Fenerbahçe Üniversitesi

BLM 101 – Bilgisayar Mühendisliğine Giriş

FB-CPU Tasarımı

Ahmet Hazar Haspolat
190301012

Ekrem Büyükkaya
190301016

Mustafa Berk Taşkın
190301021

Ömer Sait Yorulmaz
190301025

Projenin Amacı

Bu proje kapsamında FB_CPU isimli bir işlemci tasarlayıp, bu tasarım üzerinden simülatör aracılığıyla, makine dili ile yazılan kod parçacıkları çalıştırılacaktır. Yapılan tasarım sonucunda ram, kontrol üniteleri ve saklayıcılar gözlemlenebilecek ve aralarındaki ilişkiler bütünü ile birlikte bu makine kodlarının nasıl çalıştığı deneyimlenebilecektir.

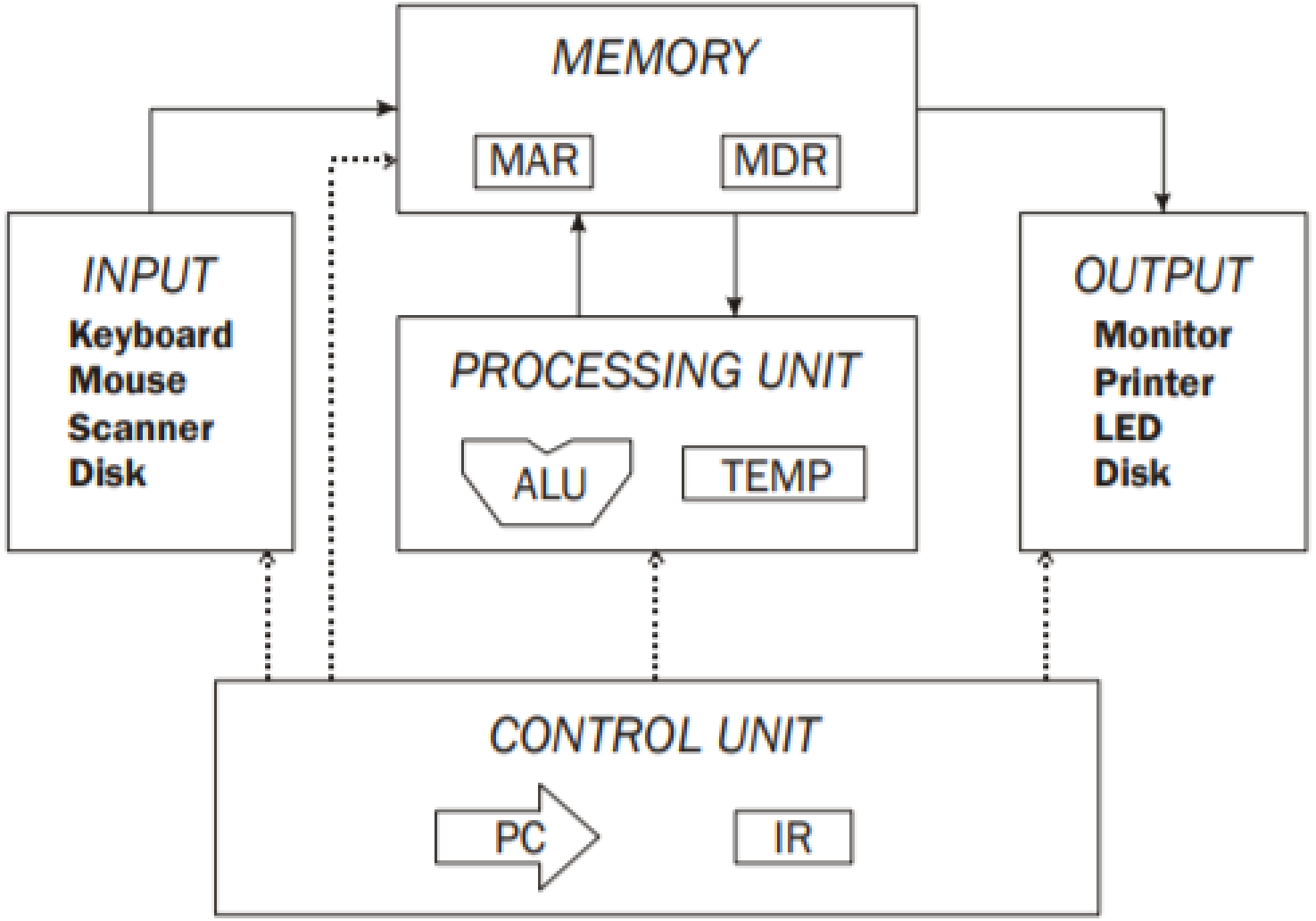
Kullanılan Araçlar

Von Neumann Simulatörü:

Tasarımı gerçekleştirilen ve veri akışlarının daha rahat bir şekilde gözlemlenebildiği simülatördür.

