Logo, company name

Description automatically generated

**TEKNOLOJİ FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ**

**Sistem Programlama – 1. Ödevi**

**Semih Özenç – B200109008**

**Yavuz Selim Şahin – B200109031**

**Muhammet Demirçelik – B200109021**

**BZK.SAU İşlemci Komutları**

Bir komut, komut seti mimarisi olarak tanımlanan işlemcinin tek bir işlemidir. Daha yaygın bir ifadeyle, yürütülen bir programın bir parçası olarak tanımlanır.

BZK.SAU komutları hakkında bilgi almak için Halit Öztekin hocamızın Yüksek Lisans ve Doktora tezleri okundu. Bu tezler arasında birkaç komutun farklı olduğu görüldü.

Oluşturduğum Hash Tablosu , Doktora tezi referans alınarak yapıldı.

BKZ.SAU işlemcisinde komutlar akümülatör ve bellek üzerinde işlem yapan komutlar, indis ve yığın göstergeci ile ilgili komutlar, dallanma ile ilgili komutlar, durum kod kaydedicisi üzerinde işlem yapan komutlar ile giriş-çıkış komutları olmak üzere dört ana bölümde toplanabilir.

İndis ve yığın göstergeci ile ilgili komutlar 8 adettir.

Dallanma komutları 22 adet olup göreceli, direkt, indis ve doğal adresleme modlarını kullanmaktadırlar.

Durum kod kaydedicisi üzerinde işlem yapan komutlar 6 adet olup bu komutların hepsi doğal adresleme modu kullanmaktadır.

Akümülatör ve bellek üzerine işlem yapan komutlar 21 adettir.

Giriş ve çıkış komutları 3 adet olup bu komutlar da doğal adresleme modunu kullanırlar.

**Bellek ve Akümülatör Komutları**

1. **ADD**: Bellekteki veri ile Akümülatördeki veriyi toplayarak akümülatöre kaydeden bir komuttur. Dört farklı adresleme moduna sahiptir.
2. **ADC**: Bellekteki veri, Akümülatördeki veri ve eldeyi toplayarak akümülatöre kaydeden bir komuttur. Dört farklı adresleme moduna sahiptir.
3. **AND**: Bellekteki veri ile Akümülatördeki veriyi lojik AND işlemine tabi tutarak sonucu akümülatöre kaydeden bir komuttur. Dört farklı adresleme moduna sahiptir
4. **CMP**: Bellekteki veri ile Akümülatördeki veriyi karşılaştırarak sonucu akümülatöre kaydeden bir komuttur. Dört farklı adresleme moduna sahiptir.
5. **DIV**: Bellekteki veriyi ile Akümülatördeki veriye bölerek sonucu akümülatöre kaydeden bir komuttur. Dört farklı adresleme moduna sahiptir.
6. **XOR**: Bellekteki veriyi ile Akümülatördeki veriyi lojik XOR işlemine tabi tutarak sonucu akümülatöre kaydeden bir komuttur. Dört farklı adresleme moduna sahiptir.
7. **OR**: Bellekteki veriyi ile Akümülatördeki veriyi lojik OR işlemine tabi tutarak sonucu akümülatöre kaydeden bir komuttur. Dört farklı adresleme moduna sahiptir.
8. **SUB**: Bellekteki veriyi ile Akümülatördeki veriden çıkartma işlemine tabi tutarak sonucu akümülatöre kaydeden bir komuttur. Dört farklı adresleme moduna sahiptir.
9. **SUBC**: Bellekteki veriyi ile Akümülatördeki veriden borç biti ile çıkartma işlemine tabi tutarak sonucu akümülatöre kaydeden bir komuttur. Dört farklı adresleme moduna sahiptir.
10. **MUL**: Bellekteki veriyi ile Akümülatördeki veriyi çarpma işlemine tabi tutarak sonucu akümülatöre kaydeden bir komuttur. Dört farklı adresleme moduna sahiptir.
11. **LDA**: Bellekteki veriyi akümülatöre yükler. Dört farklı adresleme moduna sahiptir.
12. **STA**: Akümülatördeki veriyi belleğe kaydeder. Üç farklı adresleme moduna sahiptir.
13. **CLR**: Akümülatördeki veriyi temizler ve yine akümülatöre kaydeder. Sadece doğal adresleme moduna sahiptir.
14. **DECR**: Akümülatördeki verinin bir eksiğini alarak akümülatöre kaydeder. Doğal adresleme moduna sahiptir.
15. **INCR**: Akümülatördeki verinin bir fazlasını alarak akümülatöre kaydeder. Doğal adresleme moduna sahiptir.
16. **COMR**: Akümülatördeki verinin 1’e tümleyenini alır ve akümülatöre kaydeder. Doğal adresleme moduna sahiptir.
17. **NEGR**: Akümülatördeki verinin 2’e tümleyenini alır ve akümülatöre kaydeder. Doğal adresleme moduna sahiptir.
18. **PSH**: Akümülatördeki veriyi yığın göstergecinin bellekte göstermiş olduğu yere kaydeder. Doğal adresleme moduna sahiptir.
19. **PUL**: Yığın göstergecinin bellekte göstermiş olduğu veriyi akümülatöre kaydeder. Doğal adresleme moduna sahiptir.
20. **SAR**: Akümülatördeki veriyi aritmetik olarak sağa kaydırır ve akümülatöre kaydeder. Doğal adresleme moduna sahiptir.
21. **SAL**: Akümülatördeki veriyi aritmetik olarak sola kaydırır ve akümülatöre kaydeder. Doğal adresleme moduna sahiptir.

