MİKROİŞLEMCİLERDE ÜRETİLEN DEĞERLERİN GENİŞLİKLERİNİ TAHMİN ETMEK İÇİN DONANIM TASARIMI

H. ŞEYMA ÜLKER

YÜKSEK LİSANS TEZİ

MÜHENDİSLİK ANABİLİM DALI

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

EKİM 2007

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Yücel ERCAN

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Prof. Dr. Ali YAZICI

Anabilim Dalı Başkanı

H. Şeyma ÜLKER tarafından hazırlanan MİKROİŞLEMCİLERDE ÜRETİLEN DEĞERLERİN GENİŞLİKLERİNİ TAHMİN ETMEK İÇİN DONANIM TASARIMI adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Yrd. Doç. Dr. Oğuz ERGİN

Tez Danışmanı

Tez Jüri Üyeleri

Başkan :Doç. Dr. Elif Derya ÜBEYLİ _____

Üye : Yrd. Doç. Dr. Oğuz ERGİN _____

Üye : Yrd. Doç. Dr. Y.Murat ERTEN _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

H.Şeyma ÜLKER

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri**
Anabilim Dalı : **Bilgisayar Mühendisliği**
Tez Danışmanı : **Yrd. Doç. Dr. Oğuz ERGİN**
Tez Türü ve Tarihi : **Yüksek Lisans – Ekim 2007**

H. Şeyma ÜLKER

MİKROİŞLEMCİLERDE ÜRETİLEN DEĞERLERİN GENİŞLİKLERİNİ TAHMİN ETMEK İÇİN DONANIM TASARIMI

ÖZET

Mikroişlemcilerde üretilen değerlerin büyük bir bölümü yazmaç büyüklüğünden daha az sayıda bit ile ifade edilebilen dar değerlerden oluşmaktadır. Dar değerlerden yararlanarak işlemcinin başarımını artırmaya veya güç tüketimi azaltmaya yönelik pek çok çalışma bulunmaktadır. Daha önce önerilen bu yöntemlerin daha iyi çalışmasını sağlamak amacıyla, işlemcide üretilen değerlerin genişliklerini tahmin eden etkili tahmin birimleri tasarlanmıştır. Tez çalışması kapsamında oluşturulan bu tahmin birimlerinin tasarımında bellek eşleme yöntemlerinden Doğrudan Eşleme, Tam İlişkili Eşleme ve Kümeli İlişkili Eşleme yöntemleri ile yer değiştirme algoritmalarından İlk Giren İlk Çıkar, En Uzun Zamandır Kullanılmayan ve Rastgele Seçim algoritmaları kullanılmıştır. Bu tahmin birimleri bir x86 mikroişlemci benzeştiricisi olan MSIM üzerinde SPEC 2000 sınama programları kullanılarak benzetilmiş ve tahmin birimi tasarımlarının başarımları karşılaştırılmıştır.

Anahtar Kelimeler: Mikroişlemci, benzetim, dar değerler, tahmin birimi

University : **TOBB Economics and Technology University**
Institute : **Institute of Natural and Applied Sciences**
Science Programme : **Computer Engineering**
Supervisor : **Assistant Professor Dr. Oğuz ERGİN**
Degree Awarded and Date : **M.Sc. – October 2007**

H.Şeyma ÜLKER

HARDWARE DESIGN FOR PREDICTING WIDTHS OF VALUES PRODUCED IN MICROPROCESSORS

ABSTRACT

A large percentage of the computed values in a processor are narrow values which have fewer bits compared to the width of a register. There are many techniques that improve the preformance or decrease the energy consumption of the processor by exploiting narrow width values. Effective predictors that predict the widths of the produced values have been designed to improve these proposed studies. In this master thesis Direct Mapping, Full Associative Mapping, Set Associative Mapping techniques and First In First Out, Least Recently Used and Random replacement algorithms have been used during the design of these predictors. These predictors have been simulated on a Alpha microprocessor called MSIM using SPEC 2000 benchmarks and the performance of predictor designs have been compared.

Anahtar Kelimeler: Microprocessor, simulation, narrow values, predictor

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren danışmanım Dr. Oğuz ERGİN'e yine kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine ve yüksek lisans öğrencilerine, son olarak desteğini esirgemeyen ailem ve arkadaşlarıma ve Yurt İçi Yüksek Lisans Burs Programı kapsamında tez çalışmamı destekleyen TÜBİTAK'a teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

ÖZET iii

ABSTRACT iv

TEŞEKKÜR v

İÇİNDEKİLER vi

ÇİZELGELERİN LİSTESİ viii

ŞEKİLLERİN LİSTESİ ix

KISALTMALAR xi

1. GİRİŞ 1

1.1. Giriş ve Çalışmanın Amacı 1

1.2. İşlemci Başarım Ölçütleri 1

1.3. Çok Yollu Mikroişlemci Mimarisi 3

1.3.1. Sıralı Ön Uç 6

1.3.1.1. Buyruğun Yakalanması (Okunması) 6

1.3.1.2. Buyruğun Çözülmesi 7

1.3.1.3. Yazmaçların Yeniden İsimlendirilmesi 7

1.3.2. Sırasız Yürütme (Çalıştırma) Merkezi 8

1.3.2.1. Yayın Kuyruğu 8

1.3.2.2. Yeniden Sıralama Belleği 9

1.3.2.3. Yükleme-Saklama Kuyruğu 10

1.4. Mikroişlemciler Arasındaki Farklılıklar 11

BÖLÜM 2 13

2. LİTERATÜR İNCELEMESİ 13

2.1. Dar Değerlerden Yararlanarak İşlemci Başarımını Artırmaya Yönelik Literatürdeki Uygulamalar 13

2.1.1. Dar Değerlerden Yararlanarak Başarımı Artırma 13
2.1.2. Dar Değerlerden Yararlanarak Güç Tüketimini Azaltma 15
2.1.3. Dar Değerlerden Yararlanarak Değer Tahmini Yapma 17
BÖLÜM 3 19
3. BELLEK EŞLEME YÖNTEMLERİ 19
3.1. Doğrudan Eşleme 20
3.1.1. Doğrudan Eşleme Yönteminin Artıları ve Eksileri 23
3.2. Tam İlişkili Eşleme 24
3.2.1. Tam İlişkili Eşleme Yönteminin Artıları ve Eksileri 27
3.3. Kümeli İlişkili Eşleme 27
3.3.1. Kümeli İlişkili Eşleme Yönteminin Artıları ve Eksileri 31
3.4. Yer Değişim Algoritmaları 31
3.4.1. İlk Giren İlk Çıkar Algoritması 32
3.4.2. En Uzun Zamandır Kullanılmayan Algoritması 32
3.4.3. Rastgele Seçim Algoritması 32
3.4.4. En Az Sıklıkla Kullanılan 32
BÖLÜM 4 33
4. UYGULAMA 33
4.1. Benzetim Ortamı 35
4.2. Sınama Programları 36
4.3. Benzetim Sonuçları 38
BÖLÜM 5 50
5. SONUÇLAR VE ÖNERİLER 50
KAYNAKLAR 53
ÖZGEÇMİŞ 55

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

Çizelge 4.1. Benzeştirici Yapılandırmaları 35

Çizelge 4.2. SPEC2000 Tam Sayı Sınama Programları 36

Çizelge 4.3. SPEC2000 Kayan Nokta Sınama Programları 37

Çizelge 4.4. Tahmin Birimlerinin Özelliklerine Göre Ortalama Tam Tahmin Yüzdeleri 48

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 1.1. Çok Yollu İşlemci Mimarisi 3

Şekil 1.2. Temel İşlemci Aşamaları 5

Şekil 3.1. Doğrudan Eşlemeli Önbellek Yapısı 21

Şekil 3.2. Verilen Örnek İçin Ana Bellek Adres Biçimi 22

Şekil 3.3. Genel Adres Alan Boyutları 23

Şekil 3.4.Tam İlişkili Önbellek Yapısı 25

Şekil 3.5. Tam İlişkili Eşleme için Ana Hafıza Adres Biçimi 26

Şekil 3.6. Kümeli İlişkili Önbellek Yapısı 29

Şekil 3.7. Kümeli İlişkili Eşleme İçin Ana Hafıza Adres Biçimi 30

Şekil 4.1.10 Bin Komutluk Evrelerdeki Üretilen Değerlerin Genişliklerinin Ortalamasının Sıklık Tablosu 38

Şekil 4.2. 100 Bin Komutluk Evrelerdeki Üretilen Değerlerin Genişliklerinin Ortalamasının Sıklık Tablosu 39

Şekil 4.3. 1 Milyon Komutluk Evrelerdeki Üretilen Değerlerin Genişliklerinin Ortalamasının Sıklık Tablosu 39

Şekil 4.4. 500,1000 ve 2000 Satırlık Doğrudan Eşlemeli Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları 41

Şekil 4.5. 500, 1000 ve 2000 Satırlık Tam Eşlemeli EUZK Yer Değiştirme Algoritmasının Kullanıldığı Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları 41

Şekil 4.6. 500,1000 ve 2000 Satırlık Tam Eşlemeli İGİÇ Yer Değiştirme Algoritmasının Kullanıldığı Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları 42

Şekil 4.7. 500,1000 ve 2000 Satırlık Tam Eşlemeli Rastgele Yer Değiştirme Algoritmasının Kullanıldığı Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları 42

Şekil 4.8. 500,1000 ve 2000 Satırlık Kümeli Eşlemeli EUZK Yer Değiştirme Algoritmasının Kullanıldığı Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları 43

Şekil 4.9. 500,1000 ve 2000 Satırlık Kümeli Eşlemeli İGİÇ Yer Değiştirme Algoritmasının Kullanıldığı Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları 43

Şekil 4.10. 500,1000 ve 2000 Satırlık Kümeli Eşlemeli Rastgele Yer Değiştirme Algoritmasının Kullanıldığı Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları 44

Şekil 4.11. 500 Satırlık Tam İlişkili Eşlemeli Tahmin Biriminin Başarımında Yer Değiştirme Algoritmalarının Etkisi 45

Şekil 4.12. 500 Satırlık Kümeli İlişkili Eşlemeli Tahmin Biriminin Başarımında Yer Değiştirme Algoritmalarının Etkisi 45

Şekil 4.13. 1000 Satırlık Tam İlişkili Eşlemeli Tahmin Biriminin Başarımında Yer Değiştirme Algoritmalarının Etkisi 46

Şekil 4.14. 1000 Satırlık Kümeli İlişkili Eşlemeli Tahmin Biriminin Başarımında Yer Değiştirme Algoritmalarının Etkisi 46

Şekil 4.15. 2000 Satırlık Tam İlişkili Eşlemeli Tahmin Biriminin Başarımında Yer Değiştirme Algoritmalarının Etkisi 47

Şekil 4.16. 2000 Satırlık Kümeli İlişkili Eşlemeli Tahmin Biriminin Başarımında Yer Değiştirme Algoritmalarının Etkisi 47

Şekil 4.17. 500, 100 ve 2000 Satırlık Tahmin Birimlerinin Tüm Yöntem ve Algoritmalar İçin Elde Ettiği Ortalama Başarım 49

KISALTMALAR

Kısaltmalar	Açıklama
AMB	Aritmetik Mantık Birimi
BB	Bekleme Bölgesi
ÇBB	Çevrim Başına Buyruk
EAB	En Anlamlı Bit
EAAB	En Az Anlamlı Bit
EASK	En Az Sıklıkla Kullanılan
EUZK	En Uzun Zamandır Kullanılmayan
İB	İşlem Birimi
İGİÇ	İlk Giren İlk Çıkar
MSIM	Çoklu Kullanım Benzeştiricisi
OSY	Okuma Sonrası Yazma
SPEC	Standart Başarım Değerlendirme Şirketi
SSE	Internet Streaming Single Instruction Extension
TKÇV	Tekli Komut Çoklu Veri
YK	Yayın Kuyruğu
YSB	Yeniden Sıralama Belleği
YSK	Yükleme Saklama Kuyruğu
YSO	Yazma Sonrası Okuma
YSY	Yazma Sonrası Okuma
YYA	Yazmaçların Yeniden Adlandırılması

BÖLÜM 1

1. GİRİŞ

1.1. Giriş ve Çalışmanın Amacı

Mikroişlemcilerde üretilen değerlerin büyük bir bölümü yazmaç büyüklüğünden daha az sayıda bit ile ifade edilebilen dar değerlerden oluşmaktadır. Dar değerlerden yararlanarak işlemcinin başarımını artırmaya veya güç tüketimi azaltmaya yönelik pek çok çalışma bulunmaktadır. Daha önce önerilen bu yöntemlerin daha iyi çalışmasını sağlamak amacıyla, işlemcide üretilen değerlerin genişliklerini tahmin eden etkili tahmin birimleri tasarlanmıştır. Bu tahmin birimleri mevcut yöntemlerle birlikte uygulandığında hem bu yöntemlerde kullanılan "Yazmaç Dosyası" (YD), "Yeniden Sıralama Belleği" (YSB), "İşlem Birimi" (İB) gibi birimlerin, toplamda da işlemcinin genel başarımını artıracak hem de güç tüketimini azaltacaktır.

Tez çalışmasının 1. bölümünde işlemci başarım ölçütleri ve temel işlemci mimarisi anlatılmaktadır. 2. bölümde dar değerlerden yararlanan yöntemlerle ilgili literatür taraması yer almaktadır. 3. bölümde tez kapsamında tasarlanan tahmin birimlerinin oluşturulmasında yararlanılan bellek eşleme yöntemleri ve yer değiştirme algoritmaları anlatılmaktadır. 4. bölümde tahmin birimlerinin ve benzeştiricinin ayrıntıları ile yapılan çalışmalar ve elde edilen sonuçlar yer almaktadır. Son bölümde ise sonuçlar değerlendirilmekte ve gelecek çalışmalara yönelik öneriler anlatılmaktadır.

1.2. İşlemci Başarım Ölçütleri

İşlemci başarımında en temel ölçütler komut başına düşen çevrim sayısı ve saat vurum sıklığıdır. Güncel mikroişlemciler başarımı artırmak için komut düzeyinde koşutluktan yararlanarak bir saat vuruşunda birden fazla komut işleyebilmektedir. Bu işlemciler ardışık bir programı daha paralel bir yapıya dönüştürmektedirler. Süper

ölçekli işlemciler ve boru hatlı işlemciler günümüzde yaygın olarak kullanılan ve komut düzeyinde koşutluktan yararlanan yapılara örnek olarak verilebilir. Buyrukların sırasız çalıştırılması ve yazmaçların yeniden adlandırılması gibi yöntemlerin temelinde de komut düzeyinde koşutluktan yararlanma yatmaktadır.

Bilgisayarın başarımı yapılması istenilen bir işin tamamlanmasına kadar geçen zaman ile ölçülür. Bu zamana tepki zamanı veya yürütme zamanı [1] denir ve bu süreye bellek erişim süresi, giriş/çıkış işlemlerinin süresi, işlemci süresi gibi pek çok süre dâhildir. İşlemcinin başarımı ise, çalıştırılmak istenen programın komutlarının çalıştırılması için harcanan zamanı içerir. Kısaca verilen bir iş için harcanan zaman olarak tanımlanabilecek olan işlemci başarımı pek çok parametreye bağlıdır. Bu tanımdan yola çıkarak başarım aşağıdaki gibi ifade edilebilir:

$$Başarım = 1 / Yürütme\ Zamanı \quad (1.1)$$

Tanımdan da anlaşılacağı üzere başarım ile yürütme zamanı ters orantılıdır. Bir işlemcinin yürütme zamanı ise yürütülmekte olan işe, işlemcinin frekansına (saat vuruş sıklığı) ve çevrim başına işlenen buyruk (ÇBB) sayısına bağlıdır [2]. ÇBB boru hattından geçen komut miktarını ölçmede sıkça kullanılan bir ölçüttür. Zaman zaman yanlış dallanma tahminleri ve kural dışı durumlar nedeniyle boru hattındaki buyrukların boşaltılmasından dolayı ÇBB, çevrim başına boru hattından geçen ortalama komut sayısı yerine çevrim başına ortalama emekliye ayrılan komut sayısına eşit olmaktadır.

Başarımı aynı zamanda aşağıdaki gibi tanımlamak da mümkündür:

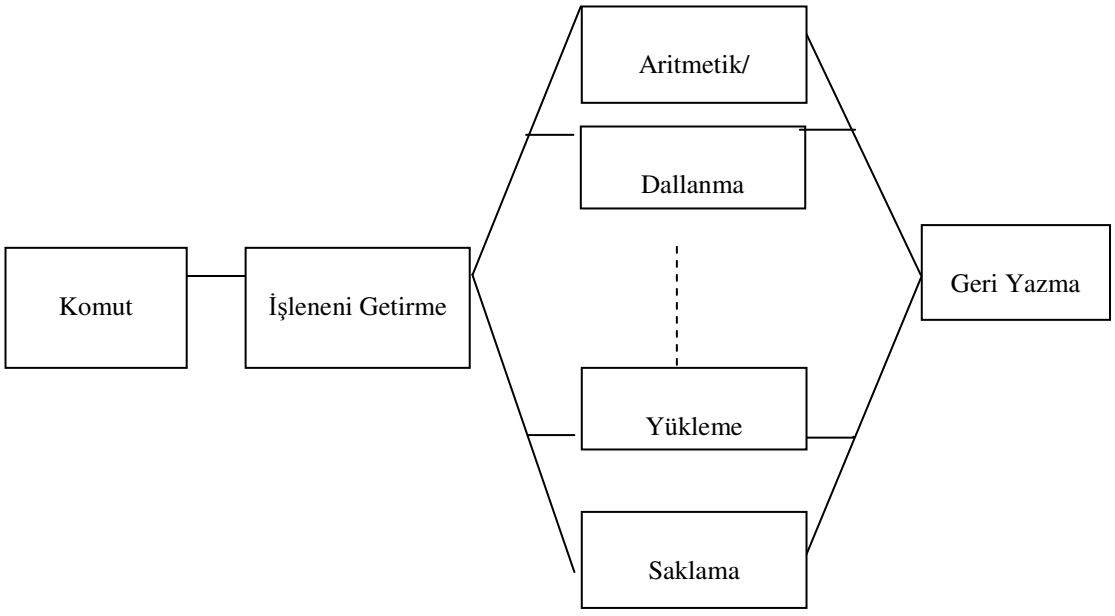
$$Başarım = (ÇBB \times frekans) / Buyruk\ Sayısı \quad (1.2)$$

Bu eşitlikten anlaşılacağı üzere başarımı artırmak için ya frekansı yani saat vuruşunun sıklığını artırmak veya çevrim başına işlenen komut sayısını artırmak

gerekmektedir. Frekans, işlemcinin bir veriyi hangi hızda işleyebildiğini belirlediği için, işlemciyi yüksek frekansta çalışacak şekilde tasarlamak genel başarımı belirlemede önemli bir bileşendir [3]. Başarımı artırmak için iki vuruş arasındaki zaman ya da programın çalışması için gereken vuruş sayısı da azaltılabilir. Yakın zamanda Intel® Pentium® M işlemcilerinde daha gerçekçi ve doğru dallanma tahminleri yapılarak işlenen komut sayısını azaltmak başarımı artırmada kullanılan bir diğer yöntemdir [4].

