BLM 101- Bilgisayar Mühendisliğine Giriş
2019-2020 Güz Dönemi

FB-CPU V1

Proje Teslim Raporu
13 Ocak 2020

İrem KALKANLI ,Deniz UZUN, Özlem ÇALI, Aysen İpek ÇAKIR

Öğretim Elemanı: Dr. Vecdi Emre Levent, emre.levent@fbu.edu.tr, İzinsiz Kopyalanamaz

1 Giriş

1.1 Projenin Amacı

Amaç makine dilinde yazılan 10 farklı operasyon kodu çalıştırabilen bir işlemci tasarımı geliştirmektir.

1.2 Proje Ekibi

İrem KALKANLI(Proje Ekip Sorumlusu):
Okul numarası:190301007
Doğum Tarihi:15.01.2000
Doğum Yeri: İstanbul
Mezun Olduğu Lise: Ataşehir 3 Doğa Koleji

Deniz UZUN:
Okul numarası:190301015
Doğum Tarihi:08.04.2001
Doğum Yeri: İstanbul
Mezun Olduğu Lise: Kavacık Uğur Anadolu Lisesi

Özlem ÇALI:
Okul numarası:190301002
Doğum Tarihi:19.05.2000
Doğum Yeri: Hatay
Mezun Olduğu Lise: Necmi Asfuroğlu Anadolu Lisesi

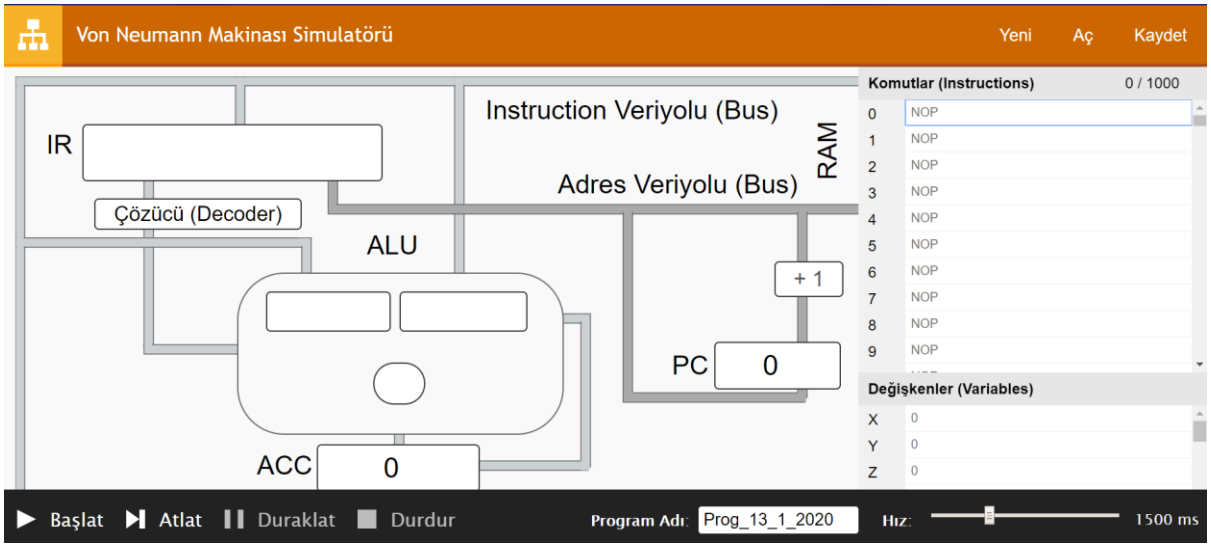
Aysen İpek ÇAKIR:
Okul numarası:190301001
Doğum Tarihi:20.03.2001
Doğum Yeri: Malatya
Mezun Olduğu Lise: Fethi Gemuhluoğlu Fen Lisesi

