TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SONATA:
ÖZELLEŞTİRİLMİŞ ADRES YAZMAÇLARI İLE ENERJİ TASARRUFU

YÜKSEK LİSANS TEZİ

Esra Nur AYAZ

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. Oğuz ERGİN

Aralık 2021

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

.

Esra Nur AYAZ

İMZA

ÖZET

Yüksek Lisans Tezi

SONATA: Özelleştirilmiş Adres Yazmaçları ile Enerji Tasarrufu

Esra Nur AYAZ

TOBB Ekonomi ve Teknoloji Üniveritesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Oğuz ERGİN

Tarih: Aralık 2021

Modern bilgisayarlar, adres çevrimini gerçekleştirmek için sayfa tabloları kullanmaktadır. Çevrimin yapılabilmesi için 5 bellek erişimi gerekmektedir. Bu sıkça gerçekleşen bellek erişimlerine ciddi bir yük bindirmektedir. Adres çevriminin kapsadığı alan kullanılan sisteme göre değişmekle birlikte genellikle sayfa boyutuyla (4KB) kelime boyutu (8 bayt) arasında büyük bir boyut farkı bulunmaktadır. Sonuç olarak yazmaç öbeğinin ciddi bir kısmı büyük kısmı tekrar edilen bellek adresleriyle dolmaktadır. Dahası, sayfa tablosu için önbellek görevi gören Etkinleştirilmiş Sayfalar Önbelleği (ESÖ), yazmaçtaki sanal adreslerin bir kopyasını zaten tutmaktadır.

Bu çalışmada sayfa tablosu tabanlı adres çevrimi yapan sistemlerde bu tekrar etme durumunu çözerek enerji tasarrufu ve performans artışı sağlamayı amaçladık. Bu nedenle SONATA adını verdiğimiz mikromimari değişikliği öneriyoruz. SONATA bu amaçla yazmaçların bir kısmını ayırıp sadece adresler için kullanır. Adreslerin sayfa numaraları ESÖ'de saklanır. SONATA mikromimaride asgari düzeyde değişiklik yapar ve yazmaç sayısını ile yazmaç öbeğine erişim mekanizmasını değiştirmez. SONATA yazmaç öbeğinin dinamik enerji tüketiminde %2.76'lık, ESÖ'nün enerji tüketiminde %1.69'luk ve AMB'nin enerji tüketiminde %0,61'lik bir tasarruf sağlar. Bunun yanında ESÖ'deki bulma oranını %3.14 artırır.

Anahtar kelimeler: Etkinleştirilmiş Sayfalar Önbelleği, Adres Çevrimi, Sayfa Tablosu, Bellek

ABSTRACT

Master of Science

SONATA:
Specialized Address Registers To Reduce Power Consumption

Esra Nur AYAZ

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Prof. Dr. Oğuz ERGİN

Date: December 2021

Current computing systems use page tables to perform address translation. Modern computers make 5 memory accesses to obtain a translation. This adds a serious amount of overhead to the memory access. The granularity of the address translation is typically defined by the page size employed in the system. In current systems, there is a significant discrepancy between the page size (typically 4 KiB) and the word size (8 bytes in a 64-bit system CPU). As a result, the register file in the processor fills up with partially redundant memory address operands that point to different words in the same virtual page. Furthermore, the Translation Lookaside Buffer (TLB), which is a cache for address translations, contains an additional copy of the same virtual page number in one of its entries.

Our goal is to exploit this redundancy to improve the performance and energy consumption of computing systems that employ page table based address translation. To this end, we introduce SONATA, a microarchitectural modification that proposes to reserve a fraction of registers for offsets and use them as a new register type: Address registers. SONATA stores base addresses belonging to address registers in the TLB. SONATA makes minimal changes to the system and does not change the number of registers or the access mechanism to the overall register file. SONATA

offers a reduction in register file’s dynamic energy consumption by 2.76%, TLB’s energy consumption by 1.69% and ALU energy consumption by 0,61%.

Keywords: TLB, Address Translation, Page Table, Memory

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Oğuz Ergin'e, kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine, çeşitli konularda çok şey öğrendiğim Kasırga Mikroişlemciler Laboratuvarı'ndaki çalışma arkadaşlarıma, tezime konu olan projede kullandığım programlarda yardımcı olan Kerem Arıkan ve Fatma Nisa Bostancı'ya, çözüm geliştirme aşamasında değerli fikir alışverişlerinde bulunduğum Ataberk Olgun'a ve destekleriyle her zaman yanımda olan aileme ve arkadaşlarıma çok teşekkür ederim.

İÇİNDEKİLER

Sayfa

ÖZET iv
ABSTRACT vi
TEŞEKKÜR viii
İÇİNDEKİLER ix
ŞEKİL LİSTESİ xi
ÇİZELGE LİSTESİ xii
KISALTMALAR xiii
SEMBOL LİSTESİ xiv
1. GİRİŞ 1
2. TEMEL BİLGİLER 5
2.1 Sanal Bellek 5
2.2 Adres Çevrimi 5
2.2.1 Sayfa Tablosu, Sayfa Tablosu Yürüyüşü 5
2.2.2 Etkinleştirilmiş Sayfalar Önbelleği 6
2.3 Yeniden Sıralanılmış İşlemci Düzeni 6
3. MOTİVASYON 9
3.1 Sayfa Numaralarının Yazmaç Öbeği'nde Yarattığı Şişkinlik 9
3.2 Sanal Sayfa Numarası Sıkıştırılabilirlik Değerleri 11
3.3 Sanal Sayfa Numaralarının Okuma Erişimi Oranları 12
4. SONATA 15
4.1 Anahtar Fikir 15
4.2 Tasarım 15
4.2.1 Yazmaç Öbeği 16
4.2.2 Etkin Sayfalar Önbelleği (ESÖ) 17
4.2.3 Yükle Sakla Kuyrukları 18
4.2.4 Yeniden Adlandırma Mekanizması 18
4.2.5 Adres Yazmaçları İçin Aritmetik Mantık Birimi 19
4.3 İşleyiş 20
4.3.1 Yükle ve Sakla Kuyrukları ile ESÖ'nün Etkileşimi 20
4.3.2 Adres Yazmacına Yazmak 21
4.3.3 Adres Yazmacından Okumak 22
4.3.4 Adres Yazmaçlarının Bırakılması 23
4.3.5 ESÖ Satırlarının Bırakılması 23
4.3.6 Yazmaç türleri Arası Geçiş 24
4.3.7 Adres Yazmacındaki Adresin Sayfasının Değişmesi 24
5. METODOLOJİ 27
6. DEĞERLENDİRME 29
6.1 Adres Yazmacı Sayısının Başarıma Etkisi 29
6.1 Adres Yazmaçlarının Kullanım Oranları 30
6.2 Enerji Kazanımı 30
6.3 ESÖ Bulma Oranına Etkisi 31
6.4 Donanım Gereksinimi 32

6.5 Güvenlik 32
7. GEÇMİŞ ÇALIŞMALAR 33
7.1 Spekülatif Adres Çevirileri 33
7.2 Yakın Zamanda Kullanılan Çevrimlerin Tekrar Kullanılmak Üzere Saklanması 33
7.3 Etkinleştirilmiş Sayfalar Önbelleği Kapsama Alanının Genişletilmesi 34
7.4 Adres Çevriminin Öne Çekilmesi 35
7.5 Sayfa Adresi ve Sayfa Başlangıcına Uzaklığın Ayrı Saklanması 35
8. GELECEK ÇALIŞMALAR. 37
9. SONUÇ 39
Kaynakça 41
ÖZGEÇMİŞ 47

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1: Yeniden Sıralanmış İşlemci Düzenine Sahip Bir Tasarım 6
Şekil 3.2: Yazmaç Öbeği'ndeki şişmenin sıkıştırılabilen ve sıkıştırılamayan kısımları. Turuncu kısım sıkıştırılabilecekler, sarı kısım sıkıştırılamayacakları göstermektedir. cactuBSSN, xz, gcc, blender kullanılan SPEC2017'nin bir parçası olan iş yükü testleridir. x ekseni her 90 milyar tiki, y ekseni YÖ'deki şişme oranını gösterir..... 9
Şekil 3.3: Veri ve Sanal Sayfa Numarası (SSN) Okuma Sayıları..... 11
Şekil 4.1: Yazmaçta saklanan bir adresin onunla aynı sayfaya düşen diğer adreslerden farklı olan tek kısmı en önemsiz 12 bitidir..... 13
Şekil 4.2: Yazmaçların orijinal tasarımda 64-bit veri saklayabilirken önerilen tasarımda 16 yazmaç 12-bit uzunluğundadır. *n* sayısı Intel Skylake için 180'dir 15
Şekil 4.3: (a) Orijinal ESÖ satırı. Her bir satır sanal ve gerçek sayfa numaralarıyla çeşitli bayrakları saklıyor. (b) Önerilen ESÖ satırı. Her bir satır orijinal tasarımdaki bilgilerin yanı sıra 16 bitlik bir bit eşlemi içeriyor. Bu bit eşlemi o satırdaki sayfa ikilisinin hangi adres yazmaçlarıyla ilişkili olduğunu gösteriyor..... 16
Şekil 4.4: Yeniden Adlandırma Mekanizması'nda Yeniden Adlandırma Tablosu'nda değişiklik olmazken Boş Yazmaç Listesi sayısı her iki yazmaç türü için ayrı ayrı tutulması gerektiğinden ikiye çıkar..... 17
Şekil 4.5: Yükle ve sakla buyruklarının akış şeması. Yükleme ve Saklama kuyrukları ayrılmamış şekilde gösterilmiş olup Yükle/Sakla Kuyruğu (YSK, -*ing.* Load Store Queue) adı altında incelenebilir..... 19
Şekil 4.6: Sanal sayfa numarası ve uzaklıktan oluşan adresin hem sanal hem de gerçek sayfa numaraları ESÖ'de saklanır. Gelen adresin uzaklık kısmı direkt atanan yazmaca yazılır..... 20
Şekil 4.7: Gerçek Adres ESÖ'den alınan Gerçek Sayfa Numarası (GSN) ve adres yazmacında saklanan *uzaklık*tan oluşur..... 21
Şekil 4.8: Ardışık adreslerdeki veriler okunurken bir sayfadan diğerine geçilmesi mümkündür..... 22
Şekil 6.1: Adres yazmacı sayısına göre alandan kazanç ve taban tüm yazmaçların adres yazmacı olması kabul edildiğinde yazmaç sayısına göre ESÖ ıskalama oranı artışı..... 29
Şekil 6.2: Kazanılan alan yüzdesi başına ESÖ ıskalama oranı artışı. Sayıların net bir şekilde görülebilmesi için 1 ve 2 sayıda adres yazmacı için değerler grafikte verilmemiştir. 1 adet adres yazmacı 65.26, 2 adet adres yazmacı 8.98 sonucuna sahiptir..... 30

ÇİZELGE LİSTESİ

Sayfa

Çizelge 3.1: SPEC2017'deki *cactuBSSN*, *xz*, *gcc* ve *blender* iş yükleri için maksimum ve minimum şişme oranları 10
Çizelge 3.2: SPEC2017'deki *cactuBSSN*, *xz*, *gcc* ve *blender* iş yükleri için maksimum ve minimum sıkıştırılabilirlik oranları..... 10
Çizelge 5.1: Simülayonu yapılan mimarinin detayları. S1 1. seviye, L2 2. seviye önbellekleri, -V veri önbelleğini, -B buyruk önbelleğini gösterir..... 25

KISALTMALAR

AMB : Aritmetik Mantık Birimi (Arithmetic Logic Unit)
ESÖ : Etkin Sayfalar Önbelleği (Translation Lookaside Buffer)
GA : Gerçek Adres (Physical Adres)
GSN : Gerçek Sayfa Numarası (Physical Page Number)
İAB : İçeriği Adreslenebilir Bellek (Content Adressable Memory)
MİB : Merkezi İşlem Birimi (Central Processing Unit)
SA : Sanal Adres (Virtual Adres)
SK : Sakla Kuyruğu (Store Queue)
SSN : Sanal Sayfa Numarası (Virtual Page Number)
TST : Tekrar Sıralama Tablosu (Reorder Buffer)
YÖ : Yazmaç Öbeği (Register File)
YK : Yükle Kuyruğu (Load Queue)
YAT : Yeniden Adlandırma Tablosu (Register Alias Table)
YSK : Yükle Sakla Kuyruğu

SEMBOL LİSTESİ

Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklama**
KB	Kilobayt (Dijital veri boyutu birimi)
MB	Megabayt (Dijital veri boyutu birimi)
GB	Gigabayt (Dijital veri boyutu birimi)
Kb	Kilobit (Dijital veri boyutu birimi)
Mb	Megabit (Dijital veri boyutu birimi)
Gb	Gigabit (Dijital veri boyutu birimi)

1. GİRİŞ

Sanal bellek kavramı onlarca yıldır yaygın olarak kullanılmaktadır ve aynı bellek sisteminin birden fazla işlem (*-ing.* process) tarafından kullanılabilmesini sağlar. Günümüz bilgisayarları genellikle sayfalanmış sanal bellek kullanmaktadır. Bu sayfalama tekniğinde, bellek *sayfa* adı verilen önceden belirlenmiş büyüklükteki bölümlere ayrılır. Her bir sanal sayfa bir gerçek sayfaya eşlenmiştir. Sanal sayfaların gerçek sayfalara dönüştürülebilmesi için bellekte sayfa tablosu tutulmaktadır. Nihai gerçek sayfa adresine erişmek *sayfa tablosu yürüyüşü* (*-ing.* page walk) adı verilen çok aşamalı bir süreç içerir. Oldukça masraflı olan sayfa tablosu yürüyüşünden kaçınabilmek amacıyla sıkça kullanılan sanal – gerçek sayfa ikililerini saklamak üzere Etkin Sayfalar Önbelleği (ESÖ, *-ing.* Translation Lookaside Buffer, TLB) kullanılmaktadır. Adres çevrimlerini hızlandırmak amacıyla ESÖ'nün kapasitesini artırmak yoluyla kapsadığı bellek alanını artırmak ilk akla gelen seçenek olsa da masraflı olması sebebiyle uygulanabilir bir seçenek değildir. ESÖ'deki kısıtlama adres çevrim sürecini geliştirmek için farklı yöntemler geliştirilmesini zorunlu kılmaktadır.

