Fenerbahçe Üniversitesi

BLM 101 – Bilgisayar Mühendisliğine Giriş

FB-CPU Tasarımı

Proje İçeriği

Veriliş Tarihi: 2.12.2019

Teslim Tarihi ve Yeri: 6.1.2020, Ders Saatlerinde, Ders Sınıfında ve Elektronik olarak

1. Tanım:

Bu proje kapsamında FB-CPU isminde bir işlemcinin tasarımı ve tasarlanan işlemci üzerinde makine dili ile yazılan çeşitli kod parçacıkları yazılacaktır. Proje sonunda basit bir işlemcideki RAM, Kontrol Ünitesi ve Saklayıcıların bir arada çalışıp, makine dilindeki kod parçacıklarını nasıl yürütebildiği gözlemlenecektir.

2. Proje Ekibi:

Proje 4 kişilik ekiplerden oluşacaktır. Her bir proje ekibinin bir sorumlusu olacaktır. Öğrenciler 4 kişilik kendi proje ekiplerini ve proje sorumlusunu belirlemelidirler.

Ekiplerin kurulması ve proje sorumlusunun belirlenmesi en geç **9.12.2019** tarihine kadar tamamlanmalıdır. Ekip sorumluları, LMS'te açılmış olan "Proje Ekip Sorumluların Takımlarını Bildirmesi" başlığının altına, ekip üyelerinin isimlerini göndermelidirler.

LMS adresi: http://levent.tc/lms/

3. Proje LAB'ı:

Proje'nin bir kısmının gerçeklemesinin nasıl olabileceği LAB esnasında yapılacaktır.

FB-CPU Donanım ve Makine Dili Tasarımı LAB'ı **16.12.2019** tarihinde yapılacaktır. Bu tarihe kadar FB-CPU'u gerçeklemeye çalışarak sorular biriktirilmelidir.

4. Kullanılacak Araçlar:

Proje kapsamında 2 araç kullanılacaktır.

4.1. Von Neumann Simulatörü:

FB-CPU'nun mimarisini görselleştiren, veri akışının gözlemlenebildiği "Von Neumann Simulatörü" kullanılacaktır.

Erişim adresi: http://levent.tc/araclar/vonneumann/

Ekranın sağ tarafında bulunan Komutlar ve Değişkenler belleğin (RAM) içerisinde bulunan sayılardır.

Komutları 0-1'lar halinde yazmak yerine, assembly denen bir dil ile ifade ediyoruz.

Komutlar bölümüne, bu işlemci için geçerli komut seti ile komutlar yazabilirsiniz.

4.2. Logisim-Evolution:

Logisim-Evolution, dijital mantık devrelerini tasarlamak ve simüle etmek için kullanılan bir eğitim aracıdır. Mantık devreleriyle ilgili temel kavramları öğrenmeyi kolaylaştırmaktadır. İşlemcinin tasarımı bu simülatör aracının içerisinde yapılacaktır.

Erişim Adresi: http://levent.tc/courses/blm101/proje/logisim-evolution.jar

Kullanım Dokümanı: http://levent.tc/courses/blm101/proje/logisimKullanimKilavuzu.pdf

