CISC, RISC ve EPIC Esasları

Mikroişlemcinin temel unsurları kaydediciler, veri yolları ve iş hatlarıdır. Bu unsurların büyüklüğü, sayısı, yapısı o işlemcinin yeteneklerini belirler ve bir mimariyi diğer mimarilerden ayırır.

Bilgisayar tarihinin başlarında, donanım fiyatlarının yüksek oluşundan dolayı çoğu bilgisayar oldukça basit komut kümesine sahipti. Sonraki yıllarda donanımı oluşturan elemanların üretimindeki artış, fiyatların düşmesine bunun sonucunda sistemde yüksek miktarda eleman kullanılmasına sebep oldu. Böylece fazla donanım kullanımı, komut kümesinin büyümesini ve sistemi çok karmaşık yapan donanımlarda kullanılmasını sağlamıştır.

Bir bilgisayarın komut kümesi, programcının makineyi programlarken kullanabileceği ilkel emirleri veya makine komutlarının tamamının oluşturduğu kümeyi belirtir. Bir komut setinin karmaşıklığı, komut ve veri formatlarına, adresleme modlarına, genel amaçlı kaydedicilere, opcode tanımlamalarına ve kullanılan akış kontrol mekanizmalarına bağlıdır. İşlemci tasarımındaki komut seti mimarileri CISC ve RISC olmak üzere iki çeşittir.

CISC Mimarisi (Complex Instruction Set Computer-Karmaşık komut kümeli Bilgisayar)

Intel'in X86 mimarisine dayalı işlemci serisinin ortaya çıktığı 70'li yıllarda, RAM'lerin pahalı ve kısıtlı olması sebebiyle bu kaynakların tasarruflu bir şekilde kullanılarak yüksek seviyeli dillerin desteklenmesini savunan bazı tasarım mimarları bir araya gelerek CISC mimarisini geliştirmişlerdir. Bu mimari, programlanması kolay ve etkin bellek kullanımı sağlayan tasarım felsefesinin bir ürünüdür. Her ne kadar performans düşüklüğüne sahip olsa ve işlemciyi karmaşık hale getirse de yazılımı basitleştirmektedir.

CISC mimarisinin karakteristik iki özelliğinden birisi, değişken uzunluktaki komutlar, diğeri ise karmaşık komutlardır. Değişken ve karmaşık uzunluktaki komutlar bellek tasarrufu sağlar. Karmaşık komutlar iki ya da daha fazla komutu tek bir komut haline getirdikleri için hem bellekten hem de programda yer alması gereken komut sayısından tasarruf sağlar. Karmaşık komut karmaşık mimariyi de beraberinde getirir. Mimarideki karmaşıklığın artması, işlemci performansında istenmeyen durumların ortaya çıkmasına sebep olur. Ancak programların yüklenmesinde ve çalıştırılmasındaki düşük bellek kullanımı bu sorunu ortadan kaldırabilir.

CISC Mimarisi (Complex Instruction Set Computer-Karmaşık komut kümeli Bilgisayar)

Tipik bir CISC komut seti, değişken komut formatı kullanan 120-350 arasında komut içerir. Bir düzineden fazla adresleme modu ile iyi bir bellek yönetimi sağlar.

CISC mimarisi çok kademeli işleme modeline dayanmaktadır. İlk kademe yüksek düzeyli dilin yazıldığı yerdir. Sonraki kademeyi makine dili oluşturur ki, yüksek düzeyli dilin derlenmesi sonucu bir dizi komutlar makine diline çevrilir. Bir sonraki kademede makine diline çevrilen komutların kodları çözülerek, mikroişlemcinin donanım birimlerini kontrol edebilen en basit işlenebilir kodlara (mikrokod) dönüştürülür. En alt kademede ise işlenebilir kodları alan donanım aracılığıyla gerekli görevler yerine getirilir.

CISC Mimarisi (Complex Instruction Set Computer-Karmaşık komut kümeli Bilgisayar)

İlk mikroişlemci tasarımları, komut kümesindeki her bir komutun şifresini çözme ve sonra işleme şeklinde çalışan adanmış mantık kullandılar. Bu düzen birkaç kaydedici içeren basit tasarımlar için iyi bir çalışmaydı. Ancak yapımı oldukça zor ve daha karmaşık mimarilerin doğmasına sebep oldu. Bu yüzden tasarımcılar işlemcinin farklı birimleri arasındaki veriyollarını kontrol etmek için birkaç basit mantık geliştirdiler.

