sağlamak. ***Mimari durum:*** o derleyici o buyruğun bittiği anda yazmaçların bir değerinin olacağını varsayıyor. Programcı bir şey yazdığında o soruna yol açan buyruk bittiğinde yazmaçlarda bir değerin olacağını varsayıyor. Bu yazmaç değerlerinin o anda tutarlı olması gerekir. Beklentimiz işlemci içindeki yazmaç değerlerinin tutarlı olması. Program sayacının da doğru adresi gösteriyor olması gerekir.
- KDD'ye yol açan buyruktan sonra gelen buyrukları geçersiz kılmak
- BKM(buyruk kümesi mimarisi) tanımlamasında yazan işlemleri gerçekleştirmek
    1.) özel yazmaç değerlerini güncellemek

2.) program sayacını KDD'yi çözümleyen kodun başına atlatmak

Program Sırası Tablosu(-ing. Re-order Buffer)

Bütün bu sıkıntılar yüzünden program sırasının işlemcinin içinde tutulmasına ihtiyaç var. Az önce incelediğimiz boru hattında dakik olağan dışı durum desteği sağlamak için kullanılabilecek yapılardan biri program sırası tablosu(PST).

Ana fikir: Buyrukları sırasız yürüt, ancak mimari durumu güncellemeden önce yeniden sırala. Mimari durumu sıralı güncelle.

- Buyruklara çözüldüklerinde PST'te sırayla güncellenen bir eleman ayrılır. PST ilk giren ilk çıkar(FIFO) mantığı ile çalışır. Nedeni program sırasıyla geldiği için ve bu sıraya uymaya çalıştığımız için bu şekilde işlenir.
- yürüt aşaması biten buyruklar hedef yazmaç değerini PST'e yazar. Yazmaca yazmak yerine PST'ye yazıyor. Şu an bu anlattığımız yöntem ***pentium 3***'te kullanılan yöntem. p6 mimarisi olarak da geçiyor.
- yazmaç değerleri PST'teki sıra(program sırası) ile güncellenir. Böylece araya bir tampon bellek koymuş gibi olur. Program sırasını tutan ve geçici olarak üretilen değerleri tutan, mimari açısından bakıldığında görünen yazmaçları değiştirmeden önce geçici olarak yaratılan değerleri tutan bir tampon bellek yaratmış oluyoruz.

Böylece bir sorun çıkarsa buradaki değerleri atabiliyoruz. Çünkü onlar henüz işlemci sistemini güncellemiş olmuyorlar. Kural dışı durum oluştuğunda bize 2 şey sağlar. Hem geçici değerleri rahatça atabilir ve sistemi güncellememiş olur, hem de program sırasını tuttuğu için bir buyruktan sonra gelen buyrukları rahatça tespit edebiliyoruz.

PST Elemanları

geçerli: geçerli mi değil mi bilgisi saklanır.1 bit saklanır.
hedef değer: yazılacak olan değer üzerine yazılıyor.
hedef yazmaç: hedef yazmacının numarasını tutuyor. Birazdan işi bitince götürüp o yazmaca yazacak.
bellek adresi: bellek ile bir işi varsa bellek adresini tutar.
program sayacı: o buyruğun program sayacını tutar. Hangi adresten geldiğini tutuyor.
Bu tablo ile başka bir çok durumun bilgisini de tutabilir. PST'nin içerisinde saklanan değerler mimariden mimariye değişir. örneğin; Pentium 4'te hedef değerler tutulmaz.

Store ve branch buyrukları geldiğinde hedef değer sütununa veri yazmaz. Bu sütunun bulunması israf olarak görülebilir. Bellek ile işi olmayan buyruklar için de aynı şey söylenebilir. Bu israfı önleyeceğini düşündüğü için IBM bu iki sütunun olduğu ayrı bir tablo tutar.