**Yığın ve İndeks Kaydedicisi Üzerine İşlem Yapan Komutları**

1. **LDAX**: Bellekteki veriyi indeks kaydedicisine yükler. Dört farklı adresleme moduna sahiptir.
2. **LDAS**: Bellekteki veriyi yığın göstergecine yükler. Dört farklı adresleme moduna sahiptir.
3. **STAX**: İndeks kaydedicisindeki değeri belleğe kaydeder. İki farklı adresleme moduna sahiptir.
4. **STAS**: Yığın göstergecindeki değeri belleğe kaydeder. İki farklı adresleme moduna sahiptir.
5. **DECX**: İndeks kaydedicisindeki veriyi bir azaltarak yine indeks kaydedicisine kaydeder. Sadece doğal adresleme moduna sahiptir.
6. **INCX**: İndeks kaydedicisindeki veriyi bir artırarak yine indeks kaydedicisine kaydeder. Sadece doğal adresleme moduna sahiptir.
7. **DECS**: Yığın göstergecindeki veriyi bir azaltarak yine yığın göstergecine kaydeder. Sadece doğal adresleme moduna sahiptir.
8. **INCS**: Yığın göstergecindeki veriyi bir artırarak yine yığın göstergecine kaydeder. Sadece doğal adresleme moduna sahiptir.

**Sıçrama ve Dallanma Komutları**

1. **BRA**: İşlem kodundan sonra gelen byte’da yer alan veriden etkin adres hesaplanıp, bu adrese şartsız olarak dallanılır.
2. **BCC**: Eğer elde biti sıfır ise hesaplanan etkin adrese dallanılır. Elde biti sıfır değilse bir sonraki komuttan program akışına devam edilir.
3. **BCS**: Eğer elde biti 1 ise hesaplanan etkin adrese dallanılır. Elde biti 1 değilse bir sonraki komuttan program akışına devam edilir.
4. **BZR**: Eğer sıfır biti 1 ise hesaplanan etkin adrese dallanılır. Sıfır biti 1 değilse bir sonraki komuttan program akışına devam edilir.
5. **BGE**: Eğer sıfırdan büyük veya eşitse etkin adrese dallanılır.
6. **BGR**: Eğer sıfırdan büyükse etkin adrese dallanılır.
7. **BHI**: Eğer daha büyük ise etkin adrese dallanılır.
8. **BLE**: Eğer sıfırdan küçük veya eşitse etkin adrese dallanılır.
9. **BLS**: Eğer daha küçük veya eşitse etkin adrese dallanılır.
10. **BLT**: Eğer sıfırdan küçükse etkin adrese dallanılır.
11. **BMI**: Eğer işaret biti 1 ise etkin adrese dallanılır.
12. **BNE**: Eğer sıfır biti 0 ise etkin adrese dallanılır.
13. **BVC**: Eğer taşma biti 0 ise etkin adrese dallanılır.
14. **BVS**: Eğer taşma biti 1 ise etkin adrese dallanılır.
15. **BPL**: Eğer işaret biti 0 ise etkin adrese dallanılır
16. **BSR**: Şartsız olarak hesaplanan etkin adresteki alt programa dallanılır
17. **RTS**: Alt programdan, program akışının kaldığı yere döner.
18. **JMP**: Verilen adrese program akışı kaydırılır. Direkt ve indis adresleme modlarına sahiptir.
19. **JSR**: Alt programdan program akışının kaldığı yere dönülmesini sağlar. Direkt ve indis adresleme modlarına sahiptir.
20. **RTI**: Kesme işlemi bittikten sonra, program akışı kesilmeden önceki durumuna geri döndürür. Doğal adresleme moduna sahiptir.
21. **NOP**: Sadece program sayıcını bir artırır
22. **HLT**: Simülasyonu durdurur