Veriyolu mantığını kontrol etmek için basitleştirilmiş komutlara mikrokod denilir ve bu tip bir uygulama mikroprogramlı uygulama olarak bilinir. Bir mikroprogramlı sistemde işlemcinin komut kodlarının her birine karşılık gelen mikrokod komut gruplarını içeren belleği (tipik olarak ROM) vardır. Bir makine kodu işlemciye eriştiğinde, işlemci kodun daha basit komutçuklara ayrılmış dizilerini icra eder.

Komutlar yerel bir ROM bellekte olduğundan ana bellekten on kat daha hızlı bulunup getirilebilirler. Bundan dolayı tasarımcılar mümkün olduğunca çok komutu mikrokod ROM'a koymaya çalışırlar. Gerçekte sık sık kullanılan ve belirli uygulamalarda yavaş yordamların yerine, bu yordamları içeren mikrokodlu mikroişlemciler üretilmektedir.

CISC Mimarisi (Complex Instruction Set Computer-Karmaşık komut kümeli Bilgisayar)

İçerisinde mikrokod bulunduran ROM bellek, ana bellekten çok daha hızlı olduğu için, mikrokod bellekteki komut serisi fazla hız kaybetmeksizin dahili sistemde yürütülebilir.

Aynı komut kümesini adanmış mantık üzerinde yürütmek yerine, yeni yongalarla yürütmek daha kolaydır ve daha az transistör gerektirir.

Bir mikroprogramlı tasarım yeni komut kümelerini işlemek için tamamen değiştirilebilir.

Yeni komutlar mikrokod halinde eskilerin üzerine eklenir. Böylece geriye dönük uyumluluk tam olarak sağlanabilir.

Bazı makineler ticari hesaplamalar için, bazıları da bilimsel hesaplamalar için elverişli hale getirildiler. Bununla birlikte tümü aynı komut kümesini paylaştığından programlar makineden makineye, temel donanımlara bağlı kalarak, performansın mümkün olan artırımı ve azaltımı ile birlikte yeniden derlenmeden taşınabilir. Bu esneklik ve güç, mikrokodlamayı yeni bilgisayarları geliştirmek için tercih edilen yol yapar.

CISC Mimarisi (Complex Instruction Set Computer-Karmaşık komut kümeli Bilgisayar)

Bir mikroprogramlı tasarımı kullanmanın sonuçlarından birisi, tasarımcıların her bir komuta daha fazla işlevsellik katabilmeleridir. Programları çalıştırabilmek için gerekli toplam komut miktarını azaltan sadece bu değildir. Bu yüzden fazla bellek kullanımı daha etkili hale gelmiş ve assembly dilinde program yazanların durumunu kolaylaştırmıştır.

Daha sonra tasarımcılar, Assembly dili programcısının amaçladığı komutlarla kendi komut kümesini artırmıştır. Bu tip artırımlar, manevra işlemleri, özel döngü yapıları ve özel adresleme modlarını içermektedir.

CISC Mimarisi (Complex Instruction Set Computer-Karmaşık komut kümeli Bilgisayar)

Devre tasarımcıları, programcıya yakın komut kümesi oluşturmaya başladıktan sonra, sıra mantıksal adımlarla yüksek-düzeyli dillerden oluşan komut kümelerini yapılandırmaya gelmiştir. Bu adım, derleyici yazarların işini kolaylaştırdığı gibi derleyicilere kaynak kodu dizisi başına daha az komut çıkarmayı sağlar.

CISC mimarili işlemciler, tek bir çağrı ile yığın çerçeveleme ve yok etme yordamlarının da dahil olduğu çeşitli tipte komutları yürütür.

CISC'ın doğuşu

CISC tasarım kararları:

- Mikrokod kullanmak
- Zengin komut kümesini oluşturmak
- Yüksek seviyeli komut kümesini oluşturmak

Bu üç karar, bütün bilgisayarları 80'lerin sonuna taşıyan CISC felsefesinin temelidir ve bu gün de hala geçerliliğini korumaktadır. CISC adı, RISC mimarisinin gelişimine kadar bilgisayar tasarımcılarının sözlüğüne girmemişti ve bilgisayarları tasarladıkları tek mimariydi.

