BİLGİSAYAR MİMARİSİ DERSİ

ARAŞTIRMA ÖDEVİ-IV

Ayben GÜLNAR-191180041

KASIM 2022

İçindekiler Tablosu

Şekiller Listesi

1.ÖZET

Mikro mimarilerinin felsefesi aynı olsa da uygulama yaklaşımlarında farklılık gösterirler. Sürekli bir rekabet dünyasında yaşıyoruz. Zaman geçtikçe, uygulamalar daha fazla işlem gücüne ihtiyaç duyuluyor ve bu mikroişlemcilerin mimarisi üzerinde de birçok yeniliğe yol açmıştır. Mimari meseleleri, sınırlamaları, mimarileri ve özelliklerini tartışıyoruz.

Komut seti mimarisi, bir makine dili programcısının (veya bir derleyicinin) o makine için doğru (zamanlamadan bağımsız) bir program yazmak için anlaması gereken bir bilgisayar yapısıdır. [1]. Komut seti uzantıları, çok çeşitli uygulama alanlarını ve programlama kullanımlarını kapsar. Intel® Gelişmiş Vektör Uzantıları 512 (Intel® AVX-512) yönergeleri olarak anılan 512-bit SIMD vektör SIMD uzantıları, kapsamlı işlevsellik seti ve Intel® Gelişmiş Vektör Uzantıları (Intel® AVX). [2] i9 işlemciler, Intel'in 2018 yılının 2.çeyreğinde i9-8950HK modeli ile piyasaya sürdüğü Intel Core İşlemci ailesinin, X serisinden sonra gelen en üst segmentteki işlemcileri olarak tanımlanabilir. Tıpkı Intel'in serisinde olduğu gibi, Ryzen de Ryzen 3, 5, 7 ve 9 olmak üzere dört katmana sahiptir. Temelde General-Purpose Instructions, Streaming SIMD Extensions Instructions, Multimedia Extension Instructions ve x87 Floating-Point Instructions olarak ayrılabilir.

Bu araştırmamda öncelikli olarak Komut setleri ve mimarileri nedir detaylıca 2. Bölümde ele aldım. Sonrasında karşılaştırılmamız istenen inteli9 ve Amd Ryzen'in çok detaylıca özelliklerini inceleyip bunları karşılaştırdım. Son bölümde ise İntel ve Ryzen için ayrı ayrı komut setleri ve mimarilerini detaylıca inceledikten sonra tekrar bu mimarileri karşılaştırdım.

2.INSTRUCTİON SET ARCHİTECTURE

Bilgisayar mimarisi dersinin temel iki bileşeni: komut seti mimarisi ve bilgisayar organizasyonunun kendisidir. ISA(Instruction Set Architecture), işlemcinin neler yapabileceğini ve ISA bunun nasıl başarılacağını belirtir. Yani komut seti mimarisi temel olarak donanımınız ve yazılımınız arasındaki arayüzdür. Donanımla etkileşim kurmanın tek yolu, işlemcinin komut setidir. Bilgisayara komut vermek için onun dilini konuşmanız gerekir ve yönergeler bir bilgisayarın dilinin sözcükleridir ve yönerge seti temel olarak onun söz dağarcığıdır. Sözcük dağarcığını bilmeden ve çok iyi bir sözcük dağarcığına sahip olmadıkça makineden iyi faydalar elde edemezsin. ISA, makinenin derleme dili programcısı veya derleyici yazarı veya uygulama programcısı tarafından görülebilen kısmıdır. Bu, sahip olduğunuz tek arabirimdir, çünkü komut kümesi mimarisi, bilgisayarın neler yapabileceğinin belirtimidir ve makinenin, ISA'nızda belirtilen her şeyi yürütecek şekilde imal edilmesi gerekir. Makinenizle konuşmanın tek yolu ISA'dır. Bu size donanım ve yazılım arasındaki arayüz hakkında bir fikir verir.

Üzerinde çalışmak istediğiniz mimariden bağımsız, C ile yazılmış üst düzey bir programınız olduğunu varsayalım. Bu üst düzey programın, belirli bir mimariye özgü bir montaj dili programına çevrilmesi gerekir. Diyelim ki bunun LOAD, STORE, ADD, vs. belirli mimari. Burada gösterilen tüm bu talimatlar, MIPS mimarisinin talimat seti mimarisinin bir parçasıdır. Bunların hepsi İngiliz gibidir ve bu işlemci tarafından anlaşılamaz çünkü işlemci sonuçta yalnızca sıfırları ve birleri anlayabilen dijital bileşenlerden oluşur. Dolayısıyla bu montaj dilinin, sıfırlardan ve birlerden oluşan nesne kodu olan makine diline hassas bir şekilde çevrilmesi gerekecektir. Bu nedenle, üst düzey dilinizden derleme dilinize ve ikili koda çevirinin derleyici ve derleyici ile yapılması gerekecektir.

Komut seti özelliklerine bakacağız ve sıfırlara ve birlere neyin gireceğini ve sıfırları ve birleri veri, komut veya adres olarak nasıl yorumlayacağımızı göreceğiz. Tasarlanan ISA birçok uygulama üzerinden devam etmeli, taşınabilir olmalı, uyumlu olmalı, çok farklı şekillerde kullanılabilmeli yani genelliğe sahip olmalı ve ayrıca diğer seviyelere uygun işlevsellik sağlamalıdır. ISA'nın taksonomisi aşağıda verilmiştir.

ISA'lar, bir işlemcideki dahili depolamaya bağlı olarak farklılık gösterir. Buna göre ISA, işlenenlerin nerede depolandığına ve açık veya örtülü olarak adlandırılıp adlandırılmadığına bağlı olarak aşağıdaki gibi sınıflandırılabilir:

Genel amaçlı kayıtlardan birini akümülatör olarak adlandıran ve onu işlenenlerden birini zorunlu olarak depolamak için kullanan tek akümülatör organizasyonu.

Bu, işlenenlerden birinin akümülatörde olduğunun ima edildiğini gösterir ve diğer işlenenin komutla birlikte belirtilmesi yeterlidir. Tüm işlenenleri açıkça belirten genel kayıt organizasyonu. İşlenenlerin bellekte veya kayıtlarda bulunup bulunmadığına bağlı olarak, ayrıca şu şekilde sınıflandırılabilir:

– Register – register: registerların işlenenleri depolamak için kullanıldığı yer. Bu tür mimarilere aslında yük depolama mimarileri de denir, çünkü yalnızca yükleme ve depolama komutları bellek işlenenlerine sahip olabilir.

– Register – memory: Birisinin register diğerinin memory olması

– Memory – memory: Tüm operandların memory operandı olması

İşlenenlerin yığına konulduğu ve işlemlerin yığının en üstünde yürütüldüğü yığın organizasyonu. İşlenenler burada dolaylı olarak belirtilir. Üç işlenenin de bellek işlenenleri olduğu A = B + C işlemini gerçekleştirmeniz gerektiğini varsayalım. Genel amaçlı yazmaçlardan birinin toplayıcı olarak atandığını ve işlenenlerden birinin toplayıcıda her zaman mevcut olacağını varsaydığımız, biriktirici tabanlı bir ISA söz konusu olduğunda, başlangıçta bir işleneni biriktiriciye yüklemeniz gerekir. ve ADD komutu sadece işlenenin adresini belirtecektir. GPR tabanlı ISA'da üç farklı sınıflandırmanız vardır. Yazmaç belleği ISA'da, bir işlenen herhangi bir yazmaca taşınmak zorundadır ve diğeri bir bellek işleneni olabilir. Kayıt – kayıt ISA'da, her iki işlenenin de iki kayda taşınması gerekecek ve ADD komutu sadece kayıtlar üzerinde çalışacaktır. Bellek – bellek ISA, her iki bellek işlenenine de izin verir. Böylece doğrudan ekleyebilirsiniz. Yığın tabanlı bir ISA'da, her şeyden önce her iki işleneni yığına itmeniz ve ardından yığının en üstteki iki öğesini ekleyecek ve ardından sonucu yığında depolayacak bir ekleme talimatı vermeniz gerekecektir. Bu örneklerden aynı işlemi gerçekleştirmenin farklı yollarına sahip olduğunuzu görebilirsiniz ve bu açıkça ISA'ya bağlıdır. Tüm bu ISA'lar arasında en popüler olan ve tüm RISC mimarilerinde kullanılan register – register ISA'dır. Şimdi komut seti mimarisini tasarlarken dikkate alınması gereken farklı özelliklerin vardır. Bunlar:

1. Komut türleri (Komut setindeki işlemler)
2. İşlenen türleri ve boyutları
3. Adresleme Modları

4. Adresleme Belleği
5. Kodlama ve Komut Formatları
6. Derleyici ile ilgili sorunlar

Her şeyden önce, talimat türlerine, yani ISA'da desteklemek istediğiniz çeşitli talimatlara karar vermelisiniz. Bir bilgisayar programı tarafından gerçekleştirilen görevler, iki sayıyı çarpmak, bir veriyi bir yazmaçtan bir bellek konumuna taşımak, sıfır gibi belirli bir durumu test etmek, giriş cihazından bir karakter okumak veya göndermek gibi bir dizi küçük adımdan oluşur. çıktı aygıtına görüntülenecek bir karakter vb. Bir bilgisayar aşağıdaki türde yönergelere sahip olmalıdır:

1. Veri aktarım talimatları
2. Veri işleme talimatları
3. Program sıralama ve kontrol talimatları
4. Giriş ve çıkış komutları

Veri aktarım talimatları, bilgisayar sistemindeki çeşitli depolama yerleri arasında veri aktarımı gerçekleştirir, yani. kayıtlar, bellek ve G/Ç. Hem komutlar hem de veriler bellekte saklandığından, işlemcinin komutları ve verileri bellekten okuması gerekir. İşlemden sonra, sonuçlar hafızada saklanmalıdır. Bu nedenle, belleği içeren iki temel işlem gereklidir, yani Yükle (veya Oku veya Getir) ve Sakla (veya Yaz). Load işlemi, verilerin bir kopyasını bellekten işlemciye aktarır ve Store işlemi, verileri işlemciden belleğe taşır. Verileri bir kayıttan diğerine veya G/Ç aygıtlarından ve işlemciden aktarmak için başka veri aktarım talimatları gereklidir. Veri manipülasyon talimatları, veriler üzerinde işlemler gerçekleştirir ve işlemci için hesaplama yeteneklerini gösterir. Bu işlemler aritmetik işlemler, mantıksal işlemler veya kaydırma işlemleri olabilir. Aritmetik işlemler toplama (taşımalı ve taşımasız), çıkarma (ödünç almalı ve almama), çarpma, bölme, artırma, eksiltme ve bir sayının tümleyenini bulmayı içerir. Mantıksal ve bit işleme komutları AND, OR, XOR, Clear carry, set carry vb. içerir. Benzer şekilde, farklı türde kaydırma ve döndürme işlemleri gerçekleştirebilirsiniz.

Genellikle sıralı bir talimat akışı olduğunu varsayarız. Yani, ardışık konumlarda saklanan komutlar birbiri ardına yürütülür. Ancak, programın akışını değiştirmenize yardımcı olan program sıralama ve kontrol talimatlarına sahipsiniz. Bu en iyi bir örnekle açıklanır. N sayıdan oluşan bir liste ekleme görevini düşünün. Olası bir sıra aşağıda verilmiştir.

```
Move DATA1, R0
Add DATA2, R0
Add DATA3, R0
```

```
Add DATAn, R0
Move R0, SUM
```

N sayı içeren hafıza yerlerinin adresleri sembolik olarak DATA1, DATA2, şeklinde verilmiştir. DATAn ve her bir Databer'ı R0 yazmacının içeriğine eklemek için ayrı bir Add komutu kullanılır. Tüm sayılar eklendikten sonra, sonuç SUM bellek konumuna yerleştirilir. Uzun bir Ekleme yönergeleri listesi kullanmak yerine, aşağıda gösterildiği gibi bir program döngüsüne tek bir Ekleme yönergesi yerleştirmek mümkündür:

```
Move N, R1
Clear R0
LOOP Determine address of “Next” number and add “Next” number to R0
Decrement R1
Branch > 0, LOOP
Move R0, SUM
```

Döngü, gerektiği kadar çok kez yürütülen düz bir talimat dizisidir. DÖNGÜ konumunda başlar ve Şube 0 komutunda biter. Bu döngüden her geçişte, bir sonraki liste girişinin adresi belirlenir ve bu giriş getirilir ve R0'a eklenir. Bir işlenenin adresi, bir sonraki bölümde açıklanacağı gibi çeşitli şekillerde belirtilebilir. Şimdilik, bir program döngüsünü nasıl oluşturacağınızı ve kontrol edeceğinizi bilmeniz gerekiyor. Listedeki giriş sayısının, n, bellek konumu N'de saklandığını varsayalım. Kayıt R1, döngünün yürütülme sayısını belirlemek için bir sayaç olarak kullanılır. Bu nedenle, N konumunun içeriği programın başlangıcında R1 kaydına yüklenir. Ardından, döngü gövdesi içinde, R1 Azaltımı komutu, döngü boyunca her seferinde R1'in içeriğini 1 azaltır. Azaltma işleminin sonucu sıfırdan büyük olduğu sürece döngünün yürütülmesi tekrarlanır.

Artık şube talimatlarını anlayabilmelisiniz. Bu tür komut, program sayacına yeni bir değer yükler. Sonuç olarak işlemci, şube talimatını sıralı adres sırasına göre takip eden konumdaki talimat yerine, şube hedefi adı verilen bu yeni adresteki talimatı alır ve yürütür. Şube talimatı koşullu veya koşulsuz olabilir. Koşulsuz dallanma komutu, herhangi bir koşuldan bağımsız olarak belirtilen adrese dallanma yapar. Koşullu bir dallanma talimatı, yalnızca belirli bir koşul karşılanırsa bir dallanmaya neden olur. Koşul karşılanmazsa, PC normal şekilde artırılır ve sıralı adres sırasındaki bir sonraki komut alınır ve yürütülür. Yukarıdaki örnekte, Şube 0 DÖNGÜ komutu (0'dan büyükse dal), R1 kaydındaki azaltılmış değer olan hemen önceki talimatın sonucu şundan büyükse, bir dalın DÖNGÜ konumuna gelmesine neden olan koşullu bir dallanma talimatıdır: sıfır. Bu, listede henüz R0'a eklenmemiş girişler olduğu sürece döngünün tekrarlanacağı anlamına gelir. Döngüden n'inci geçişin sonunda, Azaltma talimatı sıfır değeri

üretir ve bu nedenle dallanma gerçekleşmez. Bunun yerine Move komutu getirilir ve yürütülür. Nihai sonucu R0'dan SUM bellek konumuna taşır.

Bazı ISA'lar, bu tür talimatlara Atlamalar olarak atıfta bulunur. İşlemci, sonraki koşullu dal yönergeleri tarafından kullanılmak üzere çeşitli işlemlerin sonuçları hakkındaki bilgileri izler. Bu, genellikle durum kodu bayrakları olarak adlandırılan, gerekli bilgileri ayrı bitlere kaydederek gerçekleştirilir. Bu bayraklar genellikle durum kodu kaydı veya durum kaydı adı verilen özel bir işlemci kaydında gruplandırılır. Bireysel koşul kodu bayrakları, gerçekleştirilen işlemin sonucuna bağlı olarak 1'e ayarlanır veya 0'a sıfırlanır. Yaygın olarak kullanılan bayraklardan bazıları şunlardır: Sign, Zero, Overflow ve Carry. Arama ve geri dönüş komutları, alt programlarla birlikte kullanılır. Bir alt program, belirli bir hesaplama görevini gerçekleştiren bağımsız bir talimat dizisidir. Bir programın yürütülmesi sırasında, ana programın çeşitli noktalarında birçok kez işlevini yerine getirmesi için bir alt program çağrılabilir. Bir alt program her çağrıldığında, kendi komut dizisini yürütmeye başlamak için alt programın başına kadar bir dal yürütülür. Alt program yürütüldükten sonra, dönüş komutu aracılığıyla ana programa bir dallanma yapılır. Kesintiler ayrıca bir programın akışını değiştirebilir. Bir program kesintisi, harici veya dahili olarak oluşturulmuş bir talebin sonucu olarak program kontrolünün halihazırda çalışan bir programdan başka bir hizmet programına aktarılmasını ifade eder. Servis programı yürütüldükten sonra kontrol orijinal programa geri döner.