Günümüz bilgisayarları ESÖ'nün kapsadığı alanı genişletmek için birtakım teknikler kullanmaktadır. Bu tekniklerden sıkça kullanılan bir tanesi daha büyük boyutta sayfalar kullanmaktır [1-20]. Çoğunlukla kullanılan sayfa boyutu 4KB iken, daha büyük sayfalar 2MB ve 1GB olabilmektedir. Bu durumda, örneğin 2MB boyutunda bir alan için ayrı ayrı 2^9 sayfa adresi değil tek bir sayfa adresi tutulması yeterli olacaktır. Bu sayede her bir ESÖ satırı çok daha geniş bir alanı kapsayacaktır. Başka bir teknik, *Linux buddy allocator*, işlemlere ardışık bellek blokları ayırabilmektedir. Bu iki yaygın uygulama da boştaki ardışık bellekle sınırlıdır ve bellekte şişmeyle sonuçlanabilir [21]. Veri seti büyüklüklerinin geldiği nokta ve büyüme eğilimi dikkate alındığında bu tekniklerin yetersiz kaldığı görülmektedir.

Bellek erişim buyrukları, yükle ve sakla türevleri, yazmaçlarda saklanan adres değerleriyle bellek noktalarına erişmektedir. Birden fazla sayfayı gruplama yöneliminin yükselişiyle erişilen adresler sıklıkla aynı sayfaya düşmektedir [22] [23].

Bu adresin sayfa numarasının birden fazla yazmaçta aynı anda bulunduğuna işaret eder. Bu tekrarlamayı lehimize kullanarak yazmaçların bir kısmını küçültmeyi ve bu yazmaçlarda adreslerin sadece sayfa başlangıç noktasına uzaklıklarını saklamayı önermekteyiz. Bu nedenle SONATA (Storing Offsets and Notable Address Translations, -*tr*. Sayfa Uzaklıklarını ve Kayda Değer Adres Çevrimlerini Saklama) adını verdiğimiz bir çözüm öneriyoruz. SONATA asgari masraf ve mikromimari değişiklikle adres çevrim sürecini etkinleştirir.

SONATA yazmaç öbeğinde, Etkin Sayfalar Önbelleği'nde, yeniden adlandırma mekanizmasında değişiklikler yapar ve enerji tasarrufu amacıyla sisteme 12 bit işlenenlerle işlem yapan bir Aritmetik Mantık Birimi ekler. SONATA kullanan bir sistemde yazmaç öbeğindeki yazmaçlar iki tipe ayrılır: (1) *normal yazmaçlar* ve (2) *adres yazmaçları*. Normal yazmaçlar, değişiklik yapılmamış, orijinal tasarımı taşımayı sürdüren yazmaçlardır. Adres yazmaçları 12-bit boyutunda olacak şekilde değiştirilmiştir. Bu yazmaç türü yalnızca adreslerin sayfa başlangıcına uzaklıklarını saklar. Sayfa numaraları gerekli durumlarda ESÖ'den okunur.

Yaptığımız testler sonucunda SONATA'yla önerilen değişiklikler yapıldığında kayda değer bir enerji kazancı gözlemledik. Öncelikle önerdiğimiz sistemi gem5 benzeticisiyle [24] gerçekleyip SPEC CPU2006 [49] ve SPEC CPU2017 [48] testleriyle uygulamalarını yürüterek başarımlarını gözlemledik ve ESÖ'de, yazmaç öbeğinde ve aritmetik işlemlerde enerji kazanımı sağladık. Bu enerji kazanımı üç ana noktadan sağlanmaktadır: (1) Yazmaç kümesinin genel boyutunun küçülmesi, (2) ESÖ üzerindeki aramaların bir kısmının daha enerji tüketerek şekilde yapılması, (3) 64- yerine 12-bit AMB kullanılması.

Özetle, SONATA üzerine yazılmış olan bu tez aşağıdaki kazanımları sağlar:

(1) Sanal adreslerin varlığı ve yazmaç dosyasındaki desenleri analiz edilip sanal adreslerin ne oranda *sıkıştırılabilir* oldukları gözlemledik. Bu analiz ve gözlemlerimiz sonucunda sanal adreslerin iş yükü karakteristiğine bağlı olarak %10-%40 aralığında sıkıştırılabilir olduğu sonucuna vardık.

(2) Gözlemlerimiz sonucunda SONATA adını verdiğimiz tasarımı öneriyoruz. SONATA sistemin mikromimari düzeydeki parçalarına değişiklikler önerir. SONATA asgari donanımsal masrafla tekrar eden sayfa adreslerini sıkıştırır.

(3) SONATA'yı gem5'te modelleyerek önerimizin uygulanabilirliğini ve sisteme etkisini gösterdik. Önerimizin sistemin enerji etkinliğini artırdığını ortaya koyduk.

2. TEMEL BİLGİLER

2.1 Sanal Bellek

Sanal bellek çok sayıda işlemin aynı gerçek bellek uzayında çalışmasını sağlayan bir bellek eşleştirme tekniğidir. Bu, sayfaları işlemler için ayırarak ve onlara atayarak sağlanır. Bunun için sanal – gerçek sayfa adresi çiftleri önceden yeri belirlenmiş çekirdek uzayında saklanır. Sayfanın işleme ayrılmasından sonra her bir bellek erişimi masraflı bir *sayfa tablosu yürüyüşü* adı verilen adres çevrimi sürecine girer.

2.2 Adres Çevrimi

2.2.1 Sayfa Tablosu, Sayfa Tablosu Yürüyüşü

İstenilen gerçek sayfanın sayfa başlangıç adresi elde edildikten sonra bu sayfa içerisindeki konumlara erişilebilir. Günümüz bilgisayarları genellikle sayfalandırılmış sanal bellek kullanır. Bu teknikte bellek daha önceden belirlenmiş büyüklükteki sayfalara bölünmüştür. Sayfa boyutu genellikle 4KB'tır fakat daha büyük sayfa boyutları da mümkündür. Her bir sanal sayfa bir gerçek sayfaya eşlenmiştir.

Gerçek sayfanın yerine erişmek için sanal sayfaların yerini saklayan sayfa tablosu yürünmelidir. Sayfa yürüyüşü en üst seviye sayfa tablosunda başlar. Çevirilecek adresin en önemli bitleri bu çevrimde kullanılacak sayfa tablosunda hangi satırın kullanılacağını işaret eder. Ulaşılan sayfa tablosu satırı bir sonraki sayfa tablosunun yerini işaret eder. Çevrimi istenilen sayfanın gerçek adresi sonuncu seviyedeki sayfa tablosunda erişildikten sonra elde edilir. 5 seviyeli bir sayfalama tekniği kullanılan bir sistemde bir sayfa adresine erişmek için sırasıyla 5 sayfa tablosu ziyaret edilir. Adres çevrimi için yapılan 5 sayfa tablosu erişimi toplam 6 bellek erişimi gerekir. Bu zaman alıcı ve karmaşık işlemden kaçınabilmek için bazı çevrimlerin ayrı bir yerde önceden depolanması önemlidir.

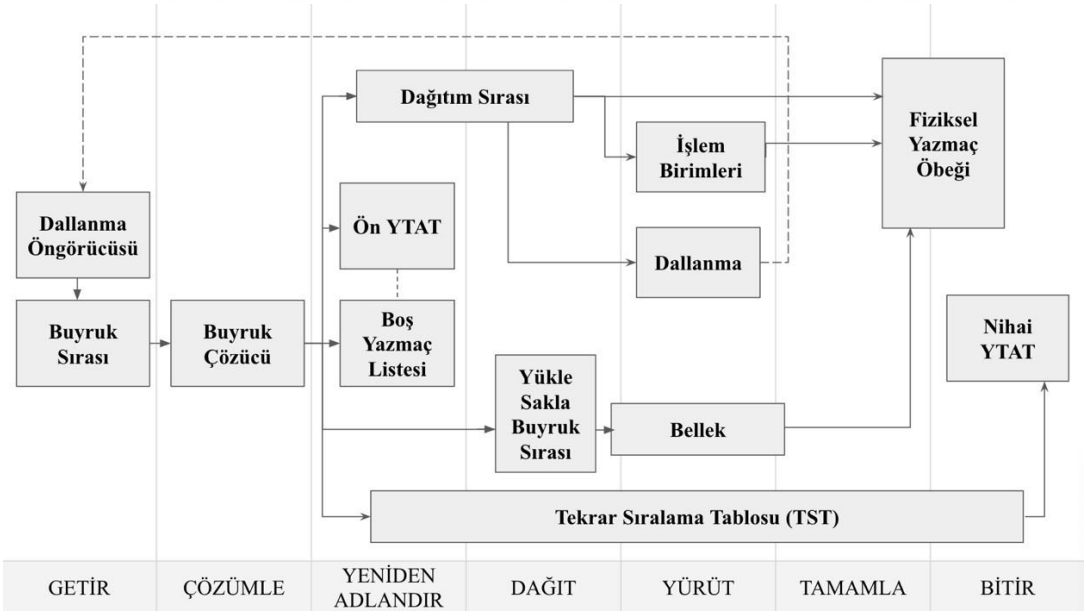
Bellek erişimleri için gereken adres çevrimlerinin zahmetli olmasının yanında bellek erişimleri de sıkça yaşanmaktadır. Merkezi işlem birimi (MİB, -*ing.* Central Processing Unit, CPU) performansını ölçmek için SPEC testleri kullanılmaktadır. SPEC2017'de belleğe erişen yükle ve sakla buyrukları tüm buyrukların %37.68'ini oluşturur [25] [26] [27]. Tüm SPEC testleri içerisinde en düşük bellek erişimi oranına

sahip olan *libquantum* %19.4, en yüksek bellek erişim oranına sahip olan *hmmer* %57 oranında bellek erişim buyruklarından oluşmaktadır.

2.2.2 Etkinleştirilmiş Sayfalar Önbelleği (ESÖ)

Günümüz bilgisayarları oldukça zahmetli olan sayfa tablosu yürüyüşünden kaçınmak için sıkça kullanılan sayfa çevrimlerini Etkinleştirilmiş Sayfalar Önbelleği'nde (*-ing.* Translation Lookaside Buffer) saklar. ESÖ sayfa tablosu için bir tür önbellek görevi görür. Her bellek erişiminde çevirilecek adres öncelikle ESÖ'de aranır. Eğer adres ESÖ'de bulunamazsa, sayfa tablosu yürüyüşü kaçınılmaz bir hale gelir. Sayfa yürüyüşü tamamlandıktan sonra çevrilen sanal – gerçek sayfa numarası ikilisi gelecekte karşılaşılabilecek çevrim isteklerini karşılamak amacıyla ESÖ'ye getirilir ancak ESÖ boyutu kısıtlıdır. Eğer yeni adres çevrimi için uygun bir yer bulunamazsa o anda ESÖ'de bulunan adres çevrimlerden biri ESÖ'den çıkarılır ve yerine yeni sanal – gerçek sayfa adresi ikilisi yerleştirilir. Yeni yeni sanal – gerçek sayfa adresi ikilileri her zaman zaman alıcı sayfa tablosu yürüyüşüyle elde edilir.

2.3 Yeniden Sıralandırılmış İşlemci Düzeni



Şekil 2.1: Yeniden Sıralanmış İşlemci Düzenine Sahip Bir Tasarım [28] [29].

İşlemciler buyrukları geliş sıralarına göre bitirmelidir fakat bu zorunluluk işlemcinin kaynaklarının atıl durumda bırakılmasına sebep olabilir. Bunun önüne geçmek için

işlemciler buyrukları sırasıyla kabul eder ve sırasıyla bitirerek sonuçlarını kaydeder fakat bu iki aşama arasında kalan kısımlarda, yani işlemlerin yapıldığı sırada, buyrukları kaynakları daha etkili kullanmak üzere tekrar düzenleyebilir. Sırası düzenlenen buyruklar, tekrar geliş düzeninde bitirilmelidir.

Bir Buyruğun Yaşam Döngüsü: Buyruklar ilk olarak getirilerek çözümlenir, ne tür bir buyruk olduğu, hangi birimleri kullanacağı tespit edilir. Buyruğa kullanacağı yazmaçlar atandıktan sonra gerekli bilgiler uygun birimlere aktarılır ve işlemler tamamlanır. Tamamlanan buyruklar geliş sırasına göre bitirilir, uygun durumlarda sonuç hedef yazmaca yazılır.

Tekrar Sıralama Tablosu (TST): Tekrar sıralama tablosu buyrukların sonuçlarının tekrar geliş düzeninde yazılmasını sağlar.