Bundan sonra bütün CISC tasarımlarının paylaştığı genel karakteristikler ve bu karakteristiklerin CISC yapısına sahip bir makinenin işlemine etkileri görülecektir.

CISC Tasarımının Özellikleri

80’li yıllara kadar çıkarılan çipler kendine has tasarım yollarını takip ettiler. Bunların çoğu “CISC tasarım kararları” denilen kurallara uydular. Bu çiplerin hepsinin benzer komut kümeleri ve donanım mimarileri vardır. Komut kümeleri, assembly dili programcılarının rahatlığı için tasarlanırlar ve donanım tasarımları oldukça karmaşıktır.

Komut Kümeleri

CISC mimarisinin gelişimine izin veren tasarım sınırlamaları az miktarda yavaş bellek ve ilk makinaların Assembly dilinde programlanması komut kümelerine bazı ortak karakteristikler yüklemiştir:

- Komutların bir kaynak ve bir hedefe sahip olması
- ADD R1,#25 komut satırında, ‘R11 hedef,’#25’ ise kaynak verisidir.
- Kaydediciden kaydediciye, kaydediciden belleğe ve bellekten kaydediciye komutlara sahip olması
- Diziler yoluyla belleği indislemek için geliştirilmiş pek çok adresleme modları içermesi
- Değişken uzunlukta komutlar içermesi (uzunluk genellikle adresleme moduna göre değişir)
- İcrası için birden fazla çevrim gerektiren komutlar. Eğer bir komut çalıştırılmadan önce ek bir bilgiye ihtiyaç duyarsa, ekstra bilgiyi toplamak fazladan çevrim gerektirecektir. Sonuç olarak bazı CISC komutlarını yürütmek, diğerlerinden daha uzun zaman alacaktır.

Donanım Mimarisi

- Pek çok adresleme modunu desteklemesi amacıyla tek bir komut tarafından karmaşık komut ve deşifre mantığı yürütülür.
- Az miktarda genel amaçlı kaydedici: Bellek üzerine doğrudan işlem yapabildiğinde, fazla genel amaçlı kaydediciye ihtiyaç yoktur. Az miktarda kaydedici, çok miktarda bellek kullanımı demektir.
- Pek çok özel amaçlı kaydedici: Çoğu CISC tasarımı, yığın işaretçisi ve kesme yöneticisi gibi özel amaçlı kaydedicileri kendisi kurar. Bu işlem donanım tasarımını bir dereceye kadar basitleştirir. Komutlar karmaşık olduğundan her parametrenin tutulması ayrı bir önem taşır.
- Bayrak kaydedicisi: Bu kaydedici, son işlemin sonucunun ne olduğunu işlemciye bildirir ve sonraki işlemin buna göre yönlendirilmesini sağlar.

İdeal CISC Mimarisi

CISC işlemciler, bir sonraki komuta başlamadan önce elindeki komutu tamamen icra etmek üzere tasarlanmıştır. Ama gerçekte böyle olmaz, çünkü komutlar çok karmaşıktır ve tek saykılda işlenmez. İş-hattında beklemelere sebep olurlar. Bu sebeple çoğu işlemci bir komutun icrasını pek çok belirli aşamaya ayırır. Bir aşama biter bitme sonuç bir sonraki aşamaya aktarılır.

- Al getir: Bir komut ana bellekten alınıp getirilir.
- Kodunu çöz: Komutun şifresi çözülür. Eğer gerekliyse işlemci bellekten ek bilgi okur.
- Çalıştır: Komut işlenir. Mikroprogramın kontrol kodu işletimi yürütecek donanım çevrimini belirler.
- Tekrar yaz: Sonuçlar belleğe yazılır.

İdeal bir CISC makinasında, her bir komut sadece bir çevrim gerektirir. Gerçekte bu bir anda bir komutu icra eden makine için mümkün olan en yüksek hızdır.

Gerçekçi bir CISC Makinesi

Gerçekte bazı komutlar aşama başına birden fazla saat çevrimi gerektirir. Bununla birlikte, CISC oluşumunun temelindeki düşünce, toplam çevrim sayısını küçük tutmak olduğundan bir CISC tasarımı bu yavaşlamayı uygun görebilir.