2 Sistem Mimarisi

2.1 Kullanılan Araçlar

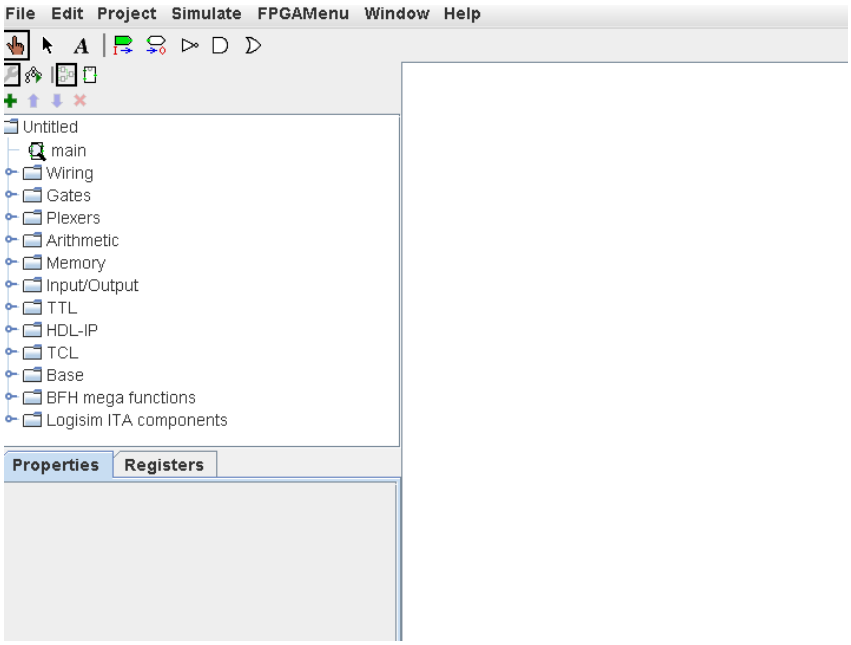
Proje kapsamında 2 araç kullanıldı.

1)Von Neumann Simulatörü: Von Neumann mimarisi veri ve komutları tek bir birimde bulunduran bilgisayar tasarı örneğidir. FB-CPU'nun mimarisini görselleştiren ve veri akışının gözlemlenebildiği bir simulatördür.



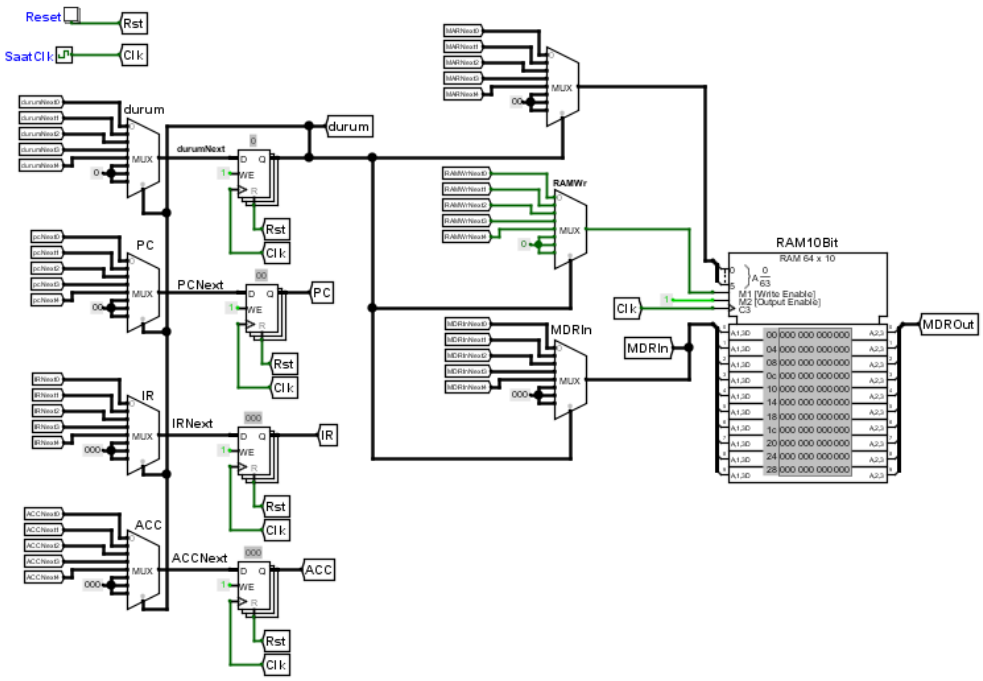
2) Logisim-Evolution: Dijital mantık devrelerini tasarlamak ve simüle etmek için kullanılan bir eğitim aracıdır. Mantık devreleriyle ilgili temel kavramları öğrenmeyi kolaylaştırmaktadır.İşlemcinin tasarımı bu simulator içerisinde yapılmıştır.