Yeniden Adlandırma Tablosu (YAT): Sanal bellek – gerçek bellek ayrımı gibi, yazmaçlar da programcının gördüğünden farklı sayıdadır. Programcı sadece tek bir yazmaç kullandığını düşünürken arkada birden fazla yazmaç kullanılması oldukça muhtemeldir.

Yükle Sakla Kuyruğu (YSK): Yükleme ve saklama buyrukları diğer buyruklardan farklı olarak belleğe eriştiği için kendi içinde sıralamalarının korunması gerekir. Yükle ve sakla kuyrukları bunu sağlar.

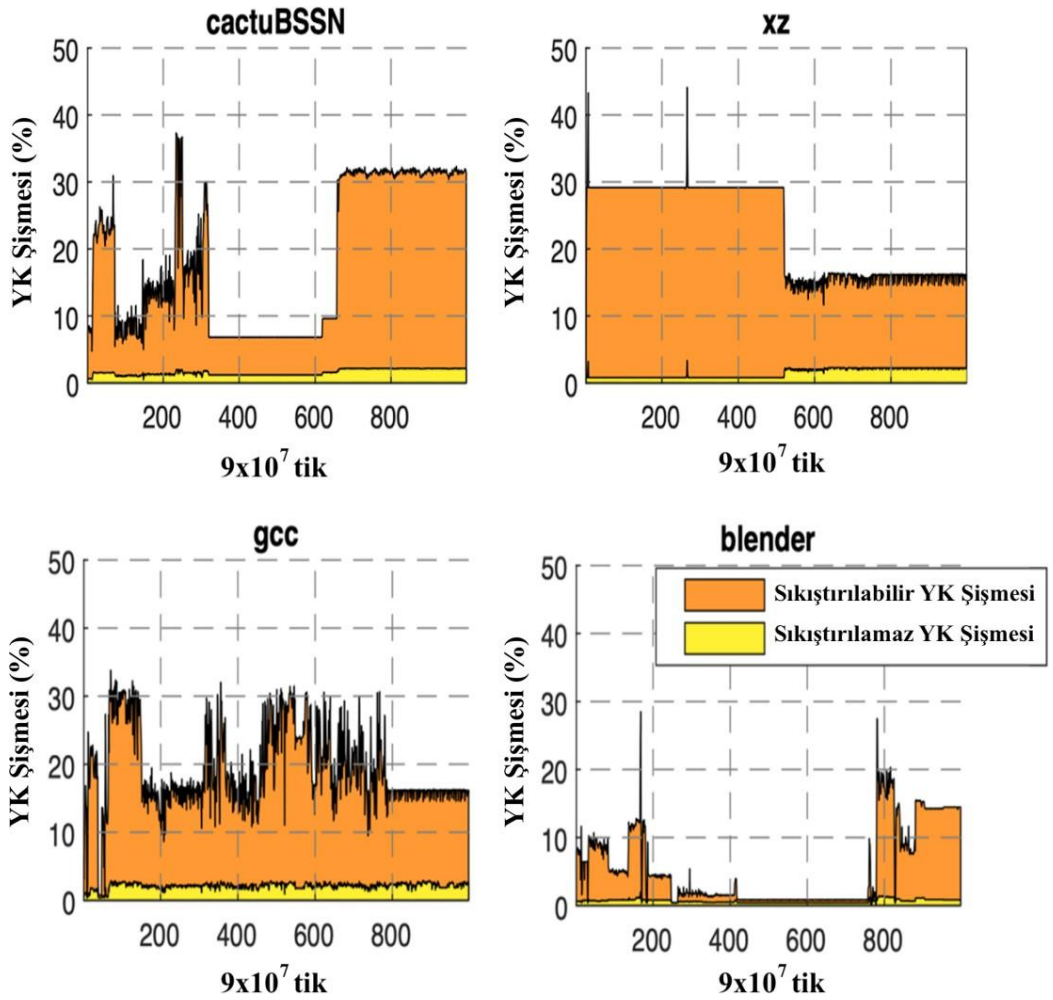
3. MOTİVASYON

Adreslerin uzaklık kısmı için ayrı bir bellek yapısının gerekliliğinden emin olmak için yazmaç öbeğinde tekrar tekrar saklanan sayfa adreslerinin azaltılmasının önemli olduğunu göstermek için sistem üzerinde bazı testler uyguladık. Bu bölümde bu sorunu 3 yönden ele alacağız: (1) Sanal sayfa numaralarının Yazmaç Öbeği'nde oluşturduğu şişme, (2) sanal sayfa numaralarının ne kadar sıkıştırılabilir olduğu, (3) okuma erişimi oranları.

3.1 Sayfa Numaralarının Yazmaç Öbeği'nde Yarattığı Şişkinlik

Günümüz bilgisayarlarında mimari meta bilgi saklamak için gereğinden fazla donanımsal kaynak kullanımı görülmektedir. Yazmaç Öbeği'ndeki (YÖ, *-ing.* Register File) sanal sayfa adreslerinin bir kısmı da bu tanıma girmektedir. Bu durumu iyileştirmek için nelerden faydalanabileceğimizi inceledik.

YÖ'ndeki sanal sayfa numaralarının yarattığı baskıdan kayda değer bir oranda kurtulabilmek için yükle veya sakla buyruklarından gelen yazma isteklerini inceledik. Bu incelemede yalnızca sanal sayfa numaraları kullanıldı. Bu tercihin arkasında YÖ üzerindeki sanal sayfa adresi baskısını açık bir şekilde görme ve gösterme isteği yatmaktadır. İş yüklerinin sıklıkta aynı sayfadaki farklı adreslere eriştiğini göstermek amaçlanmıştır. İstek gönderilen sanal sayfa numaralarını kaydettik ve daha sonra gelen istekler bu listedeki sayfa numaralarıyla karşılaştırılarak daha önceden bu sayfa için istek gönderilip gönderilmediği ile yazmaçlarda ne oranda tekrar eden sayfa numarası bulunduğu araştırdık. Şekil 3.2'de *cactusBSSN, xz, gcc, blender* iş yüklerinin ilk 90 milyar saat vuruşu sırasında ortaya çıkan dikkate değer değişikliklerin grafiği incelenebilir. Bu iş yüklerinin test edilmesi sonucunda YÖ'nin yaklaşık %40'ına kadar sıkıştırılabilir olduğunu keşfettik. Çizelge 3.1'de bahsi geçen iş yükü testleri için maksimum ve minimum şişme oranı gösterilmiştir.



Şekil 3.2: Yazmaç Öbeği'ndeki şişmenin sıkıştırılabilen ve sıkıştırılamayan kısımları. Turuncu kısım sıkıştırılabilecekler, sarı kısım sıkıştırılamayacakları göstermektedir. cactuBSSN, xz, gcc, blender kullanılan SPEC2017'nin bir parçası olan iş yükü testleridir. x ekseni her 90 milyar tiki, y ekseni YÖ'deki şişme oranını gösterir.

Minimum şişme oranları görmezden gelinebilecek düzeyde olmakla birlikte tüm süreç dikkate alındığında YÖ'deki şişme oranının %10-40 aralığında değiştiği söylenebilir. Bu oranlar üzerinde çalışılması gereken ciddi bir soruna işaret etmektedir.

Çizelge 3.1: SPEC2017'deki *cactuBSSN*, *xz*, *gcc* ve *blender* iş yükleri için maksimum ve minimum şişme oranları.

İş yükü	Maksimum Şişme (%)	Minimum Şişme (%)
cactuBSSN	37.32	4.99
xz	29.17	11.63
gcc	33.81	0.78
blender	28.51	0.43

3.2 Sanal Sayfa Numarası Sıkıştırılabilirlik Değerleri

Yazmaç Öbeği'ndeki sanal sayfa adreslerinin ne kadar alan kapladığını ve bu alanın ne kadarının sıkıştırma yapılarak serbest bırakıbileceğini inceleyeceğiz ve bu bölümde elde ettiğimiz değerler anlatılacaktır.

Önceki bölümde sayfa numaralarının YÖ'de yarattığı şişkinliği incelemiştik. Bunun tespiti için sanal sayfa numaralarının kullanıldığı bir liste kullanılmıştı. Bu listenin boyutu bize YÖ'deki muhtemel sıkıştırılabilme oranını bulmamızda da yardımcı oldu. Bu liste yardımıyla YÖ'deki adreslerin ne kadarının sıkıştırılamayacağını anlayabiliriz. Şekil 3.2'de de görüldüğü üzere ciddi bir kısmı sıkıştırılabilir durumdadır. Sıkıştırılanlar aynı değerin tekrar tekrar depolandığı ve bunlardan biri hraiç diğerlerinin kaldırılabileceği durumlardır.

Çizelge 3.2: SPEC2017'deki *cactuBSSN*, *xz*, *gcc* ve *blender* iş yükleri için maksimum ve minimum sıkıştırılabilirlik oranları.

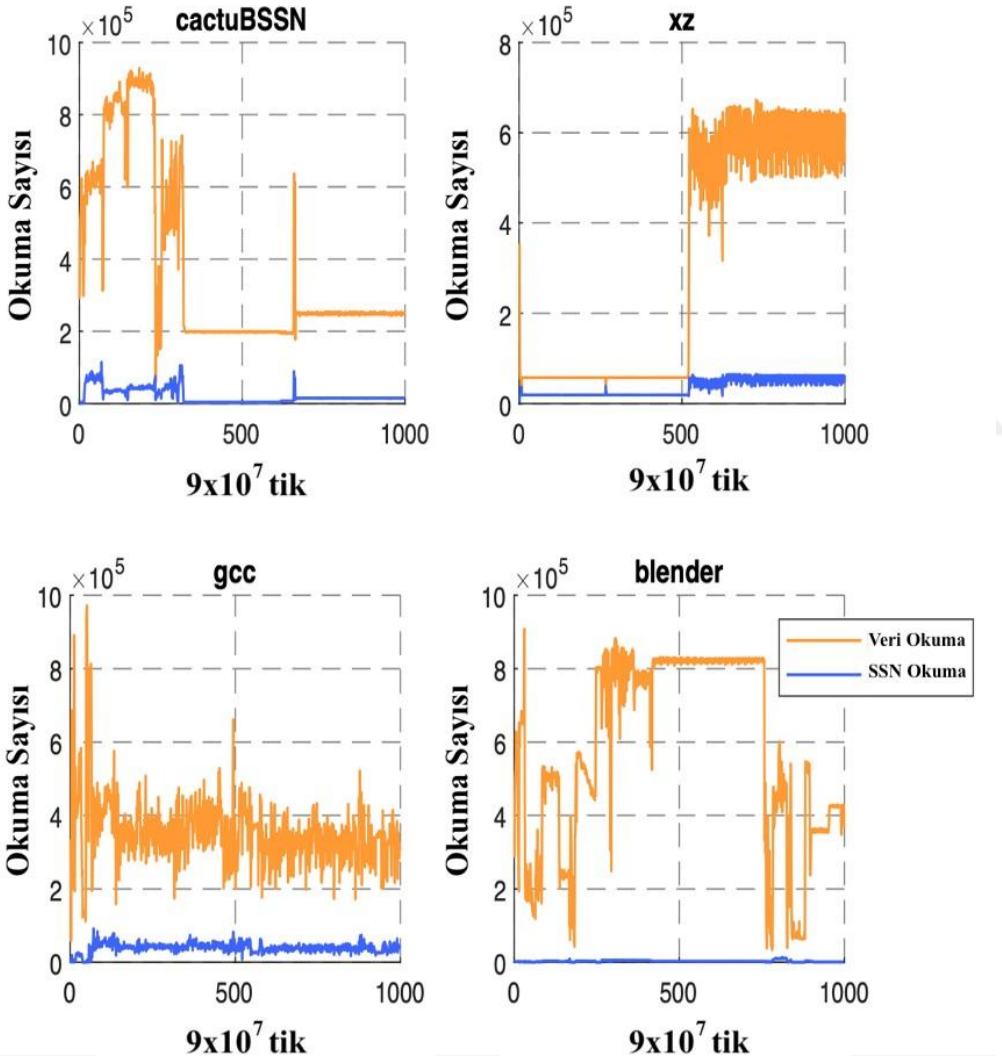
İş yükü	Maksimum Sıkıştırılabilirlik Oranı (%)	Minimum Sıkıştırılabilirlik Oranı (%)
cactuBSSN	37.32	4.99
xz	29.17	11.63
gcc	33.81	0.78
blender	28.51	0.43

Çizelge 3.2'de SPEC2017'deki *cactuBSSN*, *xz*, *gcc* ve *blender* için sıkıştırılabilirlik oranları görülebilir. Burada gösterilen, sanal sayfa numaralarının ne oranda kayıp yaşanmadan kaldırılabileceğidir. *gcc* ve *blender*da sıkıştırılabilirliğin düşük olduğu zamanlar olduğu görülmekle birlikte sanal sayfa numarasının kapladığı alanın gereken

alanın 42.46 katına kadar çıktığı görülebilir. Bu göz önüne alındığında sıklıkla boru hattı gecikmelerinden kaynaklanan ciddi sorunlarla karşılaşabilineceği anlaşılabilir.

3.3 Sanal Sayfa Numaralarının Okuma Erişim Oranları

Çalışma için dayanak oluşturacak sonuncu dikkat çekeceğimiz konu diğer verilere kıyasla sanal sayfa numaralarının okunma sıklığıdır. Bu analizin arkasında yatan sebep okuma/yazma erişimlerinin azaltılma ihtimalidir. Bu azalma güç tasarrufu sağlayacaktır.