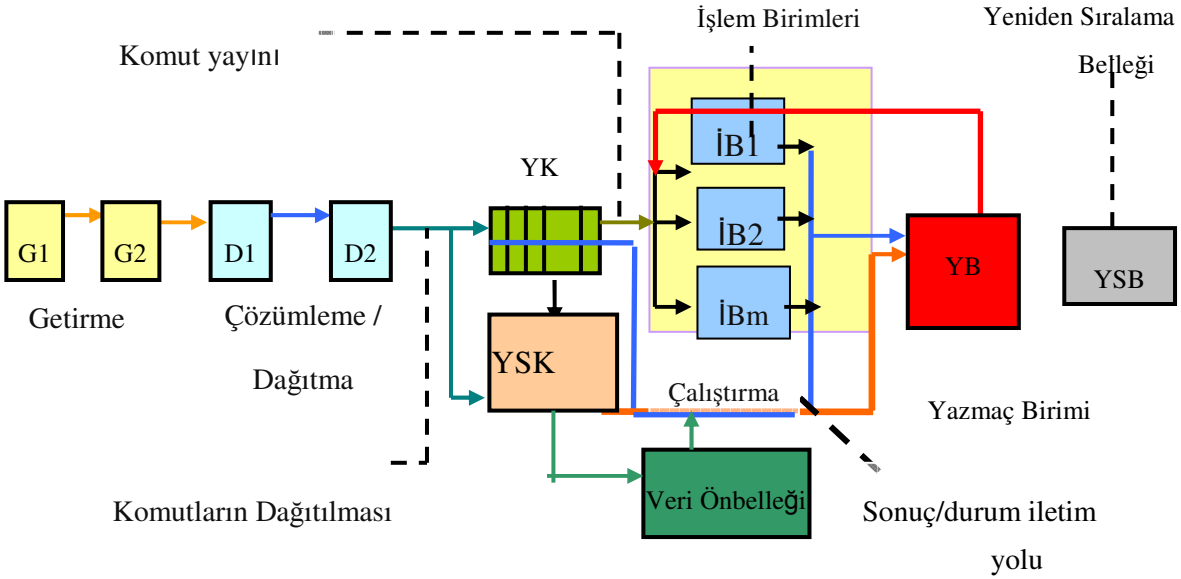
1.3. Çok Yollu Mikroişlemci Mimarisi

Tek bir boru hattının işlemciden beklenen başarımı sağlamaya yeterli olmadığı durumda boru hattı sayısı artırılarak paralel çalışmaları sağlanabilir. Çok yollu işlemci mimarisi olarak adlandırılan bu yapıda tek bir boru hattındaki komutların farklı aşamaları paralel işlenirken, farklı boru hatlarındaki komut grupları arasında da paralellik sağlanmış olmaktadır [5]. Çok yollu mikroişlemciler çevrim başına birden fazla komut işleyerek, ÇBB sayısını ve dolayısıyla toplam başarımı artırmaktadır. Tipik bir çok yollu işlemci yapısı Şekil 1.1'de görülmektedir:



Şekil 1.1. Çok Yollu İşlemci Mimarisi

Tipik bir mikroişlemcinin çalışması beş temel aşamada gerçekleşmektedir. Bu aşamalar Yakalama (Getirme), Çözme, Yürütme (Çalıştırma), Yazma, Emekliye Ayrılma (Sonlandırma) aşamalarıdır. İşlemciler arasında bazı farklılıklar bulunmakla birlikte çok yollu işlemci iç mimarileri bu temel aşamalara göre çalışmaktadır. İlerleyen bölümlerde bu aşamalar ayrıntılı olarak anlatılmaktadır. İşlemcinin temel aşamaları Şekil 1.2'de belirtilmektedir.



Şekil 1.2. Temel İşlemci Aşamaları

1.3.1. Sıralı Ön Uç

Komutlar program sırasıyla getirilir ve çözülürken, yürütme, yazma ve emekliye ayrılma aşamaları program sırasından bağımsız olarak gerçekleşebilmektedir. Yazmaçların yeniden adlandırılması tekniğiyle komutların program sırasını takip etmeden işlenmesi sağlanmakta, böylece aralarında veri bağımlılıkları olmayan komutların işlem sırası gelinceye kadar beklemeleri ile oluşan zaman kaybı önlenmektedir.

1.3.1.1. Buyruğun Yakalanması (Okunması)

Yakalama (getirme) aşamasında komutlar komut belleğinden yakalanarak "Yayın Kuyruğu (YK)" olarak adlandırılan, ilk-giren-ilk-çıkar (İGİÇ) yapılı bir kuyruğa yerleştirilmektedir. Program sırasına göre öndeki komutların sonuçları hesaplandıkça, bu sonuçlar YK'de beklemekte olan diğer komutlara iletilmektedir. YK'deki bekleyen bir komutun bütün işlenenleri hazır hale gelince bu komut işlenmeye hazır duruma geçmektedir. Hazır bekleyen bir komutun işleneceği işlem birimi de hazır olur olmaz, bu komutun işlenenleri işlem birimine aktarılarak, komut işlenmeye başlayabilmektedir.

YK'nin büyüklüğü W yollu bir makine için, hem Yakala hem de Çöz aşamalarındaki eş zamanlı erişimin mümkün olabilmesi için genellikle 2xW boyutundadır [2]. Ancak çevrim başına W komut yakalanması çok zaman mümkün olmamaktadır. Bu durum, programdaki dallanan komutlar nedeniyle, işlemcinin dallanmanın ne yöne doğru olacağını kestirememesinden kaynaklanmaktadır. Bu şekilde koşullu dallanma komutlarının varlığından kaynaklanan boru hattı tıkanmalarını engellemek amacıyla mikroişlemciler dallanma tahmin birimleri kullanarak, koşullu dallanmaların sonucunu tahmin etmeye çalışmaktadır.

Yanlış dallanma tahminleri neticesinde boru hattındaki komutlar boşaltılmakta, program sayacı (komut işaretçisi) da dallanmanın doğru olarak gerçekleşeceği yöndeki komutları yakalayacak şekilde düzeltilmektedir. Ancak bu durum gereksiz güç tüketimine yol açtığından işlemcilerin doğru dallanma tahminleri yapması oldukça önemlidir.

1.3.1.2. Buyruğun Çözülmesi

Yakala aşamasında yakalama belleğine yazılan komutlar, bir sonraki aşamaya geçerek çözülür. Çöz aşamasında komutun türü ve komutun ihtiyacı olan kaynaklardan hangilerine erişebildiği belirlenir. Yine bu aşamada komutların kaynak ve hedef işlenenleri yeniden adlandırılmaktadır. Buradaki amaç gerçek dışı veri bağımlılıklarını ortadan kaldırmaktır.

1.3.1.3. Yazmaçların Yeniden İsimlendirilmesi

Yazmaçların yeniden adlandırılması (YYA) gerçek dışı veri bağımlılıklarını ortadan kaldıran ve komutların paralel olarak yürütülmesine olanak tanıyan bir yöntemdir. Aynı yazmaca veya bellek bölümüne birden fazla komut eşlenebileceği için komutlar arasında veri bağımlılıkları oluşmaktadır [6]. İdeal durum, komutların yazma sonrası okuma (YSO) bağımlılıklarında olduğu gibi gerçek veri bağımlılıkları dikkate alınarak yürütülmesidir. Okuma sonrası yazma (OSY) riskleri; bir komutun belleğin bir bölümüne yeni bir değer yazması gerektiğinde, belleğin bu bölümünde yer alan eski değeri okuması gereken tüm komutlar okuyuncaya kadar o komutun beklemesinin gerektiği durumlarda oluşur. Yazma sonrası yazma (YSY) riski ise aynı bellek bölümünü birden fazla komutun güncellemesi gerektiğinde bu güncellemen doğru sırayla gerçekleşmesinin gerektiği durumlarda oluşur. YYA, gerçek dışı veri bağımlılıklarından olan YSY ve OSY risklerini ortadan kaldırmada kullanılan etkili bir yöntemdir.

YYA, fiziki yazmaçları (yeniden adlandırma yazmaçları) mimari yazmaçlarla (mantıksal yazmaçlar) ilişkilendirme sürecidir. Hedef yazmacı belli olan her komuta yeni bir fiziki yazmaç atanmaktadır. Her fiziki yazmaca, boşta hazır beklemekte olan yazmaç listesinden yapılan atamadan sonra sadece bir kere yazılmaktadır. Eğer sonradan gelen başka bir komut da bu yazmacın değerine ihtiyaç duyarsa bu durumda önceki komut, sonraki komut değerini yazana kadar beklemelidir. YYA'da iki temel teknik bulunmaktadır [7]. Birinci teknikte mimari ve fiziki yazmaçlar için iki ayrı yazmaç kümesi bulunmaktadır. Fiziki yazmaçlar sonuçlanan ancak emekliye ayrılmamış komutların sonuçlarını saklarken, mimari yazmaçlar emekliye ayrılmış olan komutların sonuçlarını tutmaktadır. Bir komutun emekliye ayrılması ile bu komutun sonucunun yeniden adlandırma yazmacından mimari yazmaca kopyalanması gerekmektedir. İkinci teknikte ise tek bir yazmaç kümesi bulunmakta ve mimari yazmaçlar fiziki yazmaçlara devingen olarak eşlenmektedir. Fiziki yazmaçlar hem emekliye ayrılan komutların değerlerini hem de geçici değerleri saklamaktadır. Bir komutun emekliye ayrılması ile fiziki yazmaç kalıcı hale gelir ve kopyalama işlemi gerekmemektedir.

1.3.2. Sırasız Yürütme (Çalıştırma) Merkezi

Çöz aşamasından geçen her komut, komut türüne bağlı olarak yayın kuyruğu, bekleme birimi (BB), işlem birimi, yeniden sıralama belleği ve yükleme saklama kuyruğu (YSK) gibi birimlerin işbirliği ile program sırasını dikkate alınmadan yürütülmektedir. Çözülmüş bir komutun yayın kuyruğuna gönderilebilmesi için hem yayın kuyruğunda boşta bölüm olması hem de YSB nin dolu olmaması gerekmektedir. Yükleme ve saklama komutlarının yayın kuyruğuna gönderilebilmesi için yukarıdaki şartların yanı sıra YSK'de de bir bölümün ayrılması gerekmektedir.

1.3.2.1. Yayın Kuyruğu

Yazmaçların yeniden adlandırılması aşamasından geçen her komut, yayın kuyruğu birimine kaynak ve hedef yazmaçlarına karşılık gelen etiketleri yerleştirir [2]. Bu

etiketler daha sonraki aşamalarda, yürütülmesi tamamlanan komutlar tarafından o komutun değerini kullanan tüm komutları bilgilendirmek için kullanılır. Her komutun kaynak yazmaç etiketi, tamamlanmakta olan komutun hedef fiziki yazmaç etiketi ile karşılaştırılır. Etiketler aynı ise, karşılık gelen kaynak işlenenin hazır biti birlenir.

Bir komutun kaynak yazmacına karşılık gelen tüm hazır bitleri birlenince, komut "uyanır" ve yürütme aşamasına hazır hale gelir. Tamamlanmakta olan komutun etiketlerini, YK'de bulunan tüm komutların kaynak işlenenleri ile karşılaştırma ve hazır bitlerinin güncellenerek işlenenin hazır olduğunun bildirilmesi aşaması "Uyanma Aşaması" dır. Uyanan komutlar seçim için hazır hale gelirler. Bir komutun YK birimi hazırda bekleyen bir işlem birimine gönderilmesi aşamasına "komut yayını" denir. Yayınlanacak komut sayısı yayın kuyruğunun genişliğine ve hazırda bekleyen işlem birimi sayısına bağlıdır.

YK birkaç şekilde tasarlanabilmektedir. Bazı tasarımlarda işlenenlerin veri değerleri YK birimlerinde tutulmaktadır. Bu teknikte; işlenen değerleri çözme aşamasında okunarak YK birimlerine yerleştirilmelidir. Bu değerler komutun uyanması aşamasında hazır olduğu için komutun yayınlanması sırasında yazmaç dosyasına erişim gerekmemektedir.

Diğer teknikte ise işlenenlerin veri değerleri yayın kuyruğunda tutulmadığından daha basit bir tasarım söz konusudur. Ancak komutlar yürütme için seçildiğine işlenenlerini okumak için yazmaç dosyasına erişmek zorundadır.

1.3.2.2. Yeniden Sıralama Belleği

Komutların program sırası dışında çalıştırılmalarından sonra tekrar program sırasına sokulmaları gerekmektedir. Özellikle kesme ve kural dışı durumlara karşı bir önlem olarak geliştirilen yeniden sıralama belleği komutları program sırasıyla saklamaktadır. Kesme veya kuraldışı bir durumla karşılaşıldığında; komutlar

program sırasıyla tutulduğundan, hatalı işlenen komuttan sonraki tüm komutlar kolaylıkla belirlenerek atılır ve işlenmesi gereken doğru komutlar işlenir.

Yeniden sıralama belleği, dairesel İlk Giren İlk Çıkar (İGİÇ) yapılı bir kuyruktur. Çözülme aşamasından geçen her komut için YSB dolu olmadığı sürece YSB'den bir satır ayrılmaktadır. Dolayısıyla yeniden sıralama belleğinin satır sayısı işlemcideki yürürlükteki komut sayısını belirtmektedir.

Sonucu üretilen ve yazmacına bu değeri yazılan her komutun işlem görme aşaması tamamlanmış olur ve yeniden sıralama belleğinden ayrılır. YSB'den ayrılan komutlar "emekliye ayrılma" aşamasına geçer.

1.3.2.3. Yükleme-Saklama Kuyruğu

Bellek komutlarının adresleri veya verileri sırasız olarak hesaplansa da; "kusursuz durum" olarak ifade edilen, komutların tamamının doğru olarak çalıştırıldığı, hiç br yanlış dallanma tahmininin, kesme veya kuraldışı durumun gerçekleşmediği durumu korumak için komutların belleğe erişimleri gerçek program sırasıyla olmalıdır. Program sırasına göre daha önlerde yer alan ancak bellek işlemlerinin adresleri henüz çözülmemiş olan komutların veya adresleri arasında bağımlılıklar olan komutların belleğe erişimlerini engelleyen ve komutları program sırasıyla tutan birim yükleme saklama kuyruğudur [2]. Saklama komutları emekliye ayrılıncaya kadar belleğe erişmemektedir. Böylece yanlış dallanma tahmini veya kural dışı bir durum dolayısıyla atılması gereken saklama komutlarının belleği yanlış güncellemesi engellenmektedir.

Tüm bellek komutları yayın kuyruğuna girmeden önce YSK'den bir satır ayırır. Bu komutlar; adresleri belli olduktan sonra, yükleme saklama kuyruğuna girmekte ve bellek bağımlılıklarının çözülmesini beklemektedirler. Saklama komutlarının

yayınlanabilmesi için saklanacak değerlerinin hesaplanması gerekmemektedir. Bu komutlar YSK'de iken verileri hesaplanabilir.

1.4. Mikroişlemciler Arasındaki Farklılıklar

Çok yollu mikroişlemci tasarımları arasında bazı farklılıklar bulunmaktadır. Bu farklılıklar genellikle yazmaçların yeniden adlandırılması aşaması ile yeniden sıralama belleği ve yayın kuyruğu yapılarında göze çarpmaktadır. Pentium® II ve Pentium® III işlemciler mimari yazmaçların durumlarını tutmak için ayrı bir yazmaç dosyası bulundururken YSB birimlerinde komut verilerini tutarak yeniden sırlama belleğini birer fiziksel yazmaç gibi kullanmaktadır. Bu işlemcilerde yeniden sırlama belleğinden bir satır tahsis edilen bir komuta aynı zamanda bir de yazmaç atanmaktadır. Bu durum yazmaçların atanma aşamasını kolaylaştırmakla birlikte komutların emekliye ayrılması aşamasında kusursuz durumun korunması için yazmaç değerlerinin mimari yazmaçlara da kopyalanması gerekmektedir. Pentium® 4, Alpha 21264 ve MIPS 10000 işlemcileri ile Pentium® III işlemciler arasındaki en temel fark yeniden sırlama belleğindedir. Pentium® III mimarisinde YSB birim numarası ile yazmaç numarası aynı iken Pentium® 4, Alpha 21264 ve MIPS 10000 işlemcilerde YSB ile yazmaç birimi ayrı yerlerde bulunmaktadır [6]. Bu durum özellikle çok fazla dallanma saklama komutunun olduğu programlarda YSB de gereksiz boşlukların oluşmasını önlemektedir.

Mimariler arasında yayın kuyruğu yapıları açısından da farklılıklar bulunmaktadır. Pentium® II ve Pentium® III mimarilerde yayın kuyruğu birimlerinde hem veriler hem de yazmaç numaraları tutulurken Pentium® 4, Alpha 21264 ve MIPS 10000'de yayın kuyruğunda veriler tutulmamaktadır. Bu nedenle Pentium® II ve Pentium® III de veriler ya iletilme aracılığıyla elde edilmekte ya da yazmaç dosyasına erişim çözülme aşamasında gerçekleşmektedir. Bu durum yayın kuyruğundan seçilecek komutların seçimini kolaylaştırmakla birlikte yayın kuyruğunun boyutunu arttırmaktadır. Ayrıca işlenen değerlerini okumak için hem mimari yazmaç dosyasına hem de yeniden sıralama belleğine erişim gerektiğinden çözülme aşamasını

yavaşlatmakta ve zorlaştırmaktadır. Pentium® 4, Alpha 21264 ve MIPS 10000 mimarilerde ayrı yazmaç dosyası sayesinde komutların emekliye ayrılması aşamasında veri aktarımı önlenmiş olsa da çok da karmaşık yazmaç atama-bırakma yöntemleri gerekmektedir.

BÖLÜM 2

2. LİTERATÜR İNCELEMESİ

İşlemcide üretilen değerlerin çok büyük bir kısmının dar değerler olduğu ve bu sonuçların genişliklerinin doğru tahmin edilebilme oranının oldukça yüksek olduğu pek çok araştırmacı tarafından tespit edilmiştir. Bu gözlemden yola çıkılarak, dar değerlerden yararlanmak suretiyle çok yollu işlemcilerin güç tüketimini azaltacak veya başarımını artıracak yöntemlerin tasarlanması hedeflenmiştir.