Logisim-Evolution:

Logisim Evolution, digital mantık devrelerini tasarlamak ve bunları simüle etmek için kullanılan java tabanlı bir eğitim aracıdır. Bu devrelerle alakalı kavramları öğrenmeyi kolaylaştırmakla birlikte, asıl tasarım bu araç kullanılarak yapıldı.

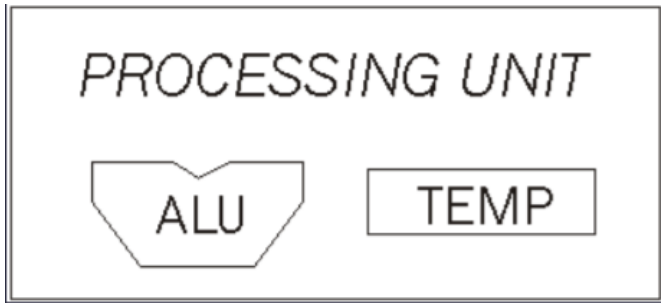

MEMORY
MAR
MDR
INPUT
Keyboard
Mouse
Scanner
Disk
PROCESSING UNIT
ALU
TEMP
OUTPUT
Monitor
Printer
LED
Disk
CONTROL UNIT
PC
IR

Bellek

• İşlemci ünitesinin belleğe erişimi iki saklayıcı ile olmaktadır:
• MAR: Memory Address Register
• MDR: Memory Data Register