Kesme prosedürü, prensip olarak, üç varyasyon dışında bir alt program çağrısına oldukça benzerdir: (1) Kesme genellikle bir talimatın yürütülmesinden ayrı olarak dahili veya harici bir sinyal tarafından başlatılır (2) kesme servis programının adresi donanım tarafından veya kesme sinyalinden veya kesmeye neden olan talimattan gelen bazı bilgilerden belirlenir; ve (3) bir kesme prosedürü genellikle yalnızca program sayacını depolamak yerine CPU'nun durumunu tanımlamak için gerekli tüm bilgileri depolar. Bu nedenle, işlemci kesintiye uğradığında, dönüş adresi, kayıt içeriği ve İşlemci Durum Sözcüğü (PSW) adı verilen durum bilgisi dahil olmak üzere işlemcinin mevcut durumunu kaydeder ve ardından kesme işleyicisine veya kesme hizmet rutinine atlar. . Bunu tamamladıktan sonra ana programa geri döner. Kesmeler, Giriş / Çıkış konusunda bir sonraki ünitede ayrıntılı olarak ele alınmaktadır.

Giriş ve Çıkış komutları, yazmaçlar, bellek ve giriş/çıkış cihazları arasında bilgi aktarımı için kullanılır. Özel olarak G/Ç transferleri gerçekleştiren özel komutlar kullanmak veya G/Ç aktarımları yapmak için bellekle ilgili talimatların kendisini kullanmak mümkündür.

Belirli bir uygulamayı gerçekleştirmesi amaçlanan gömülü bir işlemci tasarladığınızı varsayalım, o zaman kesinlikle o belirli uygulamaya özel talimatlar getirmeniz gerekecektir. Genel amaçlı bir işlemci tasarlarken, yalnızca tüm genel talimat türlerini dahil etmeye bakarsınız. Özelleştirilmiş talimatlara örnek olarak, medya ve sinyal işlemeyle ilgili talimatlar (örneğin, veri seviyesi paralelliğinden yararlanmaya çalışan vektör tipi talimatlar) verilebilir; burada aynı toplama veya çıkarma işlemi farklı veriler üzerinde yapılacak ve ardındanbakmanız gerekebilir doygun aritmetik işlemlerde, çarpma ve biriktirme komutları.

Veri türleri ve boyutları, işlemci tarafından desteklenen çeşitli veri türlerini ve bunların uzunluklarını gösterir. Yaygın işlenen türleri – Karakter (8 bit), Yarım sözcük (16 bit), Sözcük (32 bit), Tek Duyarlıklı Kayan Nokta (1 Sözcük), Çift Duyarlı Kayan Nokta (2 Sözcük), Tamsayılar – ikinin tamamlayıcısı ikili sayılar, Karakterler genellikle ASCII'de, IEEE Standardı 754'ü izleyen kayan noktalı sayılar ve Paketlenmiş ve paketlenmemiş ondalık sayılar.

2.1 Adresleme Modları

Bir talimatın işlem alanı, gerçekleştirilecek işlemi belirtir. Bu işlem, hemen verilen veya bilgisayar kayıtlarında veya bellek sözcüklerinde depolanan bazı veriler üzerinde yürütülmelidir. Program yürütme sırasında işlenenlerin seçilme şekli, komutun adresleme moduna bağlıdır. Adresleme modu, işlenene fiilen başvurulmadan önce talimatın adres alanını yorumlamak veya değiştirmek için bir kural belirtir. Bilgisayarlar, aşağıdakilerden birini veya her ikisini barındırmak amacıyla adresleme modu tekniklerini kullanır:

1. Belleğe işaretçiler, döngü kontrolü için sayaçlar, verilerin indekslenmesi ve programın yeniden konumlandırılması gibi olanaklar sağlayarak kullanıcıya programlama çok yönlülüğü vermek.

2. Komutun adresleme alanındaki bit sayısını azaltmak.

Yüksek seviyeli bir dilde programlar yazdığınızda sabitler, yerel ve global değişkenler, işaretçiler ve diziler kullanırsınız. Üst düzey bir dil programını derleme diline çevirirken, derleyici, programın çalıştırılacağı bilgisayarın komut setinde sağlanan olanakları kullanarak bu yapıları uygulayabilmelidir. Bir komutta bir işlenenin konumunun belirtildiği farklı yollara adresleme kipleri denir. Değişkenler ve sabitler en basit veri türleridir ve hemen hemen her bilgisayar programında bulunur. Assembly dilinde, bir değişken, değerini tutmak için bir kayıt veya bir bellek konumu tahsis edilerek temsil edilir.

Kayıt modu — İşlenen, bir işlemci kaydının içeriğidir; kaydın adı (adresi) talimatta verilmiştir.

Mutlak mod — İşlenen bir bellek konumundadır; bu yerin adresi talimatta açıkça verilmiştir. Buna Doğrudan da denir. Adres ve veri sabitleri, Derhal modu kullanılarak derleme dilinde temsil edilebilir.

Anında mod — İşlenen, talimatta açıkça verilir. Örneğin, Move 200immediate, R0 komutu R0 kaydına 200 değerini yerleştirir. Açıkçası, Anında mod yalnızca bir kaynak işlenenin değerini belirtmek için kullanılır. Yaygın bir kural, bu değerin acil bir işlenen olarak kullanılacağını belirtmek için değerin önünde diyez işareti (#) kullanmaktır. Bu nedenle yukarıdaki talimatı Move #200, R0 şeklinde yazıyoruz. Sabit değerler, üst düzey dil programlarında sıklıkla kullanılır. Örneğin, A = B + 6 ifadesi 6 sabitini içerir. A ve B'nin daha önce değişken olarak bildirildiğini ve Mutlak modu kullanılarak erişilebileceğini varsayarsak, bu ifade aşağıdaki gibi derlenebilir:

```
Move B, R1
Add #6, R1
Move R1, A
```

Sabitler ayrıca bir sayacı artırmak, bazı bit modellerini test etmek vb. için Assembly dilinde de kullanılır.

Dolaylı mod — Takip eden adresleme modlarında, komut, işleneni veya adresini açıkça vermez. Bunun yerine, işlenenin bellek adresinin belirlenebileceği bilgileri sağlar. Bu adrese işlenenin etkin adresi (EA) diyoruz. Bu modda, işlenenin etkin adresi, adresi talimatta görünen bir yazmaç veya bellek konumunun içeriğidir. Dolaylı yönlendirmeyi, komutta verilen yazmacın adını veya bellek adresini parantez içine alarak belirtiriz. Örneğin, Ekle (R1), R0 talimatını ele alalım. Ekle komutunu yürütmek için işlemci, işlenenin etkin adresi olarak R1 kaydındaki değeri kullanır. Bu konumun içeriğini okumak için bellekten bir okuma işlemi talep eder. Okunan değer, işlemcinin R0 yazmacının içeriğine eklediği istenen işlenendir. Add (A), R0 komutunda belirtildiği gibi, bir bellek konumu aracılığıyla dolaylı adresleme de mümkündür. Bu durumda, işlemci önce bellek konumu A'nın içeriğini okur, ardından işleneni elde etmek için bu değeri bir adres olarak kullanarak ikinci bir okuma işlemi talep eder. Bir işlenenin adresini içeren kayıt veya bellek konumuna işaretçi denir. Yönlendirme ve işaretçilerin kullanımı programlamada önemli ve güçlü kavramlardır. Örnekteki A konumunun içeriğini değiştirmek, R0 kaydına eklemek için farklı işlenenler getirir.

İndeks modu — Öğreneceğiniz bir sonraki adresleme modu, işlenenlere erişim için farklı türde bir esneklik sağlar. Listeler ve dizilerle uğraşırken kullanışlıdır. Bu modda, işlenenin etkin

adresi, bir kaydın içeriğine sabit bir değer (yer değiştirme) eklenerek üretilir. Kullanılan kayıt, bu amaç için sağlanan özel bir kayıt olabilir veya işlemcideki genel amaçlı kayıtlardan herhangi biri olabilir. Her iki durumda da, bir dizin kaydı olarak anılır. İndeks kipini sembolik olarak X(Ri) olarak belirtiyoruz, burada X komutta yer alan sabit değeri ve Ri ilgili yazmacın adını gösteriyor. İşlenenin etkin adresi EA = X + [Ri] ile verilir. İndeks kaydının içeriği, etkin adresin oluşturulması sürecinde değiştirilmez. Bir montaj dili programında, X sabiti, açık bir sayı olarak veya sayısal bir değeri temsil eden sembolik bir ad olarak verilebilir. Talimat makine koduna çevrildiğinde, X sabiti talimatın bir parçası olarak verilir ve genellikle bilgisayarın kelime uzunluğundan daha az bit ile temsil edilir. X işaretli bir tamsayı olduğundan, yazmacın içeriğine eklenmeden önce yazmaç uzunluğuna kadar işaretle genişletilmelidir.

Göreceli mod — Yukarıdaki tartışma, genel amaçlı işlemci kayıtlarını kullanan İndeks modunu tanımladı. Genel amaçlı bir kayıt yerine program sayacı PC kullanılırsa, bu modun kullanışlı bir versiyonu elde edilir. Ardından, X (PC), program sayacı tarafından halihazırda işaret edilen konumdan X bayt uzaktaki bir bellek konumunu adreslemek için kullanılabilir. Adreslenen konum, her zaman bir programdaki geçerli yürütme noktasını tanımlayan program sayacına 'göreli' olarak tanımlandığından, Göreceli mod adı bu adresleme türüyle ilişkilendirilir. Bu durumda etkin adres, genel amaçlı yazmaç Ri yerine program sayacı kullanılarak İndeks modu tarafından belirlenir. Bu adresleme modu genellikle kontrol akış komutlarıyla birlikte kullanılır.

Ancak bu mod, veri işlenenlerine erişmek için kullanılabilir. Ancak en yaygın kullanımı, dal yönergelerinde hedef adresi belirtmektir. Daha önce tartıştığımız Şube 0 DÖNGÜ gibi bir komut, dal koşulu karşılanırsa programın yürütülmesinin DÖNGÜ adıyla tanımlanan dal hedef konumuna gitmesine neden olur. Bu konum, program sayacının geçerli değerinden bir sapma olarak belirtilerek hesaplanabilir. Dallanma hedefi, dallanma komutundan önce veya sonra olabileceğinden, ofset işaretli bir sayı olarak verilir. Bir talimatın yürütülmesi sırasında işlemcinin PC'yi bir sonraki talimatı işaret edecek şekilde artırdığını hatırlayın. Çoğu bilgisayar, Göreceli modda etkin adresi hesaplamak için bu güncellenmiş değeri kullanır.

Aşağıda açıklanan iki mod, bellekteki ardışık konumlardaki veri öğelerine erişmek için kullanışlıdır.

Otomatik artırma modu — İşlenenin etkin adresi, talimatta belirtilen bir kaydın içeriğidir. İşlenene eriştikten sonra, bu kaydın içeriği listedeki bir sonraki öğeyi işaret edecek şekilde otomatik olarak artırılır. Kayıt içeriğinin etkin adres olarak kullanıldığını göstermek için belirtilen kaydı parantez içine alarak ve ardından işlenene erişildikten sonra bu içeriklerin

artırılacağını belirtmek için bir artı işareti koyarak Otomatik Arttırma modunu belirtiriz. Böylece Otomatik Arttırma modu (Ri)+ olarak yazılır.

Otomatik azaltma modu — Otomatik artırma moduna eşlik eden başka bir kullanışlı mod, bir listenin öğelerine ters sırada erişir. Otomatik azaltma modunda, komutta belirtilen bir kaydın içeriği önce otomatik olarak azaltılır ve ardından işlenenin etkin adresi olarak kullanılır. Otomatik azaltma kipini, belirtilen kaydı parantez içine alarak, önünde bir eksi işareti koyarak, kayıt içeriğinin etkili adres olarak kullanılmadan önce azaltılacağını belirtmek için belirtiriz. Böylece - (Ri) yazarız. Bu modda, işlenenlere azalan adres sırasında erişilir. Adresin neden Otomatik Azaltma modunda kullanılmadan önce azaltıldığını ve Otomatik Arttırma modunda kullanıldıktan sonra artırıldığını merak edebilirsiniz. Bunun ana nedeni, bu iki modun bir yığını gerçekleştirmek için birlikte kullanılabilmesidir.

2.2.Talimat Formatları

Önceki bölümlerde, işlemcinin farklı türde komutları çalıştırabildiğini ve işlenenleri belirlemenin farklı yolları olduğunu göstermiştik. Tüm bunlara karar verildikten sonra, bu bilgilerin işlemciye bir talimat formatı şeklinde sunulması gerekir. Komuttaki bit sayısı, alan adı verilen gruplara ayrılır. Talimat biçimlerinde bulunan en yaygın alanlar şunlardır:

1. Gerçekleştirilecek işlemi belirten bir işlem kodu alanı. Bit sayısı, gerçekleştirilebilecek işlem sayısını gösterecektir.

2. Bir bellek adresini veya bir işlemci kaydını belirten bir adres alanı. Bit sayısı, belleğin boyutuna veya kayıt sayısına bağlıdır.

3. İşlenenin veya etkin adresin belirlenme şeklini belirten bir mod alanı. Bu, işlemci tarafından desteklenen adresleme modlarının sayısına bağlıdır.

Adres alanlarının sayısı, kullanılan ISA türüne bağlı olarak üç, iki veya bir olabilir. Ayrıca, desteklenen işlenenlerin sayısına ve çeşitli alanların boyutuna bağlı olarak komutların uzunluğunun değişeceğini gözlemleyin. Bazı işlemciler tüm yönergeleri tek boyutlu bir biçime sığdırırken, diğerleri farklı boyutlardaki biçimleri kullanır. Buna göre, sabit bir formatınız veya değişken bir formatınız var.

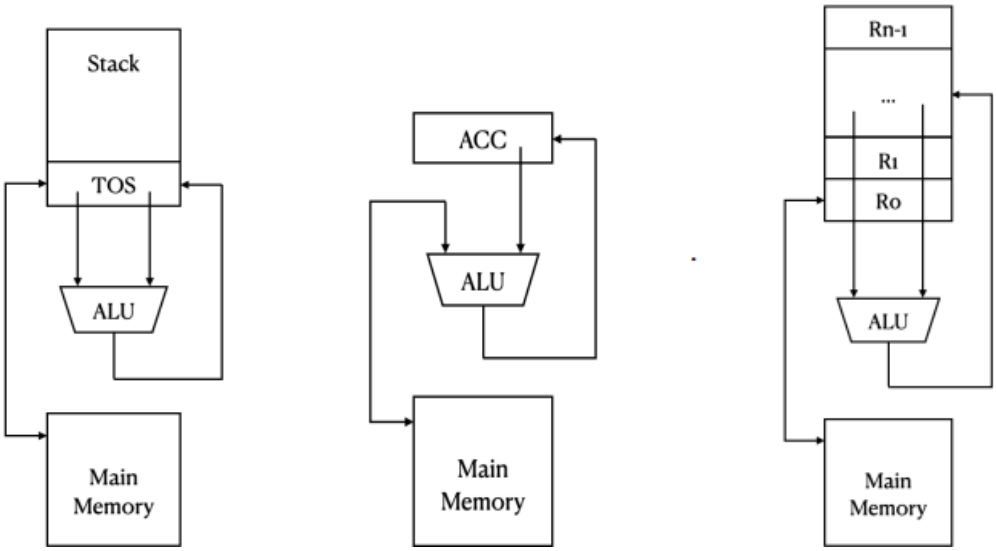
Bellek adreslerini yorumlama – temel olarak bellek adreslerinin iki tür yorumu vardır – Büyük endian düzenlemesi ve küçük endian düzenlemesi. Anılar normalde bayt olarak düzenlenir ve bir hafıza konumunun benzersiz adresi 8 bitlik bilgi depolayabilir. Ancak işlemcinin word uzunluğuna baktığınız zaman işlemcinin word uzunluğu birden fazla bayt olabilir. 32 bitlik bir

işlemciye baktığınızı varsayalım, dört bayttan oluşuyor. Bu dört bayt, dört bellek konumuna yayılır. Bir kelimenin adresini belirttiğinizde, kelimenin adresini nasıl belirleyeceksiniz – kelimenin adresi olarak en önemli baytın adresini mi belirteceksiniz (büyük uç) yoksa en az önemli baytın adresini mi belirleyeceksiniz (küçük son) kelimenin adresi olarak. Bu, büyük bir endian düzenlemesi ile küçük bir endian düzenlemesi arasında ayrım yapar. IBM, Motorola, HP büyük endian düzenlemesini ve Intel küçük endian düzenlemesini takip eder. Ayrıca bir veri farklı bellek konumlarına yayıldığında ve kelime sınırına uyan bir kelimeye ulaşmaya çalıştığınızda bir hizalama var deriz. Bir kelime sınırından başlamayan kelimelere erişmeye çalışırsanız, yine de erişebilirsiniz, ancak bunlar hizalanmamıştır. Hizalanmamış verilere erişim desteği olup olmadığı bir tasarım sorunudur. Yanlış hizalanmış verilere erişmenize izin verilse bile, verilere erişmek için normalde daha fazla sayıda bellek döngüsü gerekir.