**Durum Kod Kaydedicisi Komutları**

1. **CLC**: Elde bitini sıfırlar.
2. **CLI**: Kesme bitini sıfırlar
3. **CLO**: Taşma bitini sıfırlar.
4. **STC**: Elde bitini 1 yapar.
5. **STI**: Kesme bitini 1 yapar.
6. **STO**: Taşma bitini 1 yapar.

**Giriş-Çıkış Komutları**

1. **IN**: Giriş kaydedicisindeki veriyi akümülatöre kaydeder.
2. **OUT**: Akümülatördeki veriyi çıkış kaydedicisine kaydeder.

BZK.SAU işlemcisinde bulunan komutların adresleme modları ve onların gösterimi

**Table

Description automatically generated**

Aşağıda BZK.SAU işlemcisinin işlemci komutları bulunmaktadır. Tasarlayacağımız Hash Tablosunda aşağıdaki komutlar ve komutların makine kodları bulunacaktır.

Table

Description automatically generated

Table

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generated

**Hash Veri Yapısı (Hash Tablosu) :** Hash tablosu, veriye bir anahtar (key) yardımı ile erişilen basit bir dizi üzerine bina edilmiştir. Anahtar kullanılarak bir indeks üretilir ve bu indeks ile dizideki istenen veriye ulaşılır. Anahtar tekildir yani bir başka kayıtta aynı anahtar olamaz. Ancak veri aynı olabilir.

Hash fonksiyonu, bir anahtar (key) kullanarak tablo üzerinden bir pozisyon (index) üretir. Verilerin tablodaki yerlerini hash fonksiyonu belirler. Hash fonksiyonu girdi olarak anahtarı (key) kullanır ve çıktı olarak hash kod (hash coding) veya hash değeri üretir.

Diagram

Description automatically generated

Hash tablosu, bir anahtar-değer çiftleri koleksiyonunu tutan bir veri yapısıdır. Anahtar-değer çiftleri, belirli bir anahtar değeri tarafından benzersiz olarak tanımlanır ve bir hash işlevi kullanılarak depolanır. Bu, anahtar-değer çiftlerinin arama, ekleme ve silme işlemleri için hızlı ve verimli bir yol sağlar. Hash tablosu, arama yapmak için O(1) zaman karmaşıklığına sahiptir ve ekleme/silme işlemleri için de çoğu durumda O(1) zaman karmaşıklığına sahiptir. Bu nedenle, verilerin depolanması ve aranması hızlı olması gereken durumlarda kullanılır. Örneğin, bir veritabanında birçok öğe aranması gerektiğinde veya bir dil çevirici uygulamasında sözlük kelime eşleştirmesi yapmak için kullanılabilir. Biz bu projede BZK.SAU işlemci komutları ve onların makine kodu karşılığını bulmada kullanacağız.

**Hash tablosunun diğer arama türleriyle karşılaştırılması:**

Lineer arama => O(n)

Binary Arama => O(log n)

Hash Tablosu => O(1) ancak kötü bir hash fonksiyonu kullanıldığında tüm elemanlar aynı hash değerine sahip olabilir ve hash tablosu verimliliği O(n) seviyesine kadar düşebilir.

**Hash Fonksiyonu**

Hash fonksiyonu olarak ilk adımda komuttaki (ADD #) her bir karakterin ASCII karşılığı bulunacak ardından toplanacak ve bir değere atanacaktır. Böyle yapmamızın sebebi hash fonksiyonu için sayısal bir değere ihtiyaç olması gerektiğindendir.