5. Tasarım Gereksinimleri

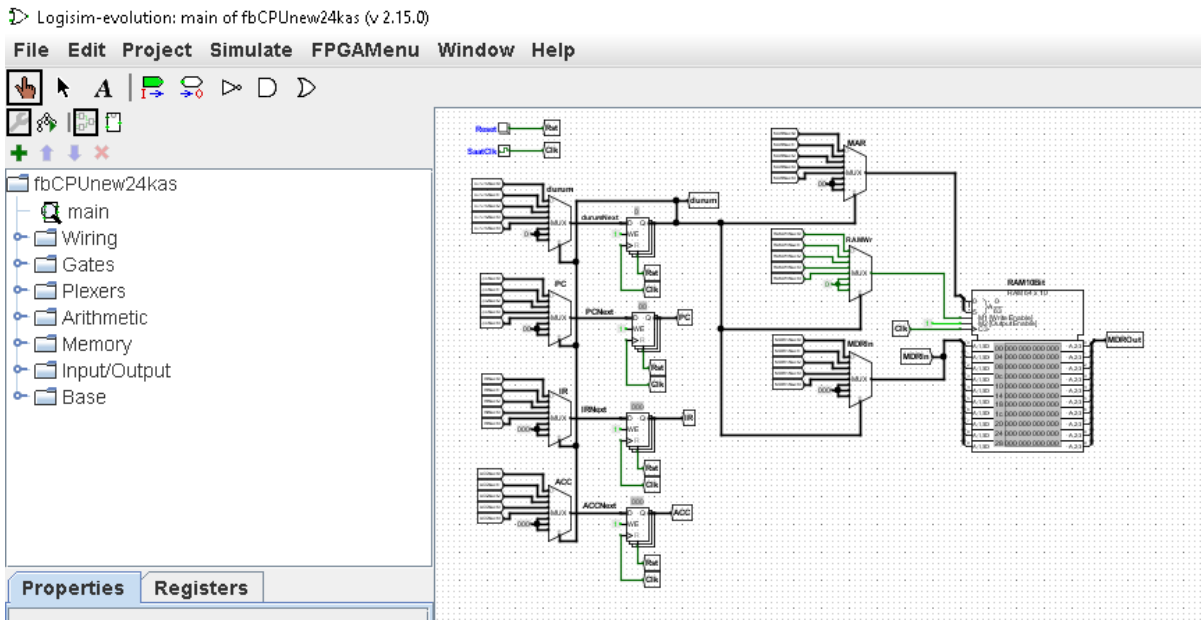
Bu başlık altında tasarlanması istenen FB-CPU'nun gereksinimleri verilmektedir.

İşlemci tasarımı için başlangıç tasarımı verilmiştir.

Başlangıç tasarımı: http://levent.tc/courses/blm101/proje/BLM101_proje_fbCPUBaslangic.circ (Açılan ekranda, sağ tıklayıp farklı kaydet'e tıklayınız)

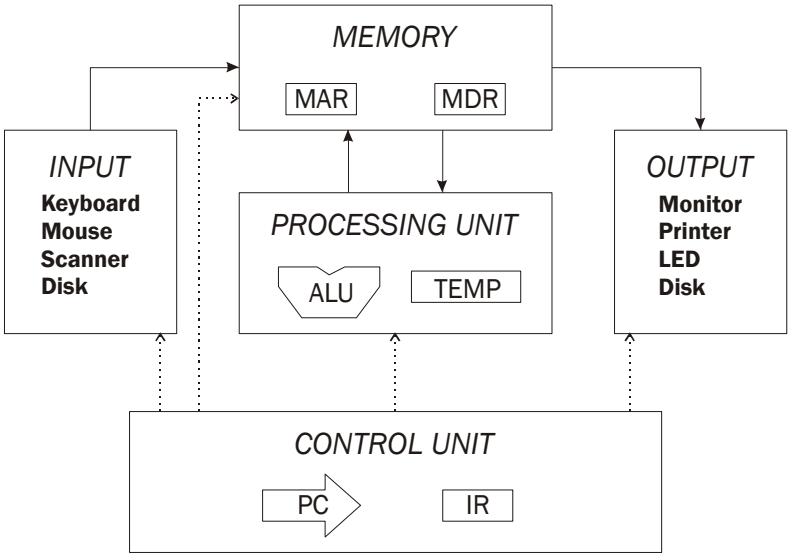
Bu dosyayı Logisim uygulamasından açarak (Uygulamada File->Open) başlatabilirsiniz.

Tasarımı açtığınızda Şekil 1'deki ekran gelmelidir.



Şekil 1. FB-CPU Başlangıç Tasarım Ekranı

Tasarlanması istenen işlemci Von Neumann mimarisindedir. Şekil 2'de Von Neumann Mimarisi verilmektedir.