Bir PST elemanında:

- Buyrukları ***program sırasına dizebilmek*** için
- mimari durumu buyruğun sonucuna göre güncelleyebilmek için
- olağan dışı bir durumda mimari durumunu geri sarabilmek/program sırasında göre tutarlı hale getirebilmek için

gerekli tüm veriler bulunur.

fmul x0 x1 x2	G	Ç	Ü	Ü	Ü	Ü	PST	Y			
add x3 x4 x5		G	Ç	Ü	PST	PST	PST	PST	Y		
add x3 x4 x5			G	Ç	Ü	PST	PST	PST	PST	Y	
fadd f1 f2 f3				G	Ç	Ü	Ü	Ü	Ü	PST	Y

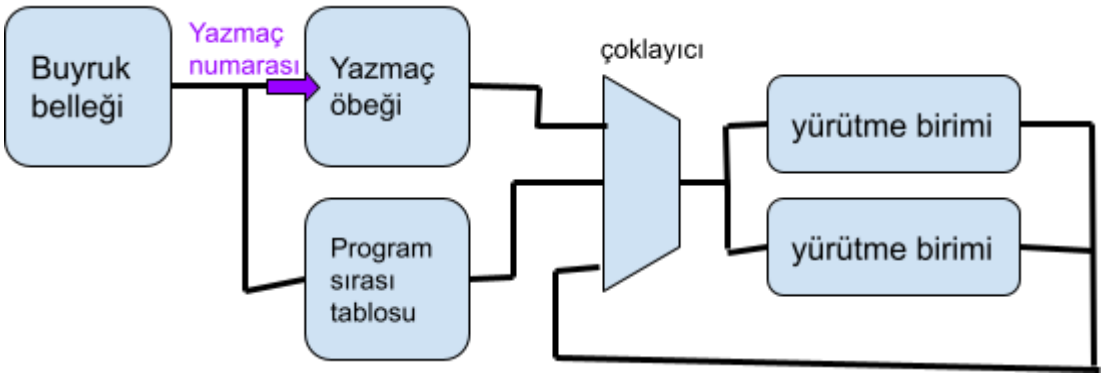
Buyruklar mimari durumu güncellemeden önce PST'yi bir ara bellek olarak kullanıyor. Bu sayede mimari durum program sırasına göre güncellenebiliyor.

Bildiğimiz boru hattı yöntemi üzerine sıralı işleme tablosu kullanıldığında buyruklar yazmaç değerlerini üç farklı yerden okuyabilir:

I. Yazmaç Öbeği: değer yazmaçlara yazılmış olabilir.
II. PST elemanları: okumak istediğimiz veri henüz yazmaçlara yazılmamış ama program sırası tablosu satırında duruyor olabilir.
III. veri yönlendirme ağı: aritmetik mantık birimlerinin işlem birimlerinde olan veri yönlendirmesinin sağlayan tellerin ağından okuyor olabilir.

Bu da karmaşıklığı arttıran bir durumdur. Okumak istediği değerin nerde olduğunu bilmesi gerekecek.

Yazmaç numarası ile program sırası tablosuna gittiğimizde aradığımız şeyi bulabilmek için bütün tabloyu kontrol etmesi gerekecek. Hangi satırda olduğu bilinmiyor.



PST'nin "içerikle aranabilir bellek" yapısında olması gerekir:(-) maliyet

Yazmaç numarası ile hem yazmaç öbeğine bakıyoruz hem de program sırası tablosunda bakıyoruz bu bir karmaşıklık yaratıyor. Bunu yapabilmemiz için bir kere ilk olarak program sırası tablosunun içeriğine göre arama yapabilmemiz gerekiyor. aramaktan çok adresleyebileceğimiz diyebiliriz. Çünkü 5 numaralı satır demiyoruz. 5 numaralı değerin olduğu satırı bulmaya çalışıyoruz. Bundan dolayı hepsi ile karşılaştırması gerekiyor. Bunu donanımda yapması zor ve gecikmesi yüksektir. Böyle olmasını istemiyoruz ve bundan dolayı;

Düşük Erişim Maliyetli Program Satırı Tablosu

Yazmaçlar geçerli olmadıklarında veriyi tutan PST adresini barındırırlar.