Şekil 3.**Error! No text of specified style in document.**.1: Veri ve Sanal Sayfa Numarası (SSN) Okuma Sayıları.

Bazı zaman aralıklarında okuma erişimleri yükselmekte bazı zaman aralıklarında düşmektedir. Okuma erişimi yükseldiğinde sıkıştırılabilirlik oranının yüksek, sıkıştırılamamazlık oranının düşük olduğunu gözlemledik. Şekil 3.3'te *xz* iş yükü testi için 500. veri noktasından sonra okuma sayılarında gözle görünür bir artış gözlemlenmektedir. Bu da sanal sayfa numaralarının sıkıştırılabilirliklerinin arttığı döneme denk gelmektedir. Bu sonuç, sıkıştırmanın okuma sayılarını etkileyebileceğini, azaltabileceğini, göstermesi açısından önemlidir.

4. SONATA

SONATA, merkezi işlem biriminde değişiklik yaparak adres çevrim sürecini daha etkili bir hale getirir. Bu bölümde SONATA'nın arkasındaki ana fikir, tasarım detayları ve işleyişi anlatılacaktır.

4.1 Anahtar fikir

SONATA'nın ele aldığı temel sorun Motivasyon bölümünde de belirtildiği üzere aynı sayfaya düşen birden fazla adresin aynı anda yazmaçlarda saklanmasıdır. Aynı sayfaya düşen adreslerin sayfa başlangıç noktasına olan uzaklığı dışındaki kısmı (sayfanın başlangıcının adresi) aynıdır. Bu, yazmacın 12 biti dışında kalan kısmının başka yazmaçtaki değerlerle aynı olabileceğini gösterir. Nitekim, yine Motivasyon'da belirtildiği üzere, yazmaçların ciddi bir kısmı aynı sayfada bulunan adresleri saklamaktadırlar.

Şekil 4.1: Yazmaçta saklanan bir adresin onunla aynı sayfaya düşen diğer adreslerden farklı olan tek kısmı en önemsiz 12 bitidir **[30]**.

SONATA, oldukça değerli olan yazmaç uzayının kayda değer bir bölümünün aynı değeri defalarca saklamak için kullanılmasının önüne geçer. Bu amaçla adresleri *taban adres* ve *uzaklık* olmak üzere ikiye ayırmayı önermekteyiz. Taban adresler defalarca saklanmak yerine tek bir yerde saklanacak ve gerekli durumlarda uzaklık değeri üzerine eklenerek bellekte istenilen noktaya erişilebilecektir.

4.2 Tasarım

SONATA taban adreslerini saklamak için Etkinleştirilmiş Sayfalar Önbelleği'ni (ESÖ) kullanmayı, uzaklıkları saklamak için özelleştirilmiş adres yazmaçlarını kullanır. ESÖ halihazırda sanal ve gerçek adres çiftlerini saklamaktadır. Bu gerçekten faydalanarak yazmaçlardaki tekrar eden sayfa adresinden kurtularak sadece uzaklıkları

yazmaçlarda saklamamız mümkündür. Gerekli durumlarda nihai adres ESÖ'den alınan taban adresle yazmaçta saklanan uzaklık kullanılarak elde edilebilir.

Bu sistemi hayata geçirmek için tasarlanan sistemde ESÖ, yazmaç öbeği ile yazmaç yeniden isimlendirme ve buyruk dağıtım mekanizmalarında değişiklik yapılmıştır. Bu bölümde yapılan değişiklikler anlatılacaktır.

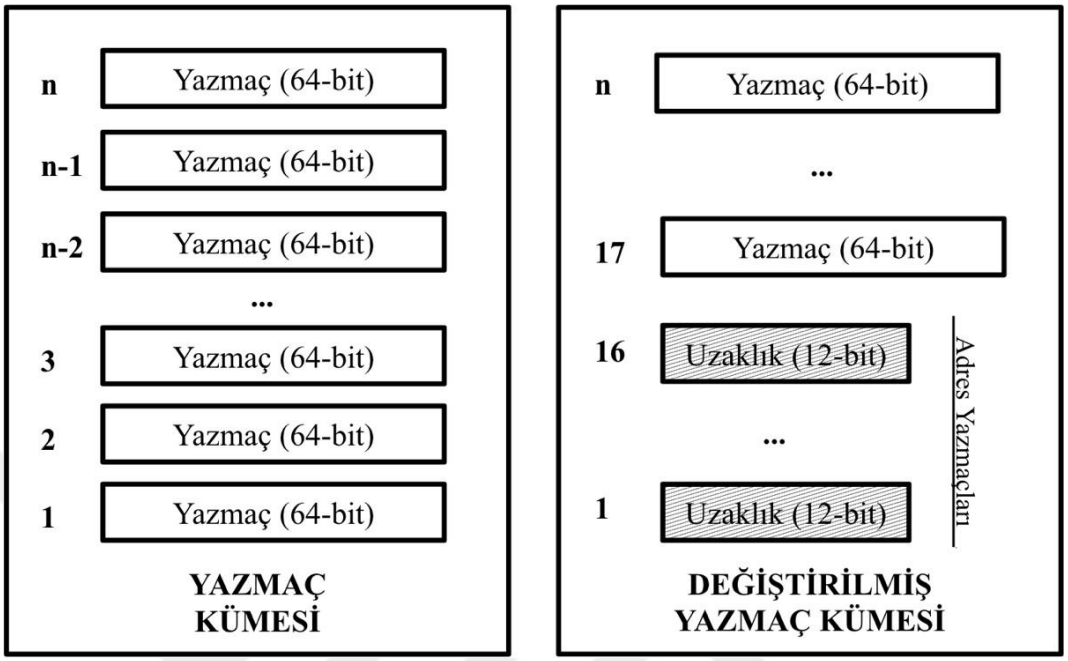
4.2.1 Yazmaç Öbeği

Intel Skylake mimarisinde 180 tam sayı (-*ing.* integer) yazmacı bulunmaktadır **[31]**. Bu yazmaçların her biri 64-bit uzunluğunda veri saklayabilir. Anahtar Fikir bölümünde belirtildiği üzere adreslerin taban adresi (bulunduğu sayfanın başlangıç adresi) sıklıkla birden fazla yazmaçta saklanmaktadır. Yeni tasarımda taban adresler yazmaçlarda değil ESÖ'de tutulmaktadır. Farklılık gösteren uzaklık değerini saklama olmak üzere yeni bir yazmaç türü olan adres yazmacını (-*ing.* address register) kullanmaktayız.

Tasarımda yazmaçlar ikiye ayrılmaktadır: *Normal yazmaç*lar (-*ing.* regular register) ve *adres yazmaç*ları (-*ing.* address register). Normal yazmaçlar üzerinde değişiklik yapılmamış yazmaçlardır. Bu yazmaçlar hem veri hem adres saklamak için kullanılabilir. Kullanımında orijinal yazmaç tasarımından farklı değildir. Adres yazmaçları 64-bit yerine sadece 12-bit uzunluğundadır ve sadece uzaklık değerini saklar. Bu yazmaçlar sadece adres için kullanılabilirler.

Bir adres yazmacı bir buyruğa atandığında adresin üst kısmı, sayfa adresi, ESÖ'de saklanır. Adresin en önemsiz 12 biti ise atandığı adres yazmacında tutulur. Bu yazmaç başına 52 bitlik bir kazanç demektir. Yerden tasarruf ettiğimiz bu 52 bit ise kayda değer bir enerji kazancı elde etmemizi sağlamaktadır. Bu kazancı Enerji Kazancı kısmında daha ayrıntılı inceleyeceğiz. Adreslerin hem sanal hem de gerçek sayfa numaraları hali hazırda ESÖ'de bulunduğu ve her çevrimde ESÖ'ye gitme

mecburiyeti bulunduğu dikkate alındığında sanal sayfa numarasının yazmaçta saklanmamasının bir kayba yol açmayacağı görülecektir.

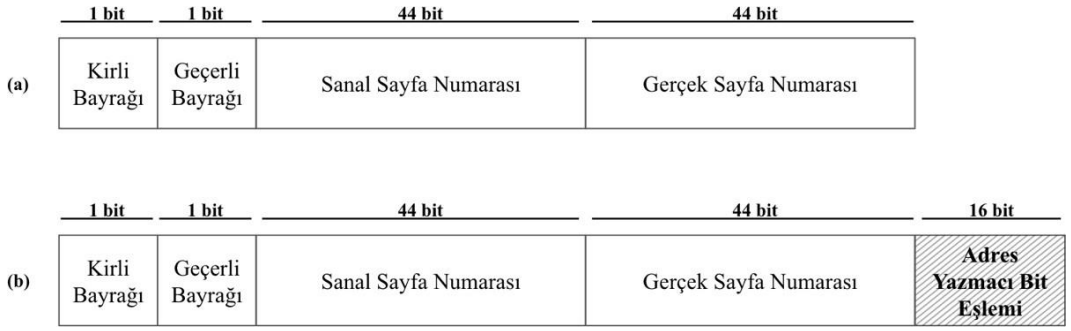


Şekil 4.2: Yazmaçların orijinal tasarımda 64-bit veri saklayabilirken önerilen tasarımda 16 yazmaç 12-bit uzunluğundadır. *n* sayısı Intel Skylake için 180'dir [51][50].

4.2.2 Etkin Sayfalar Önbelleği (ESÖ)

Etkin Sayfalar Önbelleği sıkça kullanılan sanal sayfa numaraları ile gerçek sayfa numara karşılıklarını saklar. Adres çevriminde ilk adım sanal sayfanın ESÖ'de bulunup bulunmadığını kontrol etmektir. Bunun sebebi sayfa tablosunu yürümenin çok uzun vakit almasıdır [32] [33]. Gerçek adrese daha hızlı ulaşmak için sıkça kullanılan adresler MİB'de bulunan ESÖ'de saklanmaktadır. Bu da ana bellekte bulunan sayfa tablosuna kıyasla 200 kat daha hızlıdır [29].

Her bir ESÖ satırı sanal ve gerçek sayfa numaralarıyla *kirli* (-*ing.* dirty), *geçerli* (-*ing.* valid) gibi bayrakları saklar. Bu çalışmada yararımıza kullandığımız ESÖ girdisi elemanları ESÖ'nün temel unsuru olan sanal ve gerçek adreslerdir. Bunların yanında her bir ESÖ satırına 16-bit uzunluğunda bir bit eşlem eklenmiştir. Bu her satırda %15'lik bir alan artışına denk gelmektedir.



Şekil 4.3: (a) Orijinal ESÖ satırı. Her bir satır sanal ve gerçek sayfa numaralarıyla çeşitli bayrakları saklıyor. (b) Önerilen ESÖ satırı. Her bir satır orijinal tasarımdaki bilgilerin yanı sıra 16 bitlik bir bit eşlemi içeriyor. Bu bit eşlemi o satırdaki sayfa ikilisinin hangi adres yazmaçlarıyla ilişkili olduğunu gösteriyor.

Adres yazmaçları için bit eşlem tutmak enerji tüketimi yönünde de bize fayda sağlamaktadır. Temel Bilgiler'deki ESÖ bölümünde anlatığı üzere adres çevrimi sırasında çevrilecek sayfa numarası ESÖ satırlarındaki sanal sayfa numaralarıyla karşılaştırılmakta ve eşleşme durumunda gerçek sayfa numarasına ulaşılabilmektedir. Bu karşılaştırma her ESÖ satırı için 44 bitlik bir karşılaştırma anlamına gelmektedir. Önerilen tasarımda ise bit eşlemindeki ilgili bit ayarlanmışsa aranılan gerçek sayfa numarasının o ESÖ satırında olduğu sonucu ortaya çıkmaktadır. Eğer ilgili adres yazmacı için ayarlanan bit yoksa o çevrimin sonucu ESÖ'de bulunmadığı direkt anlaşılmaktadır. 52 bitlik karşılaştırma yerine bit eşlemde tek bir ayarlanmış bit aramadığımız için burada güç tasarrufu da sağlamış oluyoruz.

4.2.3 Yükle Sakla Kuyrukları

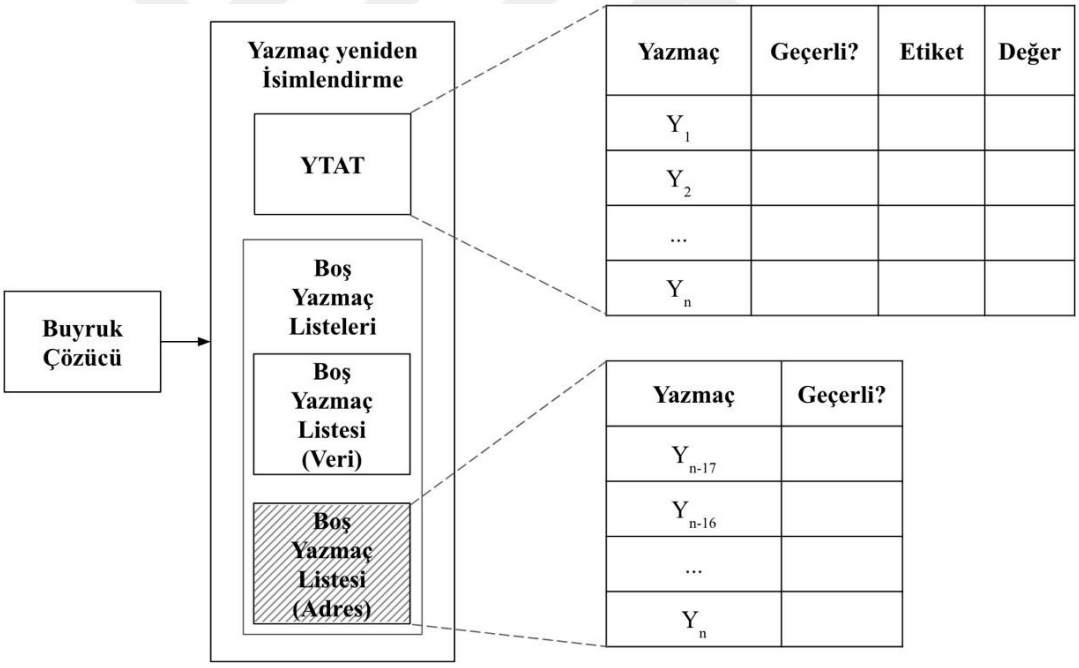
Orijinal tasarımda yükle ve sakla kuyruklarının satırları adres ile gerekli bayrakları saklar. Bu kuyruklar için önerilen mimari bir değişiklik olmamakla birlikte adres çevriminden sonra elde edilen adres ESÖ'de aranırken atanan adres yazmacına uzaklık değeri, ESÖ'deki satıra sayfa adresleri yazılır. Adres ESÖ'de bulunmuyorsa ESÖ'ye getirildikten sonra, bulunuyorsa eşleşme sağlandığında bit eşlemindeki ilgili bit ayarlanır. Bu bit adres yazmacının daha sonraki erişimlerinde çevrimin bulunduğu satıra erişmesini sağlayacaktır.