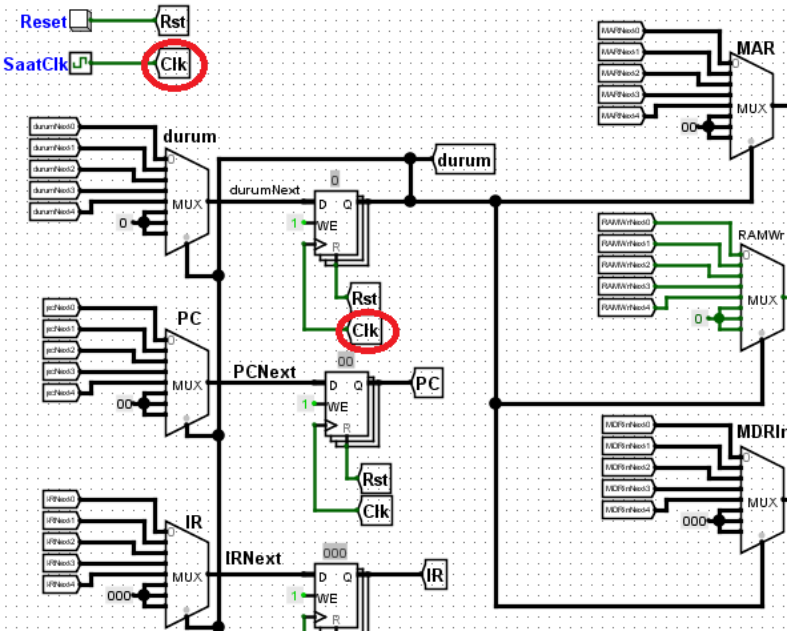


Şekil 2. Von Neumann Mimarisi

Temel olarak 4 elemanı vardır.

- Saklayıcılar (Şekil 2'de Processing Unit'in altındaki Temp değişkeni)
- Bellek (RAM)
- İşlem Ünitesi (ALU)
- Kontrol Ünitesi

Tasarım küçük olmaması ve birçok elemanı bir arada bulundurması nedeniyle, bunların birbirlerine arasında gerekli kablo bağlantılarını kurmak oldukça zordur. Bu bağlantılar kablolar ile kurulduğunda ortaya kabloları takip etmesi oldukça güç, çok karmaşık bir yapı ortaya çıkmaktadır. Bunun için Logisim aracının içerisindeki "Tunnel" denen bir mekanizma kullanılmıştır. Tunnel aracı aslında arka planda kablolama işi ile aynı görevi görmektedir. Yani Şekil 3'te kırmızı yuvarlak ile işaretlenmiş olan MAR yazısı aslında, bu iki kablonun birbirine bağlı olduğunu ifade etmektedir. Dolayısıyla ortada kabloların birbirleri üzerinden geçen bir kaos yapısı oluşmamaktadır.



Şekil 3. Tunnel Mekanizması

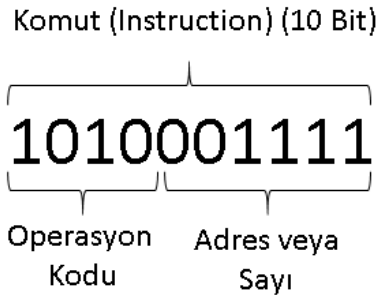
FB-CPU Von Neumann Mimarisinde tasarlanmış, bazı değişiklikler içermektedir. FB-CPU'nun desteklediği işlemler Tablo 1'de verilmektedir.