Son olarak derleyicilerin rolüne bakıldığında, komut seti mimarisini tanımlarken derleyicinin oynayacağı birçok rol vardır. İnsanların derleyiciler ve mimarilerin birbirinden bağımsız olacağını düşündüğü günler geride kaldı. Yalnızca derleyici, işlemcinin iç mimarisini bildiğinde, optimize edilmiş kod üretebilecektir. Bu nedenle, mimari kendisini derleyiciye maruz bırakmak zorunda kalacak ve derleyici de açığa çıkan donanımı kullanmak zorunda kalacak. ISA derleyici dostu olmalıdır. ISA'nın derleyiciye yardımcı olabileceği temel yollar, düzenlilik, ortogonallik ve farklı seçenekleri tartma yeteneğidir.

2.3 80×86 ve MIPS'ye ISA Özellikleri

1.ISA Sınıfı — Günümüzde neredeyse tüm ISA'lar, işlenenlerin kayıtlar veya bellek konumları olduğu genel amaçlı kayıt mimarileri olarak sınıflandırılmaktadır. 80×86'da 16 genel amaçlı yazmaç ve 16 kayan noktalı veri tutabilen, MIPS'de ise 32 genel amaçlı ve 32 kayan noktalı yazmaç bulunur. Bu sınıfın iki popüler versiyonu, birçok talimatın bir parçası olarak belleğe erişebilen 80×86 gibi kayıt-bellek ISA'ları ve belleğe yalnızca yükleme veya saklama yönergeleriyle erişebilen MIPS gibi yük depolama ISA'larıdır. Tüm yeni ISA'lar yük deposudur.



Şekil 2.3.1 Stack- Acc- Register[3]

2.Bellek adresleme — 80×86 ve MIPS dahil olmak üzere neredeyse tüm masaüstü ve sunucu bilgisayarlar, bellek işlenenlerine erişmek için bayt adresleme kullanır. MIPS gibi bazı mimariler, nesnelerin hizalanmasını gerektirir. A bayt adresinde s bayt boyutunda bir nesneye erişim, A mod s = 0 ise hizalanır. 80×86 hizalama gerektirmez, ancak işlenenler hizalanırsa erişimler genellikle daha hızlıdır.

3.Adresleme modları — Kayıtları ve sabit işlenenleri belirtmeye ek olarak, adresleme modları bir bellek nesnesinin adresini belirtir. MIPS adresleme modları, bellek adresini oluşturmak için bir kayda sabit bir ofsetin eklendiği Kayıt, Anında (sabitler için) ve Yer Değiştirme'dir. 80×86, bu üç artı üç yer değiştirme varyasyonunu destekler: kayıt yok (mutlak), iki kayıt (yer değiştirme ile indekslenmiş tabanlı), bir kaydın işlenenin bayt cinsinden boyutuyla çarpıldığı iki kayıt (ölçekli indeks ve yer değiştirmeye dayalı)). Daha çok son üçe benzer, eksi yer değiştirme alanı vardır: dolaylı, dizinlenmiş ve ölçeklenmiş dizine dayalı kayıt.

Addressing Mode	Example	Meaning
Register	`Add R4, R3`	`R4 ← R4 + R3`
Immediate	`Add R4, #3`	`R4 ← R4 + 3`
Displacement	`Add R4, 100(R1)`	`R4 ← R4 + M[100+R1]`
Register Indirect	`Add R4, (R1)`	`R4 ← R4 + M[R1]`
Indexed / Base	`Add R3, (R1 + R2)`	`R3 ← R3 + M[R1 + R2]`
Direct or Absolute	`Add R1, (1001)`	`R1 ← R1 + M[1001]`
Memory Indirect	`Add R1, @R3`	`R1 ← R1 + M[M[R3]]`
Auto-Increment	`Add R1, (R2)+`	`R1 ← R1+M[R2]; R2 ← R2+d`
Auto-Decrement	`Add R1, -(R2)`	`R2 ← R2-d; R1 ← R1+M[R2]`
Scaled	`Add R1, 100(R2)[R3]`	`R1 ← R1 + M[100+R2+R3×d]`

Şekil 2.3.2

4.İşlenenlerin türleri ve boyutları — Çoğu ISA gibi, MIPS ve 80×86, 8 bit (ASCII karakter), 16 bit (Unicode karakter veya yarım sözcük), 32 bit (tam sayı veya sözcük), 64 işlenen boyutlarını destekler -bit (çift sözcük veya uzun tamsayı) ve 32 bit (tek kesinlik) ve 64 bit (çift kesinlik) olarak IEEE 754 kayan nokta. 80×86 ayrıca 80 bit kayan noktayı (genişletilmiş çift kesinlik) destekler.

5.İşlemler — Genel işlem kategorileri veri aktarımı, aritmetik mantıksal, kontrol ve kayan noktadır. MIPS, basit ve sıralanması kolay bir komut seti mimarisidir ve 2006'da kullanılan RISC mimarilerini temsil eder. 80×86, çok daha zengin ve daha geniş bir işlem grubuna sahiptir.

6.Kontrol akışı talimatları — 80×86 ve MIPS dahil olmak üzere hemen hemen tüm ISA'lar koşullu dalları, koşulsuz atlamaları, prosedür çağrılarını ve dönüşleri destekler. Her ikisi de, şube adresinin bilgisayara eklenen bir adres alanı tarafından belirtildiği, bilgisayara bağlı adresleme kullanır. Bazı küçük farklılıklar vardır.MIPS koşullu dalları (BE, BNE, vb.) kayıtların içeriğini test ederken, 80×86 dalları(JE, JNE, vb.) aritmetik/mantık işlemlerinin yan etkileri olarak ayarlanan koşul kodu bitlerini test eder. . MIPS prosedür çağrısı (JAL), dönüş adresini bir kayda yerleştirirken, 80×86 çağrısı (CALLF) dönüş adresini bellekteki bir yığına yerleştirir.

Bir ISA'yı kodlama — Kodlamada iki temel seçenek vardır: sabit uzunluk ve değişken uzunluk. Tüm MIPS komutları 32 bit uzunluğundadır, bu da yönerge kod çözmeyi basitleştirir (aşağıda gösterilmiştir). 80×86 kodlama, 1 ila 18 bayt arasında değişen değişken uzunluktadır. Değişken uzunluklu komutlar, sabit uzunluklu komutlardan daha az yer kaplayabilir, bu nedenle 80×86 için derlenen bir program genellikle MIPS için derlenen aynı programdan daha küçüktür. Yukarıda belirtilen seçimlerin, talimatların bir ikili gösterime nasıl kodlanacağını etkileyeceğini unutmayın. Örneğin, kayıt alanı ve adresleme modu alanı tek bir komutta birçok kez görünebileceğinden, kayıt sayısı ve adresleme modu sayısının her ikisi de talimatların boyutu üzerinde önemli bir etkiye sahiptir.

Özetle ISA'ların taksonomisine ve ISA'yı tasarlarken karar verilmesi gereken çeşitli özelliklere baktık. Ayrıca örnek ISA'lara, MIPS ISA'ya ve 80×86 ISA'ya baktık. [4]

2.3.1 Stack-Accumulator ve GPR Avantaj & Dezavantaj

Stack:

Avantajları: İfade değerlendirmesinin Basit Modeli Kısa talimatlar. Dezavantajlar: Bir yığına rastgele erişilemez Bu, verimli kod üretmeyi zorlaştırır. Yığının kendisine her işlemde erişilir ve bir darboğaz haline gelir.

Accumulator:

Avantajları: Kısa talimatlar. Dezavantajlar: Akümülatör yalnızca geçici bir depolamadır, bu nedenle bellek trafiği bu yaklaşım için en yüksektir.

GPR:

Avantajları: Kod oluşturmayı kolaylaştırır. Veriler, kayıtlarda uzun süre saklanabilir. Dezavantajlar: Tüm işlenenler, daha uzun komutlara yol açacak şekilde adlandırılmalıdır. [5]

Stack	Accumulator	GPR
PUSH A	LOAD A	LOAD R1,A
PUSH B	ADD B	ADD R1,B
ADD	STORE C	STORE R1,C
POP C	-	-

Şekil 2.3.1.1

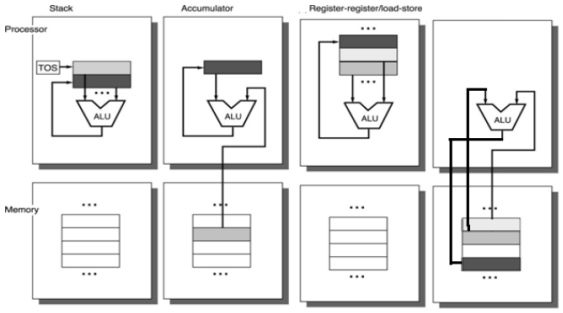
2.3.2 ISA Tasarım Konseptleri

1. Veri depolama
2. Bellek Adresleme Modları

3. Komut Setindeki İşlemler
4. Talimat Formatları
5. Komut Setini Kodlama

Stack	Accumulator	Register (Load-Store)	Memory-Memory
`Push A`	`Load A`	`Load R1, A`	`Add C, A, B`
`Push B`	`Add B`	`Load R2, B`	
`Add`	`Store C`	`Add R3, R2, R1`	
`Pop C`		`Store R3, C`	

Visually, we can represent them as follows:



Şekil 2.3.2.1 [6]

2.4 RISC ve CISC Komut Setleri

Farklı bilgisayarları birbirinden ayıran en önemli özelliklerden biri talimatlarının doğasıdır. Modern bilgisayarlar için komut setlerinin tasarımında temelde farklı iki yaklaşım vardır. Popüler bir yaklaşım, her talimatın bellekte tam olarak bir kelimeyi kaplaması ve bir talimat tarafından belirtilen belirli bir aritmetik veya mantık işlemini yürütmek için gereken tüm işlenenlerin zaten işlemci kayıtlarında olması durumunda daha yüksek performansın elde edilebileceği öncülüne dayanır. Bu yaklaşım, bir programın toplam yürütme süresini azaltmak ve bir programın toplam yürütme süresini azaltmak için bir talimat dizisini işlemek için gereken çeşitli işlemlerin "ardışık" tarzda gerçekleştirildiği bir işlem birimi uygulamasına elverişlidir. Her komutun tek bir kelimeye sığması zorunluluğu, bir bilgisayarın komut setine dahil edilebilecek farklı türde komutların karmaşıklığını ve sayısını azaltır. Bu tür bilgisayarlara İndirgenmiş Komut Kümesi Bilgisayarları (RISC) denir. RISC yaklaşımına bir alternatif, daha karmaşık komutların kullanılmasıdır. Birden fazla bellek kelimesini kapsayabilen ve daha karmaşık işlemleri belirtebilen. Bu yaklaşım, 1970'lerde RISC yaklaşımının tanıtılmasından önce yaygındı. Karmaşık yönergelerin kullanımı başlangıçta belirli bir etiketle tanımlanmamış olsa da bu fikre dayanan bilgisayarlar daha sonra Karmaşık Komut Kümesi Bilgisayarları (CISC) olarak adlandırıldı. RISC ve CISC iki farklı komut seti stilidir. Daha basit ve anlaşılması

daha kolay olduğu için önce RISC'yi tanıttık. Her iki stilin bazı temel özelliklerine baktıktan sonra, temel özelliklerini özetlemeliyiz. RISC stili aşağıdakilerle karakterize edilir:

- Basit adresleme modları
- Tek bir kelimeye sığan tüm talimatlar
- Basit adresleme modlarının bir sonucu olarak komut setinde daha az komut
- Yalnızca işlemcideki işlenenler üzerinde gerçekleştirilebilen aritmetik ve mantık işlemleri
- Bir bellek konumundan diğerine doğrudan aktarıma izin vermeyen yükleme/depolama mimarisi; bu tür aktarımlar bir işlemci kaydı aracılığıyla yapılmalıdır
- Ardışık düzen oluşturma gibi teknikleri kullanarak işleme birimi tarafından hızlı yürütmeye elverişli basit talimatlar
- Karmaşık görevleri gerçekleştirmek için daha fazla ama daha basit yönergeler gerektiğinden, boyut olarak daha büyük olma eğiliminde olan programlar

CISC stili aşağıdakilerle karakterize edilir:

- Daha karmaşık adresleme modları
- Bir talimatın birden fazla kelimeyi kapsayabileceği daha karmaşık talimatlar
- Karmaşık görevleri uygulayan birçok talimat
- İşlemci kayıtlarındaki işlenenlerin yanı sıra bellek işlenenleri üzerinde gerçekleştirilebilen aritmetik ve mantık işlemleri
- Tek bir Move talimatı kullanarak bir hafıza konumundan diğerine transferler
- Karmaşık görevleri gerçekleştirmek için daha az sayıda ancak daha karmaşık talimat gerektiğinden, boyut olarak daha küçük olma eğiliminde olan programlar

1970'lerden önce tüm bilgisayarlar CISC tipindeydi. Önemli bir amaç, donanımı oldukça karmaşık görevleri yerine getirme yeteneğine sahip hale getirerek, yani karmaşıklığı yazılım düzeyinden donanım düzeyine taşıyarak yazılımın geliştirilmesini basitleştirmekti. Bu, bilgisayar belleğinin daha küçük ve sağlanması daha pahalı olduğu zamanlarda önemli olan programları daha basit ve daha kısa hale getirmeye elverişlidir. Günümüzde bellek ucuzdur ve çoğu bilgisayarda büyük miktarlarda bulunur. RISC tarzı tasarımlar, donanımı çok basit hale getirerek çok yüksek performans elde etme girişimi olarak ortaya çıktı, böylece Bölüm 6'da tartışılacağı gibi komutlar ardışık düzen tarzında çok hızlı bir şekilde yürütülebilir. Bu, karmaşıklığın donanım düzeyinden donanım düzeyine taşınmasıyla sonuçlanır. Basit

talimatlardan oluşan kodu optimize etmek için karmaşık derleyiciler geliştirildi. Hafıza kapasiteleri arttıkça kodun boyutu daha az önemli hale geldi. RISC ve CISC stilleri, önemli ölçüde farklı iki yaklaşımı tanımlıyor gibi görünse de günümüzün işlemcileri genellikle bu yaklaşımlar arasında bir uzlaşma gibi görünen şeyler sergiliyor. Örneğin, yürütülen talimatların sayısını azaltmak için, bu yeni talimatların yürütülmesi hızlı olduğu sürece, bazı RISC olmayan talimatları bir RISC işlemcisine eklemek çekicidir. [7]

	CISC	RISC
Name	Complex Instruction Set Computer	Reduced Instruction Set Computer
Example	x86-32, IA32	MIPS, ARM
Advantages	Smaller program size, especially if memory is premium	Small and simple instruction set, easier to build/optimise hardware
Disadvantages	Complex processor implementation, difficult hardware optimisation	Burden on software to compile efficiently

Şekil 2.4.1

3.INTEL i9

Hem Intel Core i7 hem de Intel Core i9, uzman kullanıcılar için konumlandırılmış üst düzey işlemcilerdir. Bununla birlikte, ikisi arasında, ikincisi, daha iyi işleme yeteneklerinin yanı sıra gelişmiş Turbo Boost ve bazı sürümlerde Hyper-Threading özelliklerine sahip meraklı düzeydeki kullanıcılar için tasarlanmış ve pazarlanmış daha da güçlü bir CPU olarak konumlandırılmıştır.