4.2.4 Yeniden Adlandırma Mekanizması

İşlemcinin kaynaklarını etkili bir şekilde kullabilmek adına buyruklardaki yazmaçlar yeniden isimlendirilmektedir. Yazmaçların yeniden isimlendirilmesinde kontrol

edilecek birkaç unsur bulunmaktadır: (1) Buyruğun sakla ya da yükle olması, (2) uygun adres yazmacı bulunması. Buyruğun sakla ya da yükle olmasının kontrolü buyruğu çözümleme aşamasında yapılıp gerekli durumlarda yükle ve sakla buyruklarından satır ayrıldığından ayrıca bir çaba gerektirmemektedir. Buyruğun uygunluğunun kontrolü sağlandıktan sonra uygun yazmaç varlığı araştırılmalıdır.

Uygun yazmaç arayışı sistemimizde değişiklik gerektirir. Yazmaçlar yeniden adlandırılırken *Boş Yazmaç Listesi* (*-ing.* Free List) ve *Yeniden Adlandırma Tablosu* (YAT, *-ing.* Register Alias Table, RAT) kullanılır. Boş Yazmaç Listesi yazmaçların her birinin kullanımda olup olmadığını tutmak için, Yeniden Adlandırma Tablosu bir mimari yazmaç-gerçek yazmaç eşleştirmelerini tutmak için kullanılır. Yeni bir yazmaç türü eklediğimiz için Boş Yazmaç Listesi'nin ikiye ayrılması ve bir Boş Yazmaç Listesi'nin normal yazmaçlar, diğerinin adres yazmaçları için kullanılması gerekmektedir.



Şekil 4.4: Yeniden Adlandırma Mekanizması'nda Yeniden Adlandırma Tablosu'nda değişiklik olmazken Boş Yazmaç Listesi sayısı her iki yazmaç türü için ayrı ayrı tutulması gerektiğinden ikiye çıkar.

4.2.5 Adres Yazmaçları İçin Aritmetik Mantık Birimi

Adres yazmaçlarındaki uzaklıkların sadece 12-bit olduğu düşünüldüğünde 64-bit işleme kapasitesine sahip bir Aritmetik Mantık Birimi (AMB, *-ing.* Arithmetic Logic Unit, ALU) kullanmanın verimsizliği görülecektir. Bu verimsizliğin önüne geçmek

için adres yazmaçlarına hizmet veren bir 12-bit AMB SONATA tasarımına eklenmiştir. Bellekteki ardışık adreslere aynı yazmaç kullanılarak erişmek sıklıkla rastlanılan bir durumdur. Bu gibi durumlarda 64-bit AMB değil, yeni 12-bit AMB kullanılacaktır. Bu sayede hem performans hem de enerji tasarrufu sağlanacaktır.

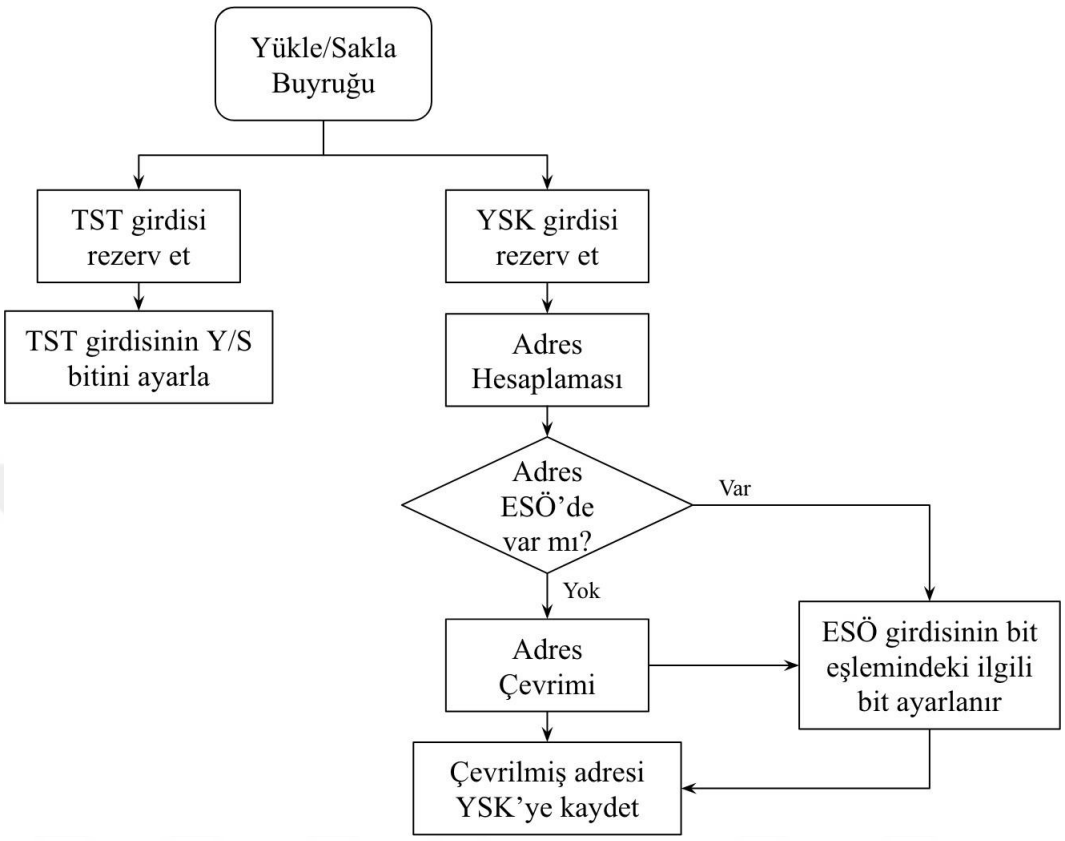
4.3 İşleyiş

Bu bölümde Tasarım bölümünde anlatılan mimari değişikliklerin işleyişleri, işlemcinin çalışmasını nasıl etkiledikleri ve birbirleriyle nasıl etkileştikleri anlatılacaktır. Öncelikle bir buyruğun yaşam döngüsündeki değişiklikleri, daha sonra adres yazmaçlarının farklı durumlarda nasıl bir yol izlenileceğini ve sorun yaratabilecek uç noktaları incelenecektir.

4.3.1 Yükle ve Sakla Kuyrukları ile ESÖ'nün Etkileşimi

Yürütülecek buyruk belirlenen şartları sağlıyorsa SONATA mekanizmasıyla işlenir. Bu şartlar buyruğun yükle ya da sakla buyruğu olması ve uygun bir adres yazmacının bulunmasıdır. Yükleme veya saklama işlemi yapmayan buyruklar önerilen mekanizmadan etkilenmez. Yükleme ya da saklama işlemi yapan bir buyruk olması durumunda Şekil 4.3.1'de de görülebilecek akış izlenir. Gelen buyruk için bir *Tekrar Sıralama Tablosu* girdisi ve yükleme işlemiyse *Yükle Kuyruğu*'nda (YK, -*ing.* Load Queue, LQ) bir girdi, saklama işlemiyse *Saklama Kuyruğu*'nda (SK, -*ing.* Store Queue, SQ) bir girdi ayrılır. TST girdisinin buyruğun yükleme veya saklama yaptığını

işaret eden bayrağını ayarlanır. YSS girdisi ayrıldıktan sonra adres hesaplama ve çevrimi sürecine girilir.

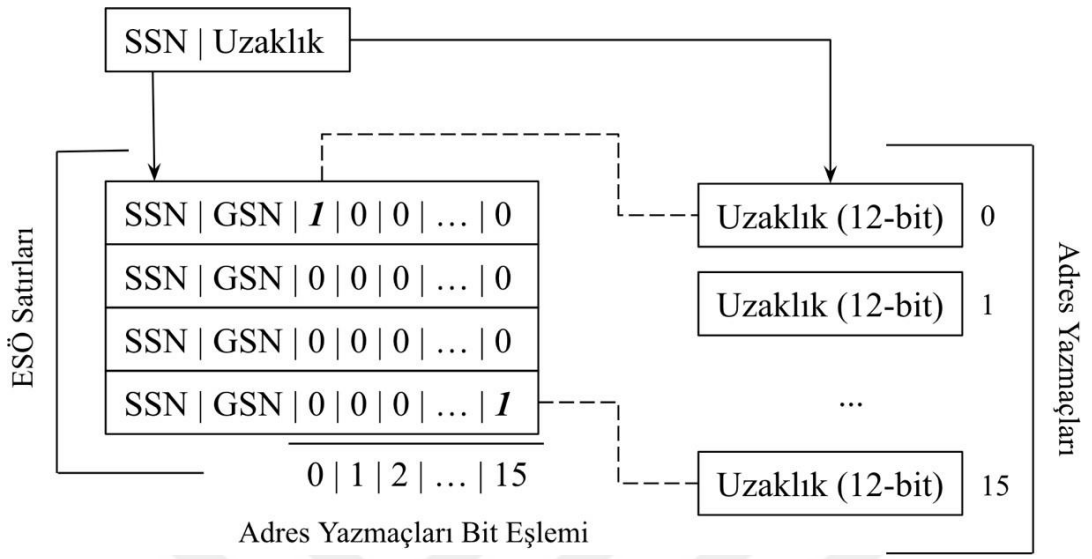


Şekil 4.5: Yükle ve sakla buyruklarının akış şeması. Yükleme ve Saklama kuyrukları ayrılmamış şekilde gösterilmiş olup Yükle/Sakla Kuyruğu (YSK, -*ing.* Load Store Queue) adı altında incelenebilir.

4.3.2 Adres Yazmacına Yazmak

Adres yazmacına ataması yapılan adresin *uzaklık* değeri adres yazmacına, sayfa numarası ESÖ'ye yazılır. Adres yazmacına atanmış adresin kontrol edilmesi gereken tek durum adresin sonraki sayfaya geçip geçmediğidir. Başka bir sayfaya geçen adresin sayfa numarası değişir. Böylece ESÖ'de ilişkili olduğu girdi de değişmelidir.

Bu durum adres yazmacına yazılacak *uzaklık* değerini değiştirmez. Eski sayfa numaraları ve yeni sayfa numaraları için ESÖ'de uygun değişiklikler yapılmalıdır.



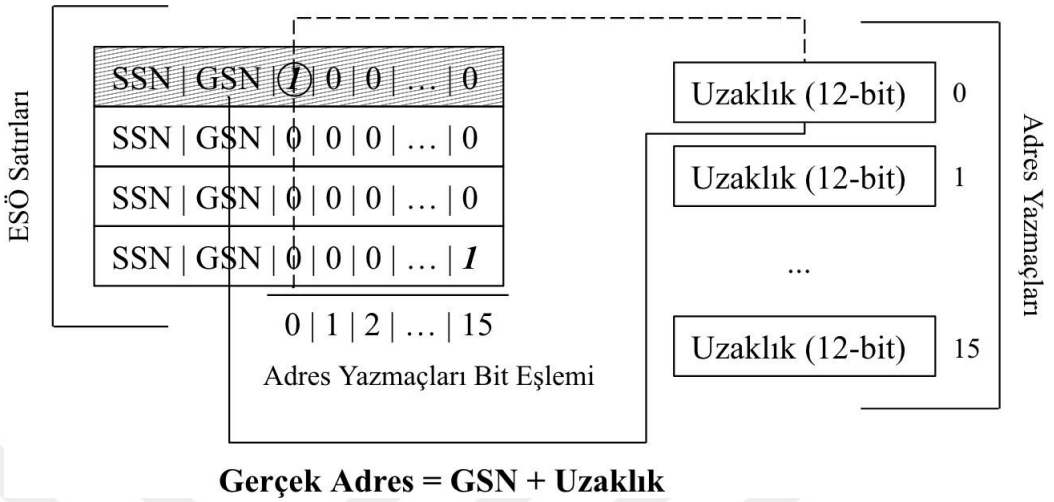
Şekil 4.6: Sanal sayfa numarası ve uzaklıktan oluşan adresin hem sanal hem de gerçek sayfa numaraları ESÖ'de saklanır. Gelen adresin uzaklık kısmı direkt atanan yazmaca yazılır.