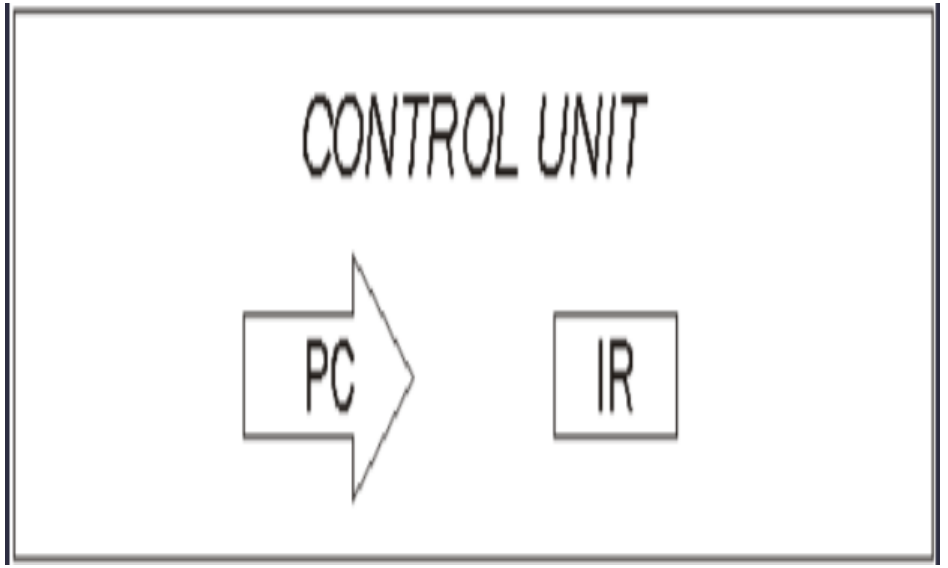
İşlemci Ünitesi

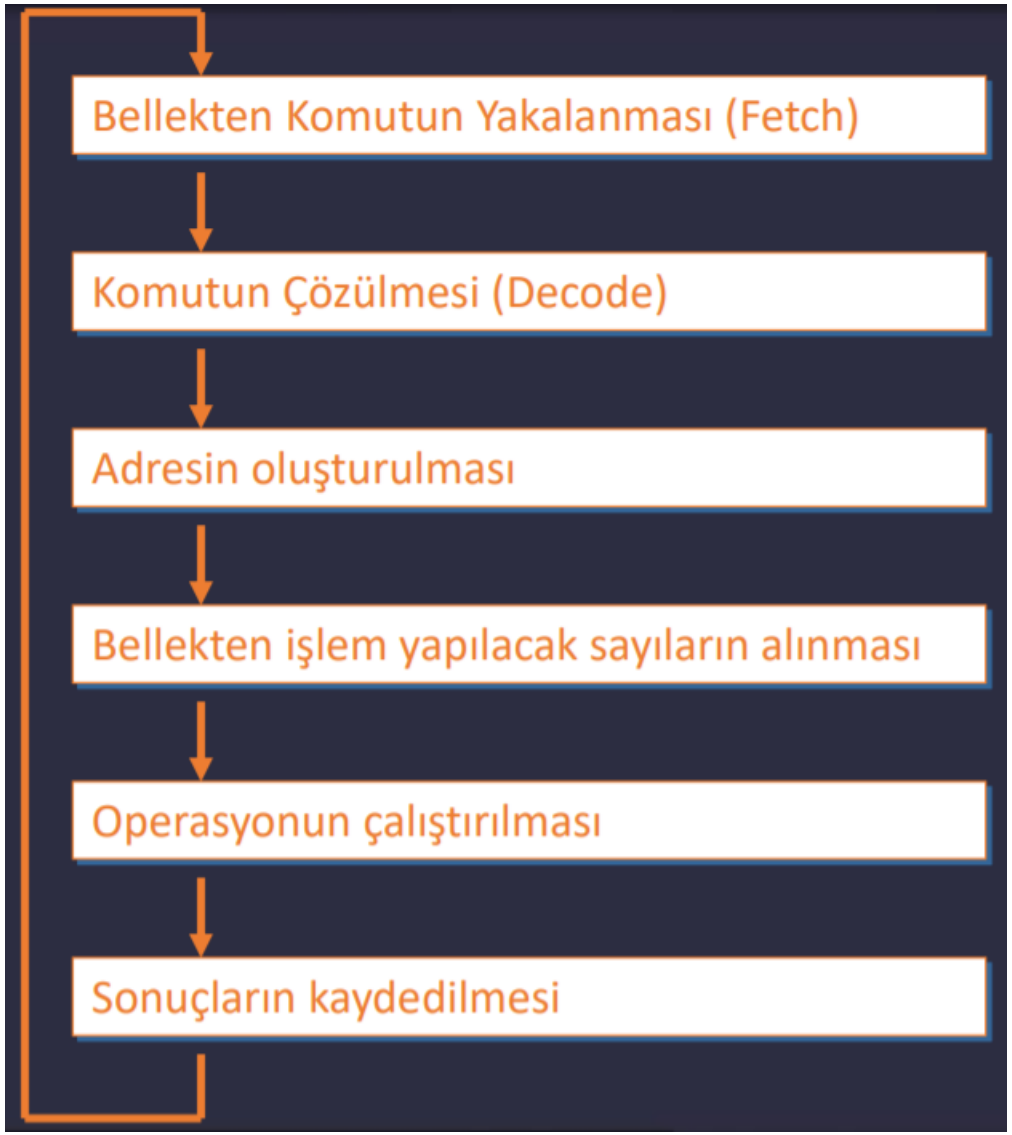
• ALU = Aritmetik ve Lojik Ünitesi

