
# *BLM 101- Bilgisayar Mühendisliğine Giriş*
# *2019-2020 Güz Dönemi*

## *FB-CPU V1*

## *Proje Teslim Raporu*
## *6 Ocak 2020*

***Hasan Demir , İlhan Ersoy , Cüneyt Balcı , Mehmet Çolak***

# 1 Giriş

## 1.1 Projenin Amacı

FB-CPU isminde bir işlemcinin tasarımı ve tasarlanan işlemci üzerinde makine dili ile yazılan çeşitli kod parçacıklarını yazmak. Proje sonunda basit bir işlemcideki RAM, Kontrol Ünitesi ve Saklayıcıların bir arada çalıştırıp, makine dilindeki kod parçacıklarını nasıl yürütebildiğini gözlemlemek.

## 1.2 Proje Ekibi

### Hasan Demir 190301004

17.01.2000 tarihinde dünyaya geldi 2018 yılında Burdur Ercan Akın Fen Lisesi'nden mezun oldu. Şu anda Fenerbahçe Üniversitesi'nde lisans eğitimi görmekte almakta.
C diliyle ilgileniyor.

### İlhan Ersoy 190301024

07.02.2000 yılında dünyaya geldi.2018 yılında Birey Temel Lisesi'nden mezun oldu.Şuanda Fenerbahçe Üniversitesi Bilgisayar Mühendisliği bölümünde lisans eğitimi almakta. C ve javaScript dilleriyle ilgileniyor.

### Cüneyt Balcı 190301019

28.08.2000 yılında doğdu.2018 yılında Maltepe Final Temel Lisesi'nden mezun oldu .Şuanda Fenerbahçe Üniversitesi Bilgisayar Mühendisliği bölümünde lisans eğitimi almakta. C ve Java dilleriyle ilgileniyor.

### Mehmet Çolak 190301022

10.07.2001 yılında Kadiköy'de dünyaya geldi. 2018 yılında Kadiköy Nazmi Arıkan Fen Bilimleri Temel Lisesi'nden mezun oldu. Şu anda Fenerbahçe Üniversitesi Bilgisayar Mühendisliği bölümünde lisans eğitimi almakta. C ve benzeri dillerle ilgilenmektedir.

# 2 Sistem Mimarisi

## 2.1 Kullanılan Araçlar

Von Neumann Simulatörü , Logisim-Evolution

## 2.2 Tasarım

FBÜ CPU durum makinasını gerçeklemeye çalıştık.

Durum saklayıcısının aldığı değere göre farklı komutlar çalışıyor.

Durum saklayıcısı 0'a eşit ise; MAR(Memory Adress Register) saklayıcısındaki veri PC(Program Counter)'a besleniyor. Sonra Ram Write(RamWR) sinyali 0 besleniyor ve durum saklayıcısı 1 artırılıyor.

Durum saklayıcısı 1'e eşit ise:

IR(Instruction Register) saklayıcısındaki veri MDR(Memory Data Register)'a besleniyor. PC(Program Counter) saklayıcısındaki değer 1 artırılıyor ve durum saklayıcısındaki değer de 1 artırılıyor.

Durum saklayıcısı 2'ye eşit ise:

IR'ın 6'dan 9'a kadar olan bitlerinin değerinin 6'dan küçük olup olmadığına bakılıyor. Eğer 6'dan küçük ise IR'ın 0'dan 5'e olan bitleri MAR'a besleniyor ve durum 3'e eşitleniyor.

Eğer 6'dan 9'a kadar olan bitlerinin değeri 6'dan küçük değil ise 6'ya eşit olup olmadığı kontrol ediliyor. Eğer 6'ya eşit ise durum saklayıcısı 0'a eşitleniyor. IR'ın 0'dan 5'e olan bitleri MAR'a besleniyor.

Eğer 6'dan 9'a kadar olan bitlerinin değeri 6'ya eşit değil ise; 7'ye eşit olup olmadığı kontrol ediliyor ve eğer ACC(Accumulator) de 0'a eşit ise IR'nin 0'dan 5'e kadar olan bitleri PC'a besleniyor ve durum 0'a eşitleniyor.