4.3.3 Adres Yazmacından Okumak

Bir adresin iki temel unsuru sayfa numarası ve sayfa başlangıç adresine olan uzaklığıdır. Adres yazmaçlarında bulunan uzaklık direkt okunabilir. Gerçek sayfa numarasını okumak ise öncelikle aranılan sayfa numarasının hangi satırda olduğunun belirlenmesini gerektirir. Sayfa numarasına erişmek için her satırda bulunan *Adres Yazmaçları Bit Eşlemi*'nin (AYBE, -*ing.* Address Register Bitmap, ARB) adres yazmacı numarasındaki bitleri bir eşleşme için aranır. *İkilik-1* ile işaretlenmiş satırdan gerçek sayfa numarası okunarak elde edilmek istenen adres ulaşılır.

Adres yazmacında bulunan *uzaklık*ın karşılığı olan bir ESÖ satırı bulunmaması ihtimali yoktur. Bu nedenle her adres yazmacı için nihai adrese ulaşılacağı düşünülebilir. Bu kesinliğin sebebi adreslerin saklanması sırasında hem adres yazmacı hem de ESÖ'nün aynı sırada doldurulmasıdır. Adres yazmacı bırakılmadığı sürece

ESÖ'deki bit eşlemde en az bir bit ayarlanmış olacaktır. Bit eşlemi tamamen sıfırlanmayan ESÖ satırları bırakılmaz.



Şekil 4.7: Gerçek Adres ESÖ'den alınan Gerçek Sayfa Numarası (GSN) ve adres yazmacında saklanan *uzaklık*tan oluşur.

4.3.4 Adres Yazmaçlarının Bırakılması

Adres yazmaçları ilgili buyruk bitirildiğinde (-*ing*. commit) bırakılır. Bu zaman zaman gelen yükle ya da sakla buyrukları için uygun adres yazmacı bulunamamasına sebep olur. Normal yazmaçlar hala hem veri hem de adresler için kullanılabildiğinden bu bir sorun meydana getirmez.

4.3.5 ESÖ Satırlarının Bırakılması

ESÖ satırları sınırlı sayıda olduğu için yeni gelen isteklere yer açmak için en az faydalı görülen satırlar bırakılarak üzerine yeni adres çevrimleri yazılır. ESÖ bırakma mekanizmasında çarpıcı bir değişiklik gerektirmemekle birlikte bir satır bırakılmadan bir satırın Adres Yazmaçları Bit Eşlemi'nin tamamen sıfırlanmış olup olmadığı kontrol edilmelidir. Sıfırlanmamış bit olması o satırı kullanan bir adres yazmacı olduğunu gösterir. Bu durum sadece adres çevrimini kaybettiğimiz için sayfa tablosunu tekrar yürümemiz gerekliliğini değil daha önemli bir sorun olan adresi kaybetmemizi önler. Bitirilmemiş buyruğun kullandığı adresin sayfa numarasını

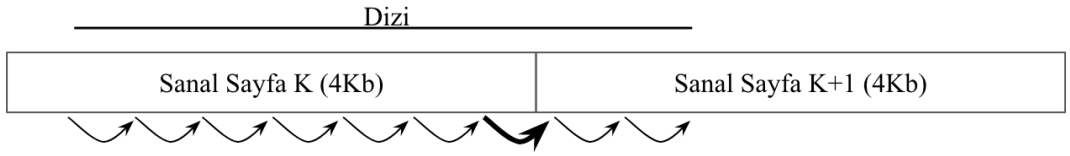
taşıyan ESÖ satırının bırakılması durumunda ilgili adresin hem gerçek hem de sanal adresi kaybedilir.

4.3.6 Yazmaç Türleri Arası Geçiş

Önerdiğimiz sistemde bulunan adres yazmaçları ve normal yazmaçlar arasında geçiş yoktur. Normal yazmaçlar hem veri hem de adresler için kullanılabilir. Adres yazmaçları sadece adres kullanımında atanır fakat üzerinde tıpkı verilerde olduğu gibi işlem yapılabilir. İşlem yapmak için 12-bit Aritmetik Mantık Birimi kullanılır. Elde edilen sonucun en önemsiz 12 biti adres yazmacına yazılır.

4.3.7 Adres Yazmacındaki Adresin Sayfasının Değişmesi

Diziler programlar tarafından yaygın bir biçimde kullanılmaktadır. Diziler bellekte ardışık konumlarda saklanmaktadır. Bu sebeple dizi elemanlarına erişirken sıklıkla dizinin başlangıç değeri artırılarak ilgili elemana ulaşılır. Bu tür art arda bellek konumlarına erişen erişim örüntülerinin, adreslerin bir sayfadan diğerine geçmesine sebep olması mümkündür. Sanal sayfa numarası değiştiğinde gerçek sayfa numarası da deişecek ve adres yazmacının bağlantılı olduğu ESÖ satırını o adres için manasız hale getirecektir. Hatta çevrimin bulunmamasının ötesinde, çevrim var kabul edildiği için yeni çevrimi yapmak da engellenmiş olacaktır.



Şekil 4.8: Ardışık adreslerdeki veriler okunurken bir sayfadan diğerine geçilmesi mümkündür.

Bu sorunun önüne geçmek için kullanılan 12-bit Aritmetik Mantık Birimi'nin (AMB, -*ing.* Arithmethic Logic Unit, ALU) taşma bayrağı (-*ing.* overflow flag) kontrol edilmelidir. 12-bit AMB sadece adres yazmaçlarına hizmet ettiği için taşmış olarak bayraklanan bir işlem sayfalar arası atlamayı işaret eder.

Sayfalar arası atlamanın tespiti durumunda öncelikle önceki sanal sayfa – gerçek sayfa çiftini tutan ESÖ satırındaki bit eşlemdeki ilgili bit sıfırlanmalı, sonrasında adres çevrimi için yeni sanal sayfa ESÖ satırlarında aranmalıdır. Bu aşamada ESÖ'deki işlemler yeni atanmış bir adres yazmacı sonrasında gerçekleşenlerle aynıdır. Sanal

sayfa numarası ESÖ'de bulunursa bu satırdaki bit eşlemin ilgili biti ayarlanır. Bulunamazsa sayfa numarası Sayfa Tablosu'nu yürüyerek çevrilir. Çevrim sonucu ESÖ'ye yazılır ve bit eşlemindeki ilgili bit ayarlanır. Adres yazmacı tarafından artık kullanılmayan ESÖ satırı bırakılmaz fakat bit eşleminde ayarlanmış biti bulunmayan satırlar bırakılabileceğinden satırın bırakılma yolu açılmış olur. Bu satır ESÖ'nün çıkarma politikası doğrultusunda bırakılabilir.

5. METODOLOJİ

Sistemi simüle ve test etmek için gem5 simülatörü kullanılmıştır [24]. Motivasyon bölümünde SPEC CPU2017 [48] iş yükleri 90 milyar saat vuruşu boyunca çalıştırılmıştır. Baz alınmak üzere çalıştırılan MİB gem5'in yeniden sıralanmış MİB'idir. Şişme oranı ve okuma oranı için sisteme yeni istatistiki ögeler eklenmiştir. Sistemin işleyişi bunlar dışında değiştirilmemiş, sabit tutulmuştur.

SONATA'nın performansnı ölçmek yine gem5 kullanılmış ve gem5'in yeniden sıralanmış MİB'ine SONATA'nın önerdiği değişiklikler uygulanmıştır. SONATA kullanan sistemden elde edilen sonuçlar baz sonuçlarla karşılaştırılmış ve performans gelişimi hesaplanmıştır.

Sistem yeniden sıralanmış yürütme özelliğine sahip iki seviyeli önbelleği bulunan bir MİB kullanarak çalıştırıldı. Simülasyondaki mimari detaylar Çizelge 5.1'den incelenebilir. Belirtilmeyen değerler gem5'in varsayılan değerleridir. Motivasyon kısmındaki simülasyonlar SPEC CPU2017 [48], performans değerlendirmeleri SPEC2006 [49] CPU iş yükleri ile test edilmiştir. Performans değerlendirmelerinde ESÖ'ye erişim sayıları, normal ve adres yazmaçlarının okuma ve yazma oranları değerlendirilmiştir.

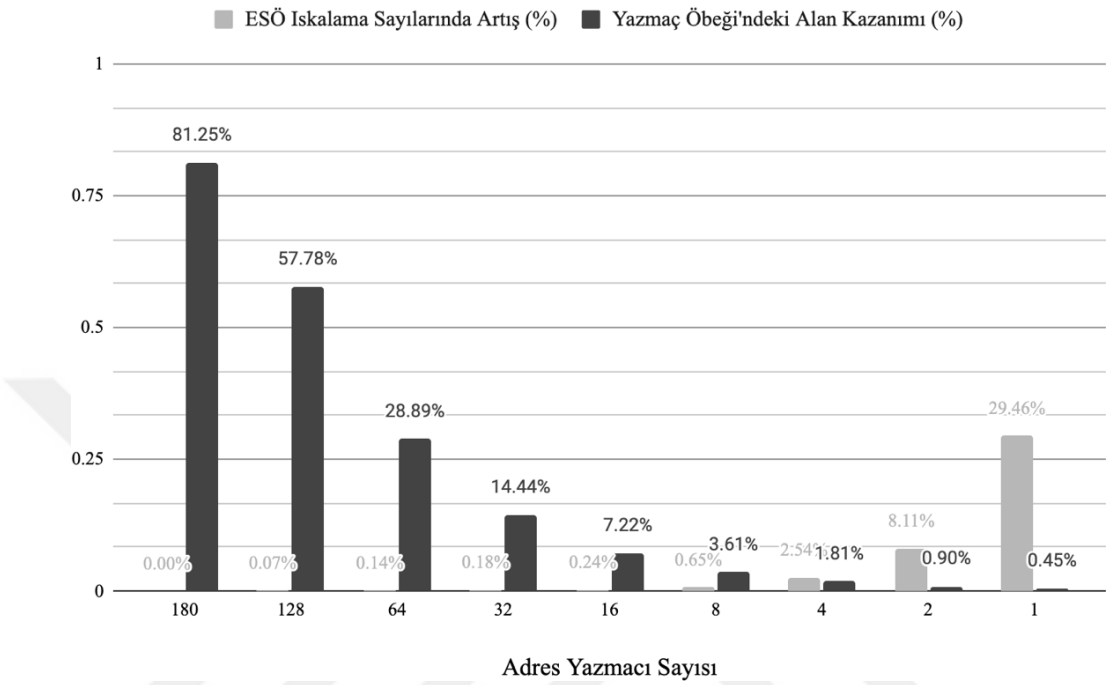
Çizelge 5.1: Simülayonu yapılan mimarinin detayları. S1 1. seviye, L2 2. seviye önbellekleri, -V veri önbelleğini, -B buyruk önbelleğini gösterir.

Eleman	Maksimum Şişme (%)
MİB	x86 Buyruk Kümesi Mimarisi, 5 Aşamalı Boru Hattı, Yeniden Sıralanmış Yürütme, Tek Çekirdek, 256 Integer Yazmaç Öbeği
S1-V Önbelleği	32KiB Boyut 8 İlişkililik
S1-B Önbelleği	32KiB Boyut 8 İlişkililik
S2 Birleşik Önbellek	256KiB Boyut 4 İlişkililik
Ana Bellek Boyutu	4GB

Enerji tasarrufunu değerlendirmek için CACTI [34] ve teorik hesaplamalar kullandık.

6. DEĞERLENDİRME

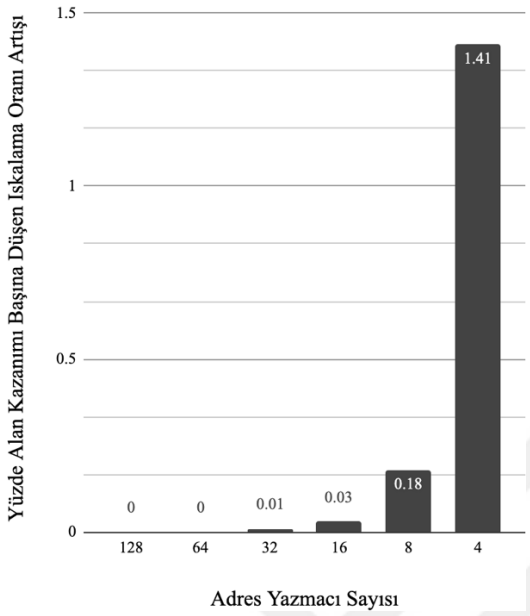
6.1 Adres Yazmacı Sayısının Başarıma Etkisi



Şekil 6.1: Adres yazmacı sayısına göre alandan kazanç ve taban tüm yazmaçların adres yazmacı olması kabul edildiğinde yazmaç sayısına göre ESÖ ıskalama oranı artışı.

SONATA'nın ana kazanımlarından olan enerji tasarrufunun en önemli parçası Yazmaç Öbeği'nin küçültülmesidir. Bu nedenle adres yazmacı sayısı belirlenirken dikkate alınan faktörlerden biri budur. Dikkate alınan bir diğer faktör ise ESÖ ıskalama ve bulma oranlarına yapılan katkıdır. Bu amaçla alan kazanımı başına düşen ıskalama oranı artışı tespit edilmiştir. Bu değerler Şekil 6.1'de görülebilir. Adres yazmaçlarının sayısının başarıma etkisini tespit etmek için tüm yazmaçların adres yazmacı olduğu durum taban kabul edilmiş olup ıskalama oranları buna göre artışı belirtmektedir. Elde edilen oranlar Şekil 6.2'de görülebilir. Görüldüğü üzere

yazmaç sayısı 16'yı geçtikten sonra ciddi bir artış olmaktadır. Bu oranı düşük tutmak amacıyla adres yazmacı sayısı 16 seçilmiştir.



