

İstanbul Üniversitesi - Cerrahpaşa

Mühendislik Fakültesi

Bilgisayar Mimarisi Dersi

Ödev Raporu

Ebru Yassı

Burhan Tarık Doğdu

İlhan Güler

Nurdamla Mutlu

**İçerik**

1. [Ödev Geliştirme Aşamaları](#_bookmark0) 3

[1.1](#_bookmark1) Ödev Konusu . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3

[1.2 Görev](#_bookmark2) Dağılımı . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.3 İcra Planı . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1. [Tasarım](#_bookmark10) ve Gerçekleme 5

[2.1](#_bookmark5) Devre Açıklaması . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . 5

[2.2 Devre Yapılarının](#_bookmark5) Çizimleri. . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

[2.3](#_bookmark3) Kontrol Sinyalleri. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

2.3.1 Kontrol Ünitesi Hakkında Açıklamalar. . . . . . . . . . . . . . . . . . . . . . . . . . 13

[2.3.1](#_bookmark3) Doğruluk Tablosu. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . 14

2.4 Doğrulama. . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

1. Benzetim ve Test 18
   1. Simülasyon Kısmındaki Durumlar . . . . . . . . . . . . . . . .. . . . . . . . . .18
   2. Test Caseler. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . 18
   3. Çıktıların Ekran Görüntüleri . . . . . . . . . . . . .. . . . . . . .. . . . . . . . . . .. . . . . .. . . . 20

# 1. Ödev Geliştirme Aşamaları

# Ödev Konusu

# Ödevde bizden bir 16 bitlik pipelined processor yapmamız ve yaptığımız processor’dan verilmiş olan test case’leri gerçeklenmemiz istendi.

# Ödevimizin rapor aşamasında yaptığımız uygulamayı detaylıca anlatıp gerçekleyeceğiz.

# Görev Dağılımı

Burhan Tarık Doğdu

Proje Yöneticisi

Mimari Yapının Belirlenmesi ve ALU

Ebru Yassı

Bilgisayar Mimarisi Proje Ödevi

Register Yapıları ve Birleştirmeler

İlhan Güler

Testing ve Kontrol

Nurdamla Mutlu

# 1.3 İcra Planı

|  |  |
| --- | --- |
| TARİH | YAPILAN İŞLEM |
| 1.05.2022 | Ödevin verilmesi |
| 5.05.2022 | Grubun kurulması |
| 10.05.2022 | Araştırmaların başlaması |
| 11.05.2022 | Zoom toplantılarının başlaması |
| 11.05.2022 | Görev dağılımlarının yapılması |
| 12.05.2022 | Devreler hakkında görüşmeler |
| 14.05.2022 | ALU’daki yapı hakkında düzeltmeler |
| 16.05.2022 | Register yapısı, kontrol ünitesi ve ALU’nun birleştirilmesi için ilk çalışmalar |
| 18.05.2022 | Register yapısı, kontrol ünitesi ve ALU’nun birleştirilmesi için 2. çalışmalar |
| 20.05.2022 | Register yapısı, kontrol ünitesi ve ALU’nun birleştirilmesi için 3. Çalışmalar(Ek1’de processor’ın pipelinesız hali mevcut. Komutları çalıştırmak için ilk denemelerimiz) |
| 21.05.2022 | Pipeline yapısının düzenlenmesi ve entegre edilebilir hale getirilmesi için ilk çalışmalar |
| 22.05.2022 | Pipeline yapısının düzenlenmesi ve entegre edilebilir hale getirilmesi için 2. çalışmalar |
| 23.05.2022 | Test caselerinin hexadecimal yapılıp testlerin başlanması |
| 24.05.2022 | Bütün verilerle raporun hazırlanması |
| 25.05.2022 | Ödevin teslimi |

# 

# (Ek 1)

# 2. Tasarım ve Gerçekleme

# 2.1 Devre Açıklaması

# Devremizi yaparken 4 adet pipeline kullandık. ALU, kontrol ünitesi, mux, Hazard ünitesi ve Register yapısı kullandık. Stall yapısı kullanmadık. Yerine kullandığımız “Constant 0”’ın performansı arttırdığı gözlemledik. Koyduğumuz RAM’den alınan bütün hexadecimal olarak verilen komutları (BEQ, BNE komutları haricinde) çalıştırabiliyoruz.

# 2.2 Devre Yapılarının Çizimleri

# Pipelined Processor’ın Main Devresi

# 

# Fetch-Decode-Jump Unit

# 

# Register Unit

# ALU

# 

# MUX

# 

# Delay Unit

# metin, tablo içeren bir resim Açıklama otomatik olarak oluşturuldu

# 1. Pipeline Yapısı

# 

# 2. Pipeline Yapısı

# tablo içeren bir resim Açıklama otomatik olarak oluşturuldu

# tablo içeren bir resim Açıklama otomatik olarak oluşturuldu

# 3. Pipeline Yapısı

# tablo içeren bir resim Açıklama otomatik olarak oluşturuldu

# tablo içeren bir resim Açıklama otomatik olarak oluşturuldu

# 4. Pipeline Yapısı

# tablo içeren bir resim Açıklama otomatik olarak oluşturuldu

# Hazard Unit

# 

# Control Unit

# 

# 2.3 Kontrol Sinyalleri

# 2.3.1 Kontrol Ünitesi Hakkında Açıklamalar

# 

# Yapıların Görevleri:

# 

# Single Register Instr = Rt'ye yazdırmak

# Single Third Register Instr = Rs'ye yazdırmak

# R1 write = R1’e yazdırma

# R7 write = R7’ye yazdırma

# Write = Registera yazdırma

# Alu Source = Alu veya Memoryden gelen değerler arasında seçim yaptırmak

# 

# (Kontrol Ünitesi)

# 2.3.2 Doğruluk Tablosu

# 

# 2.4 Doğrulama

# Devremizin çalıştığını ve doğru bir implementasyon olduğunu herhangi bir test case’i baz alarak yapmaya karar verdik. Aldığımız komut “add r1, r2, r3” oldu.

# Buna göre aldığımız çıktılar:

# 

# Kontrol Ünitesi

# tablo içeren bir resim Açıklama otomatik olarak oluşturuldu

# 2. Pipeline Yapısı

# tablo içeren bir resim Açıklama otomatik olarak oluşturuldu

# 4. Pipeline Yapısı

# 

# Fetch-Decode-Jump Ünitesi

# tablo içeren bir resim Açıklama otomatik olarak oluşturuldu

# MUX

# 

# ALU

# 

# Register Kısmı

# 3. Benzetim ve Test

# 3.1 Simülasyon Kısmındaki Durumlar

# Avantajlar: Pipelinelar sayesinde nerede hata aldığımızı daha rahat görebiliyoruz. Devremizde aldığımız hatalar gündemimize girdiğinde devremiz halihazırda temiz ve düzgün bir yapıda olduğu için ekstra bir component ve register ihtiyacında istediğimiz gibi eklemeler yapabildik.

# Dezavantajlar: Çok fazla input, outout ve bus yapısı olduğu için kaymalar oldu. Devremiz temiz olmasına rağmen bağlantılarda elimizde olmayan şekillerde bozulmalar olabildi. Ayrıca pipeline kullanımına bağlı senkronizasyon sıkıntıları yaşadığımız oldu.