Eğer 6'dan 9'a kadar olan bitlerinin değeri 7'ye eşit değil ise; 8'e eşit olup olmadığı kontrol ediliyor. Eğer 8'e eşit ise durum 0'a eşitleniyor.

Eğer 6'dan 9'a kadar olan bitlerinin değeri 8'e eşit değil ise; 9'a eşit olup olmadığı kontrol ediliyor. Eğer 9'a eşit ise durum 4'e eşitleniyor.

Durum saklayıcısı 3’e eşit ise:

Durum saklayıcısı 0’a eşitleniyor, RamWR sinyali 0’a eşitleniyor, MAR değeri 0’a eşitleniyor.

IR’ın 6’dan 9’a kadar olan bitlerinin değerinin 0’a eşit olup olmadığına bakılıyor. Eğer eşit ise MDRout saklayıcısı ACC’ye eşitleniyor.

Eğer 6’dan 9’a kadar olan bitlerinin değeri 0’ya eşit değil ise; 1’ye eşit olup olmadığı kontrol ediliyor.

Eğer 1’e eşit ise IR’ın 0’dan 5’e kadar olan bitleri MAR’a eşitleniyor ve RamWR sinyali 1’e eşitleniyor ve ACC değeri MDRin’e besleniyor.

IR’ın 6’dan 9’a kadar olan bitlerinin değerinin 1’a eşit olup olmadığına bakılıyor.

Eğer 1’e eşit değil ise 2’ye eşit olup olmadığına bakılıyor.Eğer 2’ye eşitse ACC ve MDRout toplanıp ACC’e yazılıyor.

IR’ın 6’dan 9’a kadar olan bitlerinin değerinin 1’a eşit olup olmadığına bakılıyor.

Eğer 1’e eşit değil ise 2’ye eşit olup olmadığına bakılıyor.Eğer 2'ye eşitse ACC ve MDRout toplanıp ACC’e yazılıyor.

IR’ın 6’dan 9’a kadar olan bitlerinin değerinin 2’a eşit olup olmadığına bakılıyor.

Eğer 2’e eşit değil ise 3’ye eşit olup olmadığına bakılıyor.Eğer 3'ye eşitse ACC’den MDRout çıkarılıp ACC’e yazılıyor.

IR’ın 6’dan 9’a kadar olan bitlerinin değerinin 3’a eşit olup olmadığına bakılıyor.

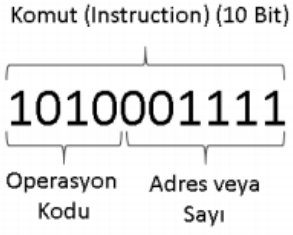
Eğer 3’e eşit değil ise 4’ye eşit olup olmadığına bakılıyor.Eğer 4'ye eşitse ACC ve MDRout çarpılıp ACC’e yazılıyor.

IR’ın 6’dan 9’a kadar olan bitlerinin değerinin 4’a eşit olup olmadığına bakılıyor.

Eğer 4’e eşit değil ise 5’ye eşit olup olmadığına bakılıyor.Eğer 5'ye eşitse ACC

MDRout ‘a bölünüp ACC’e yazılıyor

**Şekilde FB-CPU’nun 10 bitlik komutunun, operasyon ve adres için bitlerinin ayrılması gösterilmiştir.**




Şekil 4. FB-CPU Örnek Komut Binary Gösterimi

SAKLAYICILARIN GÖREVLERİ

• PC (6 Bit): RAM üzerinde hangi satırdaki komutun alınacağını belirler. 6 bit olmasının nedeni RAM'in 2^6 lokasyonu olmasındandır. Dolayısıyla PC değeri RAM'deki her yeri gösterebilmektedir.