Örnek : ADD # komutu için

A,D,D # ve (boşluk) karakterlerinin Tek tek ASCII karşılıkları hesaplanır.

(65+68+68+35+32 = 268) Bu hesaplanan değer bir değişkenine (a) atanır. Daha sonra bu değer hash fonksiyonuna girer.

Hash fonksiyonumuz: (3\*a+7) % 211

Hash fonksiyonumuz ASCII toplamı bulunmuş komut, aralarında asal olan (3,7) değerleriyle doğrusal bir fonksiyona girer ve çıkan değer komut listenin uzunluğu\*2- komut listenin uzunluğu\*3 aralığı arasında bir değer olan (211) ile modu alınarak bulunur.

Her ihtimale karşı bir çarpışma (collision) olabilir. Yani mod hesaplama sonucu çıkan indislerin aynı adreslere gelmesi durumu. Böyle bir durumda doğrusal adresleme (lineer probing) yapılır. O indis doluysa, bir sonraki boş yere kadar tabloda ilerleyerek yeni konum belirlenir. Böylece tabloda oluşan bir çakışma çözülür ve eleman yerleştirilir.

Ancak bu yöntemin dezavantajı olabilir. Ardışık çakışmalar olması durumunda performans düşer. Çünkü elemanlar birbirine yakın konumlara yerleştirildiği için tablodaki boş yerlerin kullanılması zorlaşır. Bu nedenle lineer probing kullanılırken tablonun yeterince büyük olması ve çakışmaların rastgele bir şekilde oluşması önemlidir.

Hash tablosu boyutunun belirlenmesi, çakışma zincirleri probleminin önlenmesinde önemlidir. Hash tablosu boyutu, veri kümesinin boyutuna ve beklentilere göre belirlenmelidir.

Eğer hash fonksiyonu iyi tasarlanmışsa, veri kümesindeki elemanların dağılımı dengeliyse ve çakışmalar nadir görülüyorsa, tablo boyutu veri kümesinin yaklaşık 2 katı olabilir. Ancak, çakışmaların daha sık olduğu durumlarda, tablo boyutunun veri kümesinin 3 katı veya daha fazlası olması önerilir.

Bununla birlikte, tablo boyutu ne kadar büyük olursa, bellek kullanımı da o kadar artar. Bu nedenle, veri kümesinin boyutu ve çakışmaların sıklığı göz önünde bulundurularak, uygun bir tablo boyutu belirlenmelidir.

Bu sebeplerden dolayı Hash tablomuzun boyunu = komutListesiUzunlugu \* 2 – komutListesiUzunlugu \*3 arasındadır.

Bu tasarımı Java dilinde kodladık. Buna göre bir hashTableData sınıfı oluşturduk. Bu sınıf key, value değerlerini alacak sınıftır. (Key,Value - Komut, Makine Karşılığı)

Bu sınıfın altında komutları eklemek için bir put metodu tasarladık. Put metodu key, value ikilisini alacak, key değerini yani Komutlarımızı myHashFunction() metoduna yollayacaktır.

public void put(K key, V value) {

int hash = myHashFunction(key);

int index = hash;

Entry<K, V> entry = table[index];

if (entry == null) {

table[index] = new Entry<>(key, value);

} else {

while (entry.next != null && !entry.key.equals(key)) {

entry = entry.next;

}

if (entry.key.equals(key)) {

entry.value = value;

} else {

entry.next = new Entry<>(key, value);

}

}

}

myHashFunction() metodu daha önce belirtilen hashing işlemini yapacaktır. Ardından komuta göre hesaplanan hash değeri bir kontrolden geçilecektir. Hash tablosunda böyle bir değer varsa hash değeri bir sonraki indise gidecektir. (çakışma durumunu engellemek için) En son uygun değer bulunduğunda hashDegerini put metoduna geri yollayacaktır.

private int myHashFunction(K key) {

char[] chars = key.toString().toCharArray();

int hashedDeger = 0;

int a = 0;

for (char harf : chars) { // Karakterler üzerinde döngü yap

a += (int) harf; // Karakterin ASCII değerini al

}

hashedDeger = (3 \* a + 7) % 211;

// Eğer aynı hashedDeger'e sahip başka bir eleman varsa, hashDeger'i arttırılır.

while (table[hashedDeger % 211] != null && !table[hashedDeger % 211].key.equals(key)) { //211 yerine size vardı değiştirdim

hashedDeger++;