2.1. Dar Değerlerden Yararlanarak İşlemci Başarımını Artırmaya Yönelik Literatürdeki Uygulamalar

Fiziki yazmaçları ve yazmaç dosyalarını daha etkili kullanmaya yönelik literatürde pek çok çalışma bulunmaktadır. Bu çalışmaların büyük çoğunluğu yazmaç dosyasının yükünün hafifletilmesi ve dar değerlerin paketlenmesi temeline dayanmaktadır. Yapılan birçok araştırmada işlenen değerlerinin büyük çoğunluğunun dar değerler olduğu ve bu değerlerin doğru tahmin edilme oranlarının oldukça yüksek olduğu gözlenmiştir [2,8,9].

Kendisinin ve solundaki bitlerin hepsinin sıfır (ya da bir) olduğu ilk bitin yeri, o değerin genişliğini belirtir. Veri yolunun genişliğinden (genellikle 32-bit veya 64-bit) daha küçük bir büyüklüğe sahip olan değerler dar değerlerdir. Komut belleğindeki her komuttun yanına ikişer bit eklenerek, her komut tarafından üretilen değerin genişliğinin 1–15, 16–31, 32–47 ve 48–64 bit genişliklerden hangi gruba uyduğu kolaylıkla belirlenebilir.

2.1.1. Dar Değerlerden Yararlanarak Başarımı Artırma

Komut kümesi uzantılı işlemcilerde, dar tamsayı değerli birkaç komutun tek bir işlem birimine paketlenmesi fikri uzun bir süredir kullanılmaktadır. Bu komut kümesi gelişmeleri ilk kez Intel® Pentium® II işlemcilerde MMX™ [7.10.11] komut

kümesi uzantısının kullanılması ile başlamış ve dar tam sayı değerli birkaç işlenenin geniş işlem birimlerine paketlenmesi sağlanmıştır. Daha sonraları Pentium® III de SSE (Internet Streaming Single Instruction Extension) , Pentium® 4 de SSE2, Pentium® M de SSE ve SSE2 ve Pentium® 4 ün geliştirilmiş 90nm sürümün de SSE3 sunularak kayan nokta değerli birkaç işlenenin geniş işlem birimlerine paketlenmesi sağlanmıştır [11-16]. Tek Komut Çok Veri (TKÇV) Single Instruction Multiple Data (SIMD) olarak adlandırılan bu yöntemde dar değerli komutların paketlenmesi işi daha çok programcıya bırakılmaktadır.

Gereksiz olan üst bitleri sıfır olan işlenenlerde çarpma işleminin erken sonlandırılması amacıyla ardışık sıfırları belirleyen devreler kullanılmıştır. Dar değerli işlenenler daha soldaki bitlerinin hepsinin sıfır veya hepsinin bir olduğu işlenenlerdir [9]. Dar değerlerle işlem yapan işlem birimlerinin bölümleri kapatılarak, devre dışı bırakılarak bu bölümler tarafından harcanan güç miktarının azaltılması hedeflenmiştir. Ancak bu tekniğin uygulanabilmesi için sıfır veya bir bitlerini tespit etmek amacıyla özel devreler ve diğer ek donanım gerekmektedir. Brooks ve Martonosi'nin bu çalışmasında İşlem Paketlemesi olarak isimlendirilen, dar değerli birkaç işlenenin tek bir aritmetik mantık birimine (AMB) paketlendiği bir teknik sunulmuştur [9]. Bu tekniğin kullanılmasıyla boşta kullanılmaya hazır bekleyen işlem birimi sayısında artış olacağından işlem görmeyi bekleyen komut sayısında azalma olmakla birlikte yine ek donanım maliyeti ve bu parçaların birbirlerini etkileme olasılığı söz konusudur.

"İşlem Paketlemesi" tekniğinin bir uzantısı olan "Tahmine Dayalı Paketleme"de işlenenlerden en az birinin dar değerli olması durumunda yine birkaç işlem tek bir işlem birimine paketlenmektedir. Bu teknik tahmine dayalı bir teknik olup zaman zaman doğru sonuçlar üretemeyebilmektedir. Bu durumda taşmaya neden olan komutun klasik yöntemle yeniden yayınlanması ve yürütülmesi gerekmektedir.

İşlenen genişliklerini tahmin etmek amacıyla tahmin birimi kullanarak "İşlem Paketlenmesi" tekniği geliştirilmeye çalışılmıştır [17]. Bu çalışmanın [9]'dan en

temel farkı işlenenlerin genişliklerinin tespit edilmesi yerine değerlerin tahmin edilmesidir. [17] çalışması; işlenen değerlerinin oldukça doğru tahmin edilebildiği ve dar değerlerden yararlana tekniklerde kullanılmak amacıyla, doğru tahmin eden işlem birimlerinin yapılabileceği fikrinin ilk kez ortaya atıldığı çalışma olması bakımında oldukça önemlidir.

2.1.2. Dar Değerlerden Yararlanarak Güç Tüketimini Azaltma

Araştırmacıların ilgilendiği bir diğer konu dar değerlerden yararlanarak güç tüketiminin azaltılmasıdır. Bu konuda yapılan temel çalışmalar [2,18-21]'dir.

Aggarwal ve arkadaşları çalışmalarında [20] yayınlanan pek çok komutun kaynak işlenenlerinin tamamının 8 bit veya daha küçük değerlerden oluştuğu gözlenmiştir. Bu dar değerlerin en solundaki bitlerin yazmaç dosyasına erişmeleri gerekmediğinden, bu bitlerin kapatılarak işlemcinin güç tüketiminin azaltılması hedeflenmiştir. Ayrıca yazmaç dosyasında bulunan en soldaki bitlerin bazı kapıların kaldırılması ile daha fazla enerji tasarrufu sağlanması planlanmıştır. Kapıların kaldırılması işlem birimlerindeki bu kapıların kaldırılması ile desteklenmiştir. Böylece işlem birimlerinin bazısı sadece dar değerleri işlemektedir. Bu çalışma ile yazmaç dosyasının erişim zamanını kısaltmak suretiyle enerji tasarrufu sağlanmıştır.

Gonzalez ve arkadaşları [19] yazmaç dosyası yapısını değiştirerek güç tüketimini azaltmayı hedeflemişlerdir. Bu çalışmada yazmaç dosyası basit yazmaç dosyası, kısa yazmaç dosyası ve uzun yazmaç dosyası olmak üzere üç bölüme ayrılmıştır. Yazmaç dosyasında saklanan değerlerin büyük çoğunluğunun dar değerler olduğu sonucundan hareketle dar değerler basit yazmaç dosyasına saklanmaktadır. Daha geniş bir işlenene erişim gerektiğinde basit yazmaç dosyasındaki gösterge alanı kullanılarak uzun veya kısa yazmaç dosyalarına erişim sağlanmaktadır. Üretilen değerlerin büyük çoğunluğunu dar değerler olduğu durumlarda bu değerlerin yazıldığı ve okunduğu yazmaç dosyası basit yazmaç dosyası olacağından ve bu

yazmaç dosyası tipinin boyutun küçük ve güç tüketiminin de az olması nedeniyle enerji tasarrufu sağlanmaktadır.

Benzer bir çalışma [2,18]'de gerçekleştirilmiştir. Burada dar değerli birkaç yazmacın tek bir yazmaca paketlenmesi için iki farklı teknik kullanılmaktadır. Birinci teknik daha geleneksel bir yapıda olup başlangıçta her komuta 64 bitlik birer yazmaç tahsis etmektedir. Saklanacak değerin daha küçük boyutlu bir değer olduğu anlaşılınca uygun boyutta bir yazmaç parçası tahsis edilmektedir. Böylece bir komutun sonucu hesaplanır hesaplanmaz yazma aşamasının gerçekleşmesi için gerekli boyutta bir alan sağlanmış olmaktadır.

İkinci teknik ise tahmine dayalı bir tekniktir. Burada bir tahmin birimi yardımıyla komutun sonucunun genişliğine yönelik bir tahminde bulunulmakta ve tahminin sonucuna göre bir yazmaç bölümü atanmaktadır. Sonuç üretilinceye kadar 64 bitlik bir yazmacın tutulmasının engellenmesi ve yazma aşamasında etiket yayınlanmasının önlenmesi bu tekniğin en önemli avantajlarıdır. Yine bu yöntemde doğru tahminde bulunma oranı oldukça yüksek olduğundan komutlar değerlerini gerçekte tam olarak ihtiyaç duydukları boyutlardaki yazmaç parçalarında sakladıkları için etiket yayınlarının sayısı düşmektedir. Bu çalışma ile daha küçük yazmaçlar kullanılması, dolayısıyla yazmaç dosyalarının boyutlarının küçültülmesi ve bu dosyaların daha verimli kullanılması hedeflenmiştir. Yazmaç dosyalarının boyutu ile güç tüketimi arasında doğru ilişki olduğu düşünüldüğünde bu çalışmanın da güç tüketimi üzerinde önemli etkileri olduğu anlaşılmaktadır.

Güç tüketimine yönelik çalışmalardan bir diğeri bit-bölmeli yazmaç dosyasının tasarlanmasına dayanmaktadır [21]. Bu tasarımda 64 bitlik bir veri yolunda üretilen değerlerin pek çoğunun dar değerler olduğu ve bu değerlerin atandıkları yazmacın tamamını kullanmadıkları gözlenmiştir. Bu boşlukları azaltmak amacıyla yazmaç dosyası 32 bitlik iki parçaya bölünerek yazmaç sayısı iki katına çıkarılmıştır. Yazmaçların atanması aşamasında, komutların kaynak ve hedef bölümleri için ikişer yazmaç parçası ikişer de etiket konulmaktadır. Komutların yürütme aşaması

tamamlandığında; komutun sonucu dar değerli olursa, bu durumda değerin tüm sıfır veya birlerini tutan kısmı diğer komutlar tarafından kullanılmak amacıyla bırakılmaktadır. Yazmaç dosyasının birim sayısının azalması ile güç tüketiminde azalma gözlenmektedir.

2.1.3. Dar Değerlerden Yararlanarak Değer Tahmini Yapma

Komutların sonuçlarını, komutlar çalıştırılmadan tahmin etmek amacıyla "Değer Tahmini" tekniği kullanılmaktadır [2,18,22,24]. Genellikle tüm değer tahmin yöntemlerinin uygulanmasında oldukça büyük tahmin tabloları kullanılmaktadır. Tabloların boyutları arttıkça erişim zamanları ve güç tüketimleri artmaktadır. Dar değerler kullanılarak, tahmin tablolarının boyutlarını küçültmek ve erişim zamanlarını azaltmak mümkün olmaktadır.

Sato ve Arta [22]'de değer tahmin tablosunda saklanan değerlerin çoğunluğunun 8 bitlik değerler olduğu gözleminden yola çıkarak; genişlik tahmininden yararlanarak değer tahmini için gerekli donanımı azaltmışlardır. Bu çalışmada 8 bitlik değerler için dar tahmin birimi, daha geniş değerler için ise geniş tahmin birimi kullanılmaktadır. Her tahmin aşamasında iki tahmin birimi de taranmakta, tahmin birimlerinden yalnızca birinde değere rastlanmaktadır.

[22-24] çalışmalarında dar değerler, değer tahmin tablolarının karmaşıklığını azaltmada kullanılmaktadır. Genişlik tahmin birimleri kullanılarak, değer tahmin birimlerinin boyutunu ve güç tüketimini azaltmak mümkündür. Loh, bu çalışmasında genişlik tahmin birimleri kullanarak, genişlik değerlerine göre bölümlenmiş son değer tahmin birimlerinin tüm bölümlerine erişilme ihtiyacını ortadan kaldırmaktadır. Bu sayede tüm tablolara eş zamanlı olarak erişilme zorunluluğu kalkmakta ve tahmin birimlerinin güç tüketimi azalmaktadır. Loh bu çalışmasında, küçük genişliğe sahip bir kaç değeri büyük boyutlu tek bir saklama alanına paketlemek yerine her biri farklı genişliklerde küçük boyutlu birkaç tablo kullanmayı

ve veri-genişlik tahmin birimi kullanarak bu tablolardan birini seçmeyi tercih etmektedir.

BÖLÜM 3

3. BELLEK EŞLEME YÖNTEMLERİ

Bilgisayar belleği hiyerarşik bir yapıdadır. İşlemciye en yakın olan en üst düzeyde, işlemci yazmaçları yer alır. Ardından önbelleğin bir ya da birkaç seviyesi gelir. Bu seviyeler genellikle L1, L2, … şeklinde isimlendirilir. Daha sonra ise ana bellek gelir [25].

Bellek sıradüzeninde (hiyerarşi) aşağılara inildikçe, bit başına maliyetin azaldığı, sığanın arttığı, erişim zamanının arttığı ve işlemcinin ana belleğe erişim sıklığının azaldığı gözlenir. Sadece en hızlı olan belleğin kullanılması istenilse de yüksek maliyet nedeniyle daha yavaş erişim zamanına sahip olan, büyük boyutlu, daha ucuz ve daha yavaş bellekler tercih edilmektedir. Yavaş belleklerin daha verimli kullanılması amacıyla veri ve programlardan sık kullanılanlar en hızlı bellekte tutulmaktadır.

Bilgisayar işlemcisi çok hızlıdır ve sürekli olarak bellekten veri okur. Hafıza erişim süreleri işlemci hızından daha yavaş olduğu için genellikle işlemci verinin ulaşmasını beklemek zorunda kalır. Önbellek, muhtemelen yakın gelecekte ihtiyaç duyulacak türdeki verinin saklanması için işlemci tarafından kullanılan, küçük, geçici ve hızlı bir bellektir [26].

İşlemcinin programın ilerleyen evrelerinde ana belleğe yapacağı erişimlerin büyük çoğunluğu genellikle daha önceki erişilen bellek konumlarına olacaktır. Dolayısıyla önbellek ana belleğin yakın zamanda kullanılan bazı sözcüklerinin bir kopyasını içermektedir. Bu nedenle uygun şekilde tasarlanan bir önbellek yapısı ile işlemci çok zaman önbellekte zaten mevcut olan bellek sözcüklerini talep edecektir.

Verileri getirmek için sürekli olarak ana belleğe erişilmemesi amacıyla sıkça kullanılan veri önbelleğe kopyalanır. Önbellek düzenli ya da düzensiz olabilir. Her

iki şekilde de veri erişilebilir olmalıdır. Bilgisayar hangi verinin erişilmesi muhtemel olduğunu bilemez, böylece yerellik ilkesini kullanır ve ana belleğe erişmesi gerektiğinde bütün bir bloğu önbellekten ana belleğe aktarır.

Önbellekte zamanda ve alanda yerellik olmak üzere iki tür yerellikten yararlanılır. Zamanda yerellik ilkesine göre, bir öğe bellekten okunduysa, aynı öğenin yeniden okunması olasıdır. Benzer şekilde alanda yerellik ilkesine göre, bir öğe bellekten okunduysa, yakınındaki adreslerdeki öğelerin okunması olasıdır.

Eğer bir blokta kullanılma olasılığı yüksek olan bir başka parça varsa, bu parça bütün bloğun erişim zamanını azaltır. Bu yeni bloğun önbellekteki yeri iki şeye bağlıdır: Önbellek eşleme planı ve önbellek boyutu. Birçok önbellek eşleme planına göre, istenen verinin önbellekte olup olmadığı denetlenmelidir. İstenen verinin yerinin belirlenmesi işlemini basitleştirmek için pek çok önbellek eşleme algoritması kullanılır [26].

3.1. Doğrudan Eşleme

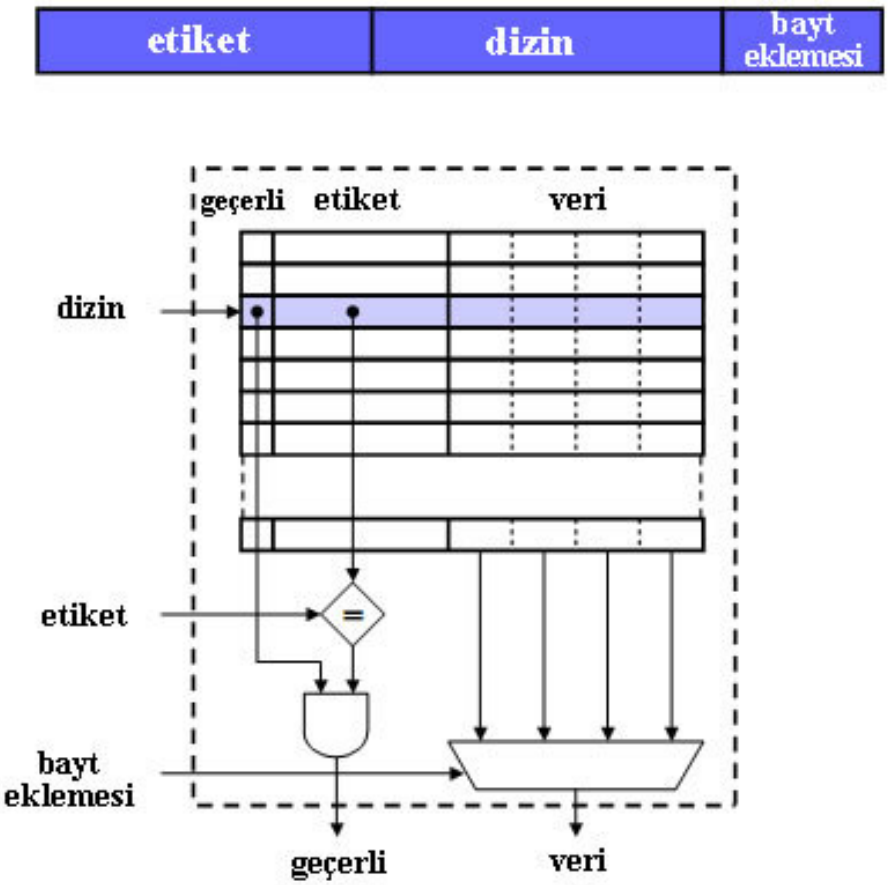
Doğrudan Eşlemede

- Ana belleğin her bloğu ön bellekte yalnızca bir satır ile eşlenir.
- Bu satır

"Önbellek Satır No" = " Ana Bellekteki Blok Sayısı" mod "Önbellek Satır Sayısı "

formülü ile hesaplanır.

Doğrudan eşlemeli önbellek yapısı Şekil 3.1'de ayrıntılı olarak görülmektedir [27].