Core i9 işlemcinin Core i7 ve Core i5'e göre avantajlarından biri, daha hızlı ve çok daha güçlü olmasıdır. Geçerli temel model, 10 çekirdekli bir yapılandırmayla başlar. Daha fazla işlem çekirdeği temelde daha iyi genel performans anlamına gelir. Her çekirdek, diğerlerinin yanı sıra kapsamlı video düzenleme, üç boyutlu işleme ve yoğun oyun oynama dahil olmak üzere işlem ağırlıklı görevlerin yapılmasına izin vermek için daha fazla bilgi işlem gücü ekler.

Intel, bu işlemciyi, performans için daha yüksek bir prim ödemeye istekli meraklı düzeydeki kullanıcılara yönelik olarak pazarlamaktadır. Bir bakıma, bu segment aynı zamanda bilgisayar donanımının kullanılabilirliği ve güvenilirliğinin bir kanıtıdır.

Core i9'un bir başka avantajı da, yeni nesil uygulamalar düşünülerek tasarlanmış olmasıdır. Şu anda yüksek tanımlı oyun, yüksek çözünürlüklü ve çok görevli video ve ses düzenleme ve yoğun kaynak kullanan uygulamaların kapsamlı ve eşzamanlı kullanımı yeteneğine sahiptir. Ek olarak, teorik olarak benzer veya daha yüksek kaynak gereksinimleri olan yeni bilgisayar uygulamalarını barındırabilecektir.

Meraklılar için bilgi işlem, genel bilgi işlemden farklıdır çünkü meraklı düzeydeki kullanıcılar, en gelişmiş donanım bileşenlerine ve özelleştirme fırsatlarına dikkat eder. Ayrıca, bu kullanıcılar, profesyonel veya rekabetçi oyuncular, animatörler ve grafik sanatçılar gibi profesyonel ticari sanatçılar ve veri analistleri ve programcılar gibi bilgisayar bilimcilerinden oluşan alt grupları içerir.

Bu işlemcinin meraklı seviyedeki kullanıcılara yönelik olduğunu unutmayın. Ortalama kullanıcılar ve önemli sayıda uzman kullanıcı, iyi saat hızlarına sahip ve iyi RAM ile eşleştirilmiş bir Intel Core i5 veya Intel Core i7 kullanmaktan yine de faydalanacaktır. Kullanım senaryosu gereksinimleri günlük ofis üretkenliği ve içerik tüketimi uygulamaları etrafında dönenler için Core i9 kesinlikle fazlalık.

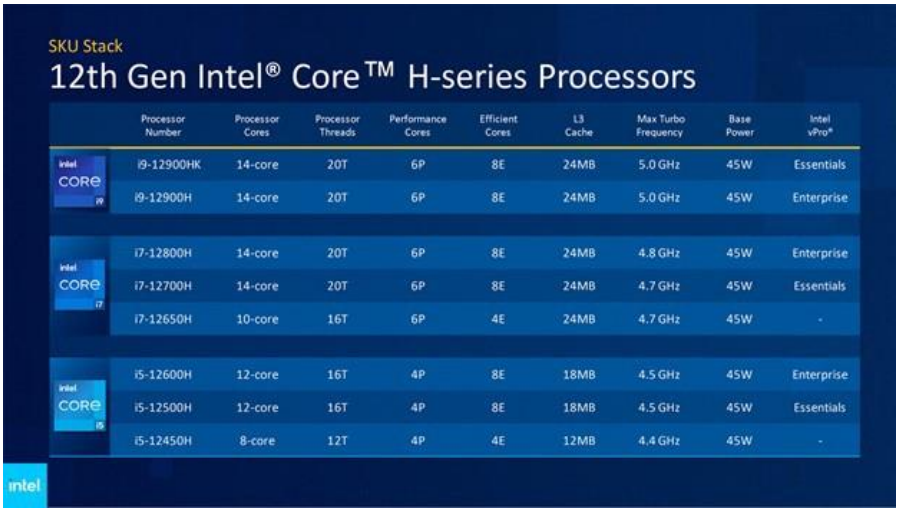
Bilgi işlem yetenekleri ne kadar iyi ve saat hızı ne kadar yüksekse, enerji gereksinimi de o kadar yüksek olur. Core i9, Intel Core işlemci ailesi arasında en fazla gücü tüketir. Ayrıca, en fazla ısıyı üretir. Daha uzun pil saatlerinin kritik olduğu taşınabilir cihazlar ve durumlar için esasen uygun değildir. Ayrıca, aşırı ısınmayı önlemek için verimli bir soğutma sistemi altında çalıştırılmalıdır.

Core i5 veya Core i7'den Core i9'a yükseltme yapmak kolay değildir. Dolayısıyla, bu işlemcinin bir başka önemli dezavantajıdır. Bir kullanıcının X2999 yonga setine sahip yeni bir anakart satın alması ve kullanması gerekir. Bunun, diğer Core i5 ve Core i5 işlemcileri destekleyen anakartlarla aynı olmadığını unutmayın. Ayrıca, daha güçlü bir işlemciye yükseltmek, mevcut soğutma sisteminin değerlendirilmesini ve yükseltilmesini gerektirecektir.

Bazı meraklı düzeydeki kullanıcılar bile bu işlemciyi kullanışsız bulabilir. Core i9'un bir başka sınırlaması, i7'den iki ila üç kat daha pahalı ve i5'ten yaklaşık yüzde 80 ila 100'den fazla daha pahalı olmasıdır. Uyumlu anakart da pahalıdır. Daha iyi bir soğutma sisteminin eklenmesi başka bir maliyet anlamına da gelir. Bu işlemin çoğu ortalama ve bazı üst düzey kullanıcılar için aşırıya kaçacağını unutmayın. [8]

İşlemci, Alder Lake amiral gemisi 12900K'nın daha yüksek ikili ve daha iyi saat hızlarına sahip yenilenmiş bir versiyonudur. Tom's Hardware'e göre buradaki öne çıkan özelliklerden biri, 12900KS'nin erişilebileceği muazzam hız - daha önce sıvı nitrojen soğutma ile akıllara durgunluk veren 7,45 GHz'de çalışıyordu. Bu CPU'nun kutudan çıktığı haliyle en yüksek artırma hızının 5,5 GHz olduğunu unutmayın, dolayısıyla bu önemli bir gelişmedir; ancak, çok önemli bir uyarı var. Spesifik olarak, bunu başaran MSI uzmanı hız aşırtma uzmanı TSAIK, bunu yalnızca bir (performans) çekirdeği devreye sokarak yaptı; bu, rakam harika olsa da, gerçek dünya performansı için her zamanki egzotik soğutma rekorunuzdan daha az pratik öneme sahip olduğu anlamına gelir.Başka bir hız aşırtma uzmanı olan Tom Hardware uzmanı - 'Splave' olarak bilinir - Cinebench, Geekbench ve HWBot kıyaslamalarında 12900KS ile çok sayıda dünya rekoru kırmayı başardı. Splave, bu farklı kriterlerde 16 dünya rekoru kırdı veya küresel olarak birinci sırayı aldı. Dikkate değer bir başarı, Geekbench 4 çok çekirdeklide 100.675'lik bir puandı; bu, 100.000 sınırını ilk kez kırarak bir önceki yüksek seviye olan 97.800'ü geçti. (Vanilya Core i9–12900K tarafından ayarlanmıştır). [9]

Core i9 ailesine talep üzerine büyük hız patlamaları için boşluk sağlar. Her iki model de çekirdekleri beslemeye yardımcı olmak için 24 MB Akıllı Önbelleğe sahiptir. Ve hız aşırtmacılar, not edin: Core i9–12900HK, Extreme Tuning Utility ile performansı en üst düzeye çıkarmak için kullanabileceğiniz kilidi açılmış bir çarpan sunar. [11]



Şekil 3.1

4.AMD RYZEN

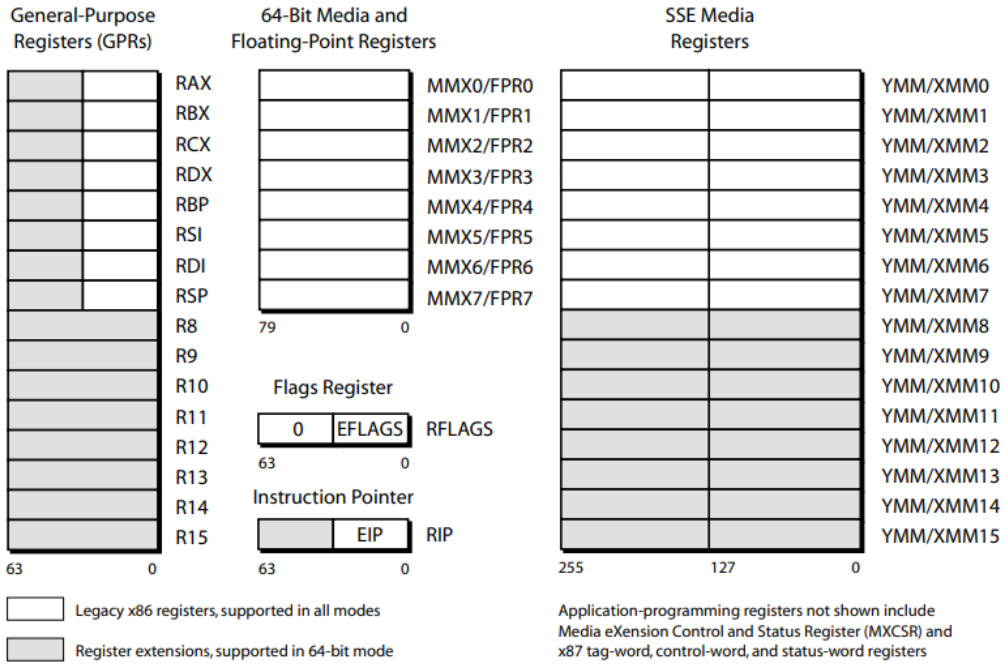
AMD64 mimarisi, endüstri standardı (eski) x86 mimarisinin basit ama güçlü bir 64-bit, geriye dönük uyumlu uzantısıdır. 64 bit adresleme ekler ve kayıt kaynaklarını genişletir. Yeniden derlenmiş 64 bit programlar için daha yüksek performansı desteklerken, eski 16 bit ve 32 bit

uygulamaları ve işletim sistemlerini değiştirmeden veya yeniden derlemeden destekler. Bu, mimari yeni işlemcilerin hem mevcut yazılımların geniş gövdesi hem de daha yüksek performanslı uygulamalar için gereken 64-bit yazılımlar için sorunsuz, yüksek performanslı destek sağlayabildiği temel 64 bit x86 mimarisine olan ihtiyaç, yüksek performanslı sunucular, veri tabanı yönetim sistemleri ve CAD gibi büyük miktarlarda sanal ve fiziksel belleği ele alan uygulamalar tarafından yönlendirilir. Bu uygulamalar hem 64 bit adreslerden hem de artan sayıda kayıttan yararlanır. Eski x86 mimarisinde bulunan az sayıda kayıt, yoğun hesaplama gerektiren uygulamalarda performansı sınırlar. AMD64 mimarisi şu özellikleri içerir:

1) Uzantıları Kaydet (bkz. Şekil 1-):
    a) 8 ek genel amaçlı kayıt (GPR'ler).
    b) 16 GPR'nin tümü 64 bit genişliğindedir.
    c) 8 ek YMM/XMM kaydı.
    d) Tüm GPR'ler için tek tip bayt kaydı adreslemesi.
    e) Bir komut öneki (REX) genişletilmiş kayıtlara erişir.
2) Uzun Mod (bkz. Tablo 1-1 sayfa 2):
    a) 64 bit'e kadar sanal adres.
    b) 64 bit talimat işaretçisi (RIP).
    c) Komut işaretçisine göre veri adresleme modu.
    d) Düz adres alanı. [12]

Table 1-1. Operating Modes

Operating Mode		Operating System Required	Application Recompile Required	Defaults		Register Extensions	Typical
				Address Size (bits)	Operand Size (bits)		GPR Width (bits)
Long Mode	64-Bit Mode	64-bit OS	yes	64	32	yes	64
	Compatibility Mode		no	32		no	32
				16	16		16
Legacy Mode	Protected Mode	Legacy 32-bit OS	no	32	32	no	32
				16	16		
	Virtual-8086 Mode			16	16		16
	Real Mode	Legacy 16-bit OS					



Şekil 4.1

AMD Ryzen, şirketin tüm zamanların en popüler işlemci serisidir. Birkaç yıldır piyasada oldukça düşük anahtarlı bir koşunun ardından AMD, Ryzen serisi CPU'larla yepyeni Zen mimarisini piyasaya sürdü. 2017'de piyasaya sürüldü ve hızla Intel'in Core serisine sağlam bir alternatif haline geldi. Ryzen serisi artık dört nesil geride. [13]

İlk olarak 2017'de piyasaya sürülen Ryzen CPU'lar ve APU'lar, AMD'nin son derece gelişmiş 'Zen' mikro mimarisi, 8. nesil AMD işlemci teknolojisi üzerine kuruludur ve şirketin yaklaşık beş yıl önce FX/A6 serisinden bu yana ilk büyük işlemci tanıtımını yapar.