Şekil 6.2: Kazanılan alan yüzdesi başına ESÖ ıskalama oranı artışı. Sayıların net bir şekilde görülebilmesi için 1 ve 2 sayıda adres yazmacı için değerler grafikte verilmemiştir. 1 adet adres yazmacı 65.26, 2 adet adres yazmacı 8.98 sonucuna sahiptir.

6.2 Adres Yazmaçlarının Kullanım Oranları

Adres yazmaçları atandıktan sonra tekrar kullanım oranları 2. kez kullanım için %96.61, 3. Kez kullanım için %89.28, 7. Kez kullanım için %73.87, 15. kez kullanım için %57.23'tür. Adres yazmacına atanan bir adres ortalama 10.41 kez aynı sayfa numarasını kullanarak çevrilmektedir. ESÖ'ye gelen isteklerin %51.28'i adres yazmaçlarından gelmektedir. İsteklerin %68.86'sı ise adres yazmaçlarıyla alakalıdır, yani ya adres yazmacından gelmekte ya da adres yazmacı ilk atandığında gelmektedir.

6.3 Enerji Kazanımı

SONATA'nın 3 yönden enerjiden tasarruf eder: (1) Adres yazmaç öbeğinin küçültülmesi, (2) ESÖ'deki aramaların 52-bitle değil tek bitle yapılması, (3) adreslerin

64-bit değil 12-bit AMB kullanması. Bu bölümde bu enerji tasarruflarını inceleyeceğiz.

Adres yazmaç öbeğinin küçültülmesi. Adres yazmaç öbeği için Intel Skylake [51] mimarisindeki tam sayı yazmaç öbeğini taban kabul ettik. Bu nedenle orijinal tasarımda 180 yazmaç olduğu kabul edildi. Bu yazmaçlardan 16'sı artık 64-bit değil 12-bit olduğundan alanda %7.22'lik bir küçülme olmuştur. Bunun dinamik enerji kullanımına etkisini CACTI [34] ile ölçtüğümüzde %2.76'lık bir enerji tasarrufu sağladık.

ESÖ'deki aramaların 52-bitle değil tek bitle yapılması. ESÖ'deki aramaların adres yazmaçlarından gelenleri sadece her ESÖ satırında bulunan adres yazmaçlarıyla ilişkili sanal – gerçek sayfa numarası çifti olup olmadığını tutan bit eşleminde aranır. Bir adres, adres yazmacını kullanıyorsa adresin ESÖ'de bulunduğu kesindir. Bu sebeple adres yazmaçlarından ESÖ'ye gelen aramalardan enerji kazancı sağlanmaktadır. Yükleme saklama buyrukları eğer adres yazmacını kullanmayacaksa ESÖ'de normal arama yapar. Adres yazmacı yeni atanmış bir yükle ya da sakla buyruğunun ESÖ'de yaptığı aramada yine bir değişiklik yoktur. Buradaki kazanç adres yazmacı atanan değerin ikinci ve sonraki kullanımlarından gelmektedir. Bu enerji tasarrufu ise %1.69'dur [35].

Adreslerin 64-bit değil 12-bit AMB kullanması. 12-bitlik AMB kullanımı %6.3 ile %12.9 arasında bir güç tasarrufu sağlamaktadır [36]. Adres yazmaçları üzerindeki işlemler AMB'de yapılan işlemlerin toplam %4.79'udur. Buradan AMB'nin toplam güç tüketimi %0.30 ile %0.61 azaltılmıştır. 12-bit AMB eklemenin olası kullanım alanları arasında adres işlemleri dışında diğer 12-bit ya da daha kısa işlemler için kullanmak vardır. AMB'nin yaptığı toplama işlemlerinin %94.1'inin bu kapsama gireceği düşünülürse bu eklemenin faydalı olacağı açıktır.

6.4 ESÖ Bulma Oranına Etkisi

Adres yazmacındaki adreslerin geçerli durumda bulunması aynı zamanda ESÖ satırlarında bulunan bit eşlemlerdeki ilgili bitleri de ayarladığından ESÖ satırlarının bırakılmasına engel olabilir. Bu sebeple bazı çevrimlerin en geç kullanıldığı halde

çıkarılmaması mümkündür. Buna karşın adres yazmacı henüz bırakılmadıysa bu adresin tekrar kullanılması muhtemeldir ve ESÖ'den çıkarılmaması bulma oranının eskiye kıyasla %3.14 yükselmesini sağlamaktadır.

6.5 Donanım Gereksinimi

SONATA ESÖ her bir ESÖ satırını 16 bit genişletir. 128 satırı olan Intel [51] mimarisi için hesaplama yaptığımızda 256 baytlık bir ek alana ihtiyaç duyduğumuz ortaya çıkmaktadır. Bunun dışında adresler üzerinde işlem yapmak için 12-bitlik bir AMB gereklidir. Önceki bölümlerde anlatıldığı üzere yapılan her eklemenin güç tasarrufu sağladığı düşünüldüğünde SONATA'nın masraflarının düşük olduğu söylenebilir.

6.6 Güvenlik

SONATA spekülatif bir bellek erişiminde bulunmadığı için Spectre ve Meltdown'a dayanıklıdır. Buna karşın adres yazmaçları kullanıldığı durumda güç tüketimi gözlemlenerek bazı sonuçlar çıkarılması ihtimali vardır. Her adres normal yazmaçta ya da adres yazmacında olmadığı için güç tüketimi farklıdır.

7. GEÇMİŞ ÇALIŞMALAR

Sanal bellek ve adres çevrimi, sisteme etkisinin büyüklüğü sebebiyle popüler bir araştırma konularıdır. Adres çevrimini geliştirmek için iki temel yönelim bulunmaktadır: (1) Adres çevrimini hızlandırarak sistemin bu süreçte daha az vakit kaybetmesini sağlamak ve (2) daha az adres çevrimi gerektirecek çözümler sunarak adres çevriminin masrafının sistemi daha az etkilemesini sağlamak [37]. Bu bölümde sözü geçen çalışmalar da çoğunlukla bu iki kategoriden birine düşmektedir.

7.1 Spekülatif Adres Çevrimleri

Spekülatif adres çevrimi kullanan uygulamalar sanal – gerçek adres çiftlerinin tahmin edilebilirliğini kullanmaktadır [38] [39] [40]. Chiueh ve Katz'ın çalışmalarında tekrar eden adresler sorununu tespit etmişlerdir [39]. Buna çözüm olarak sanal adresleri hem yazmaçlarda hem de *base address cache* adını verdikleri bir yapıda tutmaktadırlar. ESÖ adres çiftlerinin asıl saklama noktası olan sayfa tablosunun önbelleğiyken, *base address cache* de ESÖ'nün önbelleği gibi davranır. Yazmaçlar *base register ID* içerecek şekilde genişletilir. *Base register ID*ler ilgili yazmaçtaki adresin *base address cache*in hangi satırında sayfa numarası bulunduğunu tutar.

Bir diğer çalışma olan SPECTLB [38] halihazırda ESÖ'de bulunan sanal – gerçek adres çiftlerini kullanarak yeni çevrim tahminlerinde bulunur. Tahmin edilen çevrimler henüz adres çevrimi tamamlanmamışken tahmin doğrultusunda belleğe erişip verinin bellekten gelme süresini kısaltmayı amaçlar. Kesinleşmiş adres çevirisini değil tahminleri kullanmak bellekte erişilmesi istenmeyen konumlara erişilmesine sebep olabilir. Hassas noktalara izinsiz erişilmesi sistemi *Spectre* [41] ve *Meltdown*'a [42] karşı savunmasız bırakır.

7.2 Yakın Zamanda Kullanılan Çevrimlerin Tekrar Kullanılmak Üzere Saklanması

Geçmiş çalışmalar en son kullanılan sayfanın bir sonraki çevrimde dekullanılmasının çok olası olduğunu göstermektedir [23] [22]. Kadayıf, Sivasubramaniam, Kandemir,

Kandiraju ve Chen çalışmalarında en güncel sanal – gerçek sayfa çiftinin bir yazmaçta saklanmasını önermişlerdir [22]. Bu yazmaç ESÖ'de arama yapılmadan önce kontrol edilir. Bu kontrol mekanizması ESÖ'ye olan erişim sayısı ciddi miktarda azaltmıştır. Bu kontroller hem donanım hem de yazılım seviyesinde yapılabilir. Buradaki performans kısıtı erişilen adreslerin art arda olmaması durumunda yaşanmaktadır.

Bir diğer çalışma olan Jeyapaul ve Shrivastava'nın araştırmasında art arda bellek konumlarına arta arda erişimlerden faydalanabilineceği ortaya koyulmuştur [43]. Bu çalışma her ne kadar aynı sayfa içerisindeki adreslerde etkili olsa da bir sonraki sayfaya geçen durumlarda etkisiz kalmaktadır. Bu çalışmada yazılım seviyesinde döngü açma (*-ing* loop unrolling), buyruk programlama (*-ing* instruction scheduling), dizi birleştirme (*-ing.* array interleaving) gibi teknikler uygulanmıştır. Dizileri birleştirmek iki diziye sürekli art arda erişen C[i] = A[i] + B[i] gibi bir kodlamaya sahip döngülerde A ve B dizilerinin bulunduğu sayfalar arasında sürekli git-gel yapılmasının önüne geçer.

7.3 Etkinleştirilmiş Sayfalar Önbelleği Kapsama Alanının Artırılması

Adres çevrimini geliştirmek amacıyla ESÖ kapsamının genişletilmesi ilk akla gelen çözümlerden birisidir ve bu sebeple çok sayıda araştırmaya konu olmuştur. Çeşitli şekillerde kullanılan bir çözüm birden fazla sayfa boyutunun kullanılmasına izin vermektir. Tailored Page Sizes (TPS) [20] varsayılan minimum sayfa boyutundan büyük, ikinin katı olan her boyuttaki sayfayı kullanabilmeyi sağlar. Daha büyük boyuttaki sayfalar ESÖ'nün kapsadığı alanı genişletir çünkü artık ESÖ bir satırda minimum boyuttaki sayfa büyüklüğünde bir bellek alanına değil daha büyük bir alana hizmet etmektedir. Daha büyük sayfaların önündeki en büyük engellerden biri belleğin parçalı bir şekilde kullanımda olması ve istenilen büyüklükte sayfayı meydana getirecek bir bellek alanı bulunmamasıdır.

Perforated Page [19] belleğin parçalı olduğu durumlarda dahi daha büyük sayfaların kullanılmasına imkan sağlar. Perforated Page, bellekteki *delik*leri bir listeye kaydeder. Böylece daha büyük sayfalar yeterince büyük bellek alanı bulunmayan durumlarda da

kullanılabilir fakat daha büyük sayfalar kullanmak aynı sayfadan adresler tutan yazmaç sayısını azaltmadığı gibi artırır ve sorunu daha da derinleştirir.

7.4 Adres Çevriminin Öne Çekilmesi

Adres çevriminin öne çekilmesi (-*ing.* translation prefetching) yakın gelecekte kullanılacağı tahmin edilen adreslerin çevrilmesidir. Bu sayede adresin gerçekten çevrilmesi gerekirse çevrim sonucu için gereken bekleme süreci kısaltılmış olur. Kullanılacağı tahmin edilen çevrim ESÖ'de ya da özelleştirilmiş bir Öne Çekilmiş Çevrimler Arabelleği'nde (-*ing.* TLB prefetch buffer) saklanabilir. Saulsbury, Dahlgren ve Senström çalışmalarında ESÖ'den yakın zamanda çıkarılan çevrimlerin kullanılmalarının rastgele bir çevrime kıyasla daha olası olduğunu ortaya koymuşlardır [44].

ASAP [45] sayfa tablolarına paralel bir şekilde erişen bir öne çekici önerir. Böylece bir sayfa tablosuna erişmek için bir öncekine erişimin bitmesinin beklenmesi gerekmez. Bu yaklaşım sayfa tablosunu yürümenin normaldeki 5 erişimde değil, 1 erişimde tamamlanmasını sağlar. Önerilen öne çekici çevrimin ESÖ'de ıskalandığı ve sayfa yürümesinin kaçınılmaz olduğu durumlarda çalışır.

7.5 Sayfa Adresi ve Sayfa Başlangıcına Uzaklığın Ayrı Saklanması

Base Address Caching [46] yazmaçların ciddi bir kısmının adres saklamak için kullanılmasından faydalanır. *Base Address Cache* adını verdikleri hem bellekte hem de işlemcide bulunan bir yapı önermektedirler. Sayfaların taban adresleri daha kısa uzunluktaki göstergelere dönüştürülür ve sayfa numaraları yerine bu göstergeler (-*ing.* index) saklanır. Yazmaçta saklanan adres de sayfa numarası ve uzaklık yerine gösterge ve uzaklık şeklinde saklandığından yazmaçta saklanan değerin uzunluğu ve dolayısıyla yazmaç öbeğinin boyutu küçülmüş olur. Belleğe erişmek gerektiğinde gösterge bellekte bulunan *Base Address Cache* yardımıyla sayfa numarasına çevrilir.

Sayfa numarası ve yazmaçtan alınan uzaklık yardımıyla adres elde edilir ve belleğe bu yöntemle erişilir.