Şekil 3.1. Doğrudan Eşlemeli Önbellek Yapısı

Her alanın boyutu ana bellek ve önbelleğin fiziksel özelliklerine bağlıdır. Sözcük alanı belirli bir bloktan bir kelimeyi eşsiz bir şekilde tanımlar, bunun için de tahsis edilen sayıda biti içermelidir. Bu blok alanı için de geçerlidir, eşsiz bir önbellek bloğu seçmelidir. Etiket alanı ise kalan kısımdır. Ana belleğin bir bloğu önbelleğe kopyalandığında, bu etiket blokla birlikte saklanır ve eşsiz bir şekilde bu bloğu tanımlar.

Örneğin 2^{14} sözcükten oluşan bir bellek olduğu, önbellekte 16 blok ve her blokta 8 sözcük olduğu varsayılırsa bu bellekte $2^{14} / 2^{3} = 2^{11}$ blok vardır. Her ana bellek adresi 14 bite ihtiyaç duyar. Bu 14 bit adres alanının en sağdaki 3 biti sözcük alanını yansıtır (bir bloktaki 8 kelimenin birini eşsiz bir şekilde tanımlamak için 3 bite ihtiyaç duyulur). Önbellekteki belirli bir bloğu seçmek için 4 bite ihtiyaç duyulur,

böylece blok alanı ortadaki 4 bitten oluşur. Kalan 7 bit etiket alanını oluşturur. Bu alanlar boyutlarıyla birlikte Şekil 3.2'de gösterilmiştir.

7 bit	4 bit	3 bit
Etiket	Dizin	Bayt Eklemesi

Şekil 3.2. Verilen Örnek İçin Ana Bellek Adres Biçimi

Doğrudan eşlenen önbellekte önbellek bloğundan daha fazla ana bellek bloğu bulunduğundan ana bellek blokları önbellek yerleri için yarışmaktadır. Örneğin, eğer önbellek 10 blok içeriyorsa, ana bellekteki blok 0 önbellekteki blok 0'a, ana bellekteki blok 1 önbellekteki blok 1'e, ... , ana bellekteki blok 9 önbellekteki blok 9'a, ana bellekteki blok 10 önbellekteki blok 0'a eşlenir. Bundan dolayı, ana bellekteki blok 0 ve 10 (ve 20, 30 ...) önbellekteki aynı konuma (blok 0)eşlenir.

Ana bellek adresindeki alanlar ve büyüklükleri genel olarak aşağıdaki gibi ifade edilebilir [27]:

- En Az Anlamlı Bitlerin (EAAB) w tanesi bir bloktaki eşsiz bir sözcük veya baytı belirler
- En Anlamlı Bitlerin (EAB) s tanesi ana bellekteki 2^s bloktan birini belirler
    - Etiket alanı s-r (en anlamlı) bitten oluşur.
    - r bitin Satır alanı ise önbelleğin $m=2^r$ tane satırından birini belirler

Örneğin 64 KB lık bir önbellek olduğu,. anabellekle önbellek arasında 4 baytlık bloklar halinde veri aktarımı yapıldığı varsayılırsaönbellek her biri 4 baytlık $16K=2^{14}$ satırdan oluşmaktadır. 16 MB'lık ana bellek ise her biri 4 baytlık 4M bloktan oluşmaktadır. Dolayısıyla $16M= 2^{24}$ olduğundan ana belleği 24-bitlik adresler ile

doğrudan adreslemek mümkündür. Bu bitlerin ilk 2 biti sözcük belirlemede kullanılır. Kalan 24 – 2 = 22 bit ise blok belirlemede kullanılır. Bu bitlerden 14 tanesi önbelleğin satırlarını belirlemede kullanılacağı için kalan 22 – 14 = 8 bit ile etiket alanı belirlenir. Adres alanlarının genel ifadeli boyutları ve bu örnek için her alanda yer alan bit sayıları Şekil 3.3.'te görülmektedir.

Etiket Alanı (s-r)	Satır Alanı (r)	Sözcük Alanı (w)
8 bit	14 bit	2 bit

Şekil 3.3. Genel Adres Alan Boyutları

Doğrudan eşleme yönteminde, eğer yeni bir bloğun yerleştirilmesi gereken önbellek konumu doluysa, bu blok önbellekten silinerek yerine yeni blok yerleştirilir. Daha önceki blokta bir değişiklik olmuşsa ana belleğe geri yazılır, herhangi bir değişiklik olmamışsa üzerine yazılır.

Doğrudan Eşleme Yönteminde [28]:

- Adres Uzunluğu = (s+w) bit
- Adreslenebilen birim sayısı = $2^{(s+w)}$ sözcük veya bayt
- Blok boyutu = Satır boyutu = 2^{w} sözcük veya bayt
- Ana bellekteki blok sayısı = $2^{(s+w)} / 2^{w} = 2^{s}$
- Önbellekteki satır sayısı = m = 2^{r}
- Etiket boyutu = (s-r) bit

3.1.1. Doğrudan Eşleme Yönteminin Artıları ve Eksileri

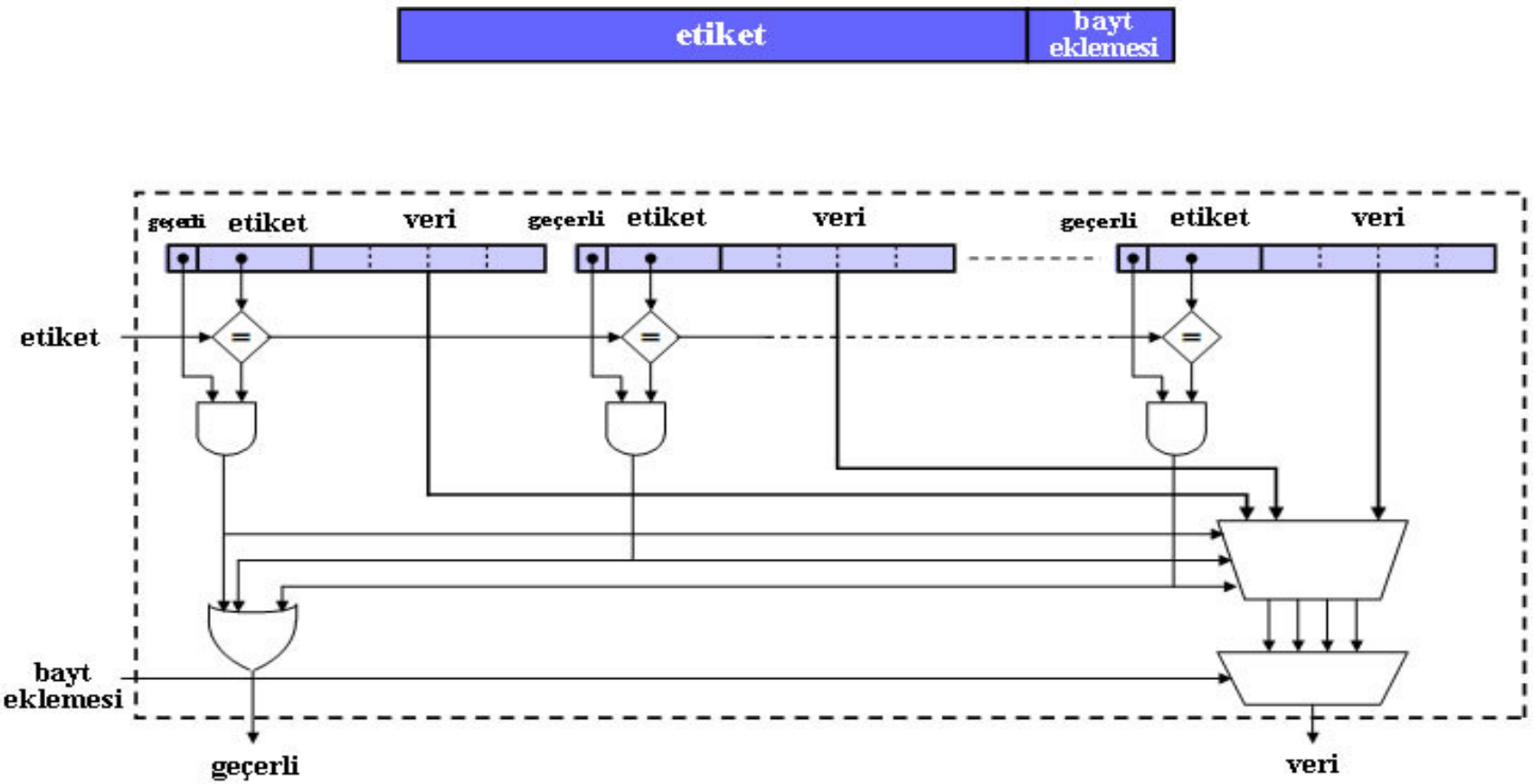
Doğrudan eşlenen önbellek diğer önbellekler kadar pahalı değildir, çünkü eşleme işleminde arama yapmaya gerek yoktur. Her ana bellek bloğunun önbellekte eşlendiği özel bir konum vardır. Bir ana bellek adresi önbellek adresine

çevrildiğinde, işlemci blok alanındaki bitleri inceleyerek bellek bloğu için önbellekte nereye bakacağını anlar [26].

Bu olumlu özelliklerin yanı sıra doğrudan eşleme yönteminin bazı olumsuz yönleri de bulunmaktadır. Örneğin program aynı satıra eşlenen iki bloğa birbiri ardına erişirse, bloklar sürekli olarak birbirinin verisinin üzerine yazmak zorunda kalacağından, bir önceki bloğun yazdığı değer kaybolacak ve aranan verinin önbellekte bulunamaması sorunu oluşacaktır. Bu durum genellikle döngülerde ortaya çıkmaktadır.

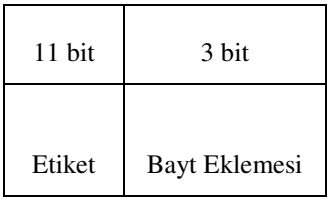
3.2. Tam İlişkili Eşleme

Doğrudan eşleme yönteminde olduğu gibi her ana bellek bloğu için eşsiz bir konum belirmek yerine ana bellek bloklarının önbellekte herhangi bir konuma yerleştirilmesine izin verilir. Bu şekilde eşlenen bir bloğu bulmanın tek yolu tüm önbelleği aramaktır. İstenen veri bloğunun önbellekte olup olmadığını anlamak için istenen etiket ile önbellekteki tüm etiketler karşılaştırılmalıdır. İlişkili eşleme yönteminde ana bellek adresi iki parçaya bölünür: Etiket ve Sözcük. Tam ilişkili önbellek yapısının ayrıntıları Şekil 3.4.'te görülmektedir:



Şekil 3.4.Tam İlişkili Önbellek Yapısı

Örneğin, bir önceki bellek yapılandırması (2^{14} sözcük, 16 bloklu önbellek, 8 sözcükten oluşan bloklar) kullanıldığında, Şekil 3.5'te görüldüğü gibi kelime alanı hala 3 bittir, etiket alanı ise 11 bittir. Bu etiket önbellekteki her blokla birlikte depolanmalıdır. Önbellek belirli bir ana bellek bloğu için arandığında, ana bellek adresinin etiket alanı önbellekteki tüm geçerli etiketlerle karşılaştırılır. Eğer eşi bulunursa blok bulunmuş demektir (Etiket eşsiz olarak bir ana bellek bloğunu ifade etmektedir.). Eğer eş bulunamazsa, blok ana bellekten aktarılmalıdır.



Şekil 3.5. Tam İlişkili Eşleme için Ana Hafıza Adres Biçimi

Tam İlişkili Eşleme Yönteminde [28]:

- Adres Uzunluğu = (s+w) bit
- Adreslenebilen birim sayısı = $2^{(s+w)}$ sözcük veya bayt
- Blok boyutu = Satır boyutu = 2^{w} sözcük veya bayt
- Ana bellekteki blok sayısı = $2^{(s+w)} / 2^{w} = 2^{s}$
- Etiket boyutu = s bit

Tamamen ilişkili eşleme yönteminde doğrudan eşleme yönteminden farklı olarak, önbellek dolu olduğunda hangi bloğu atmak gerektiğine karar vermek için bir yer değişim algoritmasına ihtiyaç duyulur. Genellikle İGİÇ, En Uzun Zamandır Kullanılamayan (EUZK) ve Rastgele yer değişim algoritmalarından biri kullanılır.

3.2.1. Tam İlişkili Eşleme Yönteminin Artıları ve Eksileri

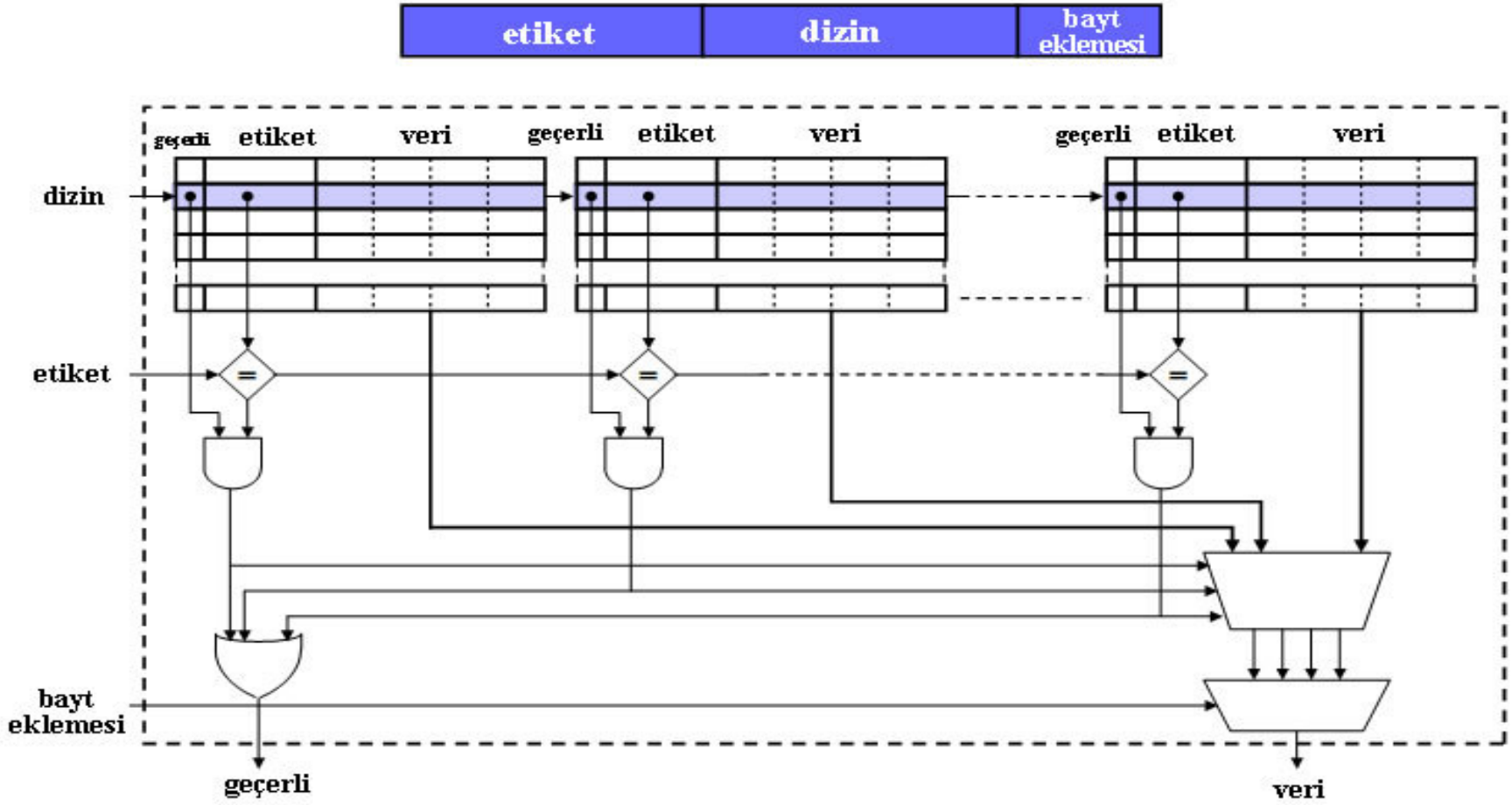
Tam İlişkili Eşlemede, yeni okunan bir blok önbellekte ilk boş bulunan yere yerleştirilebilmektedir. Önbelleğin tamamı dolduktan sonra okunan bir bloğun yerleştirilmesi aşamasında değişik yer değişim algoritmalarından yararlanarak bellekte bulma oranını artırmak mümkündür. İlişkili önbellek doğrudan eşlemeli önbelleğe göre daha hızlıdır.

Yeni okunan her blok önbelleğe yerleştirilmeden o bloğun önbellekte mevcut olup olmadığının araştırılması gerekmektedir. Bu amaçla tüm önbellek etiketlerinin her sefer baştan sona taranması gerekmektedir. Bu süreci hızlandırmak amacıyla paralel karşılaştırıcılar kullanmak mümkün olsa da bu durum oldukça karmaşık devre yapısı gerektirmektedir. Ayrıca blokla birlikte saklamak için özel devre yapısına ek olarak daha büyük bir etikete ihtiyaç duyulur. Bu nedenle, tam ilişkili eşleme yöntemini kullanan önbellekler daha büyük ve pahalıdır.

3.3. Kümeli İlişkili Eşleme

Hızlı ancak karmaşık ve pahalı olan ilişkili önbellek ve ucuz ama son derece kısıtlayıcı özelliklere sahip olan doğrudan eşlemeli önbelleğin her ikisinin de etkili yönlerine sahip, bu yapıların bir birleşimi olan N yollu kümeli ilişkili önbellek eşleme yöntemi bulunmaktadır. Bu yöntemde bir bloğu belirli bir önbellek konumuna eşlemek için doğrudan eşleme yöntemindeki gibi dizinden yararlanılır. Ancak bu yöntemden farklı olarak, adres tek bir önbellek bloğuna eşlenmek yerine birçok önbellek bloğunu içeren bir kümeye eşlenir. Genellikle N sayısı 2, 4, 8 değerlerini almaktadır. Doğrudan eşlenen önbellek, küme boyutunun 1 olan N yollu küme ilişkili önbelleğin bir özel durumudur.

N yollu kümeli ilişkili eşlemede önbellek s satırlık N tane kümeye bölünmektedir. Her kümeye bir blok tahsis edilir ve bu blok o kümedeki herhangi bir satıra eşlenebilir. Kümeli ilişkili önbellek yapısı ayrıntılı olarak Şekil 3.6.'da görülmektedir.



Şekil 3.6. Kümeli İlişkili Önbellek Yapısı

Kümeli İlişkili Eşlemede

- Ana belleğin her bloğu ön bellekte yalnızca bir küme ile eşlenir.
- Bu küme

"Önbellek Küme No" = "Ana Bellekteki Blok Sayısı" mod "Önbellek Satır Sayısı " formülü ile hesaplanır.