}

//System.out.println("Indis Adresi " + key.toString() + ": " + hashedDeger);

return hashedDeger;

}

Get() metodu ile herhangi bir komutu tabloda arayabiliriz.

public V get(K key) {

int hash = myHashFunction(key);

int index = hash; //size

Entry<K, V> entry = table[index];

while (entry != null && !entry.key.equals(key)) {

entry = entry.next;

}

if(entry == null)

throw new NullPointerException("Böyle bir değer bulunumadı");

System.***out***.println("Komutun Aranan Değeri : 0x" + entry.value);

return entry != null ? entry.value : null;

}

Yukarıda açıkladığımız metotlar sayesinde hash veri yapımızı ve hash fonksiyonumuzu oluşturduk. Örnek olarak main sınıfımızdan hash tablosuna veri ekleyip veriyi tabloda arayalım.

public static void main(String[] args) {

HashTableData<String, String> hashTableData = new HashTableData<>(211);

hashTableData.put("ADD #", "1000");

hashTableData.put("ADD $", "2000");

hashTableData.get("ADD #");

hashTableData.get("ADD $");

}

Öncelikle sınıfımızı tanımdık. Ardından put metodu ile hash tablomuza “ADD #” komutunu ve komutun makine dili karşılığını “0x1000” ekledik. Aynı şekilde bir komut daha ekledik ve ardından bu komutların get metodu ile tablodan makine karşılıklarını ekrana yazdırdık. Bu kod bloğunun ekran çıktısı aşağıdaki gibidir.

Komutun Aranan Değeri : 0x1000

Komutun Aranan Değeri : 0x2000

Yukarıdaki “ADD #” komutunun yerleşimini inceleyelim. Bu komutun ASCII çıktısı A : 65 , D : 68 , D : 68 , # : 35 ve (boşluk) : 32 değerleri bulunur . Ardından bunlar toplanır 65+ 68 + 68 + 35 + 32 = 268, Hesaplanan 268 değeri için hash değeri hesaplanır.

a değeri = 268

(3\*a+7) % 211 , (3\*268+7) = 811 değeri bulunur.

811 %211 = 178 hesaplanır. Yani “ADD #” komutu ve makine karşılığı 178 numaralı indisin boş olması durumunda buraya yerleştirilecektir. Çalıştırdığımız kodda da bu indis boş olduğundan buraya yerleştirmiş olduk.

ADD komutu

Indis Adresi ADD #: 178

Komutun Aranan Değeri : 0x1000

Tüm Komutlar Eklendikten Sonra Hash Tablosu Yerleşim Düzeni

Komutun Aranan Değeri çıktısı komutun adı kullanılarak alınmıştır. Key ile – Value değerine ulaşılmış ve ekrana yazılmıştır.