Geçerli olmama durumu: Verinin en güncel hali yazmaç öbeğinde değil:

I. veriyi hazırlayan buyruk yürütülüyor olabilir
II. veri PST'te olabilir

Tekrar: Veri Bağımlılıkları

- Yazmadan sonra okuma(YSO) Read After Write

- Okumadan Sonra Yazma(OSY) Write After Read

- Yazmadan Sonra Yazma(YSY)

OSY ve YSY neden gerçek bağımlılıklar değiller?

<u>OSY</u>
add x3 x0 x2 (I)
add **x0** x1 x4 (II)

I ve II'deki x0 değerlerinin birbirleri ile ilgisi yok. Sonsuz sayıda yazmacımız olursa ve derleyici birbiri ile ilgisi olmayan değerleri farklı yazmaçlara atarsa OSY ve YSY ile karşılaşmayız. Ama gerçek hayatta sınırlı yazmaca sahip olduğumuz için derleyici ister istemez aynı yazmaçları defalarca kez kullanabiliyor. Bundan dolayı yazmaç adından kaynaklanan bu bağımlılıklar ile karşılaşabiliyoruz.

YSO sonsuz sayıda yazmaç ile çözülebilir mi? çözülemez çünkü o gerçek bir veri bağımlılığıdır. Aynı yazmaca yazıp oradan okumaya çalıştığı için.

Yazmaç Yeniden Adlandırma

Gerçekte olmayan bağımlılıkları ortadan kaldırmak için yazmaç yeniden adlandırma tekniği kullanılır. Bu isim bağımlılığı program sırasını dağıtımına engel oluyor. Yazmaç numarası ile bir veri aramaya kalktığımızda doğru yazmaç verisini okuyacağımıza emin olamıyoruz. Dışarıdan görülen yazmaç numaraları ve içeriden görülen yazmaç numaralarını birbirinden ayırmayı düşünen bir fikir var. Bu numaraları yeniden adlandıralım.

Mimari Yazmaçlar → buyruk kümesi mimarisi tanımlamasında belirtilen yazmaçlar
Fiziksel Yazmaçlar → işlemcide bulunan yazmaçlar. gerçek yazmaçlar.

Yazmaç yeniden adlandırma birimi Mimari yazmaç → fiziksel yazmaç dönüşümlerini yönetir. Böyle olunca dışarıdan gelen bu yazmaç numaralarını içeride başka bir numaraya atamaya

başladığımızda aynı sanal bellek sisteminde kullandığımız gibi bu tablo kullanma ihtiyacı ortaya çıkıyor. Hangi numara hangi numaraya eşleştirildi diye tablo kullanma ihtiyacı oluyor.

Yazmaç yeniden adlandırma yöntemi sonsuz sayıda olmasa da BKM'de belirtilenden daha fazla yazmaç bulunduğu izlenimi verir. Yazmaç yeniden adlandırma yöntemine bağlı olarak bu yazmaç biriminin sayısının mimari yazmaç sayısının en az iki katı kadar olması gerekir. İleri doğru bir işlemin güvencesini vermek için sürekli bir yerde tıkanma olmaması için. Derleyicinin yazmaç kalmadığı için yarattığı o bağımlılıklar artık bizi sınırlamıyor.