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

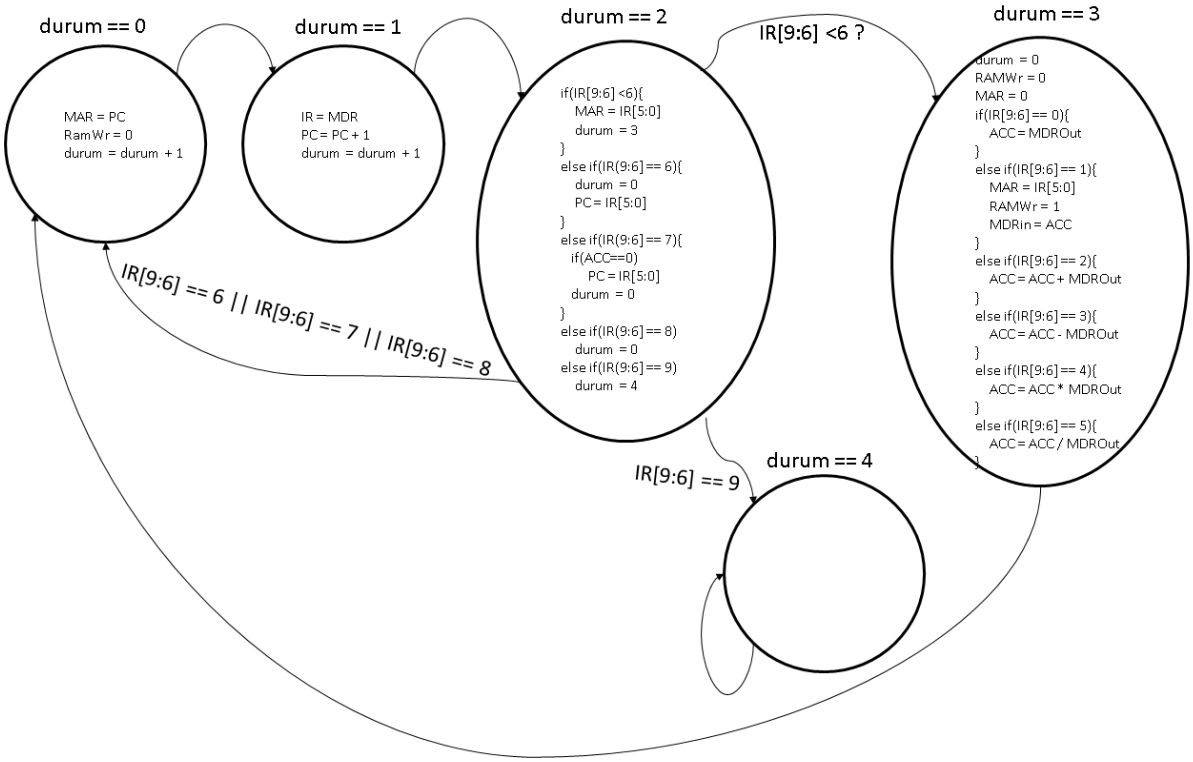
İşlemci 10 adet komutu desteklemektedir.

Şekil 4'te FB-CPU'nun 10 bitlik komutunun, operasyon ve adres için bitlerinin ayrılması gösterilmiştir.



Şekil 4. FB-CPU Örnek Komut Binary Gösterimi

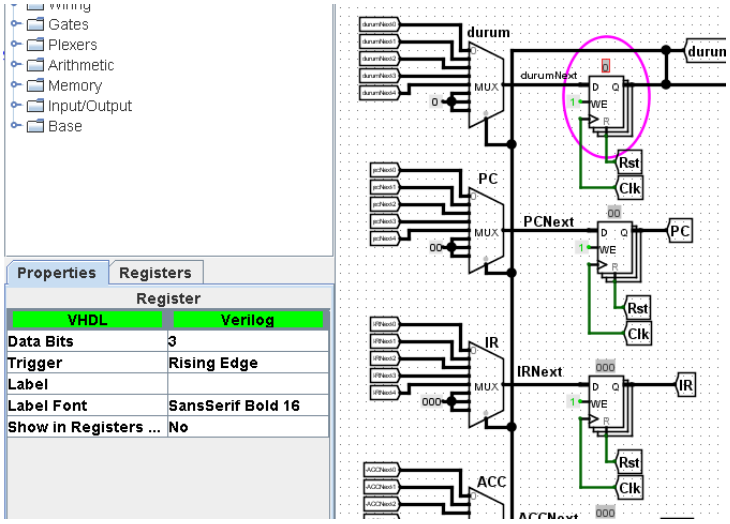
FB-CPU'nun durum diyagramı olarak ifade edilmiş hali Şekil 5'te verilmektedir. İşlemcinin adım adım yapması gereken işler bir arada göstermektedir.



Şekil 5. FB-CPU Durum Makinası Gösterimi

5.1. 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.

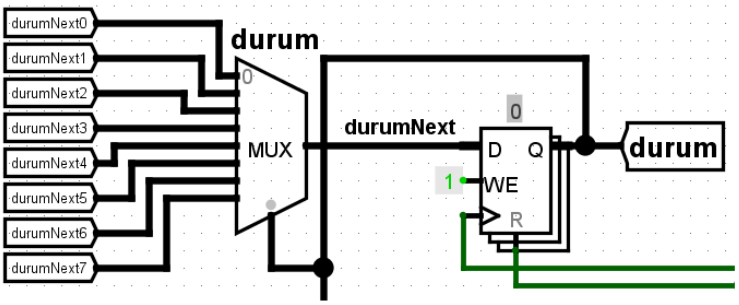


Şekil 6. Saklayıcı Bit Genişlikleri

Yeni saklayıcı eklenmeyecektir. Tüm gerekli saklayıcılar tasarımda bulunmaktadır.