Performansı belirlemek için aşağıdaki eşitlik kullanılmaktadır.

İcra süresi=$\sum_{i=1}^{N}$ komut başına çevrim sayısı[i] *çevrim süresi

N=komut sayısı

CISC Mimarisinin Üstünlükleri

- CISC makinalar ilk gelişim sıralarında bilgisayar performansını yükseltmek için mevcut teknolojileri kullandılar.

- Mikroprogramlama assembly dilinin yürütülmesi kadar kolaydır ve sistemdeki kontrol biriminden daha ucuzdur.

- Yeni komutlar ve mikrokod ROM'a eklemenin kolaylığı tasarımcılara CISC makinalarını ilk bilgisayarlar gibi çalıştırabilirler çünkü yeni bilgisayar önceki bilgisayarın komut kümelerini de içerecektir.

- Her bir komut daha yetenekli olmaya başladığından verilen bir görevi yürütmek için daha az komut kullanılır. Bu, nispeten yavaş ana belleğin daha etkili kullanımını sağlar.

- Mikroprogram komut kümeleri, yüksek seviyeli dillerine benzer biçimde yazılabildiğinden derleyici karmaşık olmak zorunda değildir.

CISC Mimarisinin Sakıncaları

- İşlemci ailesinin ilk kuşakları her yeni versiyon tarafından kabullenilmiştir. Böylece komut kodu ve çip donanımı bilgisayarların her kuşağıyla birlikte daha karmaşık hale gelmiştir.
- Mümkün olduğu kadar çok komut, mümkün olan en az zaman kaybıyla belleğe depolanabiliyor ve komutlar neredeyse her uzunlukta olabiliyor. Bunun anlamı farklı komutlar farklı miktarlarda saat çevrimi tutacaktır bu da makinanın performansını düşürecektir.
- Çoğu özel güçlü komutlar geçerliliklerini doğrulamak için yeteri kadar sık sık kullanılmıyor. Tipik bir programda mevcut komutların yaklaşık %20'si kullanılıyor.
- Komutlar genellikle bayrak (durum) kodunu komuta bir yan etki olarak kurar. Bu ise ek saykıllar yani bekleme demektir. Aynı zamanda sonraki komutlar işlem yapmadan önce bayrak bitlerinin mevcut durumunu bilmek durumundadır. Bu da yine ek saykıl demektir. Bayrakları kurmak zaman aldığı gibi, programlar takip eden komutun bayrağın durumunu değiştirmeden önce bayrak bitlerini incelemek zorundadır.

RISC Mimarisi

RISC mimarisi, CISC mimarili işlemcilerin kötü yanlarını gidermek için piyasanın tepkisi ile ona bir alternatif olarak geliştirilmiştir. RISC'ı IBM, Apple ve Motorola gibi firmalar sistematik bir şekilde geliştirmiştir. RISC felsefesinin taraftarları, bilgisayar mimarisinin tam anlamıyla bir elden geçirmeye ihtiyacı olduğunu ve neredeyse bütün geleneksel bilgisayarların mimari bakımından birtakım eksikliklere sahip olduğunu ve eskidiğini düşünüyorlardı. Bilgisayarların gittikçe daha karmaşık hale getirildiği ve hepsinin bir kenara bırakılıp en baştan geri başlamak gerektiği fikrindeydiler.

70'lerin ortalarında yarı iletken teknolojisindeki gelişmeler, ana bellek ve işlemci yongaları arasındaki hız farkını azaltmaya başladı. Bellek hızı artırıldığından ve yüksek seviyeli diller Assembly dilinin yerini aldığından, CISC'ın başlıca üstünlükleri geçersizleşmeye başladı. Bilgisayar tasarımcıları sadece donanımı hızlandırmaktan çok bilgisayar performansını iyileştirmek için başka yollar denemeye başladılar.

İlk RISC Modeli

IBM 70'lerde RISC mimarisini tanımlayan ilk şirket olarak kabul edilir. Aslında bu araştırma temel mimarisel modeller ortaya çıkarmak için Berkeley ve Standford üniversitelerince daha fazla geliştirildi. RISC'ın felsefesi üç temel prensibe dayanır.