Ryzen işlemciler, AMD'nin artık sadece fiyat açısından değil, işlemci performansı açısından da Intel ile rekabet edebileceğini söyleyen teknoloji analistleri ve eleştirmenlerinden hızlı bir şekilde beğeni topladı. Ve Lenovo, Ryzen işlemcilerini en gelişmiş masaüstü ve dizüstü bilgisayarlarından bazılarında hızlı bir şekilde sunmaya başladı ve daha fazla Ryzen donanımlı model bekleniyor. AMD'nin ilk ürün lansmanı, Ryzen işlemcinin birden çok farklı sürümünü içeriyordu:

-Ryzen: Masaüstü ve dizüstü bilgisayarlar için birden çok sürüm mevcuttur

-Ryzen PRO: Ek güvenlik ve kurumsal düzeyde yönetim özelliklerine sahiptir (kurumsal ve ticari kullanım için idealdir)

-Ryzen Mobile: Ryzen CPU'yu AMD'nin Radeon Vega grafikleriyle birleştirin

Tümleşik grafik yetenekleri ve nispeten düşük maliyetleriyle uzun süredir övülen AMD işlemciler, yakın zamanda Intel'in en yeni yongalarına karşı kıyaslama performans testlerinde başarısız olmuştu. Ancak AMD, Ryzen ile daha fazla çekirdek ekledi (temel Ryzen yongaları bile dört çekirdeklidir), orta sınıf tekliflerini altı çekirdeğe/12 iş parçacığına kadar genişletti (yüksek performanslı modellerde 8/16 veya daha fazla iş parçacığı olabilir) ve tanıtıldı Eşzamanlı Çoklu İş Parçacığı (SMT) teknolojisi, her çekirdekte daha fazla işlem yapılmasını sağlar. Bir işlemci 'çekirdeği' esasen kendi başına ayrı bir CPU'dur. Dolayısıyla, modern, dört çekirdekli veya sekiz çekirdekli bir işlemci, hesaplamalı ağır kaldırmayı yapmak için aslında dört veya sekiz işlemci yongasına sahiptir. Her çekirdek, hesaplamak, grafik oluşturmak, yazılım programlarını çalıştırmak ve diğer her şey için gerekli olan farklı komut ve yanıt kümelerini taşıyan bir veya iki 'iş parçacığına' sahiptir. Ve AMD'nin yeni Eşzamanlı Çoklu İş Parçacığı (Intel'in Hyper-Threading'ine benzer) her iş parçacığının daha verimli kullanılmasını sağlayarak işlemci üretkenliğini daha da artırır. Sektör uzmanları, AMD'nin Ryzen hakkındaki iddialarını kısa sürede doğrulayarak, yeni yongaların birçok performans kategorisinde Intel dahil diğer üreticilerin sunduğu yongalara eşit veya daha iyi olduğunu söylediler. Bu rakipler aynı fikirde olmayabilir veya rakip test sonuçları sunabilir (bu çok rekabetçi bir iştir), ancak AMD'nin devam eden fiyat avantajı göz önüne alındığında, birçok popüler teknoloji analisti, Ryzen'in pazara girişini açıklamak için 'yıkıcı' ve 'ezber bozan' gibi terimler kullandı. [14]

5.AMD VS INTEL

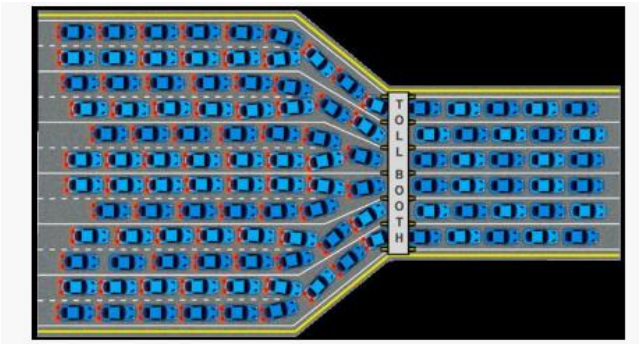


Fig 5: Intel' s approach – 12 cores with 6 shared cores

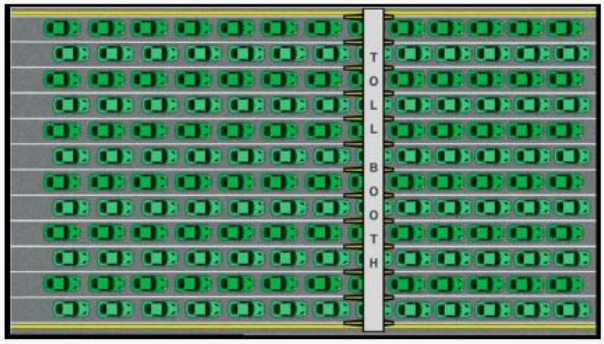


Fig 6 : AMD 's approach – 12 cores with 12 dedicated threads

Şekil 5.1.

Intel elbette uzun zamandan beri favori bir işlemci oldu. En son güncellemeler, en son piyasaya sürülen çekirdek işlemci çipi i7'de gelir. Yeni bir platform mimarisi, QPI(Quivk path Intercnnect) ile daha yüksek performanslı çok işlemcili sistemler, akıllı L3 Önbellek ile iyileştirilmiş önbellek gecikme yanıt süresi, CPU artışı geliyor

Intel boost özelliği sayesinde performans, hiper iş parçacığı aracılığıyla optimize edilmiş performanslar ve iyileştirilmiş sanallaştırma performansı. Şekil 5 ve Şekil 6, Intel'in yaklaşımını göstermektedir – sırasıyla 6 paylaşımlı çekirdeğe sahip 12 çekirdek ve 12 ayrılmış iş parçacığına sahip AMD'nin -12 çekirdek yaklaşımı.

S/N	FEATURES	AMD (OPTERON 6300)	INTEL (ITANIUM 2)
1.	Supported instruction set	MMX, SSE, SSE2, SSE3, SSE4	SSE
2.	FSB (Front Signal Bus)	1000MHz	773MHz
3.	Clock Frequency	2.4	1.66
4.	Thread count	16	2
5.	Built in Graphics	No	Yes
6.	Number of CPU cores	16	8
7.	XOP Instructions	Included	Excluded
8.	Hardware channels	DDR3	DDR3
9.	On-chip L2 + L3 cache	16+16MB	22 MB
10.	Power consumption	115W	115W

Şekil 5.2.

Mevcut trendde bir mikroişlemcinin performansına katkıda bulunan çeşitli önemli parametreleri karşılaştırdığımız bu tabloyu kullanarak üç ana Mikroişlemciyi (Apple, Intel ve AMD aileleri) karşılaştırarak sonuca varıyoruz. Burada açıklanan üç mikroişlemci ailesi gerçekten zirve için rekabet ediyor.

Modern yüksek performanslı mikroişlemcilerde nokta. En yeni Intel ve AMD mikroişlemciler performans açısından çok yakın görünüyor. Tablo 1, sınırlı sayıda ilgilenilen parametre kullanan, dikkate alınan 32 bit işlemciler arasındaki farkların sınırlı bir brifingini göstermektedir. Tabloda DDR, çift veri hızı 3'ü (bellek) temsil eder.[11]

6.INTEL İ9 INSTRUCTİON SET & ARCHİTECTURE

Tüm Intel Mimarisi talimat kodlamaları, Şekil 2-1'de gösterilen genel talimat formatının alt kümeleridir. Talimatlar, isteğe bağlı talimat ön eklerinden (herhangi bir sırayla), bir veya iki birincil işlem kodu baytından, ModR/M baytından ve bazen SIB (Ölçek-Dizin-Taban) baytından oluşan bir adresleme formu belirticisinden (gerekirse), bir yer değiştirmeden oluşur. (Gerekirse) ve bir anlık veri alanı (gerekirse). [15]

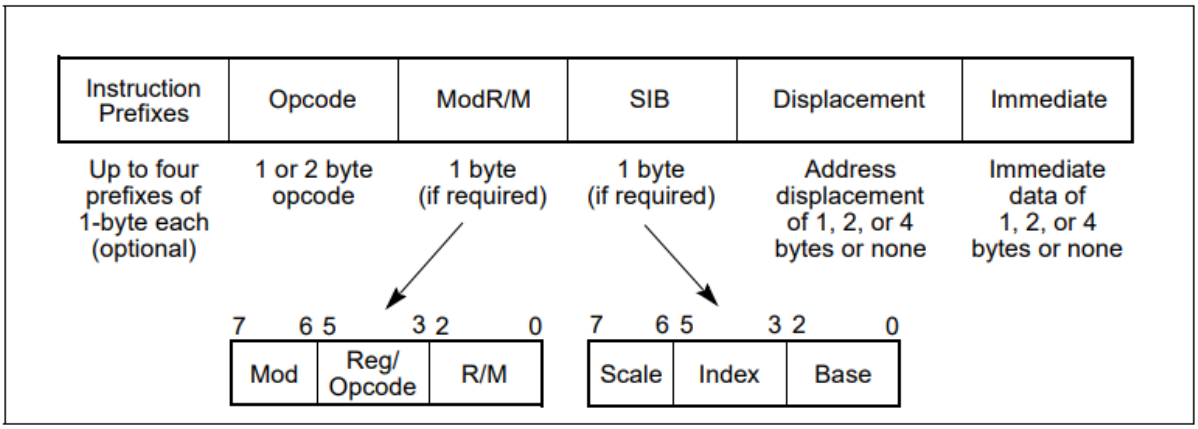


Figure 2-1. Intel Architecture Instruction Format

Şekil 6.1.

6.1 Instruction Set Extensions And Feature İntroduction İn Intel

Recent instruction set extensions and features are listed in Table 1-2. Within these groups, most instructions and features are collected into functional subgroups.

Table 1-2. Recent Instruction Set Extensions / Features Introduction in Intel® 64 and IA-32 Processors[1]

Instruction Set Architecture / Feature	Introduction
Direct stores: MOVDIRI, MOVDIR64B	Tremont, Tiger Lake, Sapphire Rapids
AVX512_BF16	Cooper Lake, Sapphire Rapids
CET: Control-flow Enforcement Technology	Tiger Lake, Sapphire Rapids
AVX512_VP2INTERSECT	Tiger Lake (not currently supported in any other processors)
Enqueue Stores: ENQCMD and ENQCMDS	Sapphire Rapids
CLDEMOTE	Tremont, Alder Lake, Sapphire Rapids
PTWRITE	Goldmont Plus, Alder Lake, Sapphire Rapids
User Wait: TPAUSE, UMONITOR, UMWAIT	Tremont, Alder Lake, Sapphire Rapids
Architectural LBRs	Alder Lake, Sapphire Rapids
HLAT	Alder Lake, Sapphire Rapids
SERIALIZE	Alder Lake, Sapphire Rapids
Intel® TSX Suspend Load Address Tracking (TSXLDTRK)	Sapphire Rapids
Intel® Advanced Matrix Extensions (Intel® AMX) Includes CPUID Leaf 1EH, "TMUL Information Main Leaf", and CPUID bits AMX-BF16, AMX-TILE, and AMX-INT8.	Sapphire Rapids
AVX-VNNI	Alder Lake[2], Sapphire Rapids
User Interrupts (UINTR)	Sapphire Rapids
Intel® Trust Domain Extensions (Intel® TDX)[3]	Future Processors
Supervisor Memory Protection Keys (PKS)[4]	Sapphire Rapids
Linear Address Masking (LAM)	Future Processors
IPI Virtualization	Sapphire Rapids

6.2. Intel® Gelişmiş Vektör Uzantıları 512 (Intel® AVX-512) Talimatları için Esaslar

Intel® Gelişmiş Vektör Uzantıları 512 (Intel® AVX-512) Yönergelerinin Esasları, 256-bit SIMD yönergelerinin çoğunu teşvik ederek Intel® Gelişmiş Vektör Uzantılarını (Intel® AVX)

ve Intel® Gelişmiş Vektör Uzantıları 2'yi (Intel® AVX2) genişletir. 512-bit sayısal işleme yetenekleri.

Intel® AVX-512 yönergeleri, Intel® AVX2 yönergeleriyle aynı programlama modelini izleyerek, yayın için gelişmiş işlevsellik, tahmini etkinleştirmek için katıştırılmış maskeleme, katıştırılmış kayan nokta yuvarlama kontrolü, katıştırılmış kayan nokta hata bastırma, dağılım yönergeleri, yüksek hızlı matematik sağlar talimatlar ve büyük yer değiştirme değerlerinin kompakt gösterimi. Performans cezaları olmadan karıştırılamayan Intel® SSE ve Intel® AVX'in aksine, Intel® AVX ve Intel® AVX-512 talimatlarının karıştırılması cezasız olarak desteklenir.

Intel® AVX-512 iç özellikleri, 32nm işlem teknolojisinden oluşturulmuş IA-32 ve Intel® 64 mimarilerinde desteklenir. Doğrudan yeni Intel® AVX-512 talimatlarına ve diğer gelişmiş 128 bit ve 256 bit SIMD talimatlarına eşlenirler. [16]

6.3. Intel® Gelişmiş Vektör Uzantıları 2 (Intel® AVX2) için Esaslar

Intel® Gelişmiş Vektör Uzantıları 2 (Intel® AVX2), 128-bit SIMD tamsayı yönergelerinin çoğunu 256-bit sayısal işleme yetenekleriyle destekleyerek Intel® Gelişmiş Vektör Uzantılarını (Intel® AVX) genişletir. Intel® AVX2 yönergeleri, Intel® AVX yönergeleriyle aynı programlama modelini izler.

Intel® AVX2 ayrıca veri öğeleri üzerinde yayın/permute işlemleri için gelişmiş işlevsellik, veri öğesi başına değişken kaydırma sayısıyla vektör kaydırma yönergeleri ve bitişik olmayan veri öğelerini bellekten almaya yönelik yönergeler sağlar.

Intel® AVX2 gerçekleri, __m128, __m128i, __m256 ve __m256i veri türlerini kullanan vektör çeşitlerine sahiptir.

Bu içsel bilgileri kullanmak için immintrin.h dosyası aşağıdaki gibi eklenir:

#immintrin.h'yi dahil edilir.

Intel® AVX2 iç özellikleri, 32nm işlem teknolojisinden oluşturulmuş IA-32 ve Intel® 64 mimarilerinde desteklenir. Doğrudan Intel® AVX2 yeni talimatlarına ve diğer geliştirilmiş 128 bit SIMD talimatlarına eşlenirler.

Intel® AVX2 yönergeleri, 128 bit tamsayı SIMD komut setlerinin büyük çoğunluğunun 256 bit geniş YMM kayıtları ile çalışmasını destekler. Intel® AVX2 yönergeleri, VEX ön eki kullanılarak kodlanmıştır ve Intel® AVX ile aynı işletim sistemi desteğini gerektirir. Genel

olarak, tanıtılan 256 bit vektör tamsayı yönergelerinin çoğu, Intel® AVX'te tanıtılan 256 bit kayan noktalı SIMD yönergelerine benzer şekilde 128 bit şerit işlemini izler.

Intel® AVX2 yönergeleri aşağıdaki gibi geniş bir şekilde kategorize edilebilir:

Intel® AVX tamamlayıcı tamsayı yönergeleri: Intel® AVX2 yönergeleri, tamsayı işlemleri için yazılan Intel® AVX yönergelerini, tamsayı veri öğeleriyle çalıştırmaya yönelik eşdeğer yönergelerin tam tamamlayıcısıyla tamamlar.

BROADCAST ve PERMUTE komutları: Bu komutlar, kayan noktalı ve tamsayılı işlemler için şeritler arası işlevsellik sağlar. Ek olarak, eski SSE komut setlerinden yükseltilen bazı Intel® AVX2 256-bit vektör tamsayı yönergeleri de şeritler arası davranış sergiliyor; örneğin, VPMOVZ/VPMOVS ailesinin talimatları.

SHIFT yönergeleri: Intel® AVX2 vektör SHIFT yönergeleri, öğe başına kaydırma sayısıyla çalışır ve 32 ve 64 bitlik veri öğesi boyutlarını destekler.

GATHER yönergeleri: Intel® AVX2 vektör GATHER yönergeleri, bitişik olmayan veri öğelerini vektör dizin bellek adreslemesi kullanarak bellekten almak için kullanılır. Bir temel yazmaçtan ve bir vektör yazmacı (XMM veya YMM) tarafından belirtilen birden fazla indeksten oluşan yeni bir bellek adresleme formu sunarlar. 32 ve 64 bit veri öğesi boyutlarının yanı sıra kayan nokta ve tamsayı öğeleri için veri türleri desteklenir. [17]

6.4 SIMD (Single Instruction Multiple Data)

SIMD, aynı işlemleri birden fazla veri şeridinde uygulayarak CPU performansını artırır. Kod, işlemcinin komut setiyle aynı hizada olduğu sürece, daha fazla şerit genellikle daha iyi performans anlamına gelir. Oyun geliştiricileri genellikle kodlarını en yaygın olarak bulunan SIMD komut seti için vektörleştirir. Sonuç olarak, Intel® Akış SIMD Uzantıları 4 (Intel® SSE4) gibi komut setleri destekten payını alır. Bu arada, Intel® Gelişmiş Vektör Uzantıları 2 (Intel® AVX2) veIntel® Gelişmiş Vektör Uzantıları 512 (Intel® AVX-512) gibi daha yeni setler nispeten kullanılmamıştır. Bazı durumlarda, sıkı oyuncular ve erken benimseyenler, son teknoloji CPU'larının neden bekledikleri performans artışını sağlamadığını merak edebilir.