Okumadan sonra yazma(OSY)
add x3 x0 x2
add x0 x1 x4
Yeniden adlandırma sonrasında:
add **P3** P0 P2
add **P5** P1 P4
yazmadan sonra yazma(YZY)
add x3 x0 x2
add x5 x3 x4
add x3 x6 x7
Yeniden adlandırma sonrasında:
add **P1** P0 P2
add P5 P1 P4
add **P8** P6 P7

gerçek bağımlılıkları ortadan kaldırmıyoruz. Zaten kaldırmamız yanlış olurdu.(P1-P1)

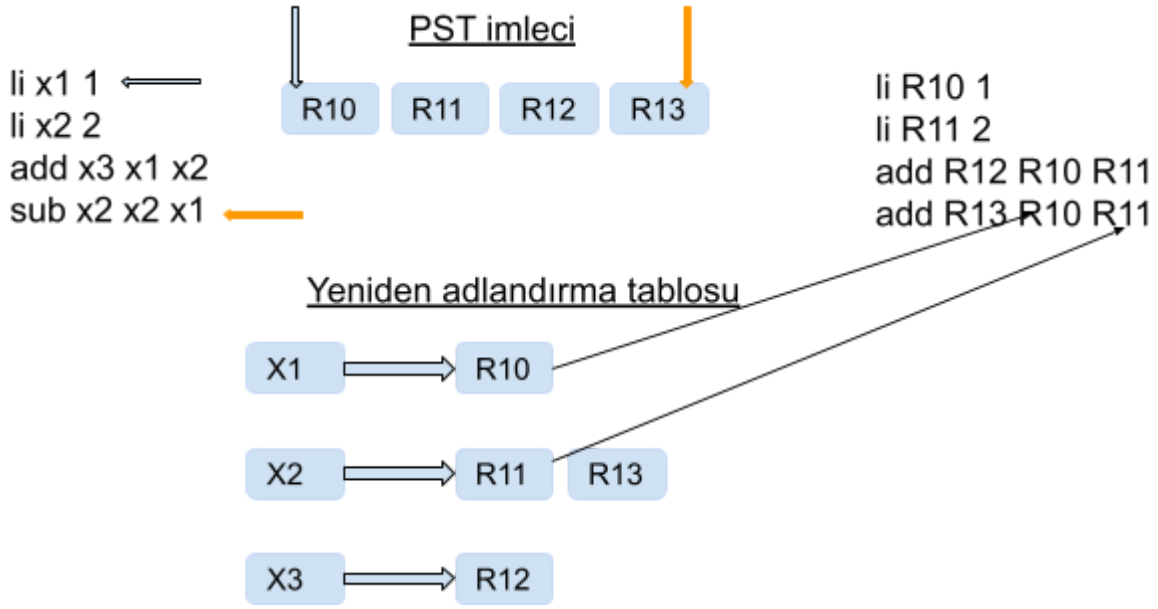
Artık isimleri farklı olduğun için yazmaçların program sırasını değiştirmek için bir engel kalmadı. Çünkü bağımlılıkları kaldırmış oldu. Gerçek veri bağımlılığı varsa bunun önünü bu şekilde alamayız.

Yazmaç yeniden adlandırma tablosu(YAT) (-ing. rename table, register aliasing table) her ***mimari yazmacın değerinin fiziksel olarak nerede tutulduğun***u(yazmaç öbeği ve ya PST olabilir) gösterir.

Çöz aşamasında:

- kaynak yazmaç numaraları YAT'a bakılarak doldurulur.
- hedef yazmaç numarasına PST imlecine bakılarak yeni bir PST elemanı atanır

Bu tablo artık işlemcinin içindeki durumu tuttuğu için aynı program sayacı ve yazmaç değerleri gibi kritik bir tablo haline gelir. Yani acil durumda kurtarılması gereken bir tablodur. Diyelim KDD oldu bu tablonun bir kopyasını bir kenarda tutmamız gerekir. Kaldığımız yerden devam edebilmek için.


PST imleci
li x1 1
li x2 2
add x3 x1 x2
sub x2 x2 x1
R10
R11
R12
R13
li R10 1
li R11 2
add R12 R10 R11
add R13 R10 R11
Yeniden adlandırma tablosu
X1
R10
X2
R11
R13
X3
R12