Gerekli mantık devreleri sol taraftaki yapılar kullanılarak elde edilebilmektedir.



2.2 Tasarım

Tasarlanması istenen işlemci Von Neumann mimarisinden tasarlanmış, birkaç farklılık içerir.



Şekilde verilen mimaride sol üstte clock sinyali belirli periyotlar üretir.Her bir periyoda da ‘ clock cycle’ denir.Kontrol ünitesi kendine gelen clock sinyali oldukça işlem yapmaya devam eder ve böylece gerekli atamalar yapılır.

Durum makinası, durum saklayıcısının değerine göre diğer saklayıcılara gerekli atamaları yapar ve istenilen operasyonlar gerçekleştirilir.Bu operasyonlar MUX yardımıyla yapılır.

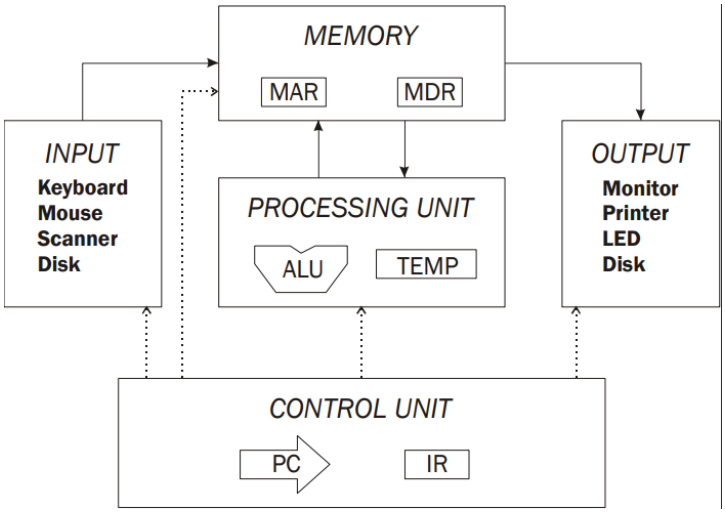
MUX yardımıyla seçme bitine gelen değere göre durum saklayıcısındaki değere göre bir sonraki durumun değeri PC,ACC,IR gibi sistemlere beslenmektedir.

PC:Şu anki komutu tutar.

IR:Bir sonraki komutu tutar.

ACC:Yapılan işlem sonucunu tutar.

Durumdan gelen değerlere göre gerekli yapı seçilerek işlemler gerçekleştirilmiş olur.



Temel olarak 4 elemanı vardır.

Saklayıcılar (Şekil 2'de Processing Unit'in altındaki Temp değişkeni)

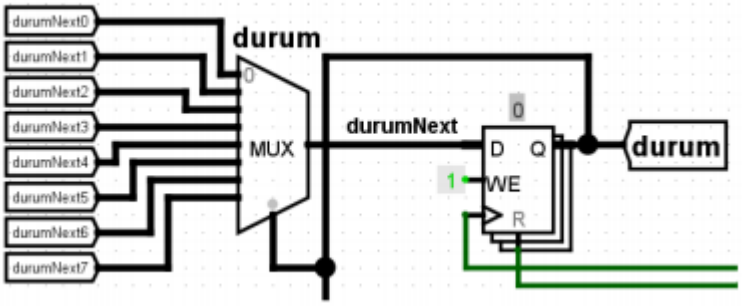
• Bellek (RAM)

• İşlem Ünitesi (ALU)

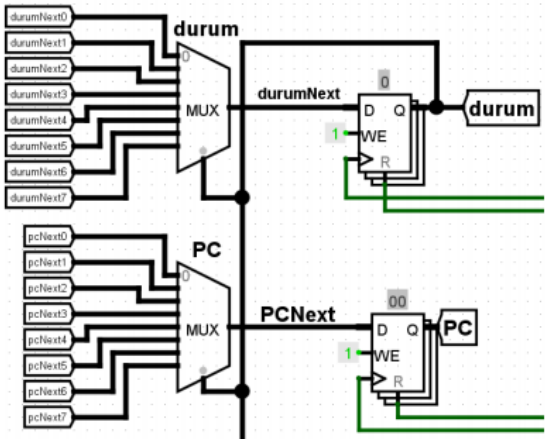
• Kontrol Ünitesi

SAKLAYICILAR

Başlangıç tasarımında 8 adet saklayıcı bulunmaktadır. 4 tane d tipi saklayıcı ve RAM'in içerisinde 4 saklayıcı bulunmaktadır.