Küme ilişkili önbellek yapısında ana bellekteki bir blok ilgili küme içerisindeki her hangi bir satırla eşlenebilir. Ana bellek adresinin küme alanı ile bloğun hangi kümede yer aldığı belirlenir.

Küme ilişkili önbellek eşlemede, ana bellek adresi Şekil 3.7'deki gibi etiket alanı, küme alanı ve sözcük alanı olmak üzere üç parçaya bölünür

Etiket ve sözcük alanları daha önceki görevlerinin aynılarını üstlenir, küme alanı ise ana belleğin eşlendiği önbellek kümelerini gösterir. Yukarıda verilen örnekte olduğu gibi 2^{14} sözcüklük ana bellek ve her bloğu 8 sözcük içeren 16 blokluk önbellekle birlikte 2 yollu küme ilişkili eşleme kullanılırsa. Eğer önbellek toplam 16 bloktan oluşursa ve her kümede 2 blok varsa, önbellekte 8 küme var demektir. Bundan dolayı, küme alanı 3 bit, kelime alanı 3 bit ve etiket alanı 8 bittir.

8 bit	3 bit	3 bit
Etiket	Küme	Bayt Eklemesi

Şekil 3.7. Kümeli İlişkili Eşleme İçin Ana Hafıza Adres Biçimi

Kümeli İlişkili Eşleme Yönteminde [28]:

- Adres Uzunluğu = (s+w) bit
- Adreslenebilen birim sayısı = $2^{(s+w)}$ sözcük veya bayt
- Blok boyutu = Satır boyutu = 2^{w} sözcük veya bayt
- Ana bellekteki blok sayısı = $2^{(s+w)} / 2^{w} = 2^{s}$
- Küme sayısı v = 2^{d}
- Önbellekteki satır sayısı = kv
- Etiket boyutu = (s-d) bit

3.3.1. Kümeli İlişkili Eşleme Yönteminin Artıları ve Eksileri

Kümeli ilişkili eşleme yöntemi doğrudan eşleme yöntemine göre daha esnek bir yöntemdir. Tam ilişkili eşlemeye göre daha küçük etiket alanı içerir ve daha az karmaşık devre yapısına sahiptir. Diğer yönden bu eşleme yöntemi doğrudan eşleme yöntemi kadar basit değildir. Ayrıca doğrudan eşleme yöntemine göre daha büyük etiket alanı içermektedir ve tam ilişkili eşleme yöntemi kadar esnek değildir.

3.4. Yer Değişim Algoritmaları

Önbellek büyüklüğü sınırlı olduğu için bir süre sonra önbellek dolacaktır. Önbellek dolduğu zaman yeni gelen bloğu yazmak için önbellekte yer boşaltılması gerekmektedir. Doğrudan eşleme yönteminde her blok tek bir satırla eşlendiği için seçme şansı bulunmamaktadır ve doğrudan ilgili dizine sahip satıra gidilerek o satırın içeriği yeni gelen blokla değiştirilir. Tam ilişkili ve kümeli ilişkili eşleme yöntemlerinde ise İGİÇ, EUZK, En Az Sıklıkla Kullanılan (EASK) Rastgele v.b pek çok yer değiştirme algoritması kullanılarak yeni gelen blok için önbellekte yer açılabilmektedir.

3.4.1. İlk Giren İlk Çıkar Algoritması

Erişilen bloklar bir kuyruk yapısına eklenir. Kuyrukta en uzun süre beklemiş olan blok çıkarılır. Bu algoritmanın uygulanması, En Uzun Zamandır Kullanılmayan bloğun çıkartıldığı yer değiştirme algoritmasına göre daha basittir.

3.4.2. En Uzun Zamandır Kullanılmayan Algoritması

Bu yöntemde, öncelikle son zamanlarda en az kullanılan, başka bir ifadeyle en uzun süredir erişilmemiş olan blok ile yeni gelen blok yer değiştirir. Algoritmanın son zamanlarda en az kullanılan bloğu attığından emin olmak için neyin ne zaman kullanıldığını takip etmek gerekmektedir. Zamanda yerellik ilkesinden yararlandığı için genellikle başarımı daha yüksektir.

3.4.3. Rastgele Seçim Algoritması

Rastgele seçilen bir blok ile yeni gelen blok yer değiştirilir. Uygulaması kolay bir algoritma olup başarımı değişkenlik göstermektedir.

3.4.4. En Az Sıklıkla Kullanılan

Bu yöntemde bir bloğun ne kadar sıklıkla gerektiği sayılır. İlk önce az sıklıkla kullanılan parçalar çıkarılır. Bu algoritma da yerellik ilkesine dayanmaktadır.

BÖLÜM 4

4. UYGULAMA

Tez çalışması kapsamında bir mikroişlemci benzeştiricisi olan MSIM [29,30] kullanılmıştır. MSIM Joseph Sharkey tarafından, SimpleScalar 3.0d benzeştiricisi üzerinde bir takım değişiklikler yapılarak geliştirilen, sadece Alpha AXP derlenmiş kodlarını desteklemekle birlikte SimpleScalar'ın üzerinde çalışabildiği tüm bilgisayar ortamlarında çalışabilen ve çok kanallı uygulamalara izin veren bir benzeştiricidir [29].

MSIM'de yer alan temel aşamalar işlem sırasına göre aşağıda yer almaktadır:

- Yakala: Komutların yakalandığı aşamadır.
- Yayınla: Komutlatın YK'ye iletildiği aşamadır.
- Yazmaçların Yeniden Adlandırılması: Yakalanan komutlardan Yeniden Adlandırma Kuyruğu'na girenlerin yeniden adlandırıldığı aşamadır. Bu aşamada komutlar önce çözülür, sonra bu komutlara YSB (yükle-sakla komutları için YSK) kaynakları tahsis edilir ve bu komutların yazmaçları yeniden adlandırılır.
- Seçim: Komutların seçimi ve bu komutların işlem birimlerine gönderildiği aşamadır.
- Uyandırma: Beklemekte olan komutlardan hedef yazmaçları hazır hale gelenlerin uyandırıldığı aşamadır.
- Yürütme: Uyandırılan komutların yürütülmeye başlandığı ve bu komutların "Yazma" olaylarının zamanlandığı aşamadır.
- YSK Güncellemesi: Bellek erişim bağımlılıklarının denetlendiği aşamadır. Bu aşamada yükleme komutları için YSK taranarak bellek bağımlılıkları çözülmüş olan hazır durumdaki komutlar belirlenir.

- Yazma: Yürütülmesi tamamlanan işlemlerin sonuçlarının işlem birimlerinden YSB'ye yazıldığı aşamadır.
- YSB İyileştirilmesi: Yanlış tahmin edilmiş olan mimari durumun atıldığı aşamadır.
- Emekliye ayrılma: Bu aşamada YSB ve YSK'deki komutlardan en erken tamamlananların sonuçları mimari yazmaç dosyasına gönderilir ve YSK'deki saklama komutlarının verileri Veri Önbelleğine gönderilir.

MSIM üzerinde SPEC 2000 [31] sınama programlarından crafty, gap ve perlbmk ve eon dışındaki programların tamamı için benzetim gerçekleştirilmiştir. Bu programlar gcc derleyicisi kullanılarak derlenmiştir.

İlk aşama olarak SPEC2000 sınama programlarından yararlanılarak her bir sınama programının benzeştirici üzerinde çalıştırılması sonucunda yazmaçlara atanan değerlerin boyutlarının evreler bazında en küçük, en büyük ve ortalama değerlerini incelemek amacıyla, ilk 1 milyar komut atlandıktan sonra 100 milyon komut çalıştırılmıştır.

100 milyon komutun; yüz, bin, on bin, yüz bin ve bir milyon komutluk evreler halinde çalıştırılması ile her evrede yazmaçlara atanan değerlere yönelik istatistiksel veriler toplanmıştır. Bu veriler ışığında yazmaçlara atanan değerlerin genişlikleri incelenmiştir.

İkinci aşamada ise tahmin birimleri tasarlanmıştır. Bu tahmin birimleri MSIM benzeştirici kodunun Yazmaçların Yeniden Adlandırılması ve Yazma aşamalarında eklemeler yapılarak oluşturulmuştur. Oluşturulan tahmin birimleri ile komutların ürettiği değerlerin genişlikleri tahmin edilmeye çalışılmıştır. Tahmin birimlerinin başarımını incelemek amacıyla her bir sınama programında 1 milyar komut atlandıktan sonra 10 milyon komut çalıştırılmıştır.

Tahmin birimlerinin tasarımında bellek eşleme yöntemlerinden Doğrudan Eşleme, Tam İlişkili Eşleme ve Kümeli İlişkili Eşleme yöntemleri kullanılarak, bu yöntemlerin tahmin biriminin başarımı üzerindeki etkileri incelenmiştir.

Ayrıca tahmin birimlerinin boyutunun tahmin biriminin başarımı üzerindeki etkisini incelemek amacıyla 500, 1000 ve 2000 satırdan oluşan tahmin birimleri kullanılmıştır. Tahmin biriminin boyutunun sınırlı tutulması nedeniyle adresleme yapılırken aynı satıra birden fazla komut yazılması mecburiyeti doğmaktadır. Bu amaçla bellek eşleme yöntemlerinden Tam İlişkili ve Kümeli İlişkili Eşleme yöntemlerinde tahmin biriminde boş satır kalmaması durumunda İGİÇ, EUZK ve Rastgele yer değiştirme yöntemleri kullanılarak tahmin birimlerinde yer boşaltılmaktadır. Benzer şekilde bu yer değiştirme algoritmalarının tahmin biriminin başarımına katkıları incelenmiştir.

4.1. Benzetim Ortamı

Mikroişlemci benzetim ortamları, donanım altyapısında işletilen komutları yazılım seviyesinde yer alan benzetimlerde işleterek, mikroişlemci veriminin ölçülmesi için kullanılan programlardır. Benzeştiricinin parametreleri Çizelge 4.1.'de gösterilmiştir:

Çizelge 4.1. Benzeştirici Yapılandırmaları

Parametre	Yapılandırma
Makine Genişliği	Her vuruşta 4 getir, 4 yayınla, 4 emekliye ayır
Pencere Boyu	32 satır yayın kuyruğu, 8 satır Yükle/Sakla, 128 satır YSB, 128 adet yazmaç
İşlem Birimleri ve Gecikmeler	tamsayı ALU (1/1), yükleme/saklama birimi (2/1), tamsayı çarpma (3/1) tamsayı bölme (20/19), kayan noktalı sayı toplama (2/1) kayan noktalı sayı çarpma (4/1), kayan noktalı sayı bölme (12/12).
L1 KomutÖnbelleği	32 KB, 2-yollu kümeli ilişkili, 32 bayt satır, 1 saat vuruşu isabet zamanı

L1 Veri Önbelleği	32 KB, 4-yollu kümeli ilişkili, 32 bayt satır, 1 saat vuruşu isabet zamanı
L2 Birleştirilmiş Önbellek	512 KB, 8-yollu kümeli ilişkili, 64 bayt satır, 6 saat vuruşu isabet zamanı
BTB	512 satır, 4-yollu kümeli ilişkili
Dallanma Tahmin	2K satır çift doruklu
Hafıza	8 bayt genişlik, ilk bölüm 100 vuruş, bölümler arası 2 vuruş
TLB	16 giriş (Komut) 4-yollu kümeli ilişkili, 32 satır (Veri) 4- yollu kümeli ilişkili, Bulamama gecikmesi 30 vuruş

4.2. Sınama Programları

Standart Başarım Değerlendirme Şirketi (SPEC) en son teknoloji yüksek –başarımlı bilgisayarlara standart yapılı sınama programlarının uygulanarak başarımlarının ölçülmesini sağlamayı hedefleyen kar amacı gütmeyen bir şirkettir.

SPEC CPU sınama programları, SPEC firması tarafından işlemci başarımını ölçmede kullanılmak üzere belirlenmiş 12 tam sayı 14 de kayan nokta programıdır. Bu programlara yönelik daha ayrıntılı bilgi Çizelge 4.2. ve Çizelge 4.3'te bulunmaktadır:

Çizelge 4.2. SPEC2000 Tam Sayı Sınama Programları

Sınama Programı	Dil	Kategori
164.gzip	C	Sıkıştırma
175.vpr	C	FPGA Devre Konum ve Yönlendirmesi
176.gcc	C	C Programlama Dili Derleyicisi
181.mcf	C	Kombinasyonel Optimizasyon
186.crafty	C	Oyun: Satranç
197.parser	C	Kelime İşlemci

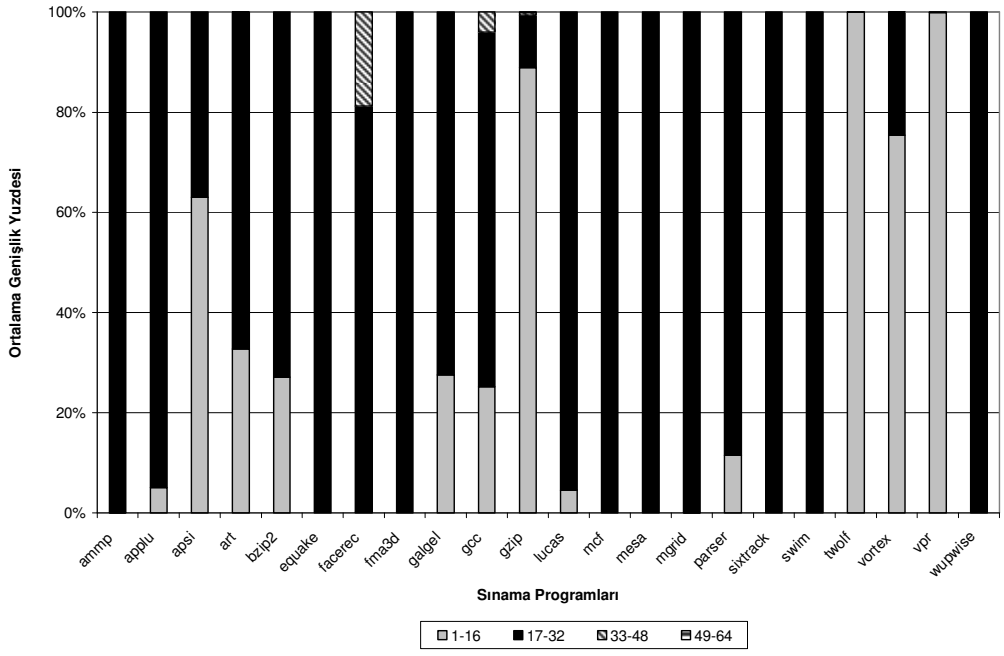
252.eon	C++	Bilgisayar Görsel Canlandırması
253.perlbmk	C	PERL Programlama Dili
254.gap	C	Grup Teorisi, Yorumlayıcı
255.vortex	C	Nesneye dayalı Veritabanı
256.bzip2	C	Sıkıştırma
300.twolf	C	Konum ve Yönlendirme Benzeştiricisi

Çizelge 4.3. SPEC2000 Kayan Nokta Sınama Programları

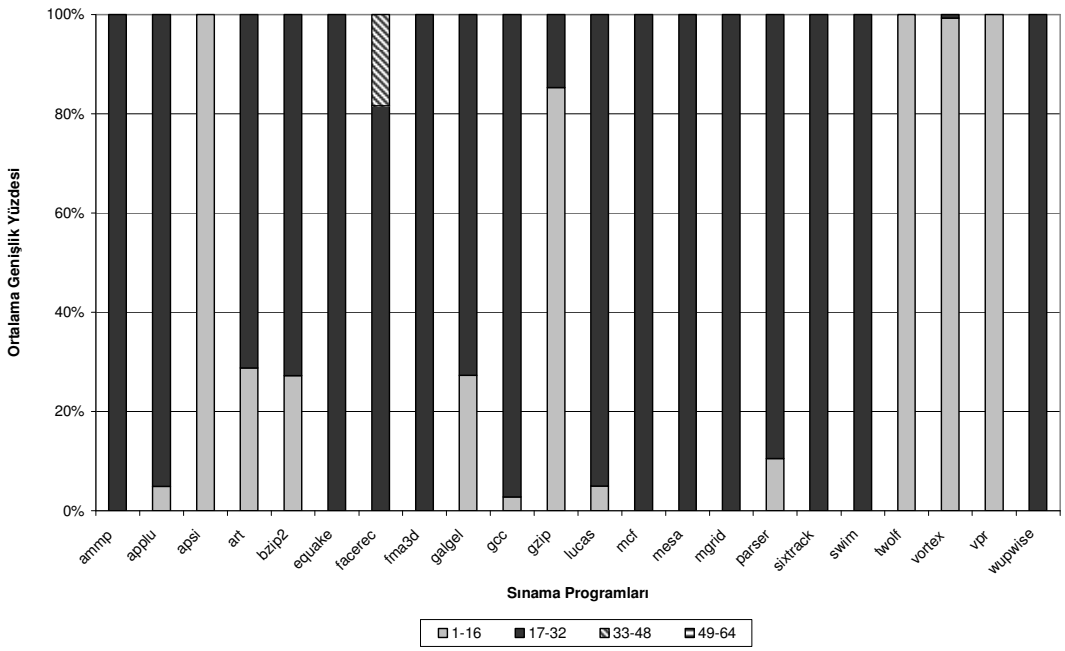
Sınama Programı	Dil	Kategori
168.wupwise	Fortran 77	Fizik / Kuantum Kromodinamik
171.swim	Fortran 77	Sığ Su Modellemesi
172.mgrid	Fortran 77	Çoklu-Izgara Çözümü: 3D Muhtemel Alan
173.applu	Fortran 77	Parabolik / Eliptik Parçalı Diferansiyel Denklemler
177.mesa	C	3-D Grafik Kütüphanesi
178.galgel	Fortran 90	Hesaplamalı Akışkan Dinamiği
179.art	C	Resim Tanıma / Sinir Ağları
183.equake	C	Sismik Dalga Yayma Simülasyonu
187.facerec	Fortran 90	Resim İşleme: Yüz Tanıma
188.ammp	C	Hesaplamalı Kimya
189.lucas	Fortran 90	Sayı Teorisi / Asal Sayı Kontrolü
191.fma3d	Fortran 90	Sonlu Eleman Çökme Benzetimi
200.sixtrack	Fortran 77	Yüksek Enerjili Nükleer Fizik Hızlandırma Tasarımı
301.apsi	Fortran 77	Meteoroloji: Kirletici Madde Dağıtımı