Bütün komutlar tek bir çevrimle çalıştırılmalıdır: Performans eşitliğinin gerekli kısmı budur. Gerçekleştirilmesi bazı özelliklerin varolmasına bağlıdır. Komut kodu harici veri yoluna eşit ya da daha küçük sabit bir genişlikte olmalı, ilave edilmek istenen operandlar desteklenmemeli ve komut kodu çözümü gecikmelerini engellemek için dikey ve basit olmalı.

Belleğe sadece "load" ve "store" komutlarıyla erişilmelidir. Bu prensip ilkinin doğal sonucudur. Eğer bir komut direkt olarak belleği kendi amacı doğrultusunda yönlendirirse onu çalıştırmak için birçok saykıl geçer. Komut alınıp getirilir ve bellek gözden geçirilir. RISC işlemcisiyle belleğe yerleşmiş veri bir kaydediciye yüklenir, kaydedici gözden geçirilir ve son olarak kaydedicinin içeriği ana belleğe yazılır. Bu seri en az üç komut alır. Kaydedici tabanlı işlem gerçekleştirmeyle performansı iyi durumda tutmak için çok sayıda genel amaçlı kaydediciye ihtiyaç vardır.

İlk RISC Modeli

Bütün icra birimleri mikrokod kullanmadan donanımdan çalıştırılmalıdır: Mikrokod kullanımı, dizi ve benzeri verileri yüklemek için çok sayıda çevrim demektir. Bu yüzden tek çevrimli icra birimlerinin yürütülmesinde kolay kullanılmaz.

Günümüzün RISC yapısına sahip ticari mikroişlemcilerinde genel olarak iki tarz görülür. *Bunlar Berkeley Modeli* ve *Standford* modelidir. Aralarındaki esas fark kaydedici kümeleri ve bunların kullanımıyla alakalıdır. Her ikisi de veri ve komutların birbirinden ayrıldığı veriye erişimin paralel yapıldığı ve komut ve veri uyuşmazlığını ortadan kaldıran *Harward harici veriyolu mimarisine* sahiptir. Eğer bu iki akış tek bir yoldan yapılmaya kalkışılırsa, herhangi bir veri çağırması komut akışını durdurur ve işlemcinin tek çevrimde işleme hedefine ulaşmasının önüne geçer.

RISC Mimarisinin Özellikleri

RISC mimarisi aynı anda birden fazla komutun birden fazla birimde işlendiği *iş hatlı tekniği* ve *süperskalar* yapılarının kullanımıyla yüksek bir performans sağlamıştır. Bu tasarım tekniği yüksek bellek ve gelişmiş derleme teknolojisi gerektirmektedir. Bu mimari küçültülen komut kümesi ve azaltılan adresleme modları sayısı yanında aşağıdaki özelliklere sahiptir.

- Bir çevrimlik zamanda bir komut işleyebilme
- Aynı uzunluk ve sabit formatta komut kümesine sahip olma
- Ana belleğe sadece load ve store komutlarıyla erişim, operasyonların sadece kaydedici üzerinde yapılması
- Bütün icra birimlerinin mikrokod kullanılmadan donanımsal çalışması
- Yüksek seviyeli dilleri destekleme
- Çok sayıda kaydediciye sahip olması

İş Hattı Tekniği

Bilgisayar donanımının bir anda birden fazla komutu işlemcinin farklı alanlarında işleyebildiği tekniğe iş hattı tekniği denir. Bir komutu ele almaya başlamadan önce bir öncekinin tamamlanmasını beklemez.

CISC tabanlı makinelerde bir komutun işlenmesi 4 adımda yapılmaktadır. Bunlar komutu bellekten alıp getirmek, kodunu çözmek, işlemek ve yeniden belleğe yazmaktır. Bu kademeler RISC tabanlı makinelerde de bulunur. Fakat paralel olarak icra edilirler. Bir kademedeki işlem biter bitmez sonucu diğer kademeye aktarılır ve diğer komut üzerinde çalışmaya başlanılır. Tek iş-hatlı sistemin performansı, bir bölümün tamamlanması için gereken zamana bağlıdır. İş-hattı tekniği kullanmayan tasarımlarda olduğu gibi tüm safhalar için geçen toplam zamana bağlı değildir.