ADD komutu

Indis Adresi ADD #: 178

Komutun Aranan Değeri : 0x1000

Indis Adresi ADD $: 181

Komutun Aranan Değeri : 0x2000

Indis Adresi ADD @: 54

Komutun Aranan Değeri : 0x3000

Indis Adresi ADD %: 184

Komutun Aranan Değeri : 0x4000

ADDC komutu

Indis Adresi ADDC #: 168

Komutun Aranan Değeri : 0x1001

Indis Adresi ADDC $: 171

Komutun Aranan Değeri : 0x2001

Indis Adresi ADDC @: 44

Komutun Aranan Değeri : 0x3001

Indis Adresi ADDC %: 174

Komutun Aranan Değeri : 0x4001

AND komutu

Indis Adresi AND #: 208

Komutun Aranan Değeri : 0x1002

Indis Adresi AND $: 0

Komutun Aranan Değeri : 0x2002

Indis Adresi AND @: 84

Komutun Aranan Değeri : 0x3002

Indis Adresi AND %: 3

Komutun Aranan Değeri : 0x4002

CMP komutu

Indis Adresi CMP #: 36

Komutun Aranan Değeri : 0x1004

Indis Adresi CMP $: 39

Komutun Aranan Değeri : 0x2004

Indis Adresi CMP @: 123

Komutun Aranan Değeri : 0x3004

Indis Adresi CMP %: 42

Komutun Aranan Değeri : 0x4004

DIV komutu

Indis Adresi DIV #: 45

Komutun Aranan Değeri : 0x1006

Indis Adresi DIV $: 48

Komutun Aranan Değeri : 0x2006

Indis Adresi DIV @: 132

Komutun Aranan Değeri : 0x3006

Indis Adresi DIV %: 51

Komutun Aranan Değeri : 0x4006

XOR komutu

Indis Adresi XOR #: 111

Komutun Aranan Değeri : 0x1007

Indis Adresi XOR $: 114

Komutun Aranan Değeri : 0x2007

Indis Adresi XOR @: 198

Komutun Aranan Değeri : 0x3007

Indis Adresi XOR %: 117

Komutun Aranan Değeri : 0x4007

LDA komutu

Indis Adresi LDA #: 202

Komutun Aranan Değeri : 0x100B

Indis Adresi LDA $: 205

Komutun Aranan Değeri : 0x200B

Indis Adresi LDA @: 78

Komutun Aranan Değeri : 0x300B

OR komutu

Indis Adresi OR #: 58

Komutun Aranan Değeri : 0x100C

Indis Adresi OR $: 61

Komutun Aranan Değeri : 0x200C

Indis Adresi OR @: 145

Komutun Aranan Değeri : 0x300C

Indis Adresi OR %: 64

Komutun Aranan Değeri : 0x400C

SUB komutu

Indis Adresi SUB #: 66

Komutun Aranan Değeri : 0x1011

Indis Adresi SUB $: 69

Komutun Aranan Değeri : 0x2011

Indis Adresi SUB @: 153

Komutun Aranan Değeri : 0x3011

Indis Adresi SUB %: 72

Komutun Aranan Değeri : 0x4011

SUBC komutu

Indis Adresi SUBC #: 56

Komutun Aranan Değeri : 0x1012

Indis Adresi SUBC $: 59

Komutun Aranan Değeri : 0x2012

Indis Adresi SUBC @: 143

Komutun Aranan Değeri : 0x3012

Indis Adresi SUBC %: 62

Komutun Aranan Değeri : 0x4012

LDAX komutu

Indis Adresi LDAX #: 46

Komutun Aranan Değeri : 0x1015

Indis Adresi LDAX $: 47

Komutun Aranan Değeri : 0x2016

Indis Adresi LDAX @: 131

Komutun Aranan Değeri : 0x3015

Indis Adresi LDAX %: 50

Komutun Aranan Değeri : 0x4015

LDAS komutu

Indis Adresi LDAS #: 29

Komutun Aranan Değeri : 0x1016

Indis Adresi LDAS $: 32

Komutun Aranan Değeri : 0x2016

Indis Adresi LDAS @: 116

Komutun Aranan Değeri : 0x3016

Indis Adresi LDAS %: 35

Komutun Aranan Değeri : 0x4016

MUL komutu

Indis Adresi MUL #: 79

Komutun Aranan Değeri : 0x1014

Indis Adresi MUL $: 81

Komutun Aranan Değeri : 0x2014

Indis Adresi MUL @: 165

Komutun Aranan Değeri : 0x3014

Indis Adresi MUL %: 85

Komutun Aranan Değeri : 0x4014

CLR komutu

Indis Adresi CLR: 49

Komutun Aranan Değeri : 0x0003

DECR komutu

Indis Adresi DECR: 21

Komutun Aranan Değeri : 0x0005

INCR komutu

Indis Adresi INCR: 63

Komutun Aranan Değeri : 0x0008

COM komutu

Indis Adresi COM: 43

Komutun Aranan Değeri : 0x0009

NEG komutu

Indis Adresi NEG: 28

Komutun Aranan Değeri : 0x000A