Chiueh de *Base Address Cache (BAC)* adını verdiği bir çözüm önerisi sunmuştur [47]. BAC taban yazmaç göstergesi – gerçek adres çiftleri için önbellek görevi görür, ESÖ'nün önbelleğine benzer bir işlevi vardır. Çevrimin burada bulunamaması durumunda ESÖ'ye erişilir. BAC tekniğinde yazmaçların tümü taban adres yazmacı olup gerçek adres çevrimini saklayabilir. Bir yazmacın hem gerçek hem sanal adres tutması mümkün olmadığından her yazmacın bir *gölge yazmaca* ihtiyacı cardır. Bu gereklilik özel bir teknik kullanılmadığı durumda yazmaç öbeği boyutunu neredeyse iki katına çıkarır.

8. GELECEK ÇALIŞMALAR

Projenin geliştirilmesine birden fazla yönde devam edilebilir. Bunlardan bazıları: (1) Buyrukları ayırma konusunda derleyicilerden yardım almak, (2) 12-bit AMB'nin diğer işlemlere açılması, (3) adreslerin tamamen ESÖ'ye kaydırılması.

Buyrukları ayırma konusunda derleyicilerden yardım almak. Buyrukların belleğe erişip erişmediği derleyici tarafından tespit edilerek adres yazmaçlarına atama yapılabilir. Derleyiciler aynı zamanda adres yazmaçlarının ve ESÖ satırlarının bırakılması konusunda daha başarılı teknikler geliştirmek için kullanılabilir.

12-bit AMB'nin diğer işlemlere açılması. SONATA eklemeyi önerdiği 12-bit AMB'yi sadece adres yazmaçlarında kullanır fakat çalışmalar göstermektedir ki 12-bit ve altı uzunlukta yapılan toplama işlemlerinin tüm toplama işlemlerine oranı %94.1'dir [36]. Bu nedenle 12-bit AMB'yi tüm buyrukların kullanımına açmak güç tüketimi açısından mantıklı bir karar olarak görülmektedir.

Adreslerin tamamen ESÖ'ye kaydırılması. Adreslerin çevrimi için daima yazmaçtaki değeri aramak için öncelikle ESÖ'ye, gerekli durumlarda sayfa tablosuna gidilir. SONATA da bu değiştirmemiştir. SONATA ile 12-bit uzunluğuna düşen adres yazmaçlarındaki uzunluk bilgisi ESÖ'ye taşınırsa *Yazmaç → ESÖ → Sayfa Tablosu* süreci *ESÖ → Sayfa Tablosu* şeklinde değiştirilebilir.

9. SONUÇ

Günümüzde sanal bellek neredeyse her bilgisayarda kullanılmaktadır ve sanal bellek kullanımı sanal adres kullanımı gerektirir. Sanal adresler belleğe erişmek için gerçek adrese çevrilmelidir. Buyrukların ciddi bir kısmını oluşturan ve gecikmesi yüksek olduğundan sistem performansı açısından önemli bir konumda bulunan yükle ve sakla buyrukları taşıdıkları sanal adreslerin gerçek adrese çevrilmesine ihtiyaç duyar. Bu adreslerin ciddi bir kısmı bellekte aynı sayfaya düşmekte ve çok değerli ve sınırlı sayıdaki yazmaçların %10-40'ının tekrarlayan sayfa numarası değerleriyle doldurmaktadır. Bu çalışma, bu müsrif yaklaşımın önüne geçme amacıyla asgari düzeyde mikromimari değişimle SONATA adını verdiğimiz bir öneri sunmaktadır. Bu önerinin uygulandığı bir yazmaç öbeği %2.76, ESÖ %1.69 ve AMB %0.61 daha az enerji tüketmektedir. Bunun yanı sıra ESÖ'deki ıskalanan aramalar %3.14 azalmıştır. Sonuç olarak SONATA adreslerin büyük kısmının tekrar etmesinden faydalanarak dikkate değer oranda enerji tasarrufu sağlamaktadır

KAYNAKLAR

[1] Talluri M, Hill M D. Surpassing the TLB performance of superpages with less operating system support. ASPLOS 1994; .

[2] Peng J, Lueh GY, Wu G, Gou X, Rakvic R. A comprehensive study of hardware/software approaches to improve TLB performance for java applications on embedded systems. Workshop on Memory System Performance and Correctness, 2006; 102–111.

[3] Karakostas V, Gandhi J, Hill MD, McKinley KS, Nemirovsky M, Swift MM, Unsal OS, et al.. Energy-efficient address translation. International Symposium on High Performance Computer Architecture (HPCA), 2016; 631–643.

[4] Papadopoulou MM, Tong X, Seznec A, Moshovos A. Prediction-based superpage-friendly TLB designs. 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), IEEE, 2015; 210–222.

[5] Swanson M, Stoller L, Carter J. Increasing TLB reach using superpages backed by shadow memory. ISCA, 1998; 204–213.

[6] Talluri M, Kong S, Hill MD, Patterson DA. Tradeoffs in supporting two page sizes. ISCA, 1992; 415–424.

[7] Navarro J, Iyer S, Druschel P, Cox A. Practical, transparent operating system support for superpages. ACM SIGOPS Operating Systems Review 2002; 36(SI):89–104.

[8] Chen L, Wang Y, Cui Z, Huang Y, Bao Y, Chen M. Scattered superpage: A case for bridging the gap between superpage and page coloring. International Conference on Computer Design (ICCD), 2013; 177–184.

[9] Pham B, Vaidyanathan V, Jaleel A, Bhattacharjee A. CoLT: Coalesced large-reach TLBs. International Symposium on Microarchitecture, 2012; 258–269.

[10] Pham B, Vesely J, Loh GH, Bhattacharjee A. Large pages and lightweight memory management in virtualized environments: can you have it both ways? International Symposium on Microarchitecture, 2015; 1–12.

[11] McCurdy C, Coxa AL, Vetter J. Investigating the TLB behavior of high-end scientific applications on commodity microprocessors. ISPASS, 2008; 95–104.

[12] Romer T H, Ohlrich W H, Karlin A R, Bershad B N. Reducing TLB and memory overhead using online superpage promotion. ACM SIGARCH Computer Architecture News, vol. 23, 1995; 176–187.

[13] Lee JH, Lee JS, Jeong SW, Kim SD. A banked-promotion TLB for high performance and low power. International Conference on Computer Design, 2001; 118–123.

[14] Kandiraju GB, Sivasubramaniam A. Characterizing the d-TLB Behavior of SPEC CPU2000 Benchmarks. SIGMETRICS, 2002; 129–139.

[15] Jacob B, Mudge T. Uniprocessor virtual memory without TLBs. IEEE Transactions on Computers 2001; 50(5):482–499.

[16] Barr TW, Cox AL, Rixner S. SpecTLB: a mechanism for speculative address translation. International Symposium on Computer Architecture (ISCA), 2011; 307–317.

[17] Seznec A. Concurrent support of multiple page sizes on a skewed associative TLB. IEEE Transactions on Computers 2004; 53(7):924–927.

[18] Park C H, Chung J, Seong BH, Roh Y, Park D. Boosting superpage utilization with the shadow memory and the partial-subblock TLB. International Conference on Supercomputing, 2000; 187–195.

[19] C. Park, S. Cha, B. Kim, Y. Kwon, D. Black-Schaffer, and J. Huh, “Perforated Page: Supporting Fragmented Memory Allocation for Large Pages,” in Proceedings of the IEEE 47th International Symposium on High Performance Computer Architecture, 2020.

[20] Faruk Guvenilir and Yale N Patt. Tailored page sizes. In 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), pages 900–912. IEEE, 2020.

[21] K.C. Knowlton, “A fast storage allocator,” Commun. ACM, vol.8, no.10, pp.623–625, Oct. 1965.

[22] Ismail Kadayif, Anand Sivasubramaniam, Mahmut Kandemir, Gokul Kandiraju, and Guangyu C. Generating physical addresses directly for saving instruction tlb energy. In 35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002.(MICRO-35).

[23] Ismail Kadayif, Partho Nath,Mahmut Kandemir, and Anand Sivasubramaniam. Reducing data tlb power via compiler-directed address generation. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 26(2):312–324, 2007.

[24] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. The gem5 simulator. ACM SIGARCH computer architecture news, 39(2):1–7, 2011.

[25] "Muhammad Hassan, Chang Hyun Park,and David Black-Schaffer. A reusable characterization of the memory system behavior of spec2017 and spec2006. ACM Transactions on Architecture and Code Optimiza- tion (TACO), 18(2):1–20, 2021.

[26] Shuang Song, Qinzhe Wu,Steven Flolid, Joseph Dean, Reena Panda, Junyong Deng, and Lizy K John. Experiments with spec cpu 2017: Similarity, balance, phase behavior and simpoints. tech. rep., TR- 180515-01, 2018.

[27] Qinzhe Wu, Steven Flolid, Shuang Song, Junyong Deng, and Lizy K John. Hot regions in spec cpu2017. In 2018 IEEE International Symposium on Workload Characterization (IISWC), 2018.

[28] M. N. Bojnordi, "Hardware Speculation" presented to CS/ECE 6810: Computer Architecture, University of Utah, 2018. [PowerPoint slides]. Available: http://www.cs.utah.edu/~bojnordi/classes/6810/s18/slides/13-spec.pdf, Accessed on: Dec. 10, 2021.

[29] Hennessy, John L and Patterson, David A. Computer architecture: a quantitative approach, 2011.

[30] Intel® 64 and IA-32 Architectures Software Developer's Manual," Intel Corporation, 2018.

[31] Inside 6th-generation intel core: New microarchitecture code-named skylake," *IEEE Micro*, vol. 37, pp. 52-62, 2017.

[32] Awad, Amro and Basu, Arkaprava and Blagodurov, Sergey and Solihin, Yan and Loh, Gabriel H. Avoiding TLB shootdowns through self-invalidating TLB entries," in *2017 26th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2017.

[33] Margaritov, Artemiy and Ustiugov, Dmitrii and Bugnion, Edouard and Grot, Boris. Virtual address translation via learned page table indexes," in *Conference on Neural Information Processing Systems*, 2018.

[34] Muralimanohar, Naveen and Balasubramonian, Rajeev and Jouppi, Norman P. CACTI 6.0: A tool to model large caches," *HP laboratories*, vol. 27, p. 28, 2009.

[35] D. Jothi and L. Saranya, "Power efficient CAM using adiabatic logic," 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015, pp. 1-4, doi: 10.1109/ICIIECS.2015.7193078.

[36] M. H. Hajkazemi and A. Baniasadi, "HICPA: A hybrid low power adder for high-performance processors," 2012 IEEE 3rd Latin American Symposium on Circuits and Systems (LASCAS), 2012, pp. 1-4, doi: 10.1109/LASCAS.2012.6180349.

[37] Mirbagher-Ajorpaz, Samira and Garza, Elba and Pokam, Gilles and Jiménez, Daniel A. Chirp: Control-flow history reuse prediction," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020.

[38] Thomas W Barr, Alan L Cox, and Scott Rixner. Spectlb: a mecha- nism for speculative address translation. ACM SIGARCH Computer Architecture News, 39(3):307–318, 2011.

[39] Tzi-cker Chiueh and Randy H Katz. Eliminating the address translation bottleneck for physical address cache. ACM SIGPLAN Notices, 27(9):137–148, 1992.

[40] Misel-Myrto Papadopoulou, Xin Tong, André Seznec, and Andreas Moshovos. Prediction-based superpage-friendly tlb designs. In 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), pages 210–222. IEEE, 2015.

[41] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher,. Spectre attacks: Exploiting speculative execution. In 2019 IEEE Symposium on Security and Privacy (SP), pages 1–19.

[42] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, et al. Meltdown: Reading kernel memory from user space. In 27th USENIX Security Symposium (USENIX Security 18).

[43] Reiley Jeyapaul and Aviral Shrivastava. Code transformations for tlb power reduction. International journal of parallel programming, 38(3):254–276, 2010.

[44] Ashley Saulsbury, Fredrik Dahlgren, and Per Stenström. Recency-based tlb preloading. In Proceedings of the 27th annual international symposium on Computer architecture, pages 117–127, 2000.

[45] Artemiy Margaritov, Dmitrii Ustiugov, Edouard Bugnion, and Boris Grot. Prefetched address translation. In Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, pages 1023–1036, 2019.

[46] Arvin Park and Matthew Farrens. Address compression through base register caching. In [1990] Proceedings of the 23rd Annual Workshop and Symposium@MICRO 23: Microprogramming and Microarchitecture, pages 193–199. IEEE, 1990.

[47] Tzi-cker Chiueh and Randy H Katz. Eliminating the address translation bottleneck for physical address cache. ACM SIGPLAN Notices, 27(9):137–148, 1992.

[48] SPEC CPU2017. [Online]. Available: www.spec.org/cpu2017. [Accessed 13 08 2021].

[49] SPEC 2006. [Online]. Available: www.spec.org/cpu2006. [Accessed 13 08 2021].

[50] Intel 64 and IA-32 Architectures Optimization Reference Manual, Intel, Apr 2018, order Number: 248966-040. [Online]. Available: https://software.intel.com/sites/default/files/managed/9e/bc/64-ia-32-architectures-optimization-manual.pdf

[51] “Intel Xeon Processor Scalable Family Technical Overview, ” Jul 2017. [Online]. Available: https://software.intel.com/en-us/articles/intel-xeon-processor-scalable-family-technical-overview