• MAR (6 Bit): Memory Address Register isminde bir saklayıcıdır. Bu saklayıcı RAM'in adres girişine bağlanmıştır. RAM'in 2^6 lokasyonu olduğu için MAR 6 bitliktir. Saklayıcı RAM'in içerisindedir.

• MDRIn (10 Bit): Memory Data Register In, RAM'e bir veri yazılacağı zaman kullanılan saklayıcıdır. RAM'in bir lokasyonu 10 bitlik olmasından ötürü, saklayıcı 10 bittir. Saklayıcı RAM'in içerisindedir.

• RAMWr (1 Bit): RAM'e veri yazılacağı durumlarda aktif edilmektedir. 1 olmadığı durumlarda RAM'e veri yazılmaz. Saklayıcı RAM'in içerisindedir.

• MDROut (10 Bit): Memory Data Register, RAM'den veri okunacağı zaman kullanılan saklayıcıdır. RAM'in bir lokasyonu 10 bit olmasından dolayı, saklayıcı 10 bittir. Saklayıcı RAM'in içerisindedir.

• IR (10 Bit): Instruction Register, RAM'den okunan kodun (instruction) saklandığı saklayıcıdır.

• ACC (10 Bit): Accumulator, aritmetik işlem sonuçlarının tutulduğu saklayıcıdır.

SİSTEMİN DONOANIMSAL İFADESİNİ SAĞLAYAN BİLEŞENLER

MUX(Multiplexer)= N adet seçme biti ve 2^N adet data biti bulunur. Seçme bitinden beslenen değere göre seçme bitlerinden bu değere karşılık gelen değer seçilir.

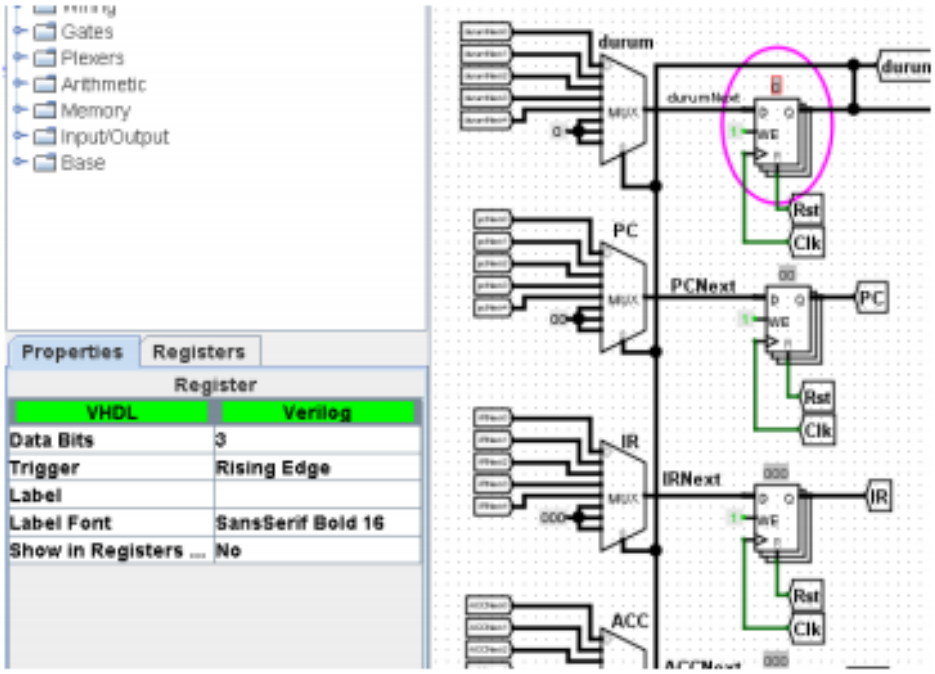
Durum saklayıcıları MUX ların seçme bitlerine bağlanı bu sayade farklı durumlarda diğer saklayıcılara başka değerler beslenebildi.

RAM(Random Acces Memory)= Komutlar ve sayılar ramin çeşitli lokasyonlarında saklandı.