4.3. Benzetim Sonuçları

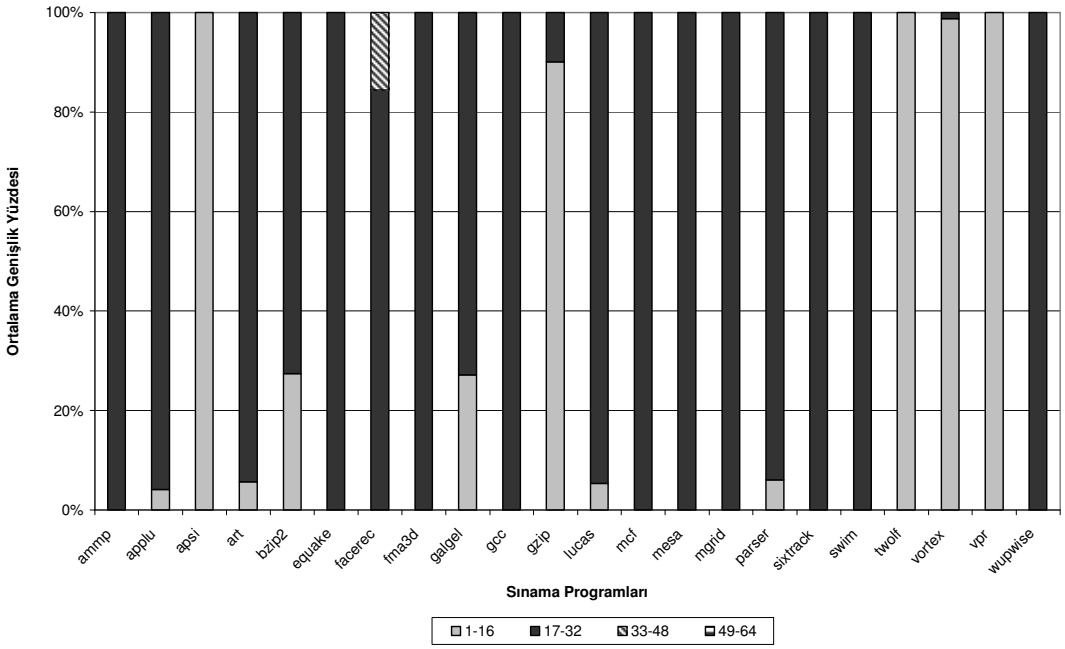
Her evre için, o evrede çalıştırılan tüm komutların yazmaçlara yazılan değerleri kullanılarak o evrenin en küçük, ortalama ve en büyük değerleri hesaplanmıştır. Bu değerlerden ortalama değerlerin evreler bazında sıklık tablosu incelendiğinde en sık tekrarlanan değerlerin 8–35 bit arasında özellikle 19 bit etrafında yoğunlaştı görülmektedir. 10 bin, 100 bin ve 1 milyon komut genişliğindeki evrelerin çalıştırılması ile evre başına hesaplanan ortalama üretilen değer genişlik değerlerinin sıklık tablosu sırasıyla Şekil 4.1, Şekil 4.2 ve Şekil 4.3'te görülmektedir:



Şekil 4.1.10 Bin Komutluk Evrelerdeki Üretilen Değerlerin Genişliklerinin Ortalamasının Sıklık Tablosu



Şekil 4.2. 100 Bin Komutluk Evrelerdeki Üretilen Değerlerin Genişliklerinin Ortalamasının Sıklık Tablosu



Şekil 4.3. 1 Milyon Komutluk Evrelerdeki Üretilen Değerlerin Genişliklerinin Ortalamasının Sıklık Tablosu

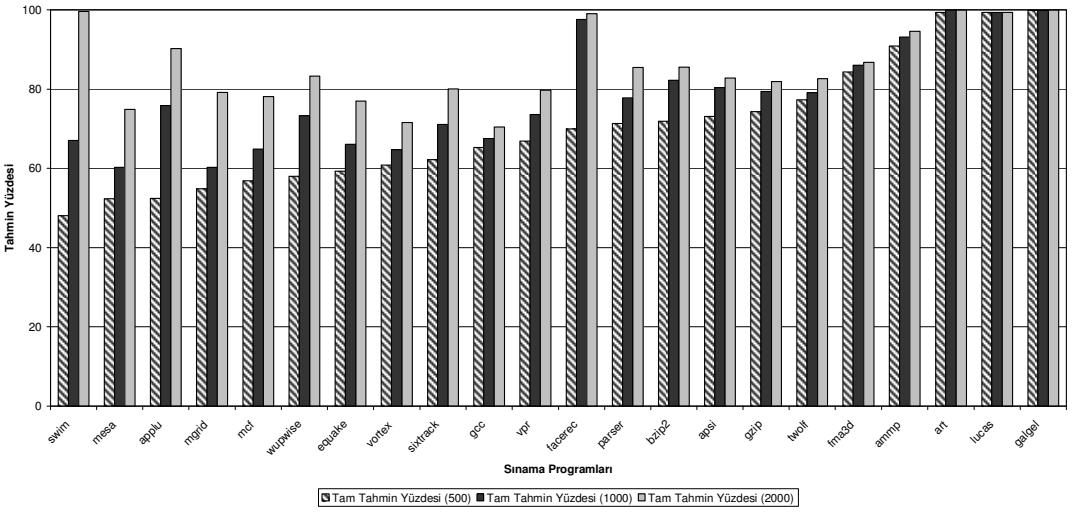
Aynı sınama programının farklı evrelerindeki değerler arasında 2–3 bit fark oluşmaktadır. Oluşan değerlerin büyük bir bölümünün yazmaç büyüklüğünden daha az sayıda bit ile ifade edilebilen dar değerleri içerdikleri gözlemlenmiştir. Bu sonuçlar ışığında 64 bitlik yazmaç yapısının yerine 16 -32 bitlik yazmaç yapıları kullanılmasının işlemci başarımı üzerinde olumlu etkisi olacağı gözlenmiştir. Bu durum aynı zamanda güç tüketimini de oldukça azaltacaktır.

Evre genişliği 10 bin komuttan oluşan ölçümlerde komut sonuçlarının ortalama değerlerin ortalamasının %25,5'inin 1 – 16 bit, %73,5'inin 17 – 32 bit, %1,0'inin 33–48 bit ve % 0,0008'inin 48–64 bitten oluştuğu gözlenmiştir.

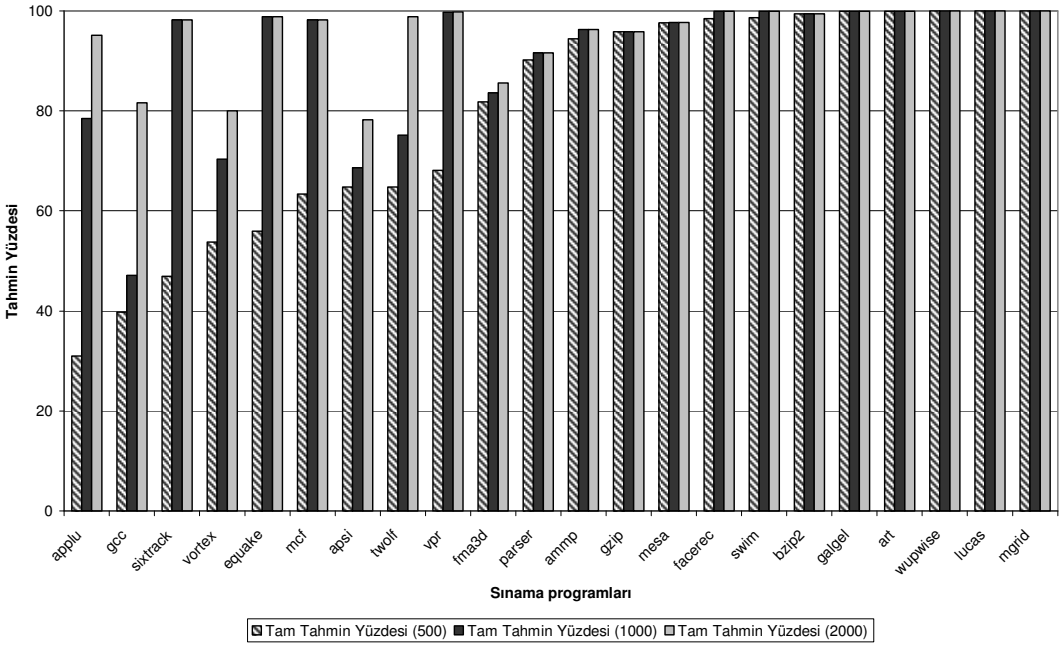
Evre genişliği 100 bin komuttan oluşan ölçümlerde komut sonuçlarının ortalama değerlerin ortalamasının %27,0'sinin 1–16 bit, %72,1'inin 17–32 bit, %0,84'ünün 33–48 bitten oluştuğu ve 48–64 bitlik değer bulunmadığı gözlenmiştir.

Evre genişliği 1 milyon komuttan oluşan ölçümlerde ise komut sonuçlarının ortalama değerlerin ortalamasının %25,6'sının 1–16 bit, %73,3'ünün 17–32 bit, %0,72'sinin 33–48 bitten oluştuğu ve 48–64 bitlik değer bulunmadığı gözlenmiştir.

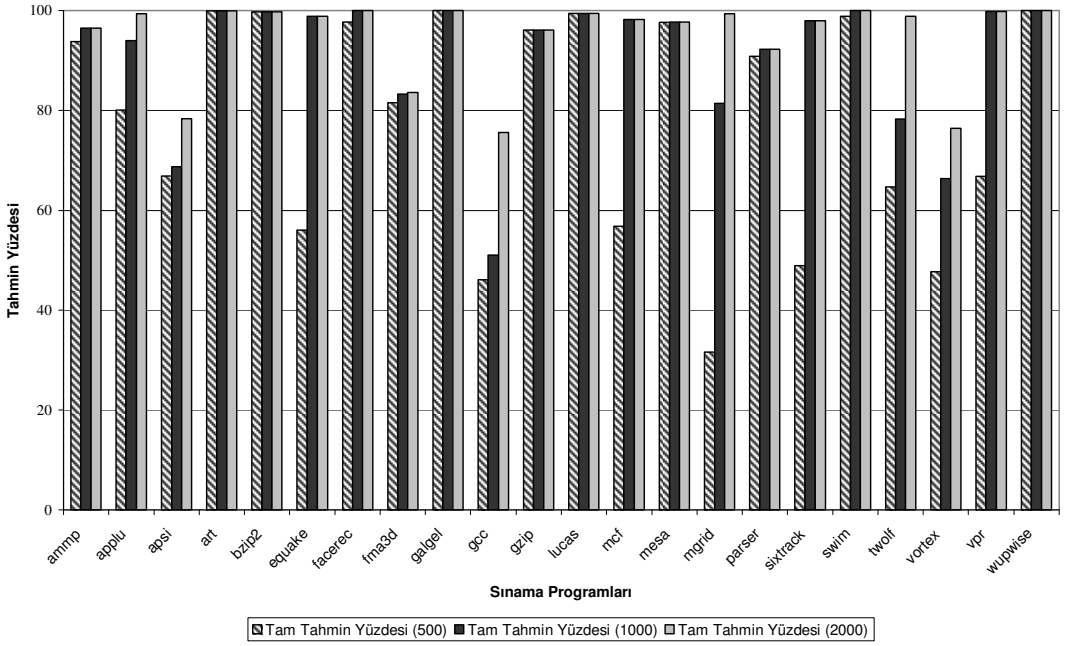
Tahmin birimlerinin tasarımında kullanılan eşleme yöntemleri ile yer değiştirme algoritmalarının ve tahmin birimlerinin boyutunun tahmin birimlerinin başarımı üzerindeki etkisi Şekil 4.4, Şekil 4.5, Şekil 4.6, Şekil 4.7, Şekil 4.8, Şekil 4.9 ve Şekil 4.10'da yer alan grafiklerde görülmektedir:



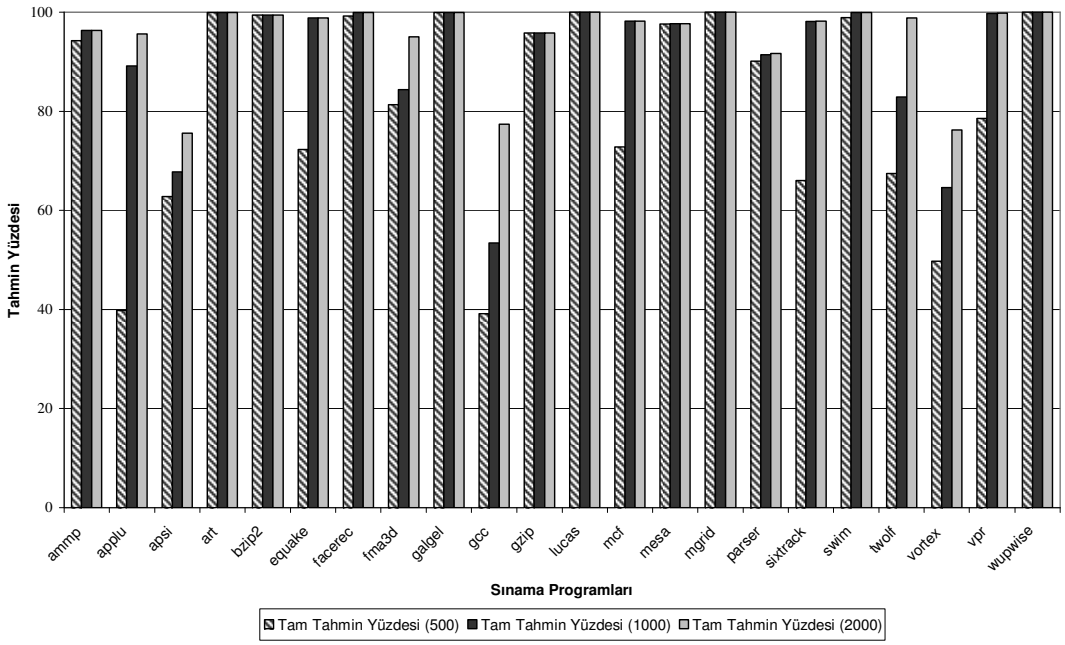
Şekil 4.4. 500,1000 ve 2000 Satırlık Doğrudan Eşlemeli Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları



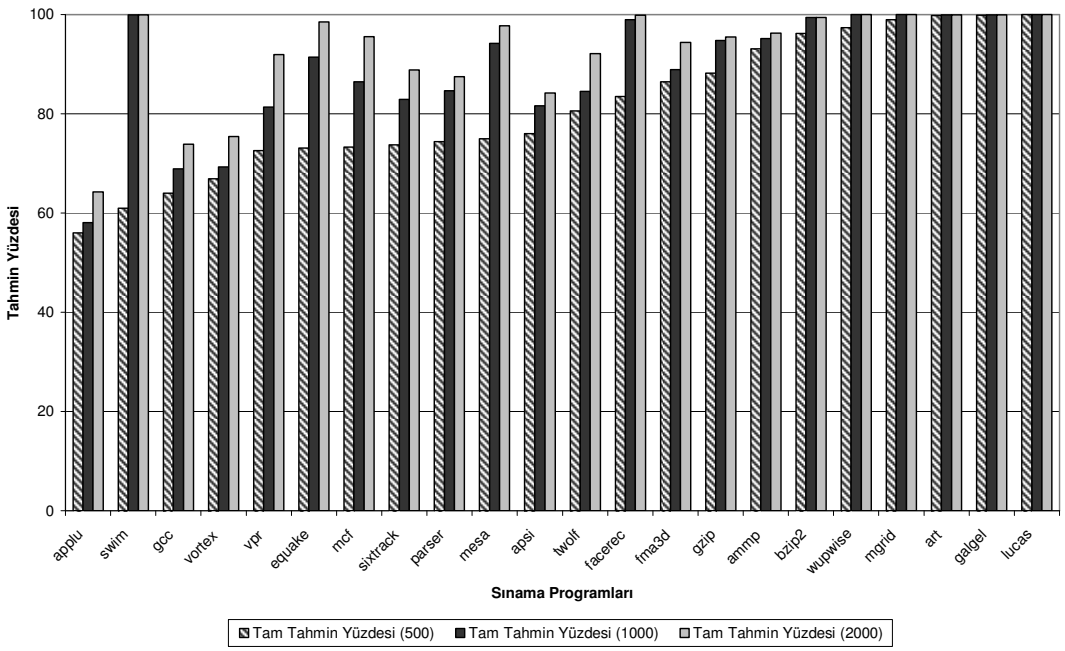
Şekil 4.5. 500, 1000 ve 2000 Satırlık Tam Eşlemeli EUZK Yer Değiştirme Algoritmasının Kullanıldığı Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları



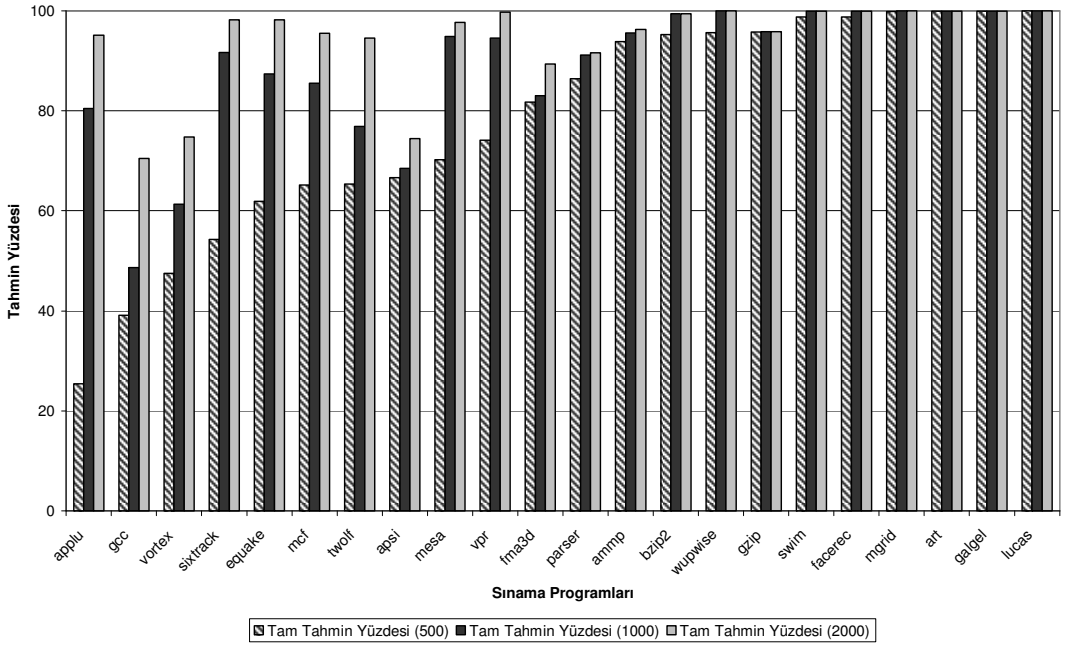
Şekil 4.6. 500,1000 ve 2000 Satırlık Tam Eşlemeli İGİÇ Yer Değiştirme Algoritmasının Kullanıldığı Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları



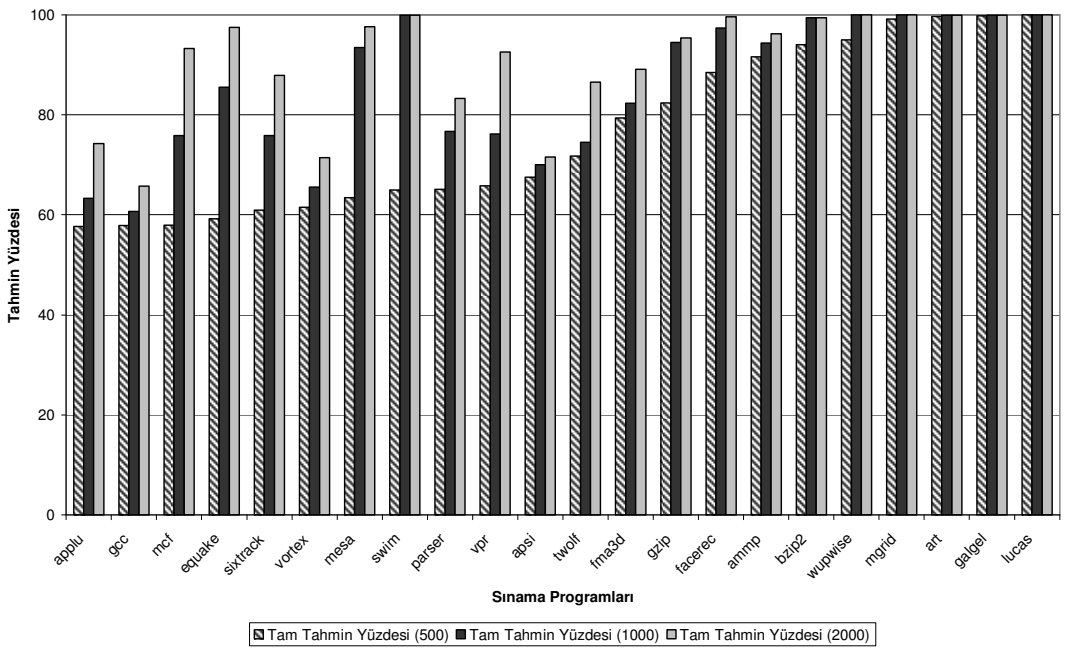
Şekil 4.7. 500,1000 ve 2000 Satırlık Tam Eşlemeli Rastgele Yer Değiştirme Algoritmasının Kullanıldığı Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları



Şekil 4.8. 500,1000 ve 2000 Satırlık Kümeli Eşlemeli EUZK Yer Değiştirme Algoritmasının Kullanıldığı Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları



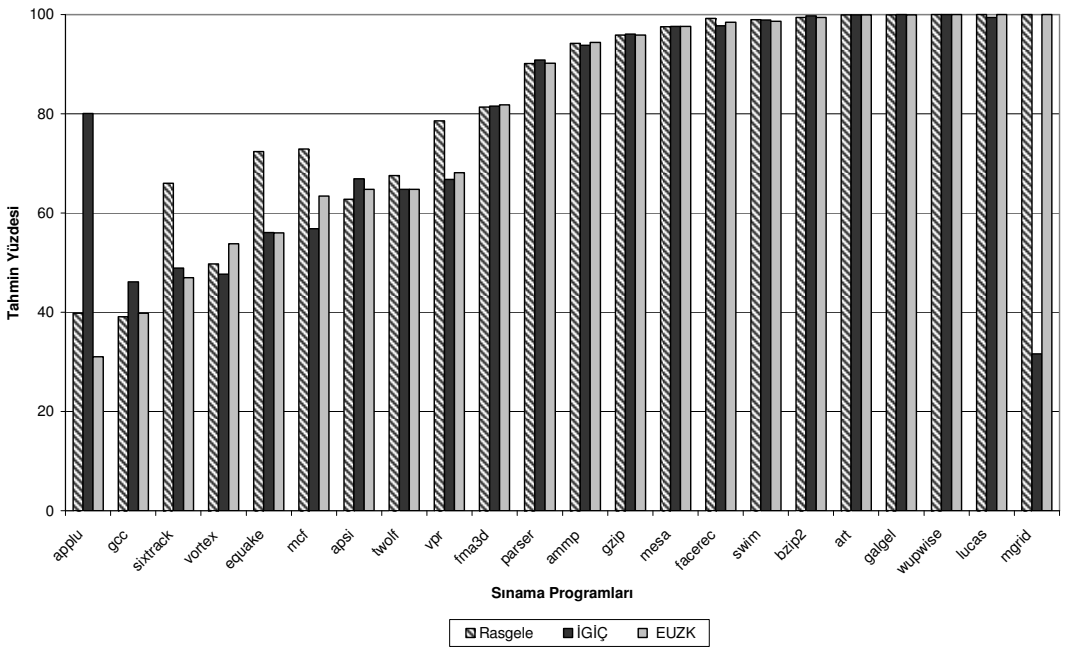
Şekil 4.9. 500,1000 ve 2000 Satırlık Kümeli Eşlemeli İGİÇ Yer Değiştirme Algoritmasının Kullanıldığı Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları



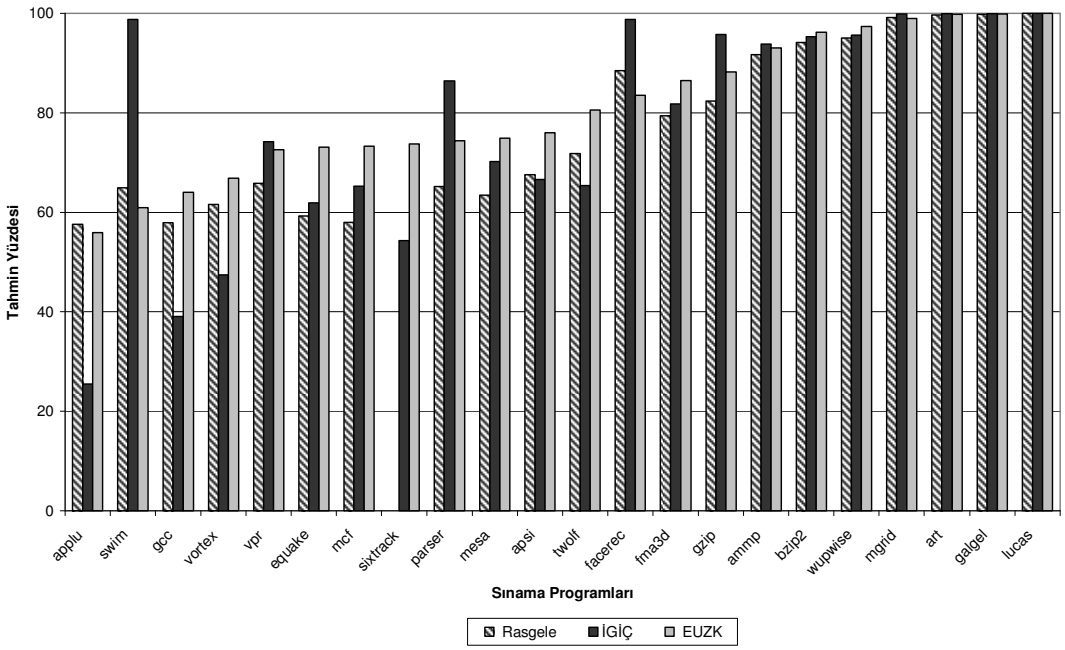
Şekil 4.10. 500,1000 ve 2000 Satırlık Kümeli Eşlemeli Rastgele Yer Değiştirme Algoritmasının Kullanıldığı Tahmin Birimi İle Elde Edilen Tam Tahmin Oranları

Bu grafikler ışığında Doğrudan, Tam İlişkili ve Kümeli İlişkili Eşleme yöntemlerinin ve İGİÇ, EUZK ve Rastgele yer değiştirme algoritmalarının kullanıldığı tahmin birimlerinin tamamında tahmin biriminin başarısının tahmin biriminin boyutu ile doğru orantılı olduğu gözlenmiştir. En yüksek doğru tahmin yüzdesi 2000 satırlık tahmin birimi ile elde edilmiştir.

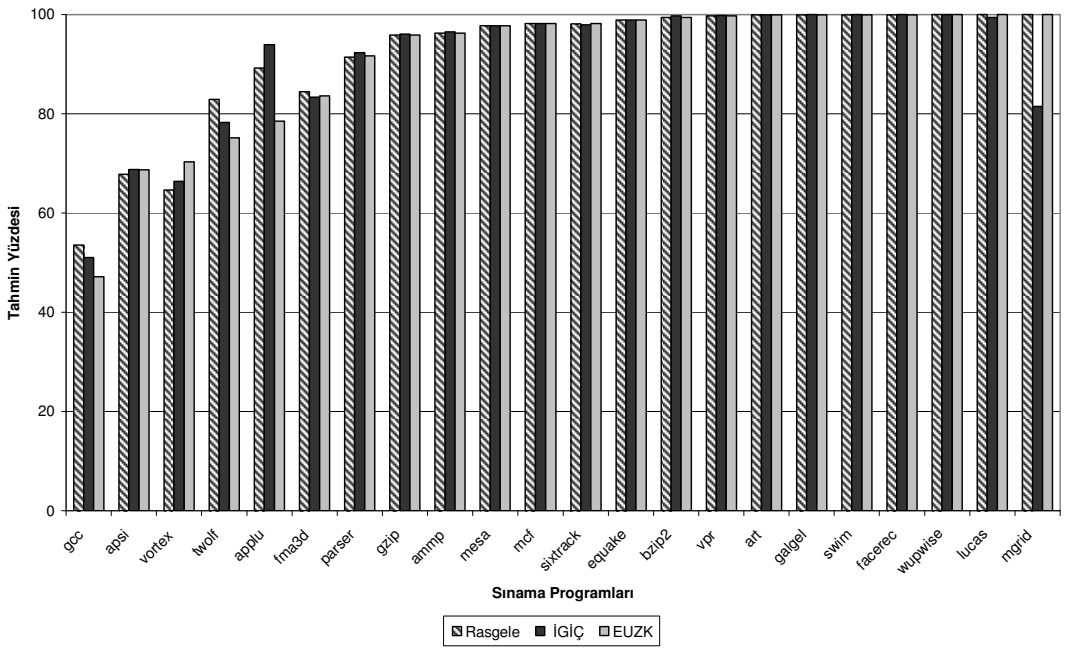
Tam İlişkili Eşlemeli ve Kümeli İlişkili Eşlemeli tahmin birimlerinde kullanılan yer değiştirme algoritmalarının tahmin biriminin boyutu ile olan ilişkileri Şekil 4.11, Şekil 4.12, Şekil 4.13, Şekil 4.14, Şekil 4.15 ve Şekil 4.16'da görülmektedir.



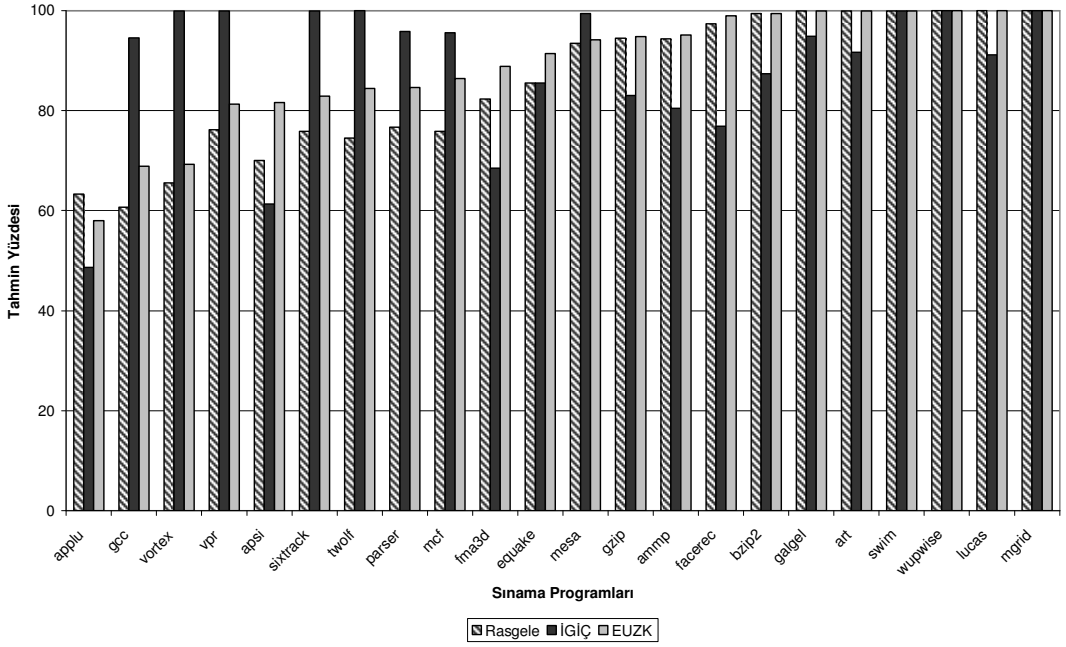
Şekil 4.11. 500 Satırlık Tam İlişkili Eşlemeli Tahmin Biriminin Başarımında Yer Değiştirme Algoritmalarının Etkisi



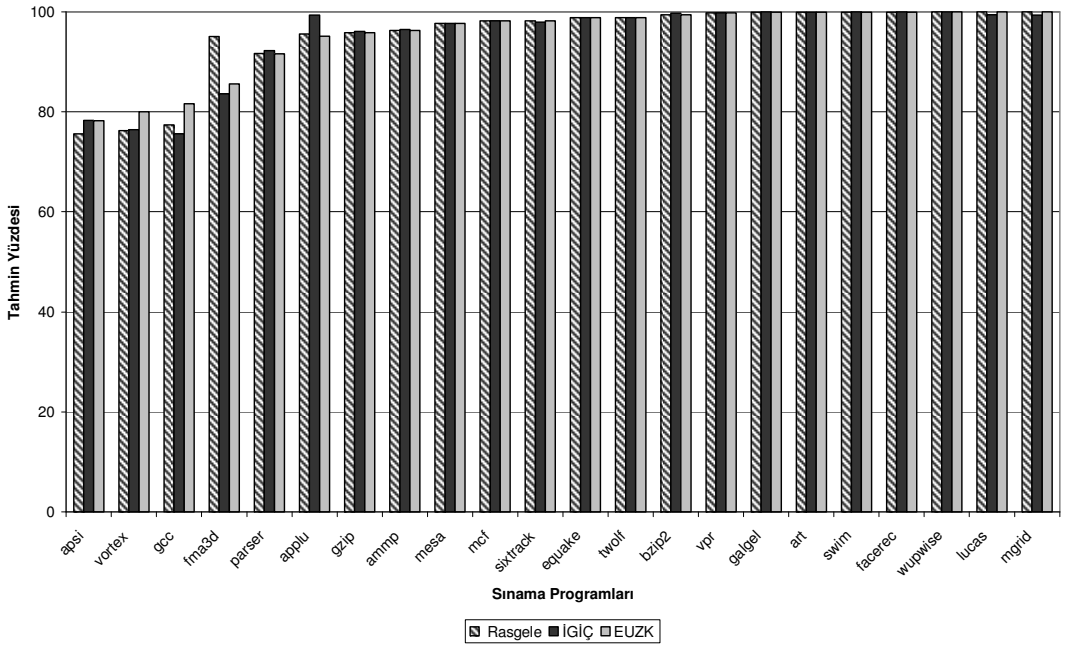
Şekil 4.12. 500 Satırlık Kümeli İlişkili Eşlemeli Tahmin Biriminin Başarımında Yer Değiştirme Algoritmalarının Etkisi



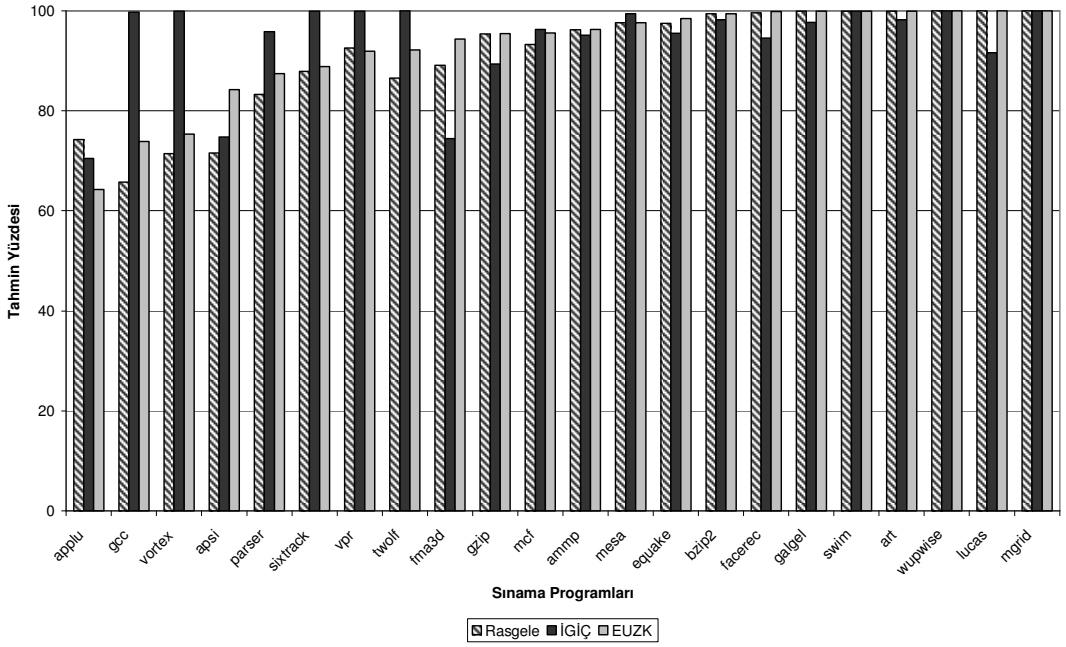
Şekil 4.13. 1000 Satırlık Tam İlişkili Eşlemeli Tahmin Biriminin Başarımında Yer Değiştirme Algoritmalarının Etkisi



Şekil 4.14. 1000 Satırlık Kümeli İlişkili Eşlemeli Tahmin Biriminin Başarımında Yer Değiştirme Algoritmalarının Etkisi