Bu saklayıcılar aşağıda görevleri ile birlikte verilmektedir.

- **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.

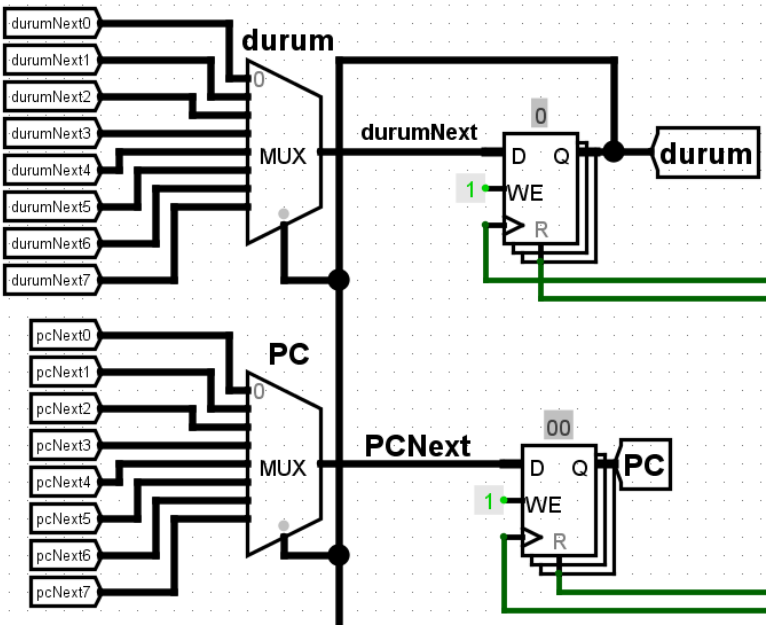
Durum == 1 ise, MUX'un girişi 0 olduğu için, durumNext yani saklayıcının D girişi durumNext2 sinyalidir.

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.

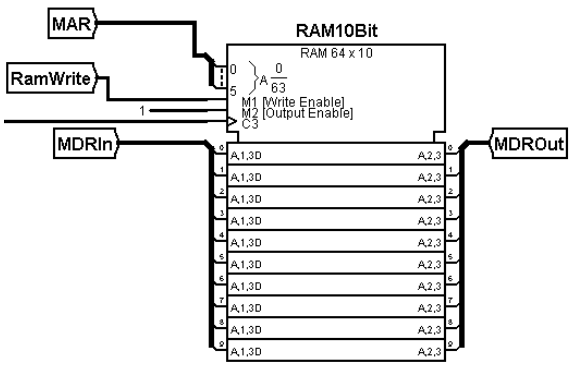


Şekil 8. Durum Saklayıcısı Diğer Tüm MUX'ların Select Girişinde

- **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.

5.2. 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

5.3. **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.

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

İşlemci tasarımı için verilen başlangıç dosyasında, İşlem ve Kontrol Ünitesi eksiktir. Bu eksik yerler, tasarımın sağında bulunan durum 0, durum 1, durum 2 … boşluklarında oluşturulacaktır.

6. Eksik Ünitelerin Tasarımı

Verilen başlangıç tasarımında durum 0 ve durum 1 için tasarım verilmiştir.

LAB'da durum 2'nin tasarımı yapılacaktır.

İşlemcinin çalışır olması için durum 3'ün tamamlanması gerekmektedir.

7. Örnek Yazılım

FB-CPU 10 bitlik komutunda, ilk 4 biti [9:6] operasyon kodunu, son 6 biti ise [5:0] adresi temsil etmektedir. Tablo 1'de verilmiş olan komutlar ile uygulamalar geliştirilmelidir. İşlemcinin doğru çalışıp çalışmadığı aşağıdaki kod parçacıkları ile test edilebilir. Tasarımdaki RAM'e sağ tıklayıp edit content'e tıklandığında, open tuşu ile verilen yazılım yüklenebilir.