# Ancak çalışmalarımızın sonucu olarak elimizden geldiğince bug free bir devre yapısı oluşturmaya çalıştık. Şimdiye kadar karşımıza gelen bir ciddi hata olmadı.

# 3.2 Test Caseler

**R-Type Komutlar**

|  |  |  |
| --- | --- | --- |
| **Komutlar** | **Hexadecimal** | **Çıktılar** |
| And r1, r2, r3 | 0053 | r1 = 0x0022 |
| Or r1, r2, r3 | 0253 | r1 = 0x0033 |
| xor r1, r2, r3 | 0453 | r1 =0x0011 |
| Nor r1, r2, r3 | 0653 | r1 =0xffcc |
| add r1, r2, r3 | 0853 | r1 =0x0055 |
| Sub r1, r2, r3 | 0a53 | r1 =0xffef |
| Slt r1, r2, r3 | 0c53 | r1 =0x0001 |
| Sltu r1, r2, r3 | 0e53 | r1 =0x0001 |

|  |  |  |
| --- | --- | --- |
| Jr r2 | 1010 | Jumps to Rt |

**I-Type Komutlar**

|  |  |  |
| --- | --- | --- |
| **Komutlar** | **Hexadecimal** | **Çıktılar** |
| Andi r3, r2, 12 | 2293 | r3 = 0x0002 |
| Ori r3, r2, 12 | 2a93 | r3 = 0x002a |
| Xori r3, r2, 12 | 3293 | r3 = 0x0028 |
| Addi r3, r2, 12 | 3a93 | r3 = 0x002c |
| Sll r3, r2, 12 | 4293 | r3 = 0x8800 |
| Srl r3, r2, 12 | 4a93 | r3 = 0x0000 |
| Sra r3, r2, 12 | 5293 | r3 = 0x0000 |
| Ror r3, r2, 12 | 5a93 | r3 = 0x0cc0 |
| Lw r3, r2, 12 | 6293 | r3=M[0.002c] |
| Sw r3, r2, 12 | 6a93 | M[0x001b]=0x0022 |
| Beq r3, r2, 12 | 7293 | Branch if r2=r3 |
| Bne r3, r2, 12 | 7a93 | Branch if r2!=r3 |

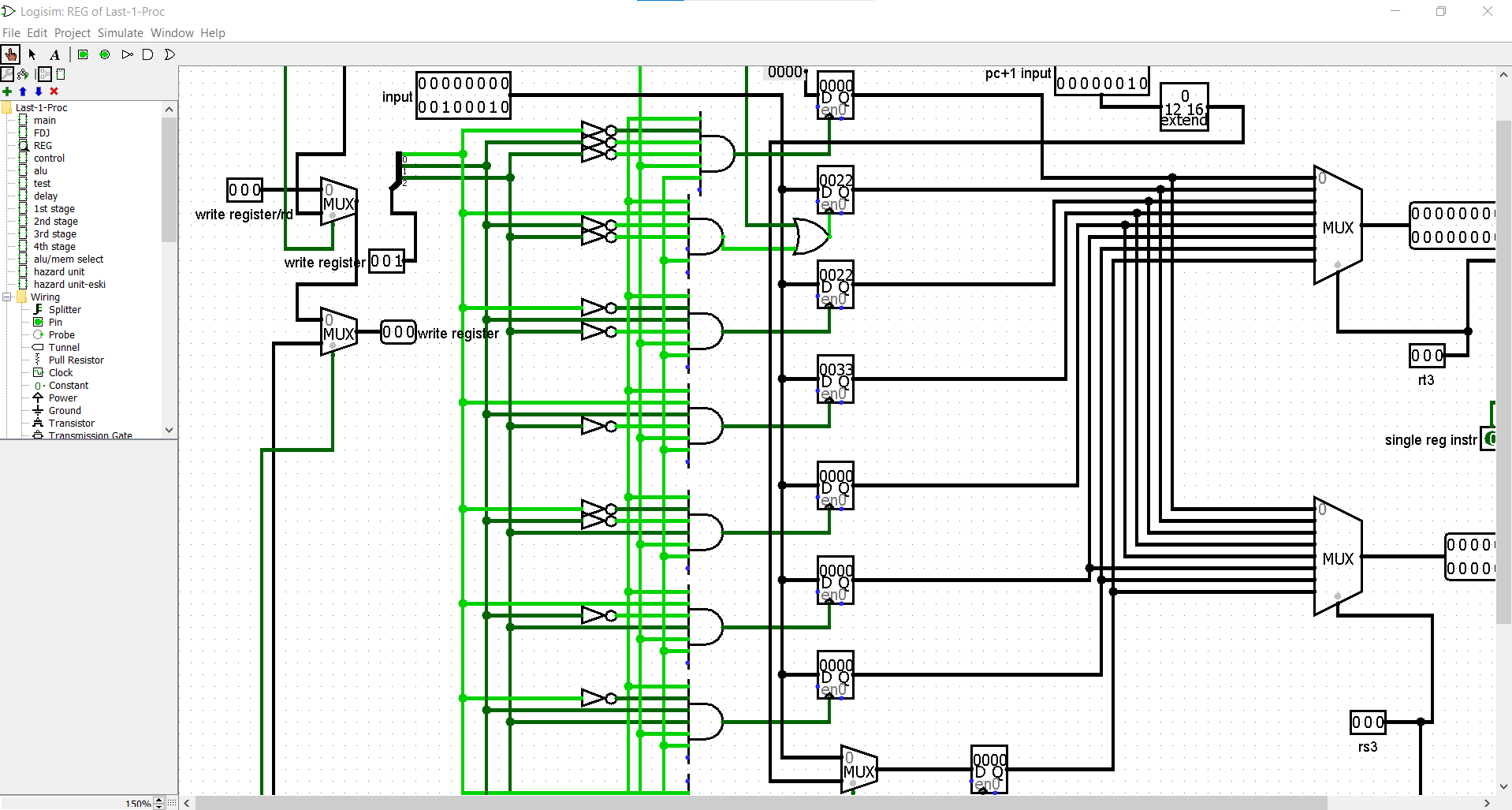
**J-Type Komutlar**

|  |  |  |
| --- | --- | --- |
| **Komutlar** | **Hexadecimal** | **Çıktılar** |
| Lui 659 | 8293 | 0x5260 |
| J 659 | f293 | PC+Imm11 |
| Jal 659 | fa93 | PC+1 or PC+Imm11 |