Birden fazla ISA'yı (komut seti mimarileri) hedeflemenin yüksek üretkenlik maliyeti nedeniyle, geliştiricilerin mevcut her SIMD seti için kodlama yapmamaları muhtemeldir. İşleri daha da karmaşık hale getirmek için, bir işlemci ailesindeki işlemcinin farklı sürümleri, farklı SIMD komut setleri sunabilir. Her komut seti için içsel kod yazmak pratik değildir.

SIMD programlama teknikleri, derleme zamanında veya bağlantı zamanında uygulanır. Bu teknikler şunları içerir:

- Derleyici tabanlı otomatik vektörleştirme
- Vektör sınıfı kitaplıklara yapılan çağrılar
- Elle kodlanmış gerçekler

Her yaklaşımın artıları ve eksileri vardır. El ile kodlanmış gerçekler - bir içsel ninja tarafından işlendiğinde - tipik olarak mükemmel sonuçlar verir. Ancak çok çeşitli SIMD komut setleri için elle kodlama karmaşıklığı artırabilir, zaman alıcıdır ve bakım maliyetlerini artırır. Örneğin, birden çok ISA'yı hedeflemek istiyorsanız, birden çok algoritma yazmanız gerekir. Bu, üretkenliği azaltır ve kod karmaşıklığını artırır. Otomatik vektörleştirme derleyicileri, birden çok ISA'yı hedeflemenin karmaşıklığını azaltabilir, ancak bunlar mükemmel olmaktan uzaktır. Derleyici, deneyimli bir programcının yerini tutamaz. Programcı genellikle SIMD esaslarını veya hat içi montajı kullanarak manuel olarak optimize etmeye bırakılır. [17]

6.5 Intel® Amx Instructıon Set Reference

Intel® Gelişmiş Matris Uzantıları (Intel® AMX), iki bileşenden oluşan yeni bir 64-bit programlama paradigmasıdır: daha büyük bir 2 boyutlu bellek görüntüsünden alt dizileri temsil eden bir dizi 2 boyutlu kayıt (döşeme) ve hızlandırıcı döşemeler üzerinde işlem yapmak için ilk uygulamaya TMUL (döşeme matrisi çarpma birimi) adı verilir. Bir Intel AMX uygulaması, programcıya bir seçenekler paleti sağlayarak döşemelerin nasıl programlanabileceğini sıralar. İki palet desteklenir; palet 0, başlatılmış durumu temsil eder ve palet 1, TMM0..TMM7 adlı 8 döşeme kaydına yayılmış 8 KB depolamadan oluşur. Her döşemenin maksimum boyutu 16 satır x 64'tür. 1 bayt, (1 KB), ancak programcı her kutucuğu kendi algoritmasına uygun olarak daha küçük boyutlara yapılandırabilir. Programcı tarafından sağlanan döşeme d boyutları (satırlar ve bytes_per_row, yani colsb), döşeme ve hızlandırıcı talimatlarının yürütülmesini sağlayan meta verilerdir. Bu şekilde, tek bir talimat, döşeme ve hızlandırıcı donanımında otonom çok döngülü yürütmeyi başlatabilir. Palet değeri (palette_id) ve meta veriler tutulur. Bir kontrol kaydında (TILECFG) dahili olarak. TILECFG içerikleri, palet_tablosunda bildirilenlerle orantılı olacaktır (mevcut parametrelerin açıklaması için Bölüm 1'deki “CPUID—CPU Tanımlaması”na bakın). Intel AMX genişletilebilir bir mimaridir. Yeni hızlandırıcılar eklenebilir veya TMUL hızlandırıcı geliştirilebilir

daha yüksek performans sağlamak için. Bu durumlarda, kutucuklar tarafından sağlanan durumun (TILEDATA) meta veri boyutlarından birinde (daha fazla satır veya sütunb) ve/veya

daha fazla adı destekleyerek büyütülmesi gerekebilir. Genişletilebilirlik, ek durumu açıklayan yeni palet girişleri eklenerek gerçekleştirilir. Yürütme meta verilerle yürütüldüğünden, mevcut bir Intel AMX ikili dosyası, CPUID tarafından belirtilen en güçlü paleti seçerek ve döngü ve işaretçi güncellemelerini buna göre ayarlayarak daha büyük depolama boyutlarından ve daha yüksek performanslı TMUL birimlerinden yararlanabilir. Şekil 3-1, Intel AMX mimarisinin kavramsal bir diyagramını göstermektedir. Bir Intel mimarisi ana bilgisayarı algoritmayı, bellek engellemeyi, döngü indekslerini ve işaretçi aritmetiğini yönetir. Fayans yükleri ve depoları ve hızlandırıcı komutları, çok döngülü yürütme birimlerine gönderilir. Gerekirse durum geri bildirilir. Intel AMX yönergeleri, Intel mimarisi yönerge akışında eşzamanlıdır ve kutucuk yönergeleri tarafından yüklenen ve depolanan bellek, ana bilgisayarın bellek erişimlerine göre tutarlıdır. Intel mimarisi ve Intel AMX kodunun serpiştirilmesi veya ana bilgisayarın Intel AMX ile paralel olarak kullanabileceği kaynaklar (örn. Intel AVX512) üzerinde herhangi bir kısıtlama yoktur. Ayrıca, Intel mimarisi ana bilgisayarının Intel mimarisi bilgi işlem yeteneği için 64 bit modunu desteklemesi dışında herhangi bir mimari gereksinim yoktur.

Intel® Gelişmiş Matris Uzantıları (Intel® AMX), iki bileşenden oluşan yeni bir 64-bit programlama paradigmasıdır: daha büyük bir 2 boyutlu bellek görüntüsünden alt dizileri temsil eden bir dizi 2 boyutlu kayıt (döşeme) ve hızlandırıcı döşemeler üzerinde işlem yapmak için ilk uygulamaya TMUL (döşeme matrisi çarpma birimi) adı verilir.

Bir Intel AMX uygulaması, programcıya bir seçenekler paleti sağlayarak döşemelerin nasıl programlanabileceğini sıralar. İki palet desteklenir; palet 0, başlatılmış durumu temsil eder ve palet 1, TMM0..TMM7 adlı 8 döşeme kaydına yayılmış 8 KB depolamadan oluşur. Her döşemenin maksimum boyutu 16 satır x 64'tür. 1 Bayt, (1 KB), ancak programcı her kutucuğu kendi algoritmasına uygun olarak daha küçük boyutlara yapılandırabilir. Programcı tarafından sağlanan döşeme d boyutları (satırlar ve bytes_per_row, yani colsb), döşeme ve hızlandırıcı talimatlarının yürütülmesini sağlayan meta verilerdir. Bu şekilde, tek bir talimat, döşeme ve hızlandırıcı donanımında otonom çok döngülü yürütmeyi başlatabilir. TILECFG içerikleri, palet tablosunda bildirilenlerle orantılı olacaktır. Intel AMX genişletilebilir bir mimaridir. Yeni hızlandırıcılar eklenebilir veya TMUL hızlandırıcı geliştirilebilir. Daha yüksek performans sağlamak için. Bu durumlarda, kutucuklar tarafından sağlanan durumun (TILEDATA) meta veri boyutlarından birinde (daha fazla satır veya sütunb) ve/veya daha fazla adı destekleyerek büyütülmesi gerekebilir. Genişletilebilirlik, ek durumu açıklayan yeni palet girişleri eklenerek gerçekleştirilir. Yürütme meta verilerle yürütüldüğünden, mevcut bir Intel AMX ikili dosyası, CPUID tarafından belirtilen en güçlü paleti seçerek ve döngü ve işaretçi güncellemelerini buna

göre ayarlayarak daha büyük depolama boyutlarından ve daha yüksek performanslı TMUL birimlerinden yararlanabilir. Şekil 3-1, Intel AMX mimarisinin kavramsal bir diyagramını göstermektedir. Bir Intel mimarisi ana bilgisayarı algoritmayı, bellek engellemeyi, döngü indekslerini ve işaretçi aritmetiğini yönetir. Fayans yükleri ve depoları ve hızlandırıcı komutları, çok döngülü yürütme birimlerine gönderilir. Gerekirse durum geri bildirilir. Intel AMX yönergeleri, Intel mimarisi yönerge akışında eşzamanlıdır ve kutucuk yönergeleri tarafından yüklenen ve depolanan bellek, ana bilgisayarın bellek erişimlerine göre tutarlıdır. Intel mimarisi ve Intel AMX kodunun serpiştirilmesi veya ana bilgisayarın Intel AMX ile paralel olarak kullanabileceği kaynaklar (örn. Intel AVX512) üzerinde herhangi bir kısıtlama yoktur. Ayrıca, Intel mimarisi ana bilgisayarının Intel mimarisi bilgi işlem yeteneği için 64 bit modunu desteklemesi dışında herhangi bir mimari gereksinim yoktur. [2]

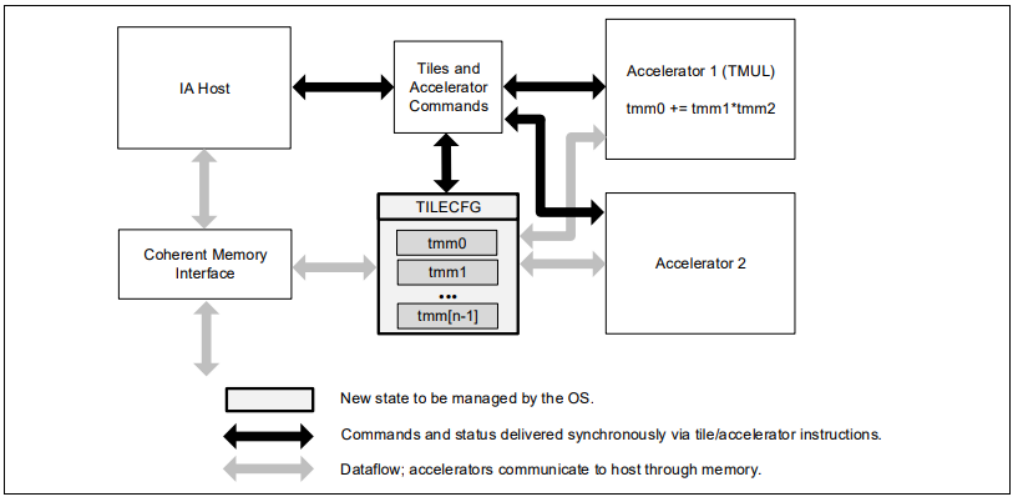


Figure 3-1. Intel® AMX Architecture

Şekil 6.5.1.

7.AMD INSTRUCTİON SET & ARCHITECTURE

AMD64 mimarisi, uzun modu desteklemek için ek talimatlarla birlikte tam eski x86 komut setini destekler. Uygulama programlama talimatları aşağıdaki gibi dört alt küme halinde düzenlenmiştir:

•**General-Purpose Programming**—Bunlar, hemen hemen tüm programlarda kullanılan temel x86 tamsayı yönergeleridir. Bu yönergelerin çoğu, genel amaçlı bilgisayarda bulunan verileri yükler, depolar veya bunlar üzerinde çalışır. Registerlar (GPR'ler) veya bellek. Talimatlardan bazıları, diğer program konumlarına dallanarak sıralı program akışını değiştirir.

• **Streaming SIMD Extensions (SSE) Programming (SSE)**—Bu talimatlar, öncelikle YMM/XMM kayıtlarında bulunan verileri yükler, depolar veya bunlar üzerinde çalışır. 128 bit

ortam yönergeleri alt kısımda çalışır. YMM kayıtlarının yarısı. SSE komutları, vektör (paketlenmiş) ve skaler veri türleri üzerinde tamsayı ve kayan nokta işlemleri gerçekleştirir. Çünkü vektör komutları bağımsız olarak ve birden fazla veri kümesi üzerinde aynı anda tek bir işlem gerçekleştirir, bunlara tek komut, çoklu veri (SIMD) yönergeleri denir. Veri blokları üzerinde çalışan yüksek performanslı medya ve bilimsel uygulamalar için kullanışlıdırlar.

• **Multimedia Extensions (MMX™) Programming—**Bunlar, MMX™ teknolojisi ve AMD 3DNow!™ teknolojisi talimatlarını içerir. Bu yönergeler, birincil olarak sunucuda bulunan verileri yükler, depolar veya bunlar üzerinde çalışır. 80 bitlik x87 kayan noktalı kayıtlara eşlenen 64 bitlik MMX kayıtları. SSE yönergeleri gibi, vektör (paketlenmiş) ve skaler veriler üzerinde tamsayı ve kayan nokta işlemleri gerçekleştirirler. Bu yönergeler, yüksek hassasiyet gerektirmeyen medya uygulamalarında kullanışlıdır. Multimedya Uzantı Yönergeleri, sonuç oluşturmayan doyurucu matematiksel işlemler kullanır. Operasyon istisnaları.

• **x87 Floating-Point Programming**—Bunlar, eski x87 uygulamalarında kullanılan kayan nokta talimatlarıdır. 80 bitlik x87 kayıtlarında bulunan verileri yükler, depolar veya bunlar üzerinde çalışırlar. Bu uygulama programlama yönergelerinden bazıları, yukarıdaki alt kümelerin iki veya daha fazlasını köprüler. Örneğin, verileri genel amaçlı yazmaçlar ile YMM/XMM veya MMX yazmaçları arasında hareket ettiren komutlar vardır ve tamsayı vektör (paketlenmiş) komutlarının çoğu, aynı anda olmasa da YMM/XMM veya MMX yazmaçlarında çalışabilir. Talimatlar iki veya daha fazla alt küme arasında köprü kuruyorsa, açıklamaları uygulandıkları tüm alt kümelerde tekrarlanır.

7.1 Media Instructions

Görüntü işleme, müzik sentezi, konuşma tanıma, tam hareket gibi medya uygulamaları

video ve 3B grafik oluşturma—belirli özellikleri paylaşır:

• Büyük miktarda veri işlerler.

• Veriler boyunca genellikle aynı işlem dizisini tekrar tekrar gerçekleştirirler.

• Veriler genellikle piksel değerleri için 8 bit, ses örnekleri için 16 bit ve kayan nokta biçiminde nesne koordinatları için 32 bit gibi küçük miktarlar olarak temsil edilir.

SSE ve MMX komutları bu uygulamaları hızlandırmak için tasarlanmıştır. Talimatlar, tek talimat, çoklu veri (SIMD) olarak bilinen bir vektör (veya paketlenmiş) paralel işleme biçimini kullanır. Bu vektör teknolojisi aşağıdaki özelliklere sahiptir:

• Tek bir kayıt birden çok bağımsız veri parçasını tutabilir. Örneğin, tek bir YMM kaydı, 32 adet 8 bitlik tamsayı veri öğesini veya sekiz adet 32 bitlik tek duyarlıklı kayan noktalı veriyi tutabilir.

• Vektör komutları, bir kayıttaki tüm veri öğeleri üzerinde bağımsız ve aynı anda çalışabilir. Örneğin, 128-bit XMM kayıtlarında iki vektör işlenenin bayt öğeleri üzerinde çalışan bir PADDB talimatı, 16 eş zamanlı toplama gerçekleştirir ve 16 döndürür. Tek bir operasyonda bağımsız sonuçlar. SSE ve MMX komutları, özel komutlar ekleyerek SIMD vektör teknolojisini bir adım öteye taşır. Medya uygulamalarında yaygın olarak bulunan işlemleri gerçekleştiren talimatlar. Örneğin, iki pikselin parlaklık değerlerini toplayan bir grafik uygulaması, sonuç hedef yazmacı aşarsa, ekleme işleminin küçük bir değere sarılmasını önlemelidir, çünkü bir taşma sonucu, parlak bir görüntünün olduğu yerde koyu bir piksel gibi beklenmeyen etkiler üretebilir. Bu talimatlar, bu tür işlemleri basitleştirmek için doyurucu aritmetik talimatları içerir. Aksi takdirde taşma veya yetersiz akış nedeniyle sarılacak bir sonuç, bunun yerine hedef kayıt defterinde temsil edilebilecek en büyük veya en küçük değerde doyuma zorlanır.