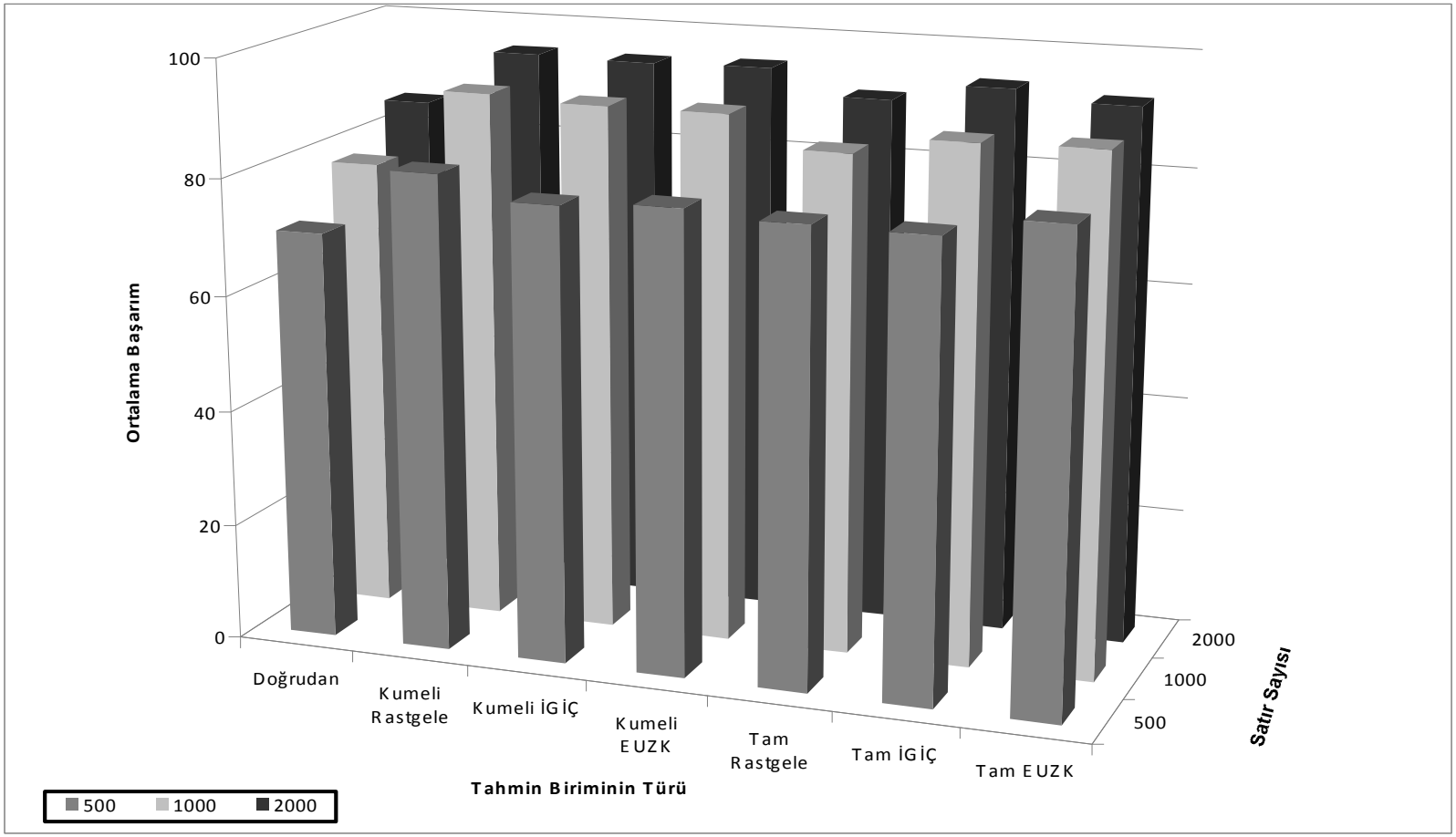
Şekil 4.15. 2000 Satırlık Tam İlişkili Eşlemeli Tahmin Biriminin Başarımında Yer Değiştirme Algoritmalarının Etkisi



Şekil 4.16. 2000 Satırlık Kümeli İlişkili Eşlemeli Tahmin Biriminin Başarımında Yer Değiştirme Algoritmalarının Etkisi

Çizelge 4.4. Tahmin Birimlerinin Özelliklerine Göre Ortalama Tam Tahmin Yüzdeleri

	Satır Sayısına Göre Ortalama TamTahmin Yüzdesi		
Tahmin Biriminin Türü	500	1000	2000
Doğrudan Eşlemeli	70,4	78,2	85,5
Tam Eşlemeli –Rastgele	82,0	91,7	95,2
Tam Eşlemeli –İGİÇ	78,2	90,9	94,9
Tam Eşlemeli –EUZK	79,3	90,9	95,2
Kümeli Eşlemeli –Rastgele	78,2	85,7	91,0
Kümeli Eşlemeli –İGİÇ	78,0	88,8	94,1
Kümeli Eşlemeli –EUZK	81,3	89,1	92,5



Şekil 4.17. 500, 100 ve 2000 Satırlık Tahmin Birimlerinin Tüm Yöntem ve Algoritmalar İçin Elde Ettiği Ortalama Başarım

BÖLÜM 5

5. SONUÇLAR VE ÖNERİLER

Yapılan çalışmada öncelikle işlemcide üretilen değerlerin genişliklerine yönelik istatistik toplanmış, ardında da üretilen bu değerlerin genişliklerini etkili bir şekilde tahmin edecek tahmin birimleri tasarlanmıştır. Çalışmaların ilk aşamasının sonucunda işlemcide üretilen değerlerin ortalamalarının yaklaşık % 26'sının 1-16 bit, % 72'sinin 16-32 bit ve kalan %2-3'ünün ise 33-64 bit genişliğinde değerler olduğu gözlenmiştir. Diğer bir ifadeyle işlemcide üretilen değerlerin ortalamasının yaklaşık % 95'i dar değerlerden oluşmaktadır.

Çalışmanın ikinci aşamasında ise işlemcide üretilen bu dar değerleri doğru olarak tahmin edecek tahmin birimleri tasarlanmıştır. Bu aşamada yedi farklı tahmin birimi tasarlanmıştır. Bu tahmin birimlerinin her birinin 500, 100 ve 2000 birimden oluştuğu durumlar için başaımları incelenmiştir. Tahmin birimlerinin hepsinin başarımının tahmin biriminin büyüklüğü ile doğru orantılı olarak arttığı gözlenmiştir. Tahmin biriminin tasarımında kullanılan eşleme yöntemine bağlı olarak bu başarım oranları %70,4 - %95,2. arasında değişmektedir. Tüm sınama programlarının ortalama tam tahmin değerleri temel alınarak incelendiğinde en düşük başarım oranlarının Doğrudan Eşlemeli, en yüksek başarımın ise Tam İlişkili Eşlemeli EUZK ve Tam İlişkili Eşlemeli Rastgele yer değiştirme algoritmalarının kullanıldığı tahmin birimlerinde elde edildiği gözlenmiştir.

500 satırlık Tam İlişkili tahmin biriminde bütün sınama programları için tüm yer değiştirme algoritmalarının tahmin oranları birbirine oldukça yakın değerler olmakla birlikte, birkaç sınama programında Rasgele ve İGİÇ algoritmaları daha iyi sonuçlar üretmiştir. 500 satırlık Kümeli İlişkili tahmin biriminde bir iki tanesi hariç bütün sınama programları için EUZK yer değiştirme algoritması daha başarılı tahmin oranları elde etmiştir.

1000 satırlık Tam İlişkili tahmin biriminde bütün sınama programları için tüm yer değiştirme algoritmalarının tahmin oranları birbirine oldukça yakın değerler olmakla birlikte, birkaç sınama programında Rasgele ve İGİÇ algoritmaları daha iyi sonuçlar elde etmiştir. 1000 satırlık Kümeli İlişkili tahmin biriminde İGİÇ ve EUZK yer değiştirme algoritmaları başa baş sonuçlar üretmiştir. 2000 satırlık Tam ve Kümeli İlişkili tahmin birimlerinde benzer sonuçlar elde edilmiştir.

İşlemci içinde üretilen değerlerin büyük çoğunluğunun dar değerler olduğu sonucundan yola çıkılarak bu tahmin birimlerinin kullanılması ile literatürde yer alan dar değerlerden yararlanan çalışmaların başarımlarının artırılması mümkündür.

64 bit lik bir yazmacın 16 şar bitlik 4 tane yazmaç parçasının birleşimi olarak tasarlanması gelecek çalışmalar arasında düşünülebilir. Böyle bir yapıda gelen sonuçların boyutları bu çalışmadaki gibi etkili bir tahmin birimi yardımıyla tahmin edilerek, yazmacın tahmin edilen büyüklükteki kısmına yerleştirilebilir. Örneğin 14 bitlik bir değer 16 bitlik kısımdan oluşan bölgeye yazılırken o bölgedeki diğer 3 tane 16 bitlik kısım kapalı tutularak gereksiz yazmaç kullanımını engellemek mümkün olacaktır. Böyle bir uygulama ile hem etkili yazmaç kullanımı sağlamak hem de yazmaç dosyasının güç tüketimini azaltmak mümkün olacaktır.

Yine bu tahmin birimlerinden sırasız yürütüm yapan mikroişlemcilerde çalıştırılan programlardaki dar değerlerin belirlenmesinde yararlanılabilir. Tahmin birimleri yardımıyla dar değerlerin üretildiği evreler belirlenerek o evreler için işlemcinin yazmaç dosyası, YSB gibi birimlerinin kullanılmayan bölümleri kapatılarak işlemcinin güç tüketimi azaltılabilir.

Ayrıca dar değerlerin tahmin edilmesi ile bu dar değerlerden birkaç tanesinin aynı işlem birimi veya yazmaca paketlenmesine yönelik önerilen çalışmaların da başarımlarını artırmak mümkün olacaktır. İki veya daha fazla seviyeden oluşan tahmin birimi veya üretilen sonuçların genişlikleri yerine sonuçların değerlerini

tahmin edecek tahmin birimi tasarımları da gelecek çalışmalar arasında düşünülebilir.

Tasarlanan tahmin birimlerinin içinde tüm sınama programları bazında en yüksek başarım oranı; Tam İlişkili eşlemeli Rasgele yer değiştirme algoritmasının ve Tam İlişkili eşlemeli EUZK yer değiştirme algoritmasının kullanıldığı tahmin birimlerinde elde edilmiştir. Bu tahmin birimlerinin başarımını Tam İlişkili eşlemeli İGİÇ yer değiştirme algoritmasının ve Kümeli İlişkili eşlemeli İGİÇ yer değiştirme algoritmasının kullanıldığı tahmin birimlerinin başarımları takip etmektedir. En düşük başarım oranları ise beklenildiği üzere Doğrudan Eşlemeli tahmin birimlerinde elde edilmiştir.

Tam İlişkili eşleme yönteminin kullanıldığı tahmin birimlerinin başarımları yüksek olmakla birlikte bu yöntemin uygulanması için gerekli olan karmaşık devre yapısı göz önünde bulundurulduğunda, Tam İlişkili eşleme yöntemi yerine Kümeli İlişkili eşleme yönteminin kullanılması daha iyi bir çözüm olabilir. Böylece daha sade bir devre yapısı ve daha az ek donanım ile Tam İlişkili eşleme yönteminin kullanıldığı tahmin biriminin başarımına yakın başarım oranları elde etmek mümkün olacaktır. Kümeli İlişkili eşleme yönteminde yeni okunan her değer tahmin birimine yerleştirilmeden, o değerin tahmin biriminde olup olmadığının araştırılması için geçen süre de Tam İlişkili eşlemeli tahmin birimine göre daha kısa olacağından tahmin birimin hızınının da artırılması sağlanacaktır.

Kümeli İlişkili eşlemeli tahmin birimlerinin başarımları yer değiştirme algoritmaları açısından incelendiği zaman en yüksek başarım oranları sırasıyla İGİÇ, EUZK ve Rasgele yerdeğiştirmeli tahmin birimlerinde elde edilmiştir. Bu yer değiştirme algoritmalarının üçünde de elde edilen başarım oranları birbirlerine oldukça yakın değerler olduğu için her üç algoritma da kullanılabilir. Ancak yine İGİÇ algoritmasının diğer iki algoritmaya göre daha basit ve sade bir tasarımının olması bu algoritmayı tercih edilir kılmaktadır.

KAYNAKLAR

[1] Patterson, D.A., Hennesy, J.L., Computer Organization and Design: The Hardware/Software Interface, Morgan Kaufmann Publishers, 2005.

[2] Ergin, O., 2005, Register File Optimizations For Superscalar Microprocessors, Doktora Tezi, Graduate School of Binghamton University, New York.

[3] Boggs, D., Bakhta, A., Hawkins, J., Marr, D. T., Miller, J.A., Roussel, P., Ronak, S., Toll, B. and Venkatraman, K. S., "The Microarchitecture of the Intel® Pentium® 4 Processor on 90nm Technology", Intel Technology Journal, 8(1), 2004.

[4] Kagan, M., Gochman, S., Orenstien, D. and Lin, D., "MMX™ Microarchitecture of Pentium® Processors with MMX Technology and Pentium® II Microprocessors", Intel Technology Journal, Q3, 1997.

[5] Ören, T., Üney, T., Çölkesen, R., Türkiye Bilişim Vakfı Türkiye Bilişim Ansiklopedisi, Papatya, 2006.

[6] Sohi, G.S., Smith, J.E., "The Microarchitecture of Superscalar Processors", in Proceedings of IEEE, Ağustos 1995.

[7] "Advanced Processor Architecture Course Page" erişi adresi: http://csd.ijs.si/courses/processor/ , erişim tarihi: 12 Mart 2007.

[8] Lipasti, M. H., Mestan, B. and Gunadi, E., "Physical Register Inlining", in Proceedings of International Symposium on Computer Architecture (ISCA-31), 2004.

[9] Brooks, D. and Martonosi, M., "Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance", in Proceedings of the International Symposium on High Performance Computer Architecture (HPCA-5), 1999.

[10] Kagan, M., Gochman, S., Orenstien, D. and Lin, D., "MMX™ Microarchitecture of Pentium® Processors with MMX Technology and Pentium® II Microprocessors", Intel Technology Journal, Q3, 1997.

[11] Gümüşkaya, H., Mikroişlemciler ve Bilgisayarlar, Intel Ailesi ve IBM PC, Alfa, 2004.

[12] "Tekli Komut Çoklu Veri" erişim adresi: http://en.wikipedia.org/wiki/Streaming_SIMD_Extensions, erişim tarihi: 23 Ocak 2007.

[13] Kesheva, J. and Pentovski, V., "Pentium® III Processor Implementation Tradeoffs", Intel Technology Journal, Q2, 1999.

[14] Hinton, G., Sager, D., Upton, M., Boggs, D., Carmean, D., Kyker, A. and Roussel, P., "The Microarchitecture of the Pentium 4 Processor", Intel Technology Journal, Q1, 2001.

[15] Gochman, S., Ronen, R., Anati, I., Berkovits, A., Kurts, T., Naveh, A., Saeed, A., Sperber, Z., Valentine, R. C., "The Intel® Pentium® M Processor: Microarchitecture and Performance", Intel Technology Journal, 7(2), Mayıs 2003.

[16] Boggs, D., Bakhta, A., Hawkins, J., Marr, D. T., Miller, J.A., Roussel, P., Ronak, S., Toll, B. and Venkatraman, K. S., "The Microarchitecture of the Intel® Pentium® 4 Processor on 90nm Technology", Intel Technology Journal, 8(1), 2004.

[17] Loh, G. H., "Exploiting Data-Width Locality to Increase Superscalar Execution Bandwidth", in Proceedings of the International Symposium on Microarchitecture (MIRCO-35), Istanbul, Turkey, Kasım 2002, 395-405.

[18] Ergin, O., Balkan, D., Ghose, K., Ponomarev, D., Register Packing: Exploiting Narrow-Width Operands for Reducing Register File Pressure, 37th International Symposium on Microarchitecture (MICRO'04), 304-315, Portland, USA, 2004.

[19] Gonzalez, R., Cristal, A., Ortega, D., Veidenbaum, A. and Valero, M., "A Content ware Integer Register File Organization", in Proceedings of International Symposium on Computer Architecture (ISCA-31), 2004.

[20] Aggarwal A. and Franklin M., "Energy Efficient Asymmetrically Ported Register Files" in Proceedings IEEE International Conference on Computer Design (ICCD-21), San Jose, USA, Ekim 2003, 2-7.

[21] Kondo, M. and Nakamura, H., "A Small, Fast and Low-Power Register File by Bit-Partitioning", in Proceedings of International Conference on High Performance Computer Architecture (HPCA-11), 2005.

[22] Sato, T. and Arita, I., "Table Size Reduction for Data Value Predictors by Exploiting Narrow Width Values", in Proceedings of International Conference on Supercomputing (ICS-14), 2000

[23] Loh, G. H., "Width Partitioned Load Value Predictors", The Journal of Instruction Level Parallelism (JILP), vol. 5, November 2003, 1-23.

[24] Loh, G. H., "Width Prediction for Reducing Value Predictor Size and Power", in 1st Value Prediction Workshop (VPW-1) held in conjunction with International Symposium on Computer Architecture (ISCA-30), San Diego, USA, June 2002, 86-93.

[25] "Ders Notları" erişim adresi: www.csc.sdstate.edu/~gamradtk/csc317/ csc317l16. pdf, erişim tarihi: Ağustos 2007.

[26] "Ders Notları" erişim adresi: http://www.avgurel.com/bilm252.html, erişim tarihi: Ağustos 2007.

[27] "Ders Notları" erişim adresi: http:// www.ucaribe.edu.mx /archivos/ freyes/IT0105/ seccion03. htm, erişim tarihi: Ağustos 2007.

[28] Stallings, W., Computer Organization and Architecture, Prentice Hall, 2005.

[29] "Mikroişlemci Benzetim Programı" erişim adresi: http://www.cs.binghamton.edu/~jsharke/m-sim/ , erişim tarihi: 24 Aralık 2006

[30] Burger, D. and Austin, T. M., "The SimpleScalar tool set: Version 2.0", Tech. Report, Dept. of CS, Univ. of Wisconsin-Madison, Haziran 1997

[31] "Sınama Programları " erişim adresi http://www.spec.org/ , erişim tarihi: 05 Eylül 2006.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı:	ÜLKER, Hatice Şeyma
Uyruğu:	T.C.
Doğum tarihi ve yeri:	13.01.1982 Ankara
Medeni hali:	Bekar
Telefon:	0 555 700 11 89
e-posta:	sulker@etu.edu.tr

Eğitim

Derece	**Eğitim Birimi**	**Mezuniyet tarihi**
Lisans	Atılım Üniversitesi/Matematik	2004
	Atılım Üniversitesi/Bilgisayar Mühendisliği(ÇAP)	2004

İş Deneyimi

Yıl	**Yer**	**Görev**
2005-2007	TOBB Ekonomi ve Teknoloji Üniversitesi	Araştırma Görevlisi

Yabancı Dil

İngilizce