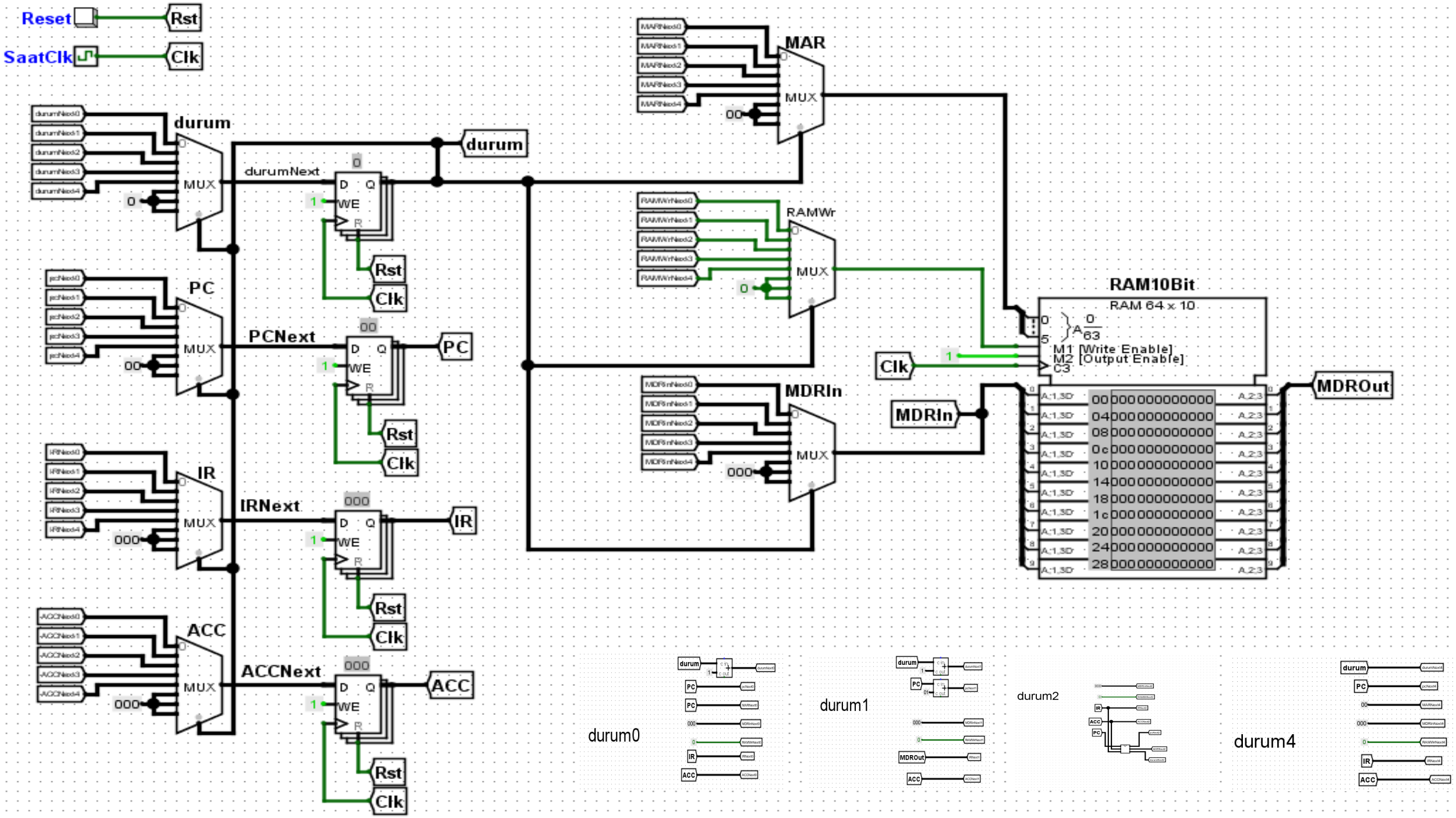


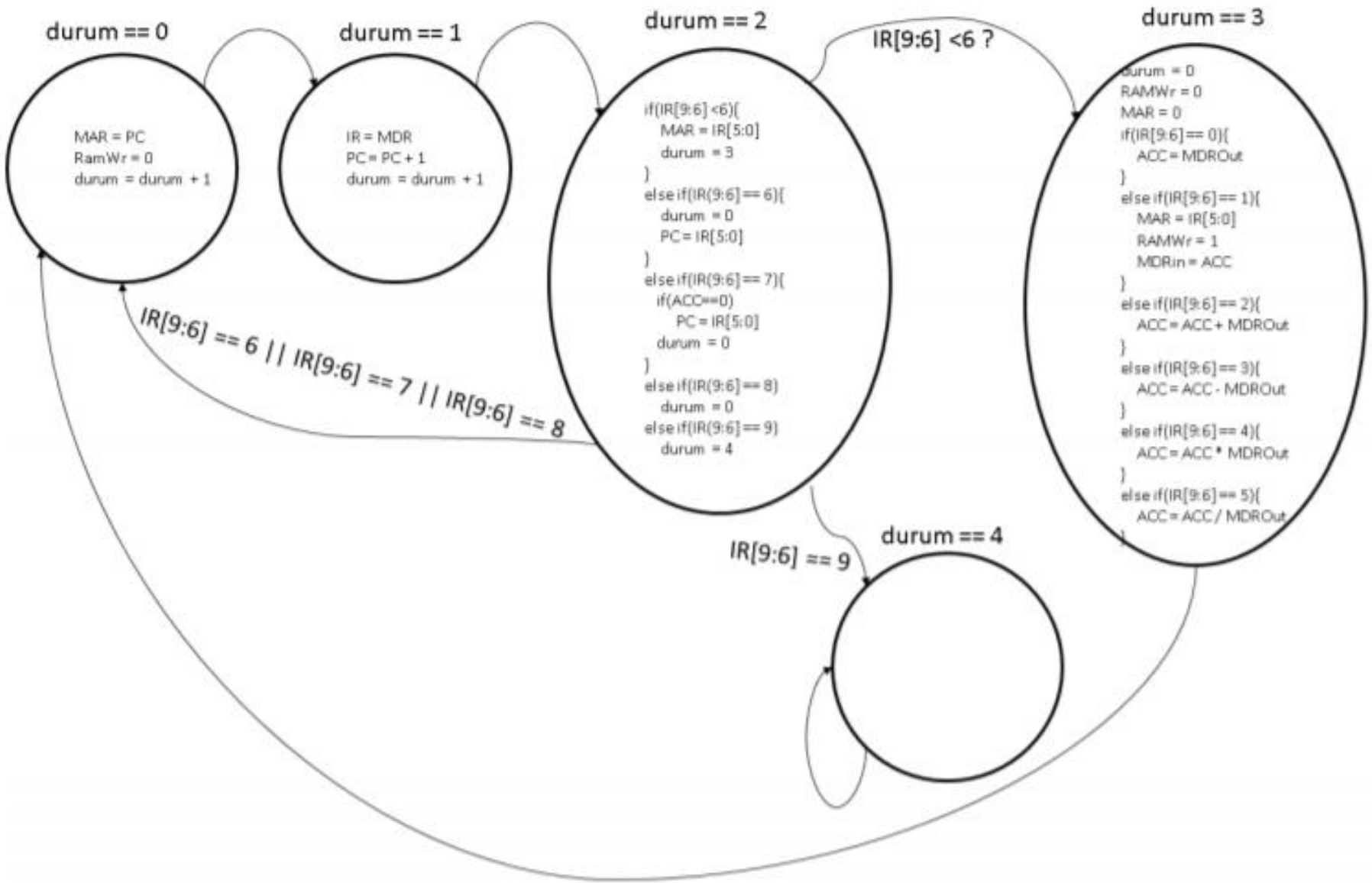
Kontrol Ünitesi:

- Bellekten program counter'ın göstediği komutu okur.
- Sistemin geri kalanına, yapılması gereken işlemleri yaptırır.
- Programın akışını yönetmektedir.
- Program Counter (PC) bir sonraki koşturulacak olan komutun adresini tutar.
- Instruction Register (IR) şu anki koşturulan komutun adresini tutmaktadır.

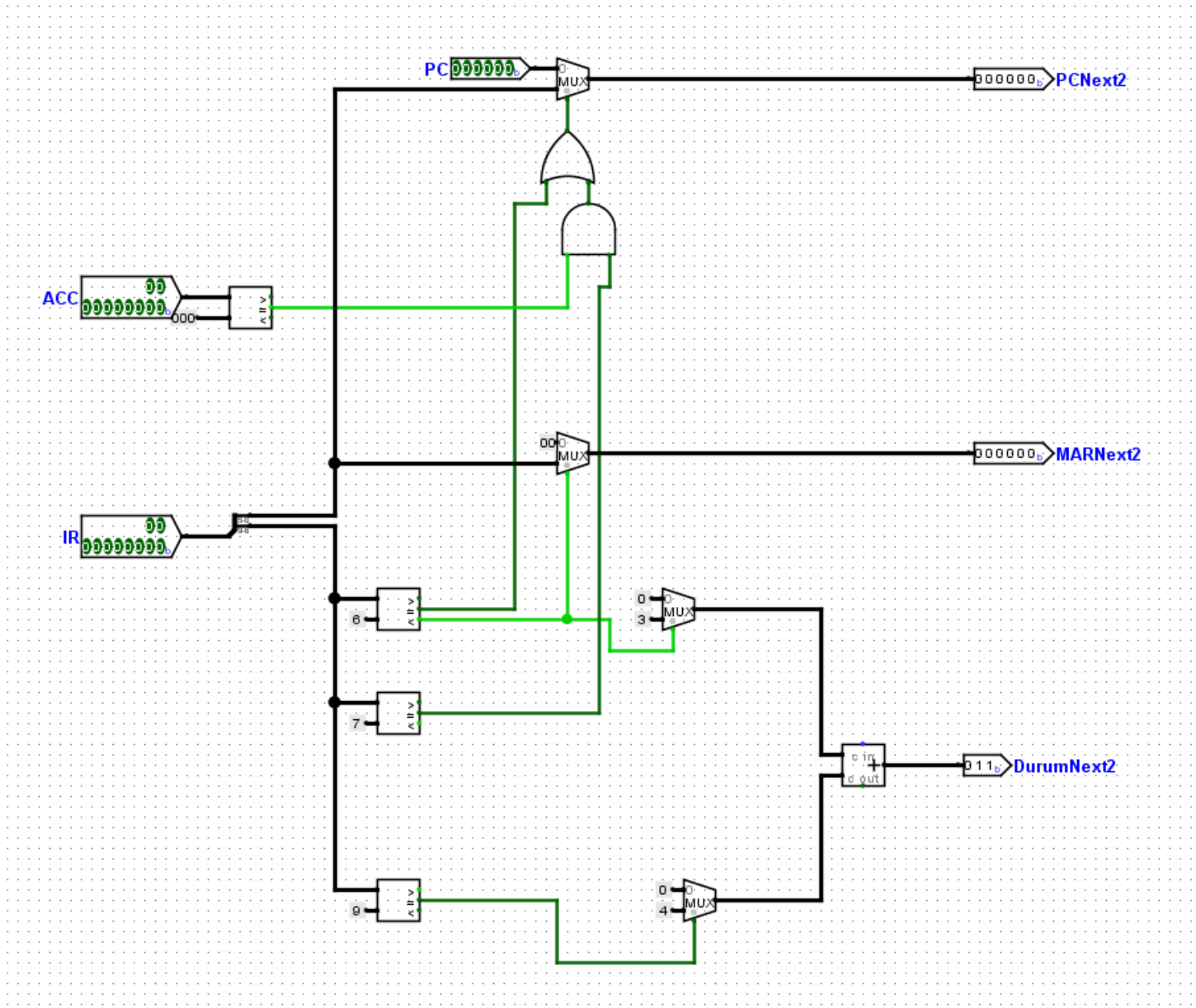



Bellekten Komutun Yakalanması (Fetch)
Komutun Çözülmesi (Decode)
Adresin oluşturulması
Bellekten işlem yapılacak sayıların alınması
Operasyonun çalıştırılması
Sonuçların kaydedilmesi