Aritmetik işlem yapan üniteler=Toplama,çarpam,çıkarma ,bölme gibi aritmetik işlemleri yapabilen donanımlar kullanıldı.

Saklayıcılar

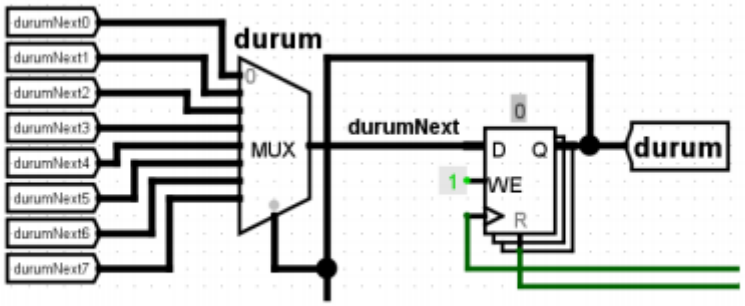
Başlangıç tasarımında 8 adet saklayıcı bulunmaktadır. Şekil 1'de sol üstten başlayıp aşağıya doğru giden 4 tane d tipi saklayıcı ve RAM'in içerisinde 4 saklayıcı bulunmaktadır. Her bir saklayıcı aslında birden çok bir araya gelmiş d tipi saklayıcılardan oluşmuştur. Yani Şekil 6'deki gibi bir saklayıcının üstüne basıldığında, sol alttaki menüde "Data Bits" yazan yer, o saklayıcının kaç bit taşıyabileceğini ifade etmektedir.




Şekil 6. Saklayıcı Bit Genişlikleri

**• Durum (3 Bit):**

FB-CPU durum makinaları yöntemi ile gerçeklenecektir. Yani bu işlemci durum ismindeki saklayıcının değerine göre 2^3 = 8 farklı durumda çalışan bir tasarımı olacaktır (İşlemcinin desteklemesi istenen işlemlerin tamamı 8 farklı durumda yapılabilmektedir). Şekil 7'e bakıldığında durum saklayıcısını ve kendisine bağlı olan MUX yapısı görülmektedir.

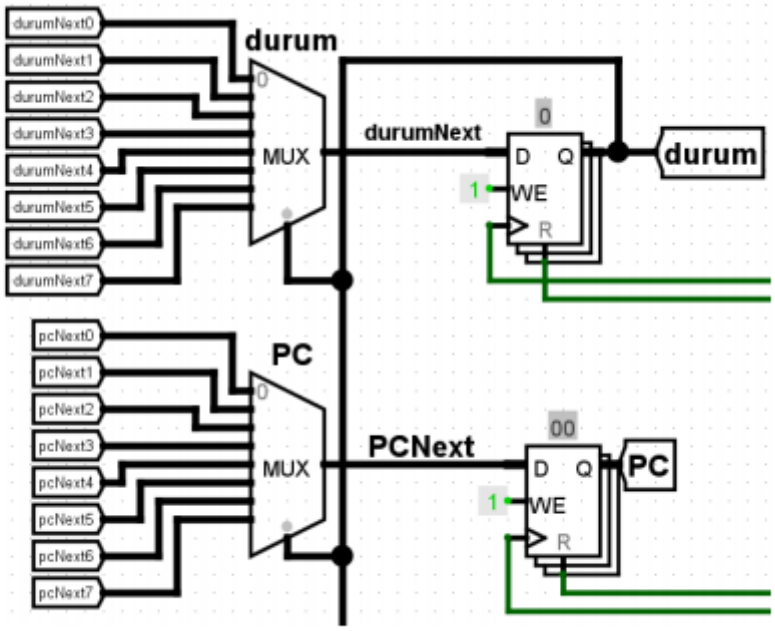



Şekil 7. Durum Saklayıcı ve MUX Yapısı

Durum saklayıcısının bir sonraki değeri, clock'un yükselen kenarında kendisine D girişinden gelen değer olacaktır. D girişinden gelecek olan değer ise MUX'un çıkışı olduğu görülmektedir. MUX yapısının select bitlerine yine durum saklayıcısının çıktısı bağlanmıştır. Bu yapı şunu sağlamaktadır:

Durum == 0 ise, MUX'un girişi 0 olduğu için, durumNext yani saklayıcının D girişi durumNext1 sinyalidir.
Durum == 1 ise, MUX'un girişi 0 olduğu için, durumNext yani saklayıcının D girişi durumNext2 sinyalidir.
Durum == 2 ise, MUX'un girişi 0 olduğu için, durumNext yani saklayıcının D girişi durumNext3 sinyalidir.
… Durum == 7 ise, MUX'un girişi 0 olduğu için, durumNext yani saklayıcının D girişi durumNext8 sinyalidir.

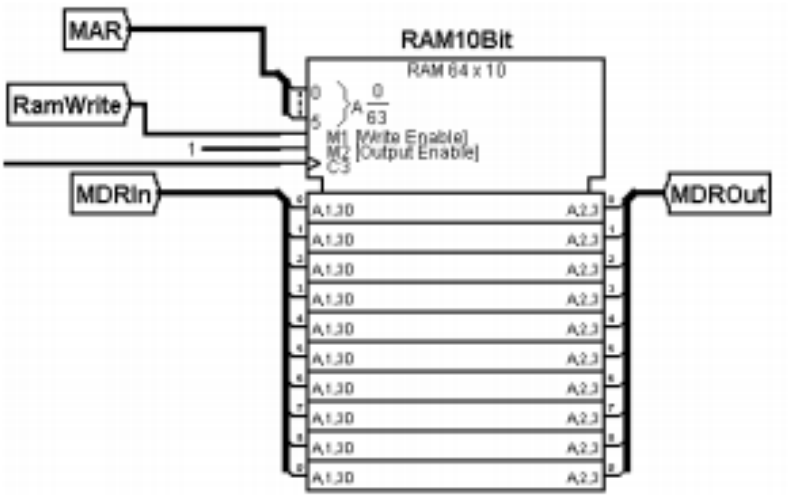
Diğer tüm saklayıcıların da MUX select bitlerine durum saklayıcısının çıktısı bağlanmıştır. Yani durum'un değerine göre tüm saklayıcıların giriş sinyalleri değişmektedir. Diğer bir değiş ile, durum saklayıcısının değerine göre saklayıcıların üzerine başka başka sinyaller atanmakta, sistemin ilerlemesi durum sinyaline bağlıdır. Şekil 8'de durum saklayıcısının PC saklayıcısının önündeki MUX'un select bitlerine bağlı olduğu görülmektedir.



Şekil 8. Durum Saklayıcısı Diğer Tüm MUX'ların Select Girişinde

Bellek (RAM, Random Access Memory):

FB-CPU'nun komutları okuyup, hesaplanan değerleri geri yazacağı bellek Şekil 9'da verilmektedir. RAM'e bağlı 4 saklayıcı ve bir clock sinyali bulunmaktadır. RAM'e bağlı saklayıcıların görevleri saklayıcılar bölümünde açıklanmıştır.




Şekil 9. RAM Yapısı

**İşlem Ünitesi (ALU, Arithmetic Logic Unit):** Aritmetik işlemlerin gerçekleştirildiği bölümdür. FB-CPU'da 4 adet aritmetik işlem vardır. Bunlar toplama, çıkartma, çarpma ve bölmedir, gelen operasyon koduna göre işlemleri gerçekleştirip ACC saklayıcısına yazmaktadır.

**Kontrol Ünitesi:** Saklayıcılar, Aritmetik İşlem Ünitesi ve RAM'e verilerin birbirleri arasında transferinden sorumludurlar. İşlemci içi veri akışını yönetir.