Bu tekniğe sahip RISC tabanlı işlemcilerde, her bir komut her kademede bir saat çevrimi harcar. Böylece işlemci her saat çevrimi başına yeni bir komut kabul edebilir.

İş Hatlı Sistemlerde Performans

İş hattı tekniği kullanan bir işlemci, bellekten veri okuma sırasındaki beklemeler, sınırlı komut kümesi tasarımı veya komutlar arasındaki uyumsuzluk gibi değişik durumlardan dolayı atıl durumda kalabilir.

Bellek Hızı

Bellek hızı sorunları çoğunlukla ön bellek kullanımıyla çözülmüştür. Statik RAM'den oluşan ön bellek *işlemci ile dinamik RAM'e sahip ana bellek arasına yerleştirilmiş* hızlı bir bellek türüdür. İşlemci ana bellekten bir alanı okumak istediğinde, bu alan aynı zamanda ön belleğe kopyalanır. Dolayısıyla, ana belleğe yazmak isterse de veri önce ön belleğe yazılır. Uygun bir zaman bulunduğunda da ana belleğe geçirilir.

Komut Gizliliği

Zayıf tasarlanmış bir komut kümesi, iş-hatlı tekniğe sahip işlemcinin sık sık durmasına sebep olabilir. Genel sorunların bazıları şöyledir:

Yüksek düzeyde şifrelenmiş komutlar: Bu tip komutlar CISC tabanlı makinelerde kullanılır ve çözmek için bir dizi testler uygulanır.
Değişken uzunlukta komutlar: Komutun tümünü getirmek belleğe çok yönlü başvurular gerektirir. Komutların bazıları bir kelime bazıları birden fazladır.
Ana belleğe giriş yapan komutlar: Ana bellek yavaş olabileceğinden işlemci durabilir.
İşlenmesi için çok fazla saat çevrimi gerektiren karmaşık komutlar: Mesela kayan noktalı işlemler
Aynı kaydediciye yazma ve okuma ihtiyacı olan komutlar: Kaydedicinin önceki okuma işleminden hala meşgul olması sebebiyle, kaydedicinin kullanılabilir olmasına kadar işlemcinin durmasına sebep olabilir.
Tek nokta kaynaklarına olan güvenilirlik: Bayrak kaydedicilerindeki durumları eğer bir komut kurarsa ve takip eden komut bu bitler okumaya çalışırsa, ikinci komut birinci komutun yazma işlemi bitene kadar beklemek zorundadır.

Güvenilirlik

RISC programcılarının problemlerinden bir tanesi, zayıf komut kümesinden dolayı işlemcinin yavaşlayabilmesidir.
Her bir komutun sonucunu depolamak için bir miktar zaman harcanmasından ve birkaç komutun aynı zamanda ele alınmasından dolayı sonraki komutların ilk komutların sonuçlarının depolanmasını beklemek zorundadır. Bununla birlikte bir program içindeki komutların basit olarak yeniden düzenlenmesi RISC programlarının bu performans kısıtlamalarını ortadan kaldırabilir.

Süper İş hattı Teknolojisi

Süper iş hatlı sistem, iş hattının her bir kademesini 2 alt devreye ayırır ve saat hızını dâhili olarak ikiye katlar. Her bir kademe saat başına 1 komut icra eder. Ancak dahili saat iki kat hızlı olduğundan iş hatlı saat darbesinin her vuruşunda iki komutu yükleyebilir.

Süperskalar Mimari

Süperskalar makineler aynı şeyi yapmakta yetenekli olan pek çok icra birimini içerir. Bu işlemcinin, birkaç benzer komutun her birini eldeki icra birimlerine dağıtarak işlemesine izin verir.

Mesela iki aritmetik birimi olan süperskalar makine iki çift sayıyı aynı anda (sonuçlarının aynı yere gitmek zorunda olmadığı durumlarda) toplayabilir. Bu makine iki çıkışlı makine olarak adlandırılır. RISC komut kümesi süperskalar mimariye tam uyumludur. Bunu sebebi çip üzerinde daha az yer tutan ve dolayısıyla bir veya daha fazla kez kopyası çıkartılabilen basit işlem birimlerinde her bir komutun icra edilebilmesidir çünkü, komutlar arasındaki bağımlılık önemsenmemektedir. (Eğer iki komut bir kaydedici veya bir bayrak kodları gibi aynı kaynakları isterse, bunlar aynı kaynakları isterse, bunlar aynı anda yürütülemezler).