# Not: Çıktılar alınırken r1 = 0x0011, r2 = 0x0022, r3 = 0x0033 şeklinde atanarak yapılmıştır.

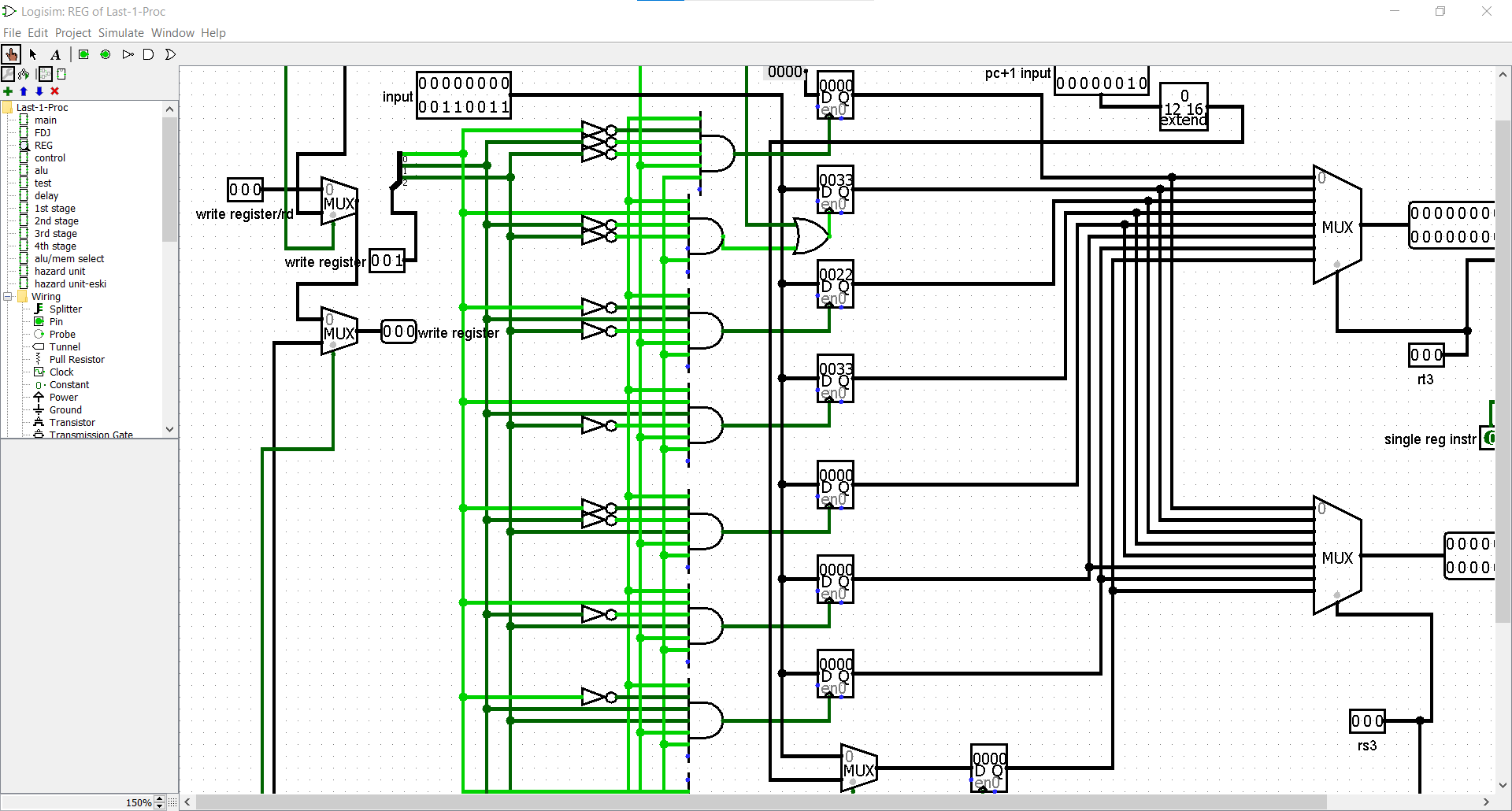