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. 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.



• 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.

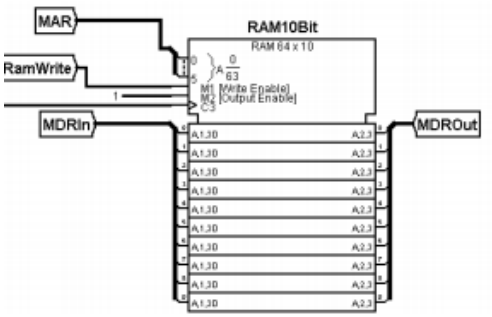
• IR (10 Bit): Instruction Register, RAM'den okunan kodun (instruction) saklandığı saklayıcıdır.

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

. Bellek (RAM, Random Access Memory):

FB-CPU'nun komutları okuyup, hesaplanan değerleri geri yazacağı bellek şekilde verilmektedir. RAM'e bağlı 4 saklayıcı ve bir clock sinyali bulunmaktadır

Bu yapı komut ve adreslerin tutulduğu bellektir.



İşlemci ünitesinin belleğe erişimi iki saklayıcı ile olmaktadır:

• MAR: Memory Address Register

• MDR: Memory Data Register

• **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.

: • Bellekten program counter'ın göstediği komutu okur.

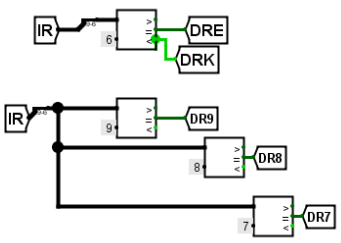
• Sistemin geri kalanına, yapılması gereken işlemleri yaptırır.

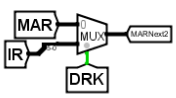
• Bir komut birden çok cycle sürebilir.

Instruction Register (IR) şu anki koşturulan komutun adresini tutmaktadır.

• Program Counter (PC) bir sonraki koşturulacak olan komutun adresini tutar.

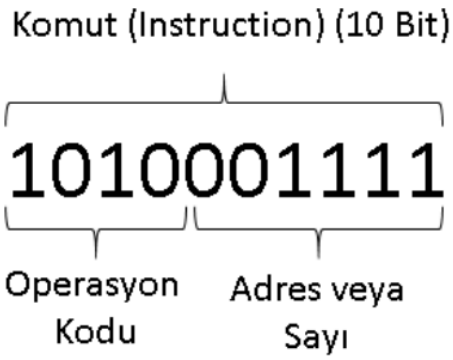
Bunu tasarlarken oldukça zor ve karmaşık yapılar ortaya çıktığı için 'TÜNEL'denilen küçük bağlantılar kullandık. Örnek olarak aşağıdaki şekilde tüneller verilmiştir.Bu tüneller tek bir yapıda kullanıldığında karmaşıklığa ve anlaşılmazlığa yol açabileceği için bu tür kullanımlar çok kullanışlı ve uygun olmuştur.





Tablo 1. FB-CPU ISA (Instruction Set Architecture)