7.1. Test Yazılımı 1

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

```
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
```

İndirme adresi: http://levent.tc/courses/blm101/proje/BLM101_proje_testYazilim1

7.2. Test Yazılımı 2

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

```
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
```

İndirme adresi: http://levent.tc/courses/blm101/proje/BLM101_proje_testYazilim2

7.3. Test Yazılımı 3

FB-CPU için bellekte 50 ve 51 adresteki iki sayının çarpımını 52 no'lu adrese kaydeden uygulamayı geliştiriniz. Ancak çarpma operasyonunu kullanmayınız. Çarpma işlemi için 50'deki sayıyı 51'deki sayı defa toplayıp 52 no'lu adrese yazınız. Gerekli değişkenler için istediğiniz adresleri kullanabilirsiniz.

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

İndirme adresi: http://levent.tc/courses/blm101/proje/BLM101_proje_testYazilim3

8. Notlandırma ve Proje Teslimi:

Bu başlık FB-CPU'nun proje teslimi ve notlandırılması hakkında bilgiler içermektedir.

8.1. Notlandırma:

Projenin **iki** ana değerlendirme kriteri vardır. Her iki kriter 50 şer puandır.

İlk kriter FB-CPU'nun komutlarının (instructions) **doğru çalıştırılmasıdır**. FB-CPU'nun 10 komutu (Instruction)'u vardır. Her bir komut'un doğru sonuç üretip üretememesine göre değerlendirme yapılacaktır.

İkinci kriter ise **Proje Teslim Dokümanı ve Sunumdur**.

- **Proje Teslim Dokümanı:**

Öğrenciler, proje raporlarını verilen "Proje Teslim Dokümanı" 'nın içerisini doldurarak yapacaklardır.

Proje Teslim Dokümanı: http://levent.tc/courses/blm101/proje/BLM101_proje_teslim_dokumani.doc

- **Proje Sunumu:**

Powerpoint üzerinde ortalama 5 dakika (4-6 dakika arası) sürecek bir sunum hazırlayarak kayıt etmelidirler. Kayıt işlemi, cep telefonu veya bilgisayar ekran kayıt yazılımları (Screen-Recorder, Bandicam vb…) ile yapılabilir.

Sunum, ekip üyeleri içinden biri tarafından, projenin nasıl yapıldığı, işlemcinin nasıl çalıştığı vb.. konularının powerpoint slaytları üzerinden anlatılırken kaydedilmesi ile olmalıdır. Sunum video'sunda powerpoint slaytları okunabilir ve konuşmacının sesinin anlaşılır olması gerekmektedir. Powerpoint slayt görünüm tasarımı istenildiği gibi yapılabilir.

Proje ekibinin tamamı, notlarını bu değerlendirmeye göre alırlar.

8.2. Teslim:

Projenin teslimi için aşağıdaki adımların gerçekleştirilmesi gerekmektedir. İstenen dosyaları sadece proje ekip sorumlusunun getirmesi, LMS ve Github (Çok yaygın bir açık kaynak kod paylaşım platformudur)'a yüklemelidir.

Proje Teslim Dokümanının, ders sınıfı ve saatinde, çıktılarının alınarak teslim edilmesi gerekmektedir.

Ayrıca LMS'te açılmış olan "Proje Teslim" sayfasına aşağıdaki dosyaların yüklenmesi gerekmektedir.

- Logisim simulatoru üzerinde yapılan işlemci tasarımı (.circ uzantılı dosya)
- Makine dilindeki iki yazılım (.fbcpusoftware uzantılı dosyalar)
- Hazırlanan powerpoint sunum dosyası (.ppt uzantılı dosya)
- Proje Teslim Dokümanı (Word formatında yüklenmelidir)
    - Dokümanın alt başlıkları doldurulmalıdır
    - Kaydedilen powerpoint sunum video'su youtube'a yüklenip, adresi, dokümanın sonuçlar bölümündeki açılmış yere link'i yazılmalıdır (Video'nun herkes'e görünür olmamasını istiyorsanız, youtube'a yükledikten sonra liste dışı seçeneğini seçerek, sadece link'e sahip olan kişilerin görmesini sağlayabilirsiniz).
    - LMS'e yüklenen tüm dosyalar (Logisim işlemci tasarım dosyası, makine dilindeki iki yazılım, ppt uzantılı sunum dosyası ve Proje Teslim Dokümanını (PDF formatında)), github.com sitesine üye olup, yüklenip, dokümanın sonuçlar bölümündeki yere link'i yazılmalıdır.