# 3.3 Çıktıların Ekran Görüntüleri

**And r1, r2, r3 Komutu Komutu**

****

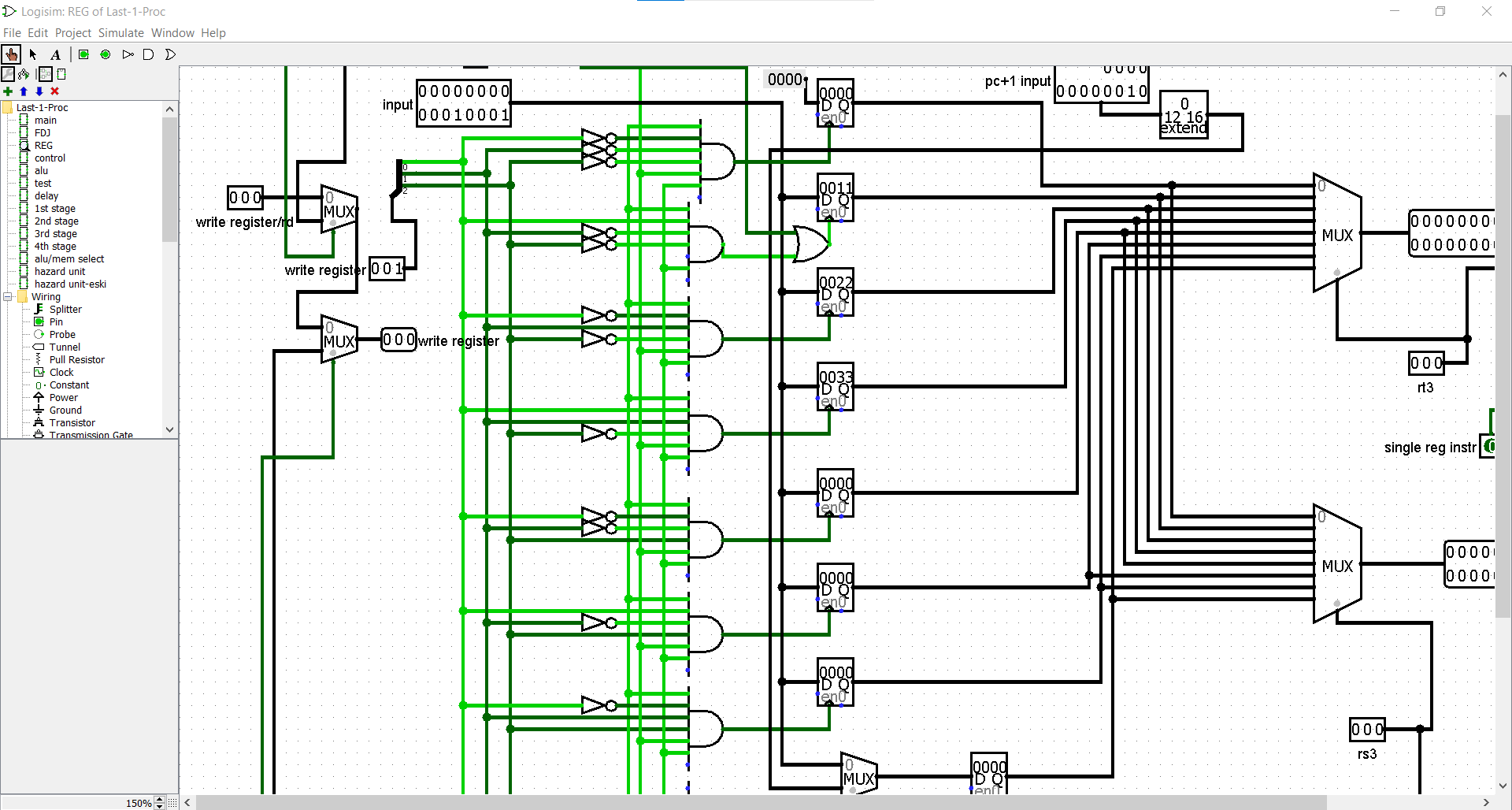
Durum Çıktısı

**Or r1, r2, r3 Komutu**

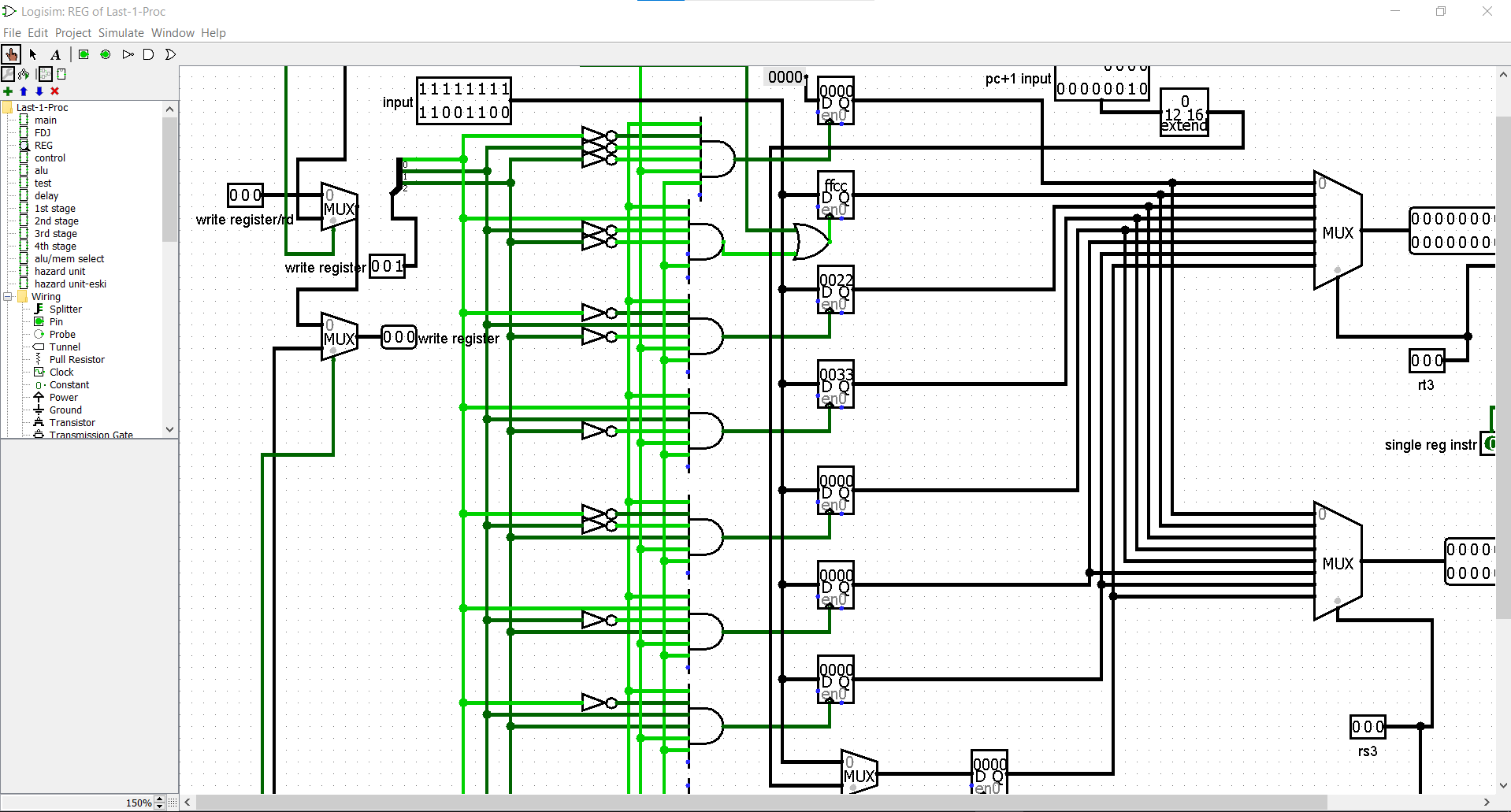