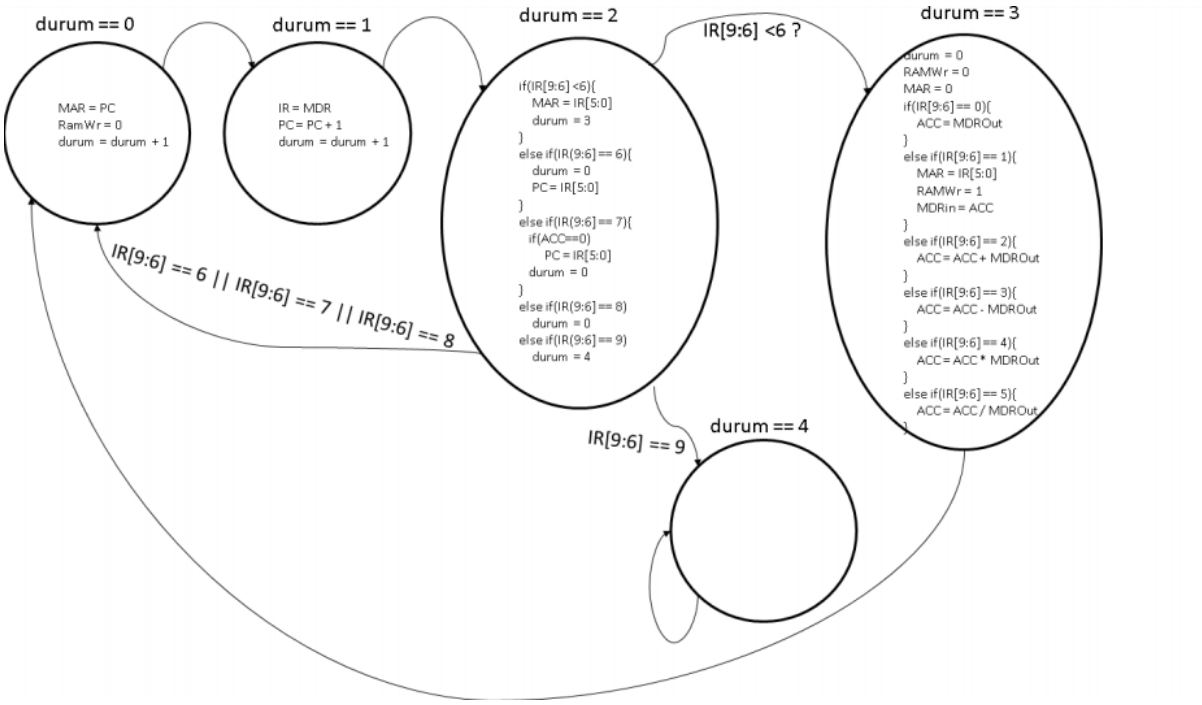
Komut Adı	Görevi	Operasyon Kodu
LOD ADDR	Yükleme (Load), Bellekteki verilen adresin içerisinden değeri alıp, ACC saklayıcısına yerleştirir. ACC = *(ADDR)	0000
STO ADDR	Kaydetme (Store), ACC'nin içerisindeki değeri alıp, bellekte verilen adrese yazar. *(ADDR) = ACC	0001
ADD ADDR	Bellekteki verilen adresteki değeri alır, ACC ile toplayıp, ACC'nin üzerine yazar. ACC = ACC +*(ADDR)	0010
SUB ADDR	Bellekteki verilen adresteki değeri alır, ACC ile çıkartıp, ACC'nin üzerine yazar. ACC = ACC - *(ADDR)	0011
MUL ADDR	Bellekteki verilen adresteki değeri alır, ACC ile çarpıp, ACC'nin üzerine yazar. ACC = ACC * (*(ADDR))	0100
DIV ADDR	Bellekteki verilen adresteki değeri alır, ACC ile bölüp, ACC'nin üzerine yazar. 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



Komutların bellekte okunması ve çözülmesi için durum makinesine 10 bitlik komutunda ilk 4 biti yani [9:6] operasyon kodunu belirtmekte,son 6 biti[5:0] adresi veya sayıyı temsil etmektedir.İşleme sokulacak olan sayı bellekteki bir adresten,komutun içerisinden veya bir saklayıcıdan alınabilir.

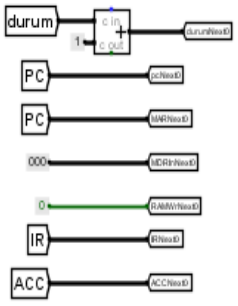
FBU-CPU tasarımı 10 adet komudu yapabilecek şekilde tasarlanmıştır.Yukardaki tabloda işlemcinin desteklediği operasyonlar ve operasyon kodları verilmiştir.

İşlemci belirtilen komutları yerine getirmek için gerekli durum değerlerini sağlanmalıdır.

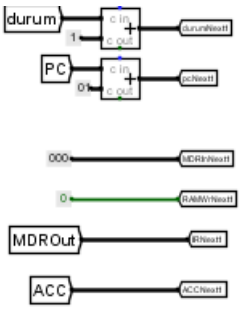


durum==0 durum==1 ve durum==4 verilerek bizden durum==2 ve durum==3 yapılması istenmiştir.