7.2 Floating-Point Instructions

AMD64 mimarisi, üç farklı kayıt kümesi kullanarak üç kayan noktalı komut alt kümesi sağlar:

• SSE yönergeleri, tamsayı işlemlerine ek olarak 32 bit tek duyarlıklı ve 64 bit çift duyarlıklı kayan nokta işlemlerini destekler. Hem vektör verileri hem de skaler veriler üzerindeki işlemler, özel bir kayan nokta istisna raporlama mekanizmasıyla desteklenir. Bu kayan nokta işlemleri, IEEE-754 standardına uygundur.

• MMX Komutları, tek duyarlıklı kayan nokta işlemlerini destekler. Hem vektör verileri hem de skaler veriler üzerindeki işlemler desteklenir, ancak bu talimatlar kayan nokta istisna raporlamasını desteklemez.

• x87 Kayan Nokta Yönergeleri, tek kesinlikli, çift kesinlikli ve 80 bit genişletilmiş kesinlikli kayan nokta işlemlerini destekler. Özel bir kayan nokta istisna raporlama mekanizmasıyla yalnızca skaler veriler desteklenir. x87 kayan nokta komutları, trigonometrik ve logaritmik transandantal işlemleri gerçekleştirmek için özel talimatlar içerir.

Tek duyarlıklı ve çift duyarlıklı kayan nokta işlemleri, IEEE-754 standardına uygundur. 256 bit ortam yönergeleri kullanılarak maksimum kayan nokta performansı elde edilebilir. Bu vektör komutlarından biri paralel olarak sekiz adede kadar tek duyarlıklı (veya dört çift duyarlıklı)

işlemi destekleyebilir. 64 bit modunda bulunan toplam 16 adet 256 bit YMM kaydı, ara sonuçları tutmak için daha fazla kayıt sağlayarak uygulamaları hızlandırır, böylece bu sonuçların bellekte saklanması ihtiyacını azaltır.

7.3 Modes of Operation

Uzun Mod

Uzun mod, eski korumalı modun bir uzantısıdır. Uzun mod iki alt moddan oluşur: 64 bit modu ve uyumluluk modu. 64 bit modu, tüm özellikleri ve kayıt uzantılarını destekler. AMD64 mimarisi. Uyumluluk modu, mevcut 16 bit ve 32 bit uygulamalarla ikili uyumluluğu destekler. Uzun mod, eski gerçek modu veya eski sanal-8086 modunu desteklemez ve donanım görevi değiştirmeyi desteklemez. Bu belge boyunca, uzun moda yapılan atıflar hem 64 bit moduna hem de uyumluluk moduna atıfta bulunur. Bir fonksiyon bu alt modlardan herhangi birine özgüyse, o zaman belirli alt modun adı kullanılır.

64-Bit Modu

Uzun modun bir alt modu olan 64 bit modu, 64 bit sanal adresleme ve kayıt uzantısı özelliklerinin tamamını destekler. Bu mod, işletim sistemi tarafından ayrı bir kod segmenti bazında etkinleştirilir. 64 bit modu, 64 bit sanal adres alanını desteklediğinden, 64 bit işletim sistemi ve araç zinciri gerektirir. Mevcut uygulama ikili dosyaları, 64 bit modunda çalışan bir işletim sistemi altında uyumluluk modunda yeniden derleme olmadan çalışabilir veya uygulamalar 64 bit modunda çalışacak şekilde yeniden derlenebilir. Adresleme özellikleri arasında 64 bitlik bir yönerge işaretçisi (RIP) ve RIP'e bağlı bir veri adresleme bulunur. Bu mod, tek kod, veri ve yığın alanı ile yalnızca düz bir adres alanını destekleyerek modern işletim sistemlerini barındırır. Uzantıları Kaydedin. 64 bit modu, REX önekleri adı verilen bir grup talimat öneki aracılığıyla kayıt uzantılarını uygular. Bu uzantılar sekiz GPR (R8–R15) ekler, tüm GPR'leri 64 bite genişletir ve sekiz YMM/XMM kaydı (YMM/XMM8–15) ekler.REX komut ön ekleri ayrıca on altı GPR'den herhangi birinin düşük baytını bayt işlemleri için kullanılabilir kılan bir bayt kaydı yeteneği sağlar. Bu, derleyici kayıt tahsisi için daha uygun olan tek tip bir bayt, sözcük, çift sözcük ve dört sözcük kaydıyla sonuçlanır. 64-Bit Adresler ve İşlenenler. 64 bit modunda, varsayılan sanal adres boyutu 64 bittir (uygulamalarda daha az olabilir). Çoğu talimat için varsayılan işlenen boyutu 32 bittir. Çoğu talimat için, bu varsayılanlar talimat önekleri kullanılarak talimat bazında geçersiz kılınabilir. REX önekleri, 64 bit işlenen boyutunu ve kayıt uzantılarını belirtir.RIP-Göreceli Veri Adresleme. 64 bit modu, 64 bit yönerge işaretçisine (RIP) göre veri adreslemeyi destekler. Eski x86 mimarisi, yalnızca

kontrol aktarım yönergelerinde IP'ye bağlı adreslemeyi destekler. RIP'ye göre adresleme, konumdan bağımsız kodun ve küresel verileri ele alan kodun verimliliğini artırır. İşlem kodları. Kayıt uzantılarına ve 64 bit adreslemeye izin vermek için birkaç talimat işlem kodu ve önek baytı yeniden tanımlanır.

Uyumluluk Modu

Uzun modun ikinci alt modu olan uyumluluk modu, 64 bit işletim sistemlerinin mevcut 16 bit ve 32 bit x86 uygulamalarını çalıştırmasına izin verir. Bu eski uygulamalar, yeniden derleme olmadan uyumluluk modunda çalışır

Eski Mod

Eski mod, ikili uyumluluğu yalnızca mevcut 16 bit ve 32 bit uygulamalarla değil, aynı zamanda mevcut 16 bit ve 32 bit işletim sistemleriyle de korur. Eski mod aşağıdaki üç bölümden oluşur

Alt modlar:

• **Korumalı Mod**—Korumalı mod, bellek bölümleme, isteğe bağlı sayfalama ve ayrıcalık denetimi ile 16 bit ve 32 bit programları destekler. Korumalı modda çalışan programlar, 4 GB'a kadar bellek alanına erişebilir.

• **Virtual-8086 Modu**—Virtual-8086 modu, korumalı modda görev olarak çalışan 16 bitlik gerçek mod programlarını destekler. Basit bir bellek bölümleme biçimi, isteğe bağlı sayfalama ve sınırlı koruma denetimi kullanır. Virtual-8086 kipinde çalışan programlar 1 MB'a kadar bellek alanına erişebilir.

• **Gerçek Mod**—Gerçek mod, basit kayıt tabanlı bellek segmentasyonu kullanan 16 bit programları destekler. Sayfalamayı veya koruma denetimini desteklemez. Gerçek modda çalışan programlar, 1 MB'a kadar bellek alanına erişebilir. Eski mod, x86 mimarisinin mevcut 32 bit işlemci uygulamalarıyla uyumludur. AMD64 mimarisini uygulayan işlemciler, tıpkı eski x86 mimarisini uygulayan işlemciler gibi eski gerçek modda açılır.

7.4 Genel Amaçlılar

Genel amaçlı programlama modeli, genel amaçlı kayıtları (GPR'ler), tamsayı komutlarını ve GPR'leri kullanan işlenenleri, program akışı kontrol yöntemlerini, bellek optimizasyon yöntemlerini ve G/Ç'yi içerir. Bu programlama modeli, orijinal x86 tamsayı programlama mimarisini, ayrıca 64 bit uzantıları ve birkaç ek talimatı içerir. Bu bölümde yalnızca uygulama programlama talimatları ve kaynakları açıklanmaktadır. Tüm ayrıcalıklı komutlar dahil sistem

programlamada tipik olarak kullanılan tamsayı komutları, diğer sistem programlama konularıyla birlikte Cilt 2'de açıklanmaktadır. Genel amaçlı programlama modeli, başlıca 256-bit veya 128-bit ortam yönergeleri, 64-bit ortam yönergeleri, x87 kayan nokta yönergeleri veya sistem yönergelerinden oluşan programlar dahil olmak üzere hemen hemen tüm programlar tarafından bir dereceye kadar kullanılır. Bu nedenle, AMD64 komut seti mimarisini kullanan herhangi bir programlama çalışması için genel amaçlı programlama modelinin anlaşılması önemlidir.

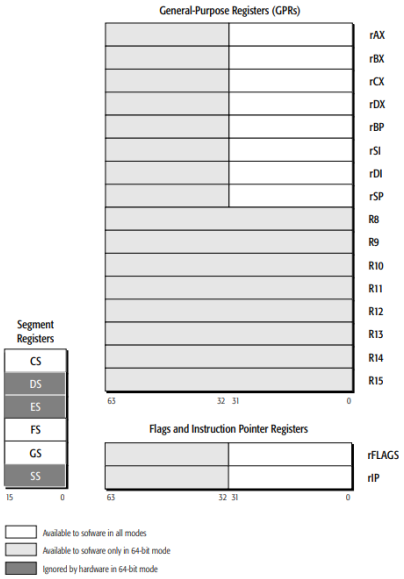


Figure 3-1. General-Purpose Programming Registers

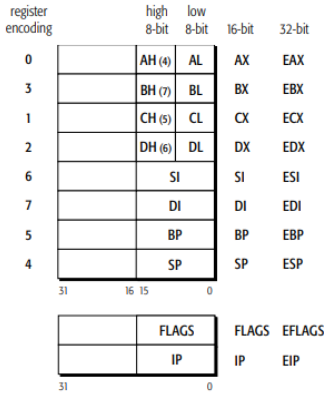


Figure 3-2. General Registers in Legacy and Compatibility Modes

Şekil 7.4.1.

Genel amaçlı programlama ortamında aşağıdaki veri türleri desteklenir:

- İşaretli (ikiye tümleyen) tamsayılar.
- İşaretsiz tamsayılar.
- BCD haneleri.
- Paketlenmiş BCD rakamları.
- Bit dizileri dahil diziler.
- Türlenmemiş veri nesneleri.

Şekil 3-6, çoğu genel amaçlı yönerge tarafından kullanılan veri türlerini göstermektedir. Yazılım, veri türlerini gösterilenden farklı şekillerde tanımlayabilir, ancak AMD64 mimarisi bu tür yorumları doğrudan desteklemez ve yazılımın bunları tamamen kendi başına halletmesi gerekir. Bit konumlarının 0'dan başlayıp uzunluk-1 ile biten sağdan sola doğru

numaralandırıldığına dikkat edin. Türlenmemiş veri nesneleri bit, yarım bayt, bayt, kelime, çift kelime, dört kelime ve dışa doğru gösterilmez.

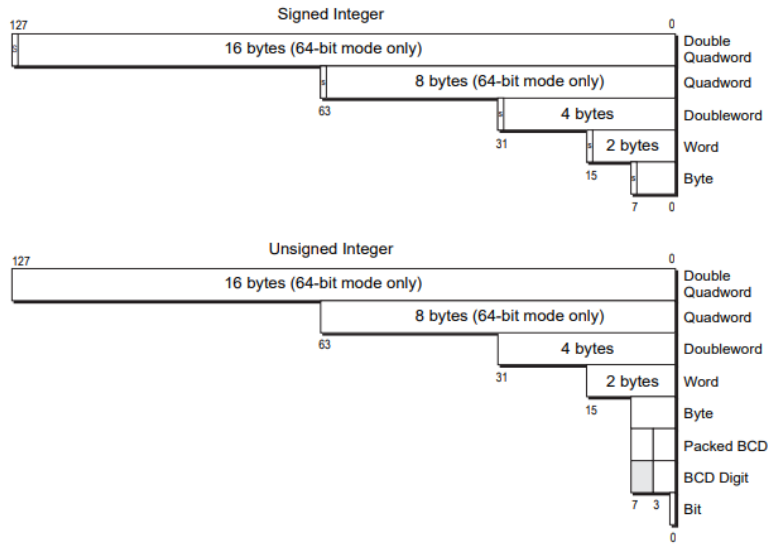


Figure 3-6. General-Purpose Data Types

Şekil 7.4.2.

7.4.1 Syntax

Her talimatın, kaynak ve hedef (sonuç) verileri için kullanılacak işlemi ve işlenenleri belirtmek için derleyiciler tarafından kullanılan anımsatıcı bir sözdizimi vardır. Şekil 3-7, bir karşılaştırma (CMP) talimatı için anımsatıcı sözdiziminin bir örneğini göstermektedir. Bu örnekte, CMP anımsatıcısını iki işlenen takip eder; bir 32 bitlik kayıt veya bellek işleneni ve bir 8 bitlik anlık işlenen.

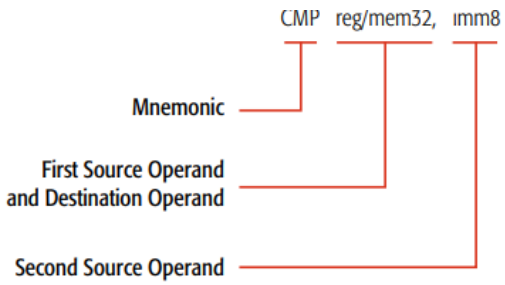


Figure 3-7. Mnemonic Syntax Example

Şekil 7.4.1.1.

7.4.2 Instructions

Data Transfer: The data-transfer instructions copy data between registers and memory.

Move

• MOV—Move

• MOVBE—Move Big-Endian

• MOVSX—Move with Sign-Extend

• MOVZX—Move with Zero-Extend

• MOVD—Move Doubleword or Quadword

• MOVNTI—Move Non-temporal Doubleword or Quadword

Stack Operations

• POP—Pop Stack

• POPA—Pop All to GPR Words

• POPAD—Pop All to GPR Doublewords

• PUSH—Push onto Stack

• PUSHA—Push All GPR Words onto Stack

• PUSHAD—Push All GPR Doublewords onto Stack

• ENTER—Create Procedure Stack Frame

• LEAVE—Delete Procedure Stack Frame

Data Conversion

The data-conversion instructions perform various transformations of data, such as operand-size doubling by sign extension, conversion of little-endian to big-endian format, extraction of sign masks,searching a table, and support for operations with decimal numbers.

Sign Extension

• CBW—Convert Byte to Word

• CWDE—Convert Word to Doubleword

• CDQE—Convert Doubleword to Quadword

• CWD—Convert Word to Doubleword

• CDQ—Convert Doubleword to Quadword

• CQO—Convert Quadword to Octword

Add and Subtract

• ADC—Add with Carry

• ADD—Signed or Unsigned Add

• SBB—Subtract with Borrow

• SUB—Subtract

• NEG—Two’s Complement Negation

Multiply and Divide

• MUL—Multiply Unsigned

• IMUL—Signed Multiply

• DIV—Unsigned Divide

• IDIV—Signed Divide

Increment and Decrement

• DEC—Decrement by 1

• INC—Increment by 1

Rotate

• RCL—Rotate Through Carry Left

• RCR—Rotate Through Carry Right

• ROL—Rotate Left

• ROR—Rotate Right

Shift

• SAL—Shift Arithmetic Left

• SAR—Shift Arithmetic Right

• SHL—Shift Left

• SHR—Shift Right

• SHLD—Shift Left Double

• SHRD—Shift Right Double

Bit Manipulation

The bit manipulation instructions manipulate individual bits in a register for purposes such as controlling low-level devices, correcting algorithms, and detecting errors.

• BEXTR—Bit Field Extract (register form is a BMI instruction)

• BEXTR—Bit Field Extract (immediate version is a TBM instruction)

Mask Bit

• BLSMSK—Mask from Lowest Set Bit

• BLCMSK—Mask From Lowest Clear Bit

• T1MSKC — Inverse Mask From Trailing Ones

• TZMSK—Mask From Trailing Zeros

Reset Bit

• BLSR—Reset Lowest Set Bit

Scan Bit

• BSF—Bit Scan Forward

• BSR—Bit Scan Reverse

Logical

The logical instructions perform bitwise operations.