****

Durum Çıktısı

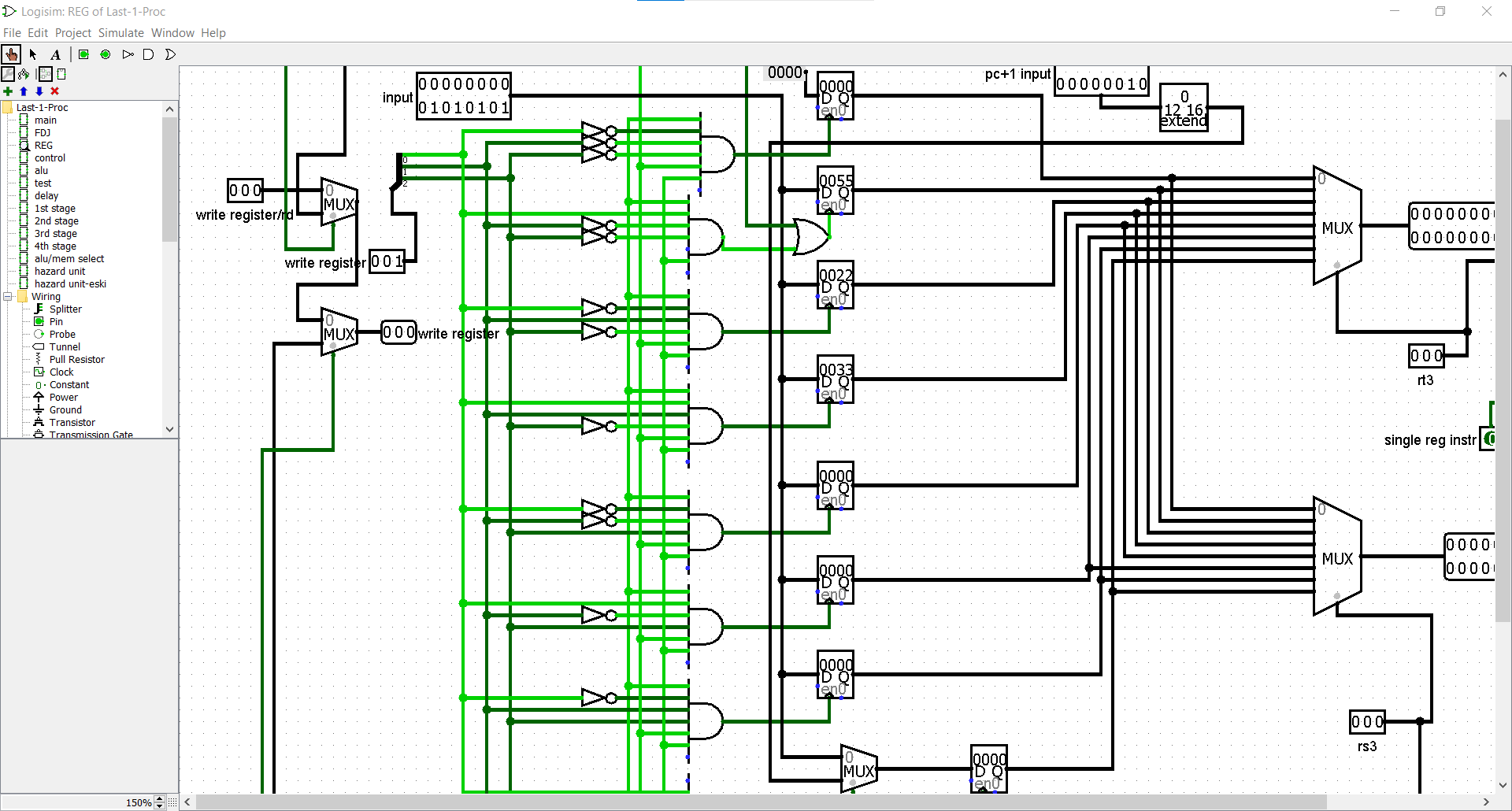
**Xor r1, r2, r3 Komutu**



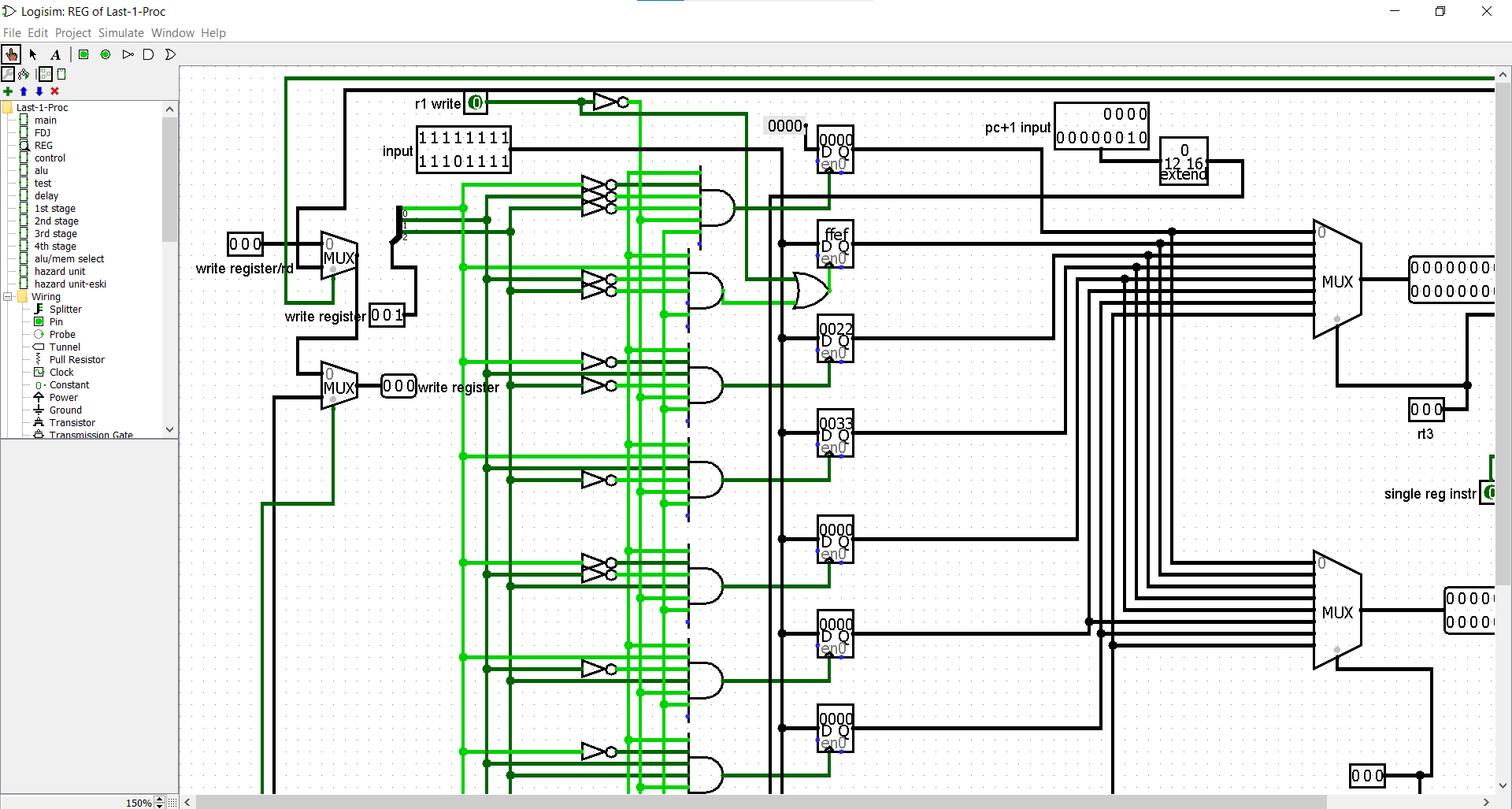