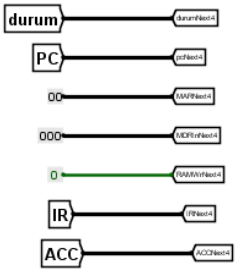
durum0



durum1



durum4



DURUM=2

Durum==2 ‘ye baktığımız zaman operasyon koduna göre olaylar gerçekleşir.

Eğer operasyon kodu 6’dan küçükse direkt durum==3 ‘e geçilerek işlemler yapılır.

Ama 6’ya eşit yada 6’dan büyük bir değer almışsa ona göre farklı atamalar olur.

Örneğin operasyon kodu 6’ya eşitse operasyon kodu PC’ye aktarılarak yani PC=sayı olarak JMP işlemi yapılmış olur.

Operasyon kodu 7’ye eşitse JMZ yapılır. ACC =0 ise verilen sayı değerini PC’e atar,değilse işlem yapmaz.Aynı zamanda durum=0 olur.

Operayon kodu 8’e eşitse NOP işlemi gerçekleşir.Yani hiçbir işlem yapılmaz.

Durum=0 olur.

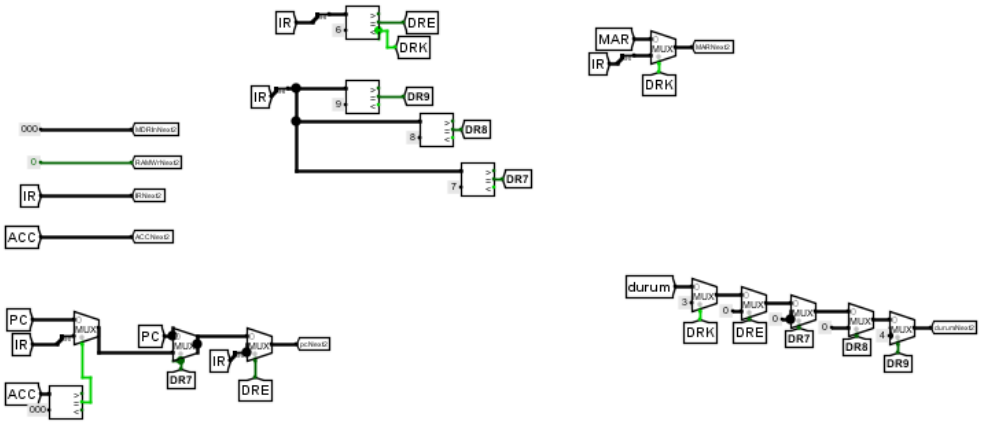
Operasyon kodu 9’a eşitse HLT olur.Uygulama durur.

Durum=4 olarak döngüye girer.

Şimdi tasarladığımız işlemcide durum=2 şemasını gösterelim.

---

durum2



DURUM=3

Durum=3 ‘e eşit olduğu zaman girilen operasyon koduna göre gerekli işlemler yapılır.

İlk başta durum=0 , RAMWr=0,MAR=0 verilmiştir.

Operasyon kodu 0’a eşitse MDROut’un değeri ACC’ye eşit olur.

Operasyon kodu 1’e eşitse IR[5:0] kodu MAR ‘a yazılır.RAMWr=1 olur.ACC içerisindeki değer MDRin ‘e yazılır.