RISC'de kaydedici sayısı çoktur. Dolayısıyla aynı kaydediciye nadir talep görülür. Bir çok-çıkışlı makine genellikle bir algetir kademesine ve bir kod-çözme kademesine sahiptir. Fakat bu kademeler her saat çevriminin bir parçası içinde işlenir. Böylece bunlar makinenin tüm hızını sınırlamazlar.

RISC Mimarisinin Üstünlükleri

RISC tasarımı olan bir mikroişlemciyi kullanmak, karşılaştırılabilir bir CISC tasarımını kullanamaya göre pek çok avantaj sağlar.

Hız: Azaltılmış komut kümesi, kanal ve süperskalar tasarıma izin verdiğinden RISC işlemciler genellikle karşılaştırılabilir yarı iletken teknolojisi ve aynı saat oranları kullanılan CISC işlemcilerinin performansının 2 veya 4 katı daha yüksek performans gösterirler.

Basit Donanım: RISC işlemcinin komut kümesi çok basit olduğundan çok az çip uzayı kullanırlar. Ekstra fonksiyonlar, bellek kontrol birimleri veya kayan noktalı aritmetik birimleri de aynı çip üzerine yerleştirilir.

Kısa Tasarım Zamanı: RISC işlemciler CISC işlemcilere göre daha basit olduğundan daha çabuk tasarlanabilirler ve diğer teknolojik gelişmelerin avantajlarını CISC tasarımlarına göre daha çabuk kabul edebilirler.

RISC Mimarisinin Sakıncaları

CISC tasarım stratejisinden RISC tasarım stratejisine yapılan geçiş kendi problemlerini de beraberinde getirmiştir. Donanım mühendisleri kodları CISC işlemcisinden RISC işlemcisine aktarırken anahtar işlemleri göz önünde bulundurmak zorundadır.

Kod Özelliği

Bir RISC işlemcisinin performansı işlediği kodun algoritmasına çok bağlıdır. Eğer programcı veya derleyici, komut programlamada zayıf iş çıkarırsa, işlemci atıl durumda kalarak bir parça zaman harcayabilir.

Programlama kuralları karmaşık olabileceğinden çoğu programcılar yüksek düzeyli bir dil kullanırlar ve komut düzenlemeyi derleyiciye bırakırlar.

RISC uygulamasının performansı, derleyici tarafından oluşturulan kodun özelliğine bağlı olduğundan dolayı geliştiriciler işlenmiş kodun özelliğine dayanan derleyicileri dikkatle seçmek zorundadırlar.

Hatalardan Arındırma

Komut planlaması dikkatli yapılmazsa hatalardan arındırmayı zorlaştırabilir. Makine dili komutlarının karıştırılması kodu okumayı zorlaştırır. Bunun için programcı kodlama yaparken dikkatli olmak zorundadır.

Kod Büyümesi

Kod genişlemesi CISC makinesi için derlenen ve RISC makinesi için tekrar derlenen bir programın aradaki göreceli uzunluk farkını işaret eder. Tam genişleme aslında derleyicinin niteliğine ve makinenin komut kümesi yapısına bağlıdır.

CISC tabanlı makinelerin karmaşık işlemlerinin tek bir komut ile yürütülmesinden dolayı kod genişlemesi problem olabilir.

Sistem Tasarımı

RISC makinelerin diğer bir problemi de komutlarını beslemek için çok hızlı bellek sistemleri gerektirmeleridir. CISC tabanlı sistemler genellikle kendi çipleri üzerinde L1 Ön bellek denilen bellekleri taşırlar.

EPIC Mimarisi (Explicitly Parallel Instruction Computing) Belirtilmiş Paralel Komutlarla Hesaplama

Çok uzun kelimeli (VLIW) bilgisayarlar, yazılımın paralelizme ilişkin kesin bilgi sağladığı mimari örneklerdir. Derleyici programdaki paralelliği tanımlar ve hangi işlemlerin bir başkasından bağımsız olduğunu belirterek donanıma bildirir. Bu bilgi, aynı çevrimde hangi işlerin başlatılabileceğiyle ilgili daha fazla denetim olmadan donanımla doğrudan değerlendirilir.