**Nor r1, r2, r3 Komutu**

****

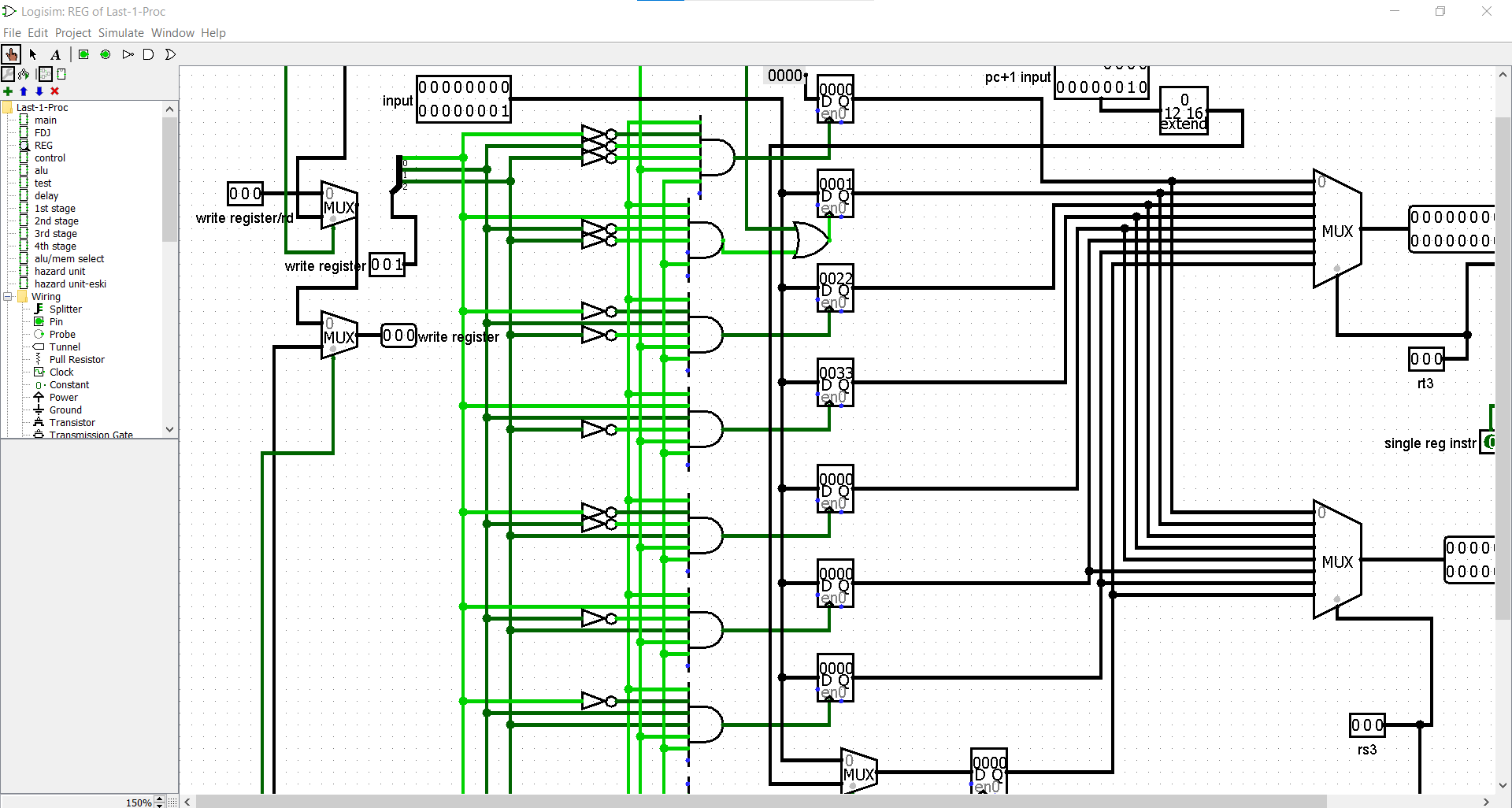
**Add r1, r2, r3 Komutu**

****

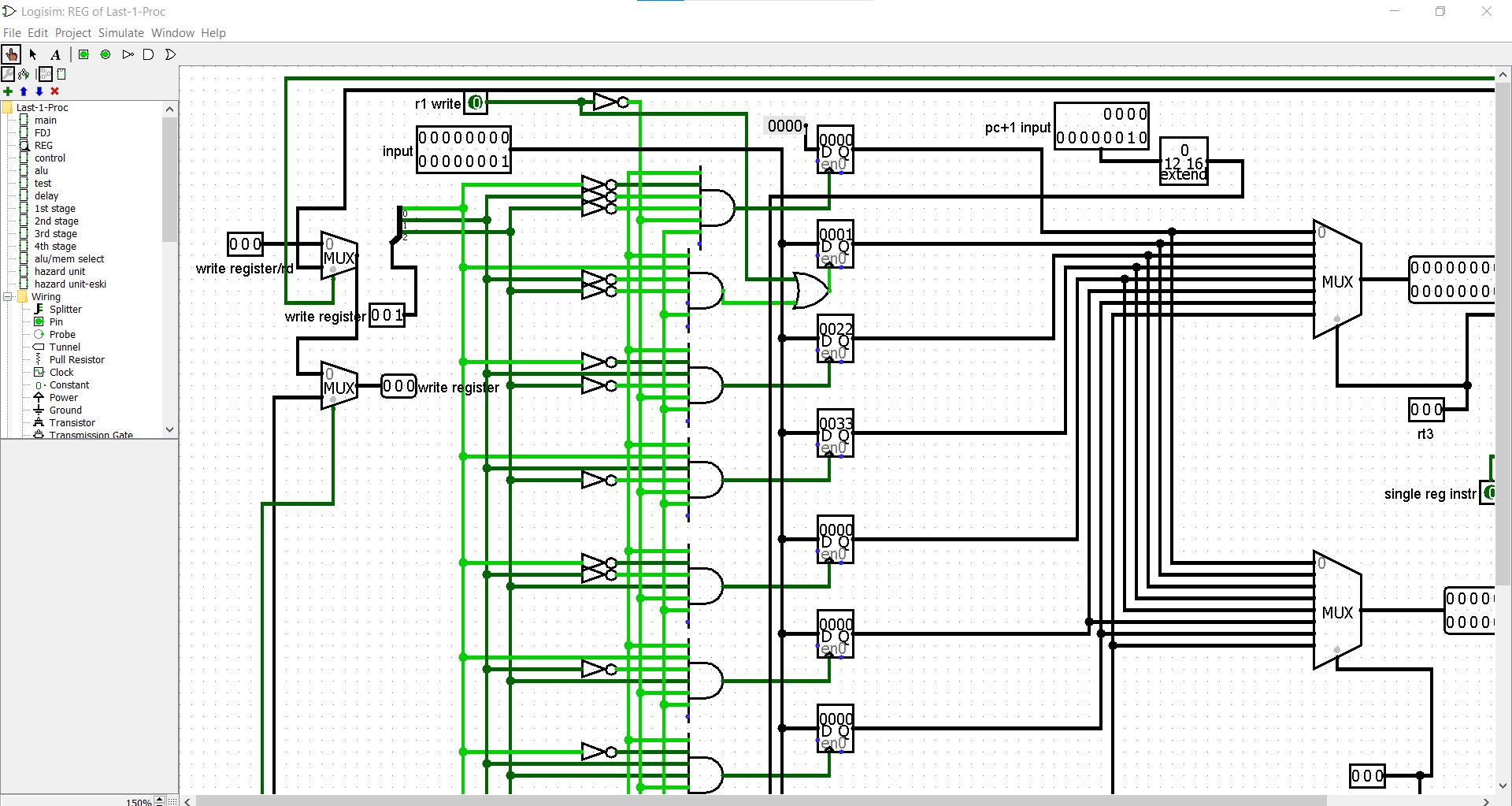