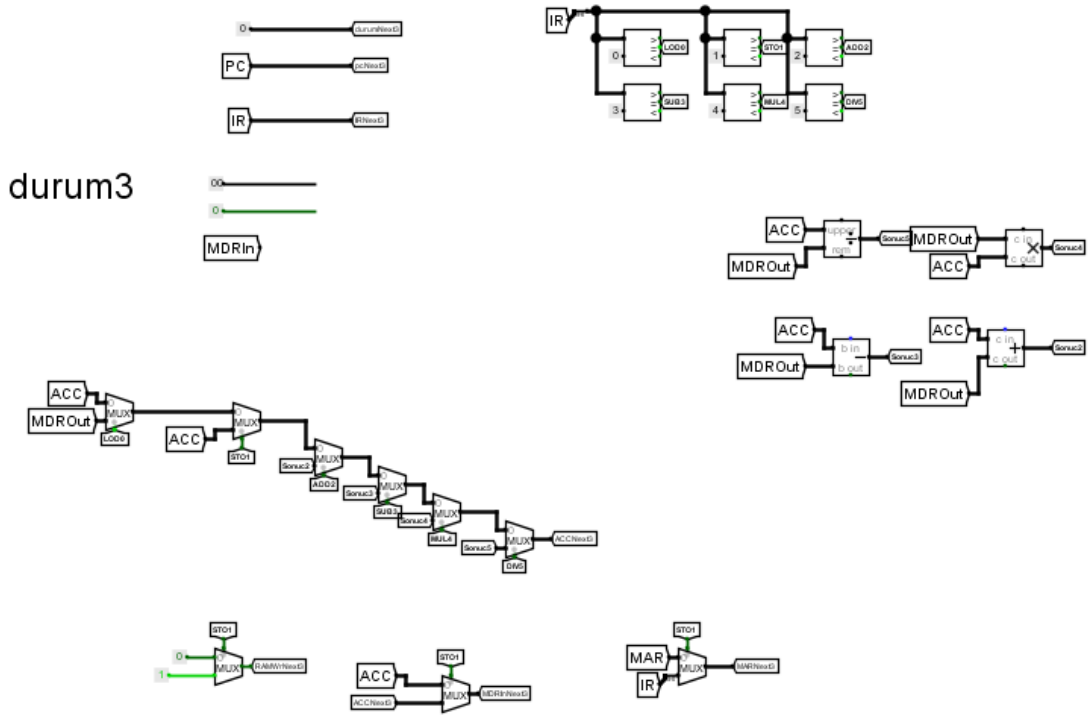
Operasyon kodu 2’ye eşitse bu toplama işlemi yapılacağı anlamına gelir.Bellekteki verilen adresteki değeri alır,ACC ile toplayarak ACC’nin üstüne yazar.

Operasyon kodu 3’e eşitse bu çıkarma işlemi yapılacağı demektir. Bellekteki girilen adresteki değeri alıp ACC ‘den çıkartarak tekrar ACC’nin üstüne yazar.

Operasyon kodu 4’e eşitse çarpma işlemi yapılacağı anlamına gelir. Bellekteki verilen adresteki değeri alıp ACC ile çarparak tekrar ACC’nin üstüne yazar.

Operasyon kodu 5’e eşitse bölme işlemi yapılacağı anlamına gelir. Bellekteki verilen adresteki değeri alıp ACC ‘ye bölerek tekrar ACC’nin üstüne yazar.

Verdiğimiz kodun işlevlerini açıkladık. Kodun çalışması için tasarladığımız işlemcinin durum 3 kısmı şekilde verilmiştir.



3 Geliştirilen Yazılım

Yaptığımız FBU-CPU için yapılan test yazılımlarının işlemcide nasıl çalışacağına bakalım.

Öncelikle tasarımdaki RAM'e sağ tıklayıp edit content'e tıklandığında,open tuşu ile verilen yazılım yüklenir.

Test Yazılımı 1

FB-CPU için bellekte 50 ve 51 adresteki iki sayının toplamını 52 no'lu adrese kaydeden uygulamayı inceleyelim.

0: 0000_110010 // LOD 50, (ACC = *50), Hex = 32

1: 0010_110011 // ADD 51, ACC = ACC + (*51), Hex = B3

2: 0001_110100 // STO 52, (*52) = ACC, Hex = 74

3: 1001_000000 // Halt, Hex = 240

50: 0000000101 // Hex = 5

51: 0000001010 // Hex = A

0.adreste ilk 4 bit operasyon kodunu belirtmekte olup LOD olduğu anlaşılıyor.Son 6 bit ise sayıyı ifade etmektedir. Yani bu aşamada 50 sayısını bellekteki verilen adresten alıp ACC saklayıcısına yerleştirir.

1.adreste ilk 4 bitteki operasyon kodu toplama işlemini ifade etmekte olup sayıyı alıp ACC ile toplayıp ACC 'ye kaydediyor.

2.adreste ilk 4 bit STO'yu belirtip,son 6 bitteki sayıyı yani ACC'deki değeri alıp bellekte verilen adrese yazar.

3.adreste ilk 4 bitteki operasyon kodu HALT işlemini belirtip uygulamayı durdurur.

50.adreste hexadecimal olarak 5 sayısını belirtir.

51.adreste hexadecimal olarak A yani ondalık sayı olarak 10 'yı belirtmektedir.

52.adreste de sonuç hexadecimal olarak F yazılmalıdır.