• AND—Logical AND

• OR—Logical OR

• XOR—Exclusive OR

• NOT—One's Complement Negation

• ANDN—And Not

Compare Strings

• CMPS—Compare Strings

• CMPSB—Compare Strings by Byte

• CMPSW—Compare Strings by Word

• CMPSD—Compare Strings by Doubleword

• CMPSQ—Compare Strings by Quadword

Scan String

• SCAS—Scan String

• SCASB—Scan String as Bytes

• SCASW—Scan String as Words

• SCASD—Scan String as Doubleword

• SCASQ—Scan String as Quadword

Move String

• MOVS—Move String

• MOVSB—Move String Byte

• MOVSW—Move String Word

• MOVSD—Move String Doubleword

• MOVSQ—Move String Quadword

Store String

• STOS—Store String

• STOSB—Store String Bytes

• STOSW—Store String Words

• STOSD—Store String Doublewords

• STOSQ—Store String Quadword

Kontrol Transferi

Kontrol-aktarma yönergeleri veya dallar, döngüler boyunca yineleme yapmak ve koşullu program mantığında hareket etmek için kullanılır.

Jump

• JMP: Belirtilen adrese koşulsuz bir atlama gerçekleştirir. Hedef adresi belirtmenin birkaç yolu vardır.

• Göreli Kısa Atlama ve Göreceli Yakın Atlama—Hedef adresi, JMP'yi izleyen talimatın rIP'sine 8 bit (kısa atlama) veya 16 bit veya 32 bit (yakın atlama) işaretli yer değiştirme eklenerek belirlenir. Atlama, geçerli kod bölümü (CS) içinde gerçekleştirilir.

• Kayıt-Dolaylı ve Bellek-Dolaylı Yakın Atlama—Hedef rIP değeri bir kayıtta veya bir bellek konumunda bulunur. Atlama, geçerli CS içinde gerçekleştirilir.

• Doğrudan Uzak Atlama—Tüm uzak atlamalar için, hedef adres geçerli kod bölümünün dışındadır. Burada talimat, 16 bitlik hedef adres kodu segmentini ve 16 bitlik veya 32 bitlik uzaklığı anlık bir değer olarak belirtir. Doğrudan uzak atlama formu 64 bit modunda geçersizdir.

• Bellek Dolaylı Uzak Atlama—Bu form için, hedef adres (CS:rIP) geçerli kod bölümünün dışındaki bir adrestedir. Belirli bir bellek konumundaki 32 bitlik veya 48 bitlik bir uzak işaretçi, hedef adresi işaret eder.

Döngü

• LOOPcc—Döngü if koşulu

LOOPcc talimatları LOOPE, LOOPNE, LOOPNZ ve LOOPZ'yi içerir. Bu talimatlar, herhangi bir bayrağı değiştirmeden rCX kaydını 1 azaltır ve ardından döngü koşulunun karşılanıp karşılanmadığını kontrol eder. Koşul karşılanırsa, program belirtilen hedef koda atlar.

Call

• CALL—Prosedür Çağrısı

CALL komutu, adresi işlenende belirtilen bir prosedüre çağrı gerçekleştirir. Dönüş adresi, CALL tarafından yığına yerleştirilir ve CALL'dan hemen sonraki talimatı işaret eder. Çağrılan prosedür yürütmeyi bitirdiğinde ve bir dönüş talimatı kullanılarak çıkıldığında, kontrol yığında kayıtlı dönüş adresine aktarılır.

Return

• RET—Çağrıdan Dönüş

RET komutu, orijinal olarak CALL komutu kullanılarak çağrılan bir prosedürden geri döner. CALL, yığına bir dönüş adresi (CALL'dan sonraki talimata işaret eden) yerleştirir. RET, yığından dönüş adresini alır ve kontrolü bu adreste bulunan talimata aktarır.

Push and Pop Flags

• POPF—Pop to FLAGS Word

• POPFD—Pop to EFLAGS Doubleword

• POPFQ—Pop to RFLAGS Quadword

• PUSHF—Push FLAGS Word onto Stack

• PUSHFD—Push EFLAGS Doubleword onto Stack

• PUSHFQ—Push RFLAGS Quadword onto Stack

Set and Clear Flags

• CLC—Clear Carry Flag

• CMC—Complement Carry Flag

• STC—Set Carry Flag

• CLD—Clear Direction Flag

• STD—Set Direction Flag

• CLI—Clear Interrupt Flag

• STI—Set Interrupt Flag

7.5 Streaming SIMD Extensions Media and Scientific Programming

SSE aritmetik komutlarının çoğu, vektör çiftleri üzerinde paralel işlemler gerçekleştirir. Vektör işlemleri ayrıca paketlenmiş veya SIMD (tek komut, çoklu veri) işlemleri olarak da adlandırılır. Birden çok öğeden oluşan vektör işlenenlerini alırlar ve tüm öğeler paralel olarak çalıştırılır. Bazı SSE komutları vektörler yerine skalerler üzerinde çalışır. Paketten çıkarma talimatlarıyla birlikte taşıma talimatları, ortam prosedürlerinde en sık kullanılan talimatlar arasındadır. XMM kayıtları arasında veya bir XMM kaydı ile bellek arasında hareket ederken, her bir tamsayı taşıma talimatı 16 bayta kadar veri kopyalayabilir. Bir XMM kaydı ile bir MMX veya GPR kaydı arasında hareket ederken, bir tamsayı taşıma talimatı 8 bayta kadar veri taşıyabilir. Paketlenmiş kayan nokta taşıma komutları, paralel olarak dört tek duyarlıklı veya iki çift duyarlıklı kayan nokta işlenenin vektörlerini kopyalayabilir.

SSE talimatları, YMM/MXM kayıtlarında, MMX™ kayıtlarında, GPR kayıtlarında veya bellekte bulunan tamsayı ve kayan noktalı veri türleri arasındaki dönüşümler ve öğe sıralaması veya kesinlik dönüştürmeleri dahil olmak üzere vektör öğelerinin veri dönüştürmesini destekler. Örneğin, paketten çıkarma yönergeleri iki vektör işleneni alır ve düşük veya yüksek öğelerini serpiştirir. Sözcük boyutundaki öğeler üzerinde paketten çıkarma ve serpiştirme işlemini göstermektedir. Bu durumda, V)PUNPCKLWD. İşlenen 1, işaretsiz tam sayıların bir vektörüyse ve sol taraftaki kaynak işlenen, değeri sıfır olan öğelere sahipse, işlem, işlenen 1'in alt yarısındaki her öğeyi şuna dönüştürür:

Orijinal genişliğinin iki katı olan bir tamsayı veri türü. Bu, bir vektör çarpma işlemi sırasında taşma olmamasını sağlamak için iki tamsayı vektörünü birlikte çarpmadan önce faydalı bir adım olacaktır. SSE aritmetik komutlarının çoğu, vektör çiftleri üzerinde paralel işlemler gerçekleştirir. Vektör işlemleri ayrıca paketlenmiş veya SIMD (tek komut, çoklu veri) işlemleri olarak da adlandırılır. Birden çok öğeden oluşan vektör işlenenlerini alırlar ve tüm öğeler paralel olarak çalıştırılır. Bazı SSE komutları vektörler yerine skalerler üzerinde çalışır.

Paketten çıkarma talimatlarıyla birlikte taşıma talimatları, ortam prosedürlerinde en sık kullanılan talimatlar arasındadır. XMM kayıtları arasında veya bir XMM kaydı ile bellek arasında hareket ederken, her bir tamsayı taşıma talimatı 16 bayta kadar veri kopyalayabilir. Bir XMM kaydı ile bir MMX veya GPR kaydı arasında hareket ederken, bir tamsayı taşıma talimatı 8 bayta kadar veri taşıyabilir. Paketlenmiş kayan nokta taşıma komutları, paralel olarak dört tek duyarlıklı veya iki çift duyarlıklı kayan nokta işlenenin vektörlerini kopyalayabilir.

SSE talimatları, YMM/MXM kayıtlarında, MMX™ kayıtlarında, GPR kayıtlarında veya bellekte bulunan tamsayı ve kayan noktalı veri türleri arasındaki dönüşümler ve öğe sıralaması veya kesinlik dönüştürmeleri dahil olmak üzere vektör öğelerinin veri dönüştürmesini destekler. Örneğin, paketten çıkarma yönergeleri iki vektör işleneni alır ve düşük veya yüksek öğelerini serpiştirir. Şekil 4-16, sözcük boyutundaki öğeler üzerinde paketten çıkarma ve serpiştirme işlemini göstermektedir. Bu durumda, V)PUNPCKLWD. İşlenen 1, işaretsiz tam sayıların bir vektörüyse ve sol taraftaki kaynak işlenen, değeri sıfır olan öğelere sahipse, işlem, işlenen 1'in alt yarısındaki her öğeyi şuna dönüştürür:

Orijinal genişliğinin iki katı olan bir tamsayı veri türü. Bu, bir vektör çarpma işlemi sırasında taşma olmamasını sağlamak için iki tamsayı vektörünü birlikte çarpmadan önce faydalı bir adım olacaktır.

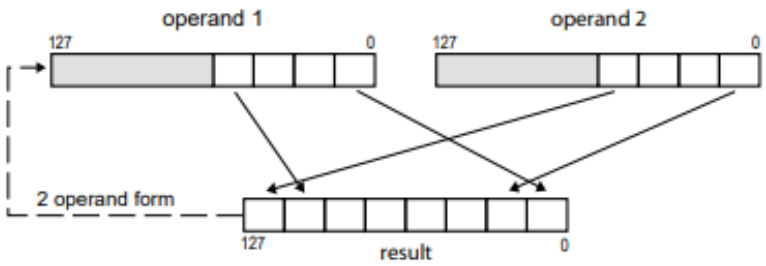


Figure 4-16. Unpack and Interleave Operation

Şekil 7.5.1. [18]

8. INSTRUCTİON SET INTEL VE AMD İÇİN KARŞILAŞTIRILMASI

Intel i9 işlemcilerinde Intel® Gelişmiş Vektör Uzantıları 2, Intel® Gelişmiş Vektör Uzantıları 512, Intel® AVX2 gibi komut setlerini kullanılanlara örnek verebiliriz. AMD Ryzen işlemcilerinde SSE (SSE4a) ve diğer çeşitli setlerden faydalandığını söyleyebiliriz. SSE komut seti ile AVX2 komut setinin amaçları hemen hemen aynıdır, AVX2 SSE komutlarının 256 bitlik geliştirilmiş versiyonudur.

Hem AMD hem de Intel'in sunduğu, ikisi arasında ortak olmayan bazı özel (isteğe bağlı) yönergeler vardır ve hangi programlar gerektiğinde nasıl tanınacağını ve kullanılacağını bilir. Ancak çoğunlukla komut seti aynıdır. Intel ve AMD'nin, x86_64 ISA'nın (komut setinin adı) paylaşılabilmesini ve iki satıcı arasında uyumlu kalmasını sağlayan bir çapraz lisans anlaşması vardır. Yani her işlemci aslında kendi iç komut setiyle farklı çalışır, ancak dış dünyayla uyumludur.

AMD ve Intel'in aslında %100 aynı komut setini kullanmadıklarını bilmek önemlidir: hepsi aynı talimat setlerine sahip olsalar da, farklı şekilde uygulanırlar ve bu nedenle, kullanım için optimize edilmiş uygulamalardır. Intel'in özel talimatları, AMD'nin işlemcisinde daha yavaş çalışacak ve özellikle AMD'nin talimatları için yapılan uygulamalar Intel'de daha yavaş çalışacaktır. Buna bir örnek AVX2 talimatlarıdır; AMD Ryzen işlemcide bir AVX2 talimatını tamamlamak iki saat döngüsü sürerken, en yeni Intel işlemcilerde yalnızca bir saat döngüsü sürer (doğru hatırlıyorsam AVX2 ilk olarak Intel'in Sandy Bridge işlemcilerinde uygulandı) ve AMD yalnızca Ryzen'de uygulanmıştır). Aynı talimatlardır, ancak farklı şekilde uygulanırlar. [19]

9.SONUÇ

Komut setinin neler yapabileceğini ve derleyicinin bu yönergeleri nasıl kullandığını anlamak, geliştiricilerin daha verimli kod yazmasına yardımcı olabilir. Ayrıca hata ayıklama için yararlı olabilecek derleyicinin çıktısını anlamalarına yardımcı olabilir. Geliştiriciler, lisans sahiplerinin kendi özel yönergelerini oluşturmasına izin vererek, özel iş yüklerini hızlandırabilir. Komut seti, Intel tarafından oluşturulan ve AMD tarafından benimsenen 32 bitlik uzantılar ve AMD tarafından oluşturulan ve Intel tarafından benimsenen 64 bitlik uzantılar dahil olmak üzere yıllar içinde birçok kez genişletilmiştir. . Intel'in özel talimatları, AMD'nin işlemcisinde daha yavaş çalışacak ve özellikle AMD'nin talimatları için yapılan uygulamalar Intel'de daha yavaş çalışacaktır. Aynı talimatlardır, ancak farklı şekilde uygulanmaktadırlar.

Bu araştırmada Instruction Setler ve Architectureları detaylıca öğrenildi. Adresleme Modları, Talimat Formatları, ISA özellikleri ve RISC-CISC komut setleri ilk kısımda ele alındı. Sonrasında ise Intel i9 ve AMD Ryzen genel olarak araştırılıp karşılaştırıldıktan sonra instruction setleri ve yapıları öğrenilip setler üzerinden yeniden karşılaştırılıp yukardaki sonuçlara ulaşıldı.

Özetlemek gerekirse Instruction setler ve yapıları hakkında çok detaylı bilgilere sahip olundu.

10. KAYNAKÇA

[1] Internet, https://datacadamia.com/computer/cpu/x86

[2] Intel® Architecture Instruction Set Extensions and Future Features , September 2022

[3] Internet, https://cs.colby.edu/courses/F20/cs232/notes/

[4]Internet,https://www.cs.umd.edu/~meesh/411/CA-online/chapter/instruction-set-architecture/index.html

[5] Internet, http://www.cs.kent.edu/~durand/CS0/Notes/Chapter05/isa.html

[6] Internet, https://www.comp.nus.edu.sg/~adi-yoga/CS2100/ch07/

[7]Internet,https://www.cs.swarthmore.edu/~richardw/classes/cs31/s18/resources/0073380652_Ch2.pdf

[8] Internet, https://www.konsyse.com/articles/intel-core-i9-advantages-and-disadvantages/

[9]Internet,https://medium.com/@infomita/intels-core-i9-12900ks-overclocked-at-7-45ghz-world-records-broken-85c1377ed55

[10] Surakshitha S, Shifa A, Ameen Rekha P M, 2016, A Comparative Study of 64 Bit Microprocessors Apple, Intel and AMD, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) ICACT – 2016 (Volume 4 – Issue 22),

[11]Internet, https://medium.com/intel-tech/12th-gen-intel-core-h-series-processors-redefine-mobile-performance-940c8242ac51

[12] AMD64 Technology AMD64 Architecture Programmer's Manual Volume 1: Application Programming

[13]Internet, https://www.androidauthority.com/amd-cpu-guide-1222438/

[14]Internet,https://www.lenovo.com/us/en/faqs/amd/amdRyzen/?orgRef=https%253A%252F%252Fwww.google.com%252F

[15] Internet, https://www.cs.cmu.edu/~410/doc/intel-isr.pdf

[16]Internet,https://www.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference/top/compiler-reference/intrinsics/intrinsics-for-avx-512-instructions.html

[17]Internet,https://www.intel.com/content/www/us/en/developer/articles/technical/simd-made-easy-with-intel-ispc.html

[18] Internet, https://www.amd.com/system/files/TechDocs/24592.pdf

[19]Internet,https://www.quora.com/Do-Intel-and-AMD-processors-have-the-same-instruction-set