PSH komutu

Indis Adresi PSH: 80

Komutun Aranan Değeri : 0x000D

PUL komutu

Indis Adresi PUL: 97

Komutun Aranan Değeri : 0x000E

SAR komutu

Indis Adresi SAR: 65

Komutun Aranan Değeri : 0x000F

SAL komutu

Indis Adresi SAL: 52

Komutun Aranan Değeri : 0x0010

DECX komutu

Indis Adresi DECX: 40

Komutun Aranan Değeri : 0x0019

INCX komutu

Indis Adresi INCX: 82

Komutun Aranan Değeri : 0x001A

DECS komutu

Indis Adresi DECS: 24

Komutun Aranan Değeri : 0x001B

INCS komutu

Indis Adresi INCS: 67

Komutun Aranan Değeri : 0x001C

HLT komutu

Indis Adresi HLT: 70

Komutun Aranan Değeri : 0x0032

IN komutu

Indis Adresi IN: 38

Komutun Aranan Değeri : 0x0039

OUT komutu

Indis Adresi OUT: 118

Komutun Aranan Değeri : 0x003A

RTI komutu

Indis Adresi RTI: 91

Komutun Aranan Değeri : 0x0030

RTS komutu

Indis Adresi RTS: 121

Komutun Aranan Değeri : 0x002D

CLC komutu

Indis Adresi CLC: 4

Komutun Aranan Değeri : 0x0033

CLI komutu

Indis Adresi CLI: 22

Komutun Aranan Değeri : 0x0034

CLV komutu

Indis Adresi CLV: 68

Komutun Aranan Değeri : 0x0035

STC komutu

Indis Adresi STC: 76

Komutun Aranan Değeri : 0x0036

STI komutu

Indis Adresi STI: 94

Komutun Aranan Değeri : 0x0037

STV komutu

Indis Adresi STV: 133

Komutun Aranan Değeri : 0x0038

NOP komutu

Indis Adresi NOP: 86

Komutun Aranan Değeri : 0x0031

BRA \* komutu

Indis Adresi BRA \*: 25

Komutun Aranan Değeri : 0x501D

BCC \* komutu

Indis Adresi BCC \*: 196

Komutun Aranan Değeri : 0x501E

BCS \* komutu

Indis Adresi BCS \*: 33

Komutun Aranan Değeri : 0x501F

BZR \* komutu

Indis Adresi BZR \*: 99

Komutun Aranan Değeri : 0x5020

BGE \* komutu

Indis Adresi BGE \*: 5

Komutun Aranan Değeri : 0x5021

BGR \* komutu

Indis Adresi BGR \*: 53

Komutun Aranan Değeri : 0x5022

BHI \* komutu

Indis Adresi BHI \*: 18

Komutun Aranan Değeri : 0x5023

BLE \* komutu

Indis Adresi BLE \*: 19

Komutun Aranan Değeri : 0x5024

BLS \* komutu

Indis Adresi BLS \*: 60

Komutun Aranan Değeri : 0x5025

BLT \* komutu

Indis Adresi BLT \*: 71

Komutun Aranan Değeri : 0x5026

BMI \* komutu

Indis Adresi BMI \*: 34

Komutun Aranan Değeri : 0x5027

BNE \* komutu

Indis Adresi BNE \*: 26

Komutun Aranan Değeri : 0x5028

BVC \* komutu

Indis Adresi BVC \*: 55

Komutun Aranan Değeri : 0x5029

BVS \* komutu

Indis Adresi BVS \*: 90

Komutun Aranan Değeri : 0x502A

BPL \* komutu

Indis Adresi BPL \*: 57

Komutun Aranan Değeri : 0x502B

BSR \* komutu

Indis Adresi BSR \*: 83

Komutun Aranan Değeri : 0x502C

STA komutu

Indis Adresi STA $: 73

Komutun Aranan Değeri : 0x2013

Indis Adresi STA @: 147

Komutun Aranan Değeri : 0x3013

Indis Adresi STA %: 74

Komutun Aranan Değeri : 0x4013

STAX komutu

Indis Adresi STAX $: 119

Komutun Aranan Değeri : 0x2017

Indis Adresi STAX @: 200

Komutun Aranan Değeri : 0x3017

STAS komutu

Indis Adresi STAS $: 101

Komutun Aranan Değeri : 0x2018

Indis Adresi STAS @: 185

Komutun Aranan Değeri : 0x3018

JMP komutu

Indis Adresi JMP $: 75

Komutun Aranan Değeri : 0x102E

Indis Adresi JMP @: 144

Komutun Aranan Değeri : 0x202E

Indis Adresi JMP %: 77

Komutun Aranan Değeri : 0x402E

JSR komutu

Indis Adresi JSR $: 87

Komutun Aranan Değeri : 0x102F

Indis Adresi JSR %: 88

Komutun Aranan Değeri : 0x402F

BPO \* komutu

Indis Adresi BPO \*: 89

Komutun Aranan Değeri : 0x502D

BPE \* komutu

Indis Adresi BPE \*: 30

Komutun Aranan Değeri : 0x502E