**Sub r1, r2, r3 Komutu**

****

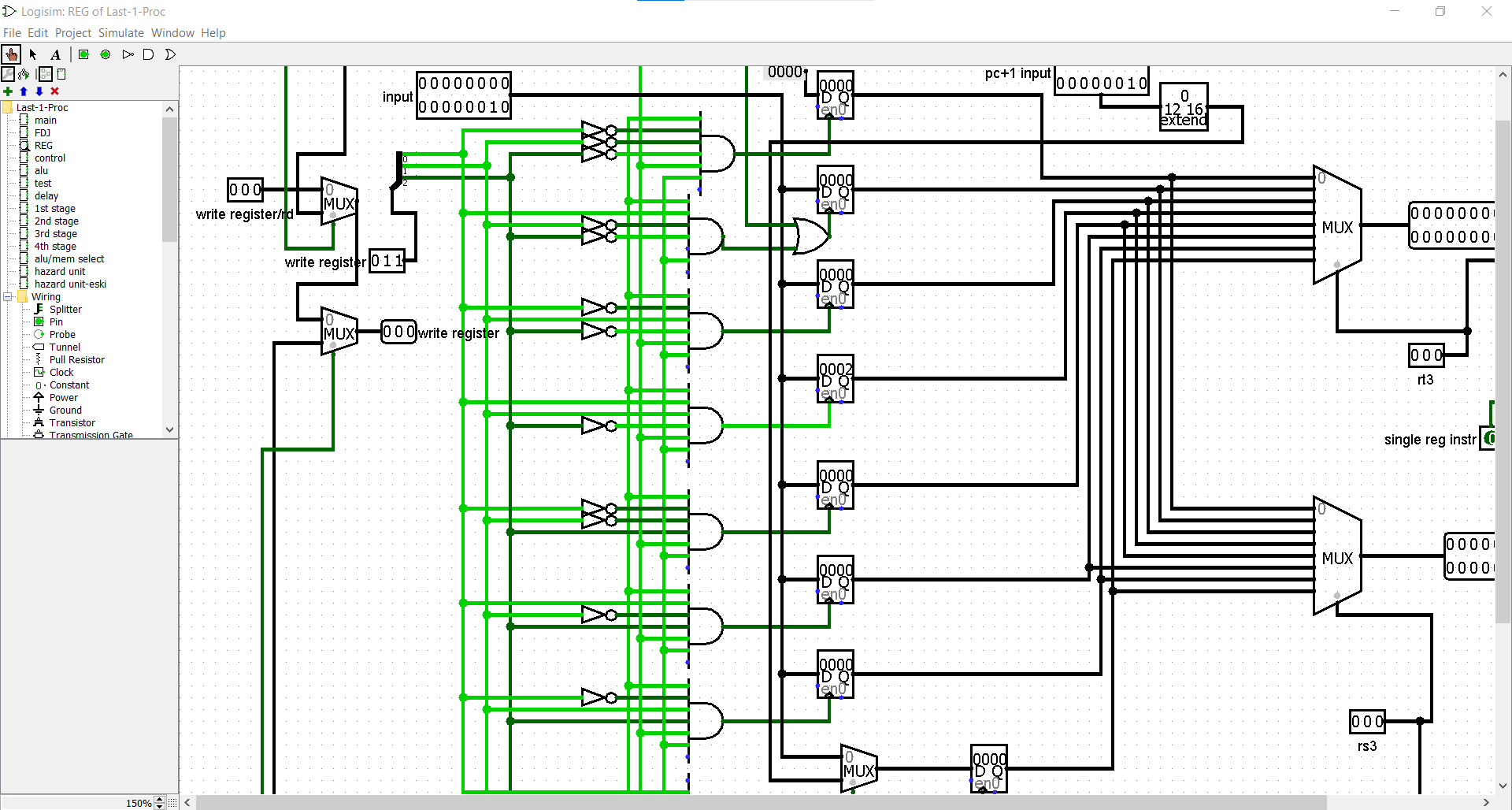
**Slt r1, r2, r3 Komutu**

****

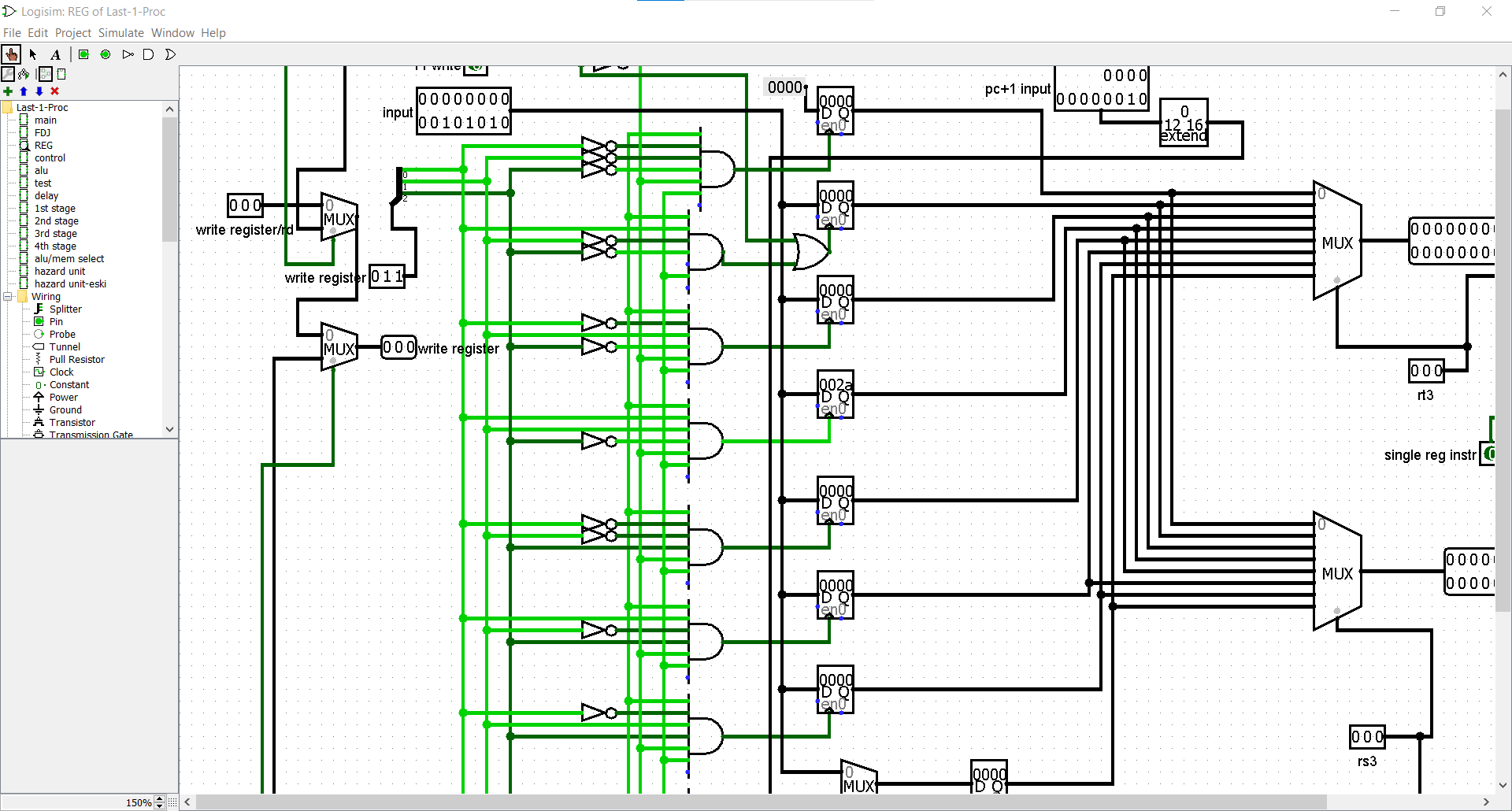
**Sltu r1, r2, r3 Komutu**