EPIC tarzı mimari, VLIW tekniğinin geliştirilmiş bir modelidir denilebilir. Süperskalar işlemcilerin en iyi yönlerinin bir çoğu EPIC felsefesine adapte edilmiştir. Çok belirgin RISC mimarileri olduğu gibi EPIC yapısı içinde bir komut kümesi mimarisinden fazlası vardır.

EPIC Mimarisinin Avantajları

- Paralel çalıştırma (çevrim başına birden çok komut çalıştırma)
- Tahmin kullanımı
- Spekülasyon kullanımı
- Derleme anında paralelizmi tanıyan derleyiciler
- 128 kayan nokta, 128 tamsayı, 64 tahminli büyük kaydedici kümesi
- Dallanma tahmini ve bellek gecikmesi problemlerine karşı üstün başarı
- Gelişme ve yeni birimlerin eklenmesine verilen doğal yapıdan kaynaklanan destek ve eskiye karşı uyumluluk

EPIC Mimarisinin Avantajları

- Paralel çalıştırma (çevrim başına birden çok komut çalıştırma)
- Tahmin kullanımı
- Spekülasyon kullanımı
- Derleme anında paralelizmi tanıyan derleyiciler
- 128 kayan nokta, 128 tamsayı, 64 tahminli büyük kaydedici kümesi
- Dallanma tahmini ve bellek gecikmesi problemlerine karşı üstün başarı
- Gelişme ve yeni birimlerin eklenmesine verilen doğal yapıdan kaynaklanan destek ve eskiye karşı uyumluluk

X86 Komut Yapısı

Komutlar ve talimatlardan meydana gelen assembly dilinde yazılmış kaynak kodlarının her bir satırında, dört ayrı alan tanımlanabilir. Bunlar etiket alanı, komut alanı, operand alanı ve açıklama alanlarıdır. Etiket, komut ve operand alanları birbirinden bir veya daha fazla boşluk ya da tab ile ayrılırlar. Açıklama alanı ; ile ayrılmalıdır.

Açıklama Alanları

Açıklama programın belli yerlerine ileride dikkat çekmek amacıyla kullanılan bir tanımdır. Programcı için seçimliktir. Eğer isterse programcı bu satırları kullanmayabilir.

```
MOV AX,15H ;15H sayısını AX kaydedicisine yükle
ADD BX,AX ;AX kaydedicisindeki veriyi BX'e yükle
```

Etiket Alanı

Bu alan sembolik bir isimdir ve komut satırının ilk başına konur. Etiketin ilk karakteri sayısal olmamak üzere tüm karakterleri içerebilir. Etiket adı maksimum 32 karakter uzunluğunda kullanılabilir.

BASLA: JMP ANA

- Etiket adları mümkünse kısa ve anlamlı olmalıdır.
- Etiket alanında bir birine benzeyen isimler kullanılmamalıdır.
- Etiket adında birbirine benzeyen karakterler bir arada kullanılmamalıdır. 0 ile O, S ile 5 gibi

Komut Alanı

Mikroişlemciye bir işi tarif eden ve mühendisler tarafından komut cümlesinin kısaltılarak mnemonik hale getirilmiş anlamlı kelimelerin kullanıldığı bu alana komut alanı, aksiyon alanı veya mnemonik alanı denilir. Etiket alanından sonra bir tab veya boşlukla girilen bu alanda bu komutlar bulunur.

CMP
JC
XCHG

Operand Alanı

Operand alanı işlemciye işlenecek verinin nerede oluğunu söyleyen kısımdır. Bu operandlara üzerinde iş yapılan veri denir. Bu alanda komut alanı ile arada bir veya iki karakterlik boşluk bırakılır. İki operand arasına virgül konulur.

```
CMP AX,BX
ADD AX,[BX]
MOV CX,00
NOT AX
JNE BASLA
```

İki operandın bulunduğu alanda ilki hedef operandı, ikincisi kaynak operandı temsil etmektedir.

Operand alanında kullanılan doğrudan verilerin sonunda B,H,D ve O gibi son takılar yazılır. Bunlar o sayıların hangi tabanda olduğunu gösterir. (B:binary, H:Hex,D: Desimal, O:Oktal)