Reset
Rst
SaatClk
Clk
durum
durumNext
PC
PCNext
IR
IRNext
ACC
ACCNext
MAR
RAMWr
MDRIn
MDROut
RAM10Bit
RAM 64 x 10
M1 [Write Enable]
M2 [Output Enable]
C3
MUX
D Q
WE
R
durum0
durum1
durum2
durum4


durum == 0
MAR = PC
RamWr = 0
durum = durum + 1
durum == 1
IR = MDR
PC = PC + 1
durum = durum + 1
durum == 2
if(IR[9:6] <6){
MAR = IR[5:0]
durum = 3
}
else if(IR[9:6] == 6){
durum = 0
PC = IR[5:0]
}
else if(IR[9:6] == 7){
if(ACC==0)
PC = IR[5:0]
durum = 0
}
else if(IR[9:6] == 8)
durum = 0
else if(IR[9:6] == 9)
durum = 4
IR[9:6] <6 ?
durum == 3
durum = 0
RAMWr = 0
MAR = 0
if(IR[9:6] == 0){
ACC = MDROut
}
else if(IR[9:6] == 1){
MAR = IR[5:0]
RAMWr = 1
MDRin = ACC
}
else if(IR[9:6] == 2){
ACC = ACC + MDROut
}
else if(IR[9:6] == 3){
ACC = ACC - MDROut
}
else if(IR[9:6] == 4){
ACC = ACC * MDROut
}
else if(IR[9:6] == 5){
ACC = ACC / MDROut
IR[9:6] == 6 || IR[9:6] == 7 || IR[9:6] == 8
IR[9:6] == 9
durum == 4

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



durum == 2

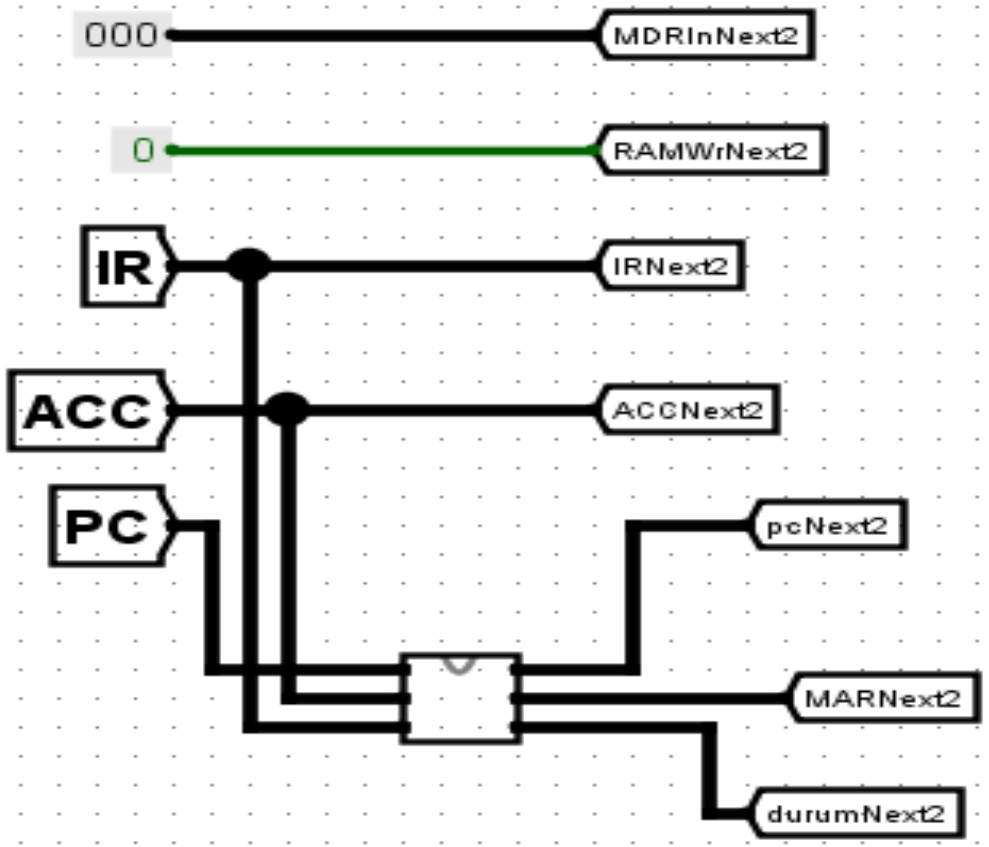
```
if(IR[9:6] <6){
   MAR = IR[5:0]
   durum  = 3
}
else if(IR(9:6] == 6){
   durum  = 0
   PC = IR[5:0]
}
else if(IR(9:6] == 7){
  if(ACC==0)
     PC = IR[5:0]
  durum  = 0
}
else if(IR(9:6] == 8)
   durum  = 0
else if(IR(9:6] == 9)
   durum  = 4
```