Test Yazılımı 2

FB-CPU için bellekte 50 ve 51 adresteki iki sayının çarpımını 52 no'lu adrese kaydeden uygulamayı inceleyelim.

0: 0000_110010 // LOD 50, (ACC = *50), Hex = 32

1: 0100_110011 // ADD 51, ACC = ACC * (*51), Hex = 133

2: 0001_110100 // STO 52, (*52) = ACC, Hex = 74

3: 1001_000000 // Halt, Hex = 240

50: 0000000101 // Hex = 5

51: 0000001010 // Hex = A

0.adreste ilk 4 bit operasyon kodunu belirtmekte olup LOD olduğu anlaşılıyor.Son 6 bit ise sayıyı ifade etmektedir. Yani bu aşamada 50 sayısını bellekteki verilen adresten alıp ACC saklayıcısına yerleştirir.

1.adreste ilk 4 bitteki operasyon kodu çarpma işlemini ifade etmekte olup sayıyı alıp ACC ile çarpıp ACC 'ye kaydediyor.

2.adreste ilk 4 bit STO'yu belirtip,son 6 bitteki sayıyı yani ACC'deki değeri alıp bellekte verilen adrese yazar.

3.adreste ilk 4 bitteki operasyon kodu HALT işlemini belirtip uygulamayı durdurur.

50.adreste hexadecimal olarak 5 sayısını belirtir.

51.adreste hexadecimal olarak A yani ondalık sayı olarak 10 'yı belirtmektedir.

52.adreste de sonuç olarak 10*5=50 yazılmalıdır.

Test Yazılımı 3

FB-CPU için bellekte 50 ve 51 adresteki iki sayının çarpımını 52 no'lu adrese kaydeden uygulamayı inceleyelim.Gerekli değişkenler için boş adresler kullandık.

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

12: 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

0. adreste 51.adresteki değer yüklenir.

1. adreste 49.adresteki değer ACC'dan çıkarılır.

2.adreste ACC değeri sıfır olana kadar döngüde kalır.Sıfır olunca 10.adrese atlar.(döngünün kırılmasını bu komut sağlar.)

3. adreste geçici değişkeni saklamak için kullanılan ilk değeri 0 olan 48.adres yüklenir.

4. adreste 50. Adresteki değer ACC'ın üstüne eklenir.

5.adreste ACC değeri geçici değişkeni saklamak amaçlı kullanılan 48. Adrese kaydedilir.

6.adreste, içinde 0 bulunan ve indeks olarak tanımladığımız 49.adresi yükleriz.

7. adreste ACC'a 46.adreste bulunan 1 değeri eklenir.

8.adreste ACC indeks olarak belirlediğimiz 49.adrese kaydedilir.

9. adreste JMP 0 komutuyla 0.adrese geri dönülür (döngü oluşmasını bu komut sağlar.)

10.adreste 48.adresteki değer yüklenir.

11.adreste ACC'daki değer 52.adrese kaydedilir.

12.adreste HLT komutu olan bitirme işlemi gerçekleşir.

46.adreste 1 değeri bulunur.

48.adreste ilk 0 değeri bulunur.(Geçici değişken)

49.adreste ilk 0 değeri bulunur.(İndeks)

50.adreste hexadecimal olarak 5 sayısını belirtir.

51.adreste hexadecimal olarak A yani ondalık sayı olarak 10 ‘yı belirtmektedir.

52.adreste de sonuç olarak 10*5=50 yazılmalıdır.

4 Sonuçlar

Geliştirilen FBU-CPU işlemcisi gerekli durum koşullarını sağladığında 10 adet komutu yerine getirip,4 adet işlem yapabilmektedir.

Elde ettiğimiz kazanımlara gelirsek, bu işlemciyi yapabilmek için saklayıcılar,bellek,işlem ünitesi ve kontrol ünitesi hakkında gerekli bilgileri edindik.

Bu bilgileri verilen Von Neumann mimarisinde inceleyerek yapı hakkında bilgi sahibi olduk.

Sonuç olarak FBU-CPU işlemcisi makine dilindeki kodlarını istenen operasyonları düzgün bir şeklide gerçekleştirebilmektedir.

Hazırlanan sunum video'su adresi:
https://www.youtube.com/watch?v=V003GQMbjgk&feature=youtu.be

Dosyaların github adresi: https://github.com/iremkalkanli/BLM-101-Projesi-FBU-CPU