# 3 Geliştirilen Yazılım

FB-CPU için bellekte 50 ve 51 adresteki iki sayının çarpımını 52 no'lu adrese kaydeden uygulama:

```
0: 0000_110011 // LOD 51, ACC = *51, Hex = 33
1: 0011_110001 // SUB 49, ACC = ACC - *49, Hex = F1
2: 0111_001010 // JMZ 10, döngü bittiyse, döngüden çıkartacaktır (ACC-49 == 0), 10. Satır, Hex = 1CA
3: 0000_110000 // LOD 48, temp değerini yükle, başlangıçta 0, Hex = 30
4: 0010_110010 // ADD 50, ikinci sayıyı ACC'nin üstüne ekle, Hex = B2
5: 0001_110000 // STO 48, ACC'nin değerini temp'e ata, Hex = 70
6: 0000_110001 // LOD 49, ACC = i, Hex = 31
7: 0010_101110 // ADD 46, ACC = i + 1, Hex = AE
8: 0001_110001 // STO 49, i = i + 1, Hex = 71
9: 0110_000000 // JMP 0, döngünün başına dön 0. satır, Hex = 180
10: 0000_110000 // LOD 48, ACC = temp, Hex = 30
11: 0001_110100 // STO 52, *52 = ACC, Hex = 74
10: 1001_000000// HLT, bitirme, Hex = 240

46: 1 // 1 sayısı
48: 0 // Hex = 0, temp
49: 0 // Hex = 0, i index'i için
50: 0000000101 // Hex = 5
51: 0000001010 // Hex = A
```

CPU Çarpma işlemi için 50'deki sayıyı 51'deki sayı defa toplayıp 52 no'lu adrese yazılıyor.

Yazılım çalıştırılırken durum makinesindeki yöntem izleniyor.

Ram den alınan her bir komut okunuyor operasyon kodu ve sayı veya adres belirleniyor. Operasyon kodlarına göre aritmetik ve mantıksal işlemler yapılıyor bu şekilde tüm komutlar uygulandığında program görevini tamamlamış oluyor.

# 4 Sonuçlar

**FBU CPU ‘nun desteklediği komutlar:**

| Komut Adı | Görevi | Operasyon Kodu |
|---|---|---|
| LOD ADDR | Yükleme (Load), Bellekteki verilen adresin içerisinden değeri alıp, ACC saklayıcısına yerleştirir.<br>ACC = *(ADDR) | 0000 |
| STO ADDR | Kaydetme (Store), ACC’nin içerisindeki değeri alıp, bellekte verilen adrese yazar.<br>*(ADDR) = ACC | 0001 |
| ADD ADDR | Bellekteki verilen adresteki değeri alır, ACC ile toplayıp, ACC’nin üzerine yazar.<br>ACC = ACC +*(ADDR) | 0010 |
| SUB ADDR | Bellekteki verilen adresteki değeri alır, ACC ile çıkartıp, ACC’nin üzerine yazar.<br>ACC = ACC - *(ADDR) | 0011 |
| MUL ADDR | Bellekteki verilen adresteki değeri alır, ACC ile çarpıp, ACC’nin üzerine yazar.<br>ACC = ACC * (*(ADDR)) | 0100 |
| DIV ADDR | Bellekteki verilen adresteki değeri alır, ACC ile bölüp, ACC’nin üzerine yazar.<br>ACC = ACC / (*(ADDR)) | 0101 |
| JMP SAYI | PC = Sayı olur. | 0110 |
| JMZ SAYI | ACC’ın değeri 0 ise, verilen sayı değerini PC’e atar, değilse işlem yapmaz. | 0111 |
| NOP | No Operation, hiçbir işlem yapılmaz. | 1000 |
| HLT | Uygulama durur | 1001 |

İşlemci 10 adet komutu desteklemektedir.

,
Bu proje vasıtasıyla bir işlemcinin temel bileşenleri , çalışma prensipleri ve çalışma mantığı deneyerek ve araştırılarak öğrenmiş olduk .

Hazırlanan sunum video’su adresi:

https://www.youtube.com/watch?v=d-rsz4AfrNw

Dosyaların github adresi:

https://github.com/ilhan-ersoy/FBU-CPU-TASARIM-