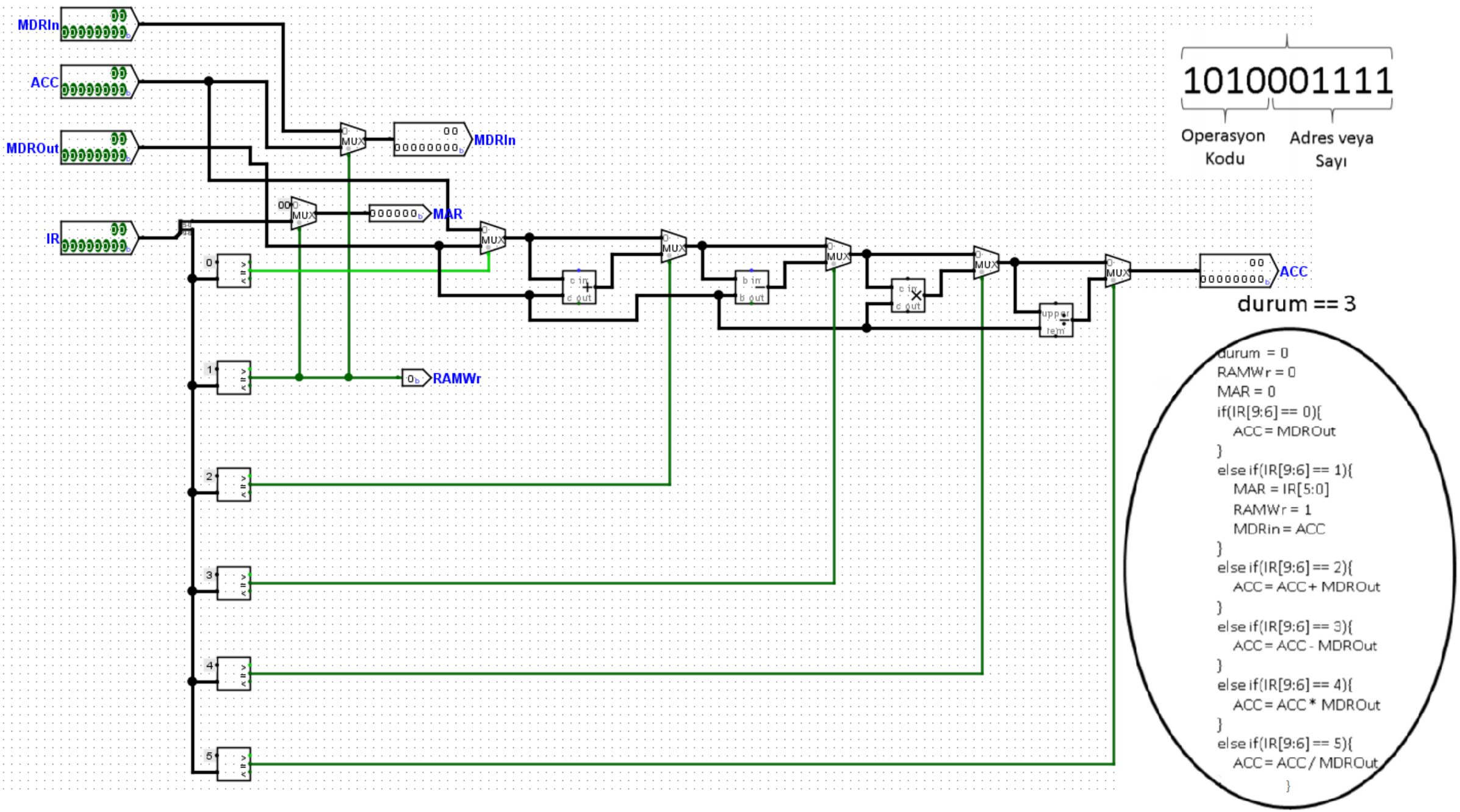
1010001111

Operasyon Kodu

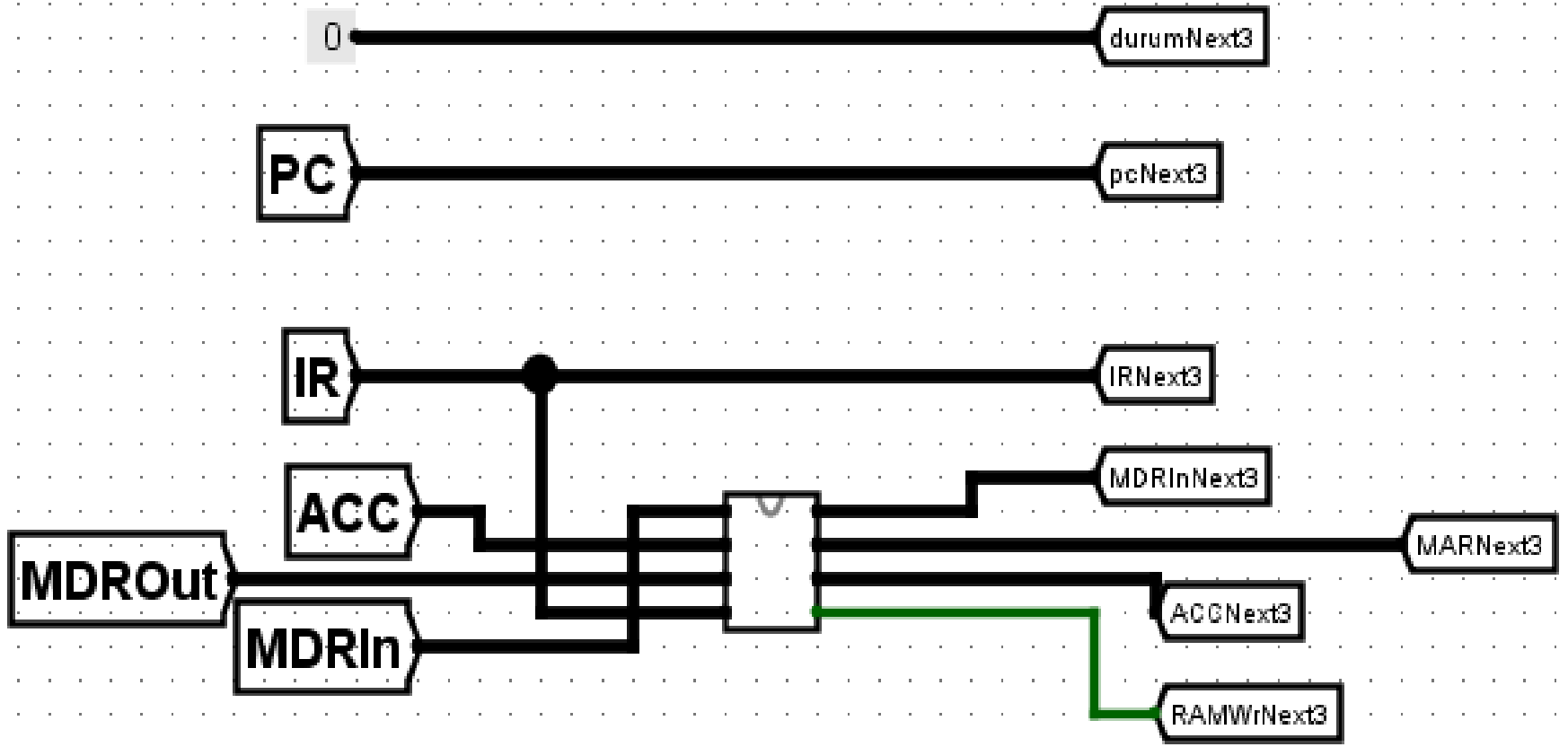
Adres veya Sayı

durum2


000
MDRInNext2
0
RAMWrNext2
IR
IRNext2
ACC
ACCNext2
PC
pcNext2
MARNext2
durumNext2


MDRIn
ACC
MDROut
IR
MUX
MDRIn
MAR
RAMWr
ACC
1010001111
Operasyon Kodu
Adres veya Sayı
durum == 3
durum = 0
RAMWr = 0
MAR = 0
if(IR[9:6] == 0){
ACC = MDROut
}
else if(IR[9:6] == 1){
MAR = IR[5:0]
RAMWr = 1
MDRin = ACC
}
else if(IR[9:6] == 2){
ACC = ACC + MDROut
}
else if(IR[9:6] == 3){
ACC = ACC - MDROut
}
else if(IR[9:6] == 4){
ACC = ACC * MDROut
}
else if(IR[9:6] == 5){
ACC = ACC / MDROut
}

durum3


0
durumNext3
PC
pcNext3
IR
IRNext3
ACC
MDRInNext3
MDROut
MARNext3
ACCNext3
MDRIn
RAMWrNext3

SONUÇLAR

Yapılan bu sistem ile, işlem süreçlerinin görselleştirilmesi sonucunda gerekli iş kırımları daha net gözlemlenebilmiş ve etkili öğrenme çıktıları alınmıştır.

Dosyaların github adresi: https://github.com/fbuni/BLM101