****

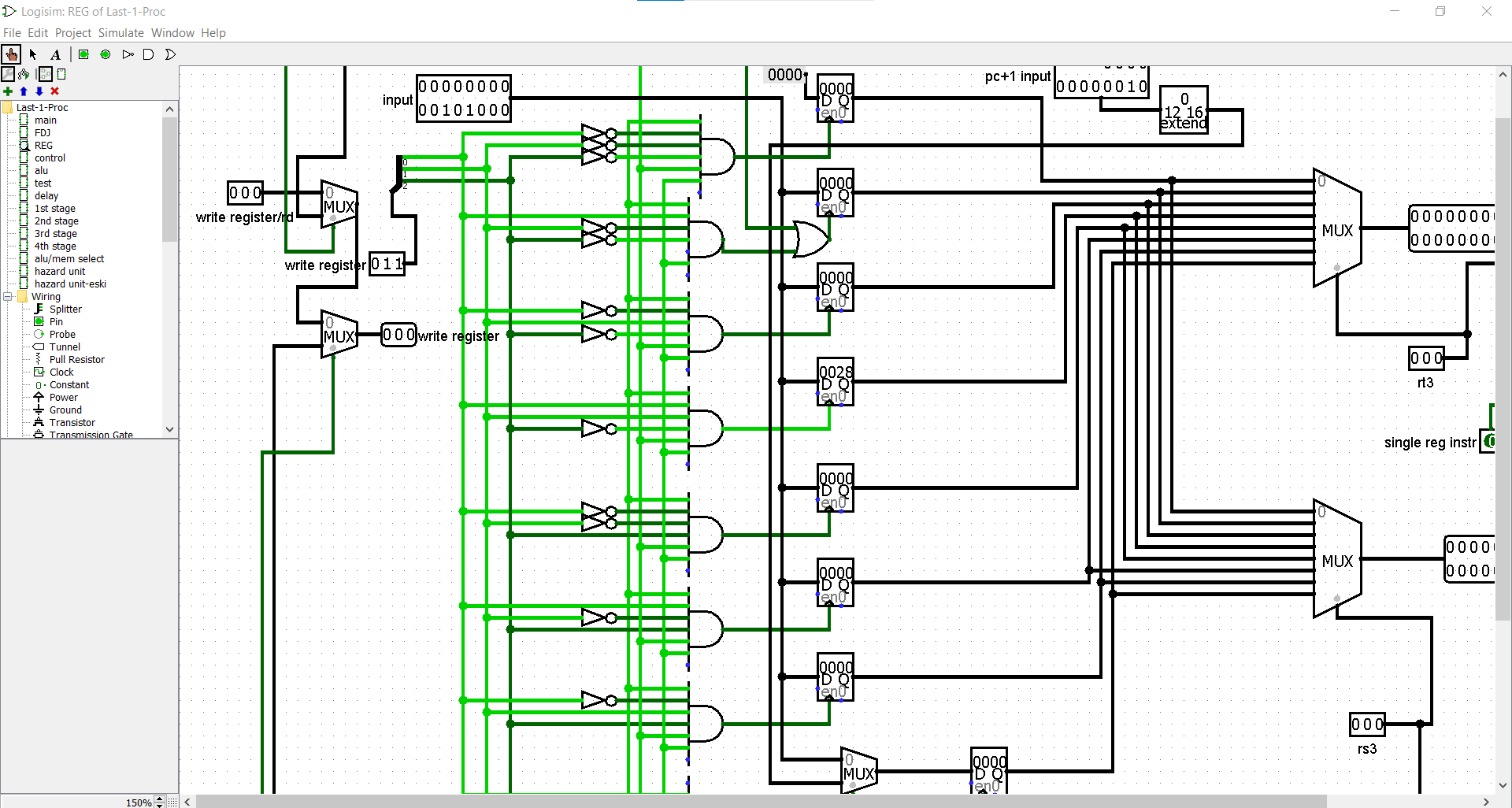
**Andi r3, r2, 12 Komutu**

****

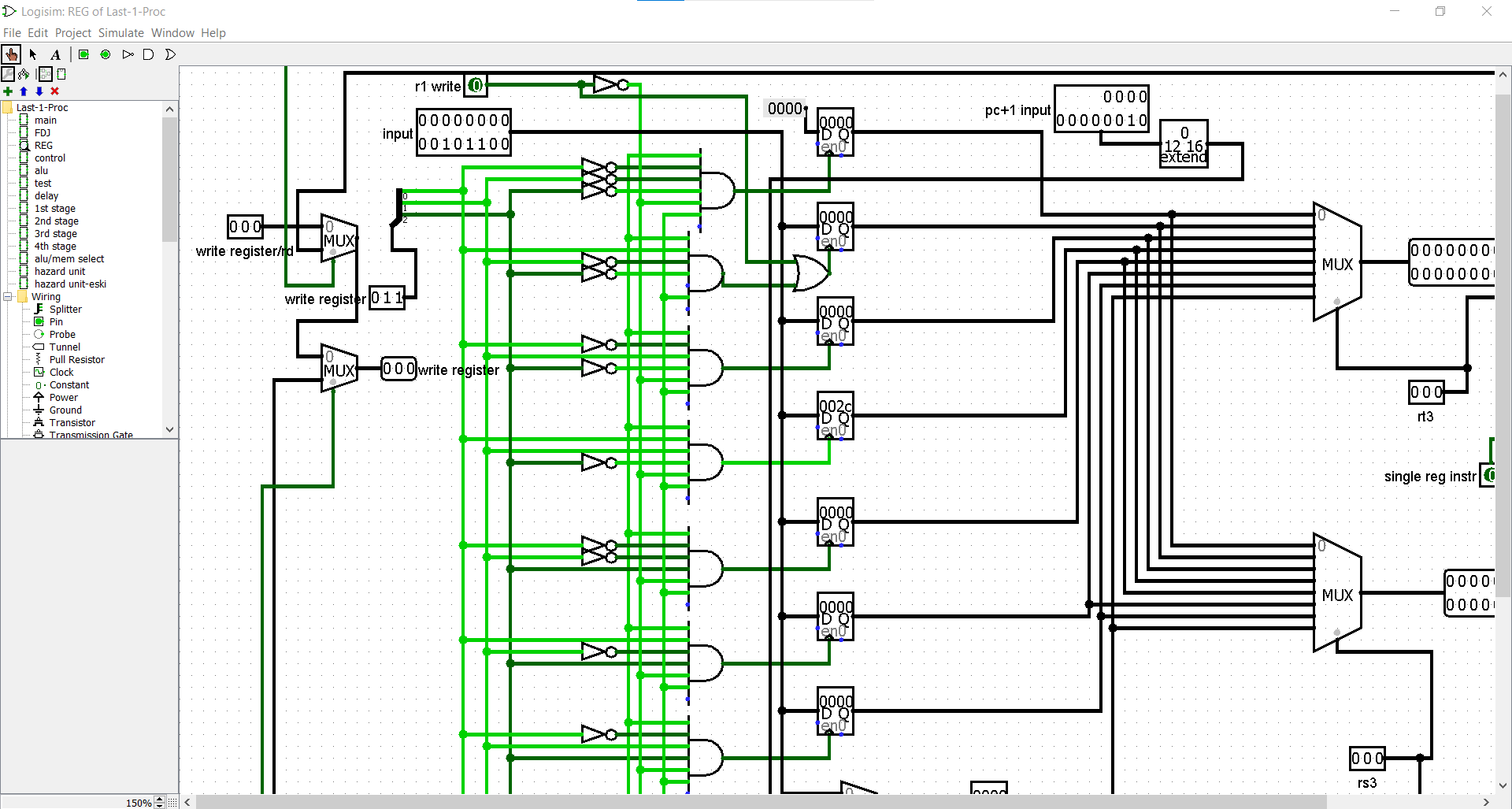
**Ori r3, r2, 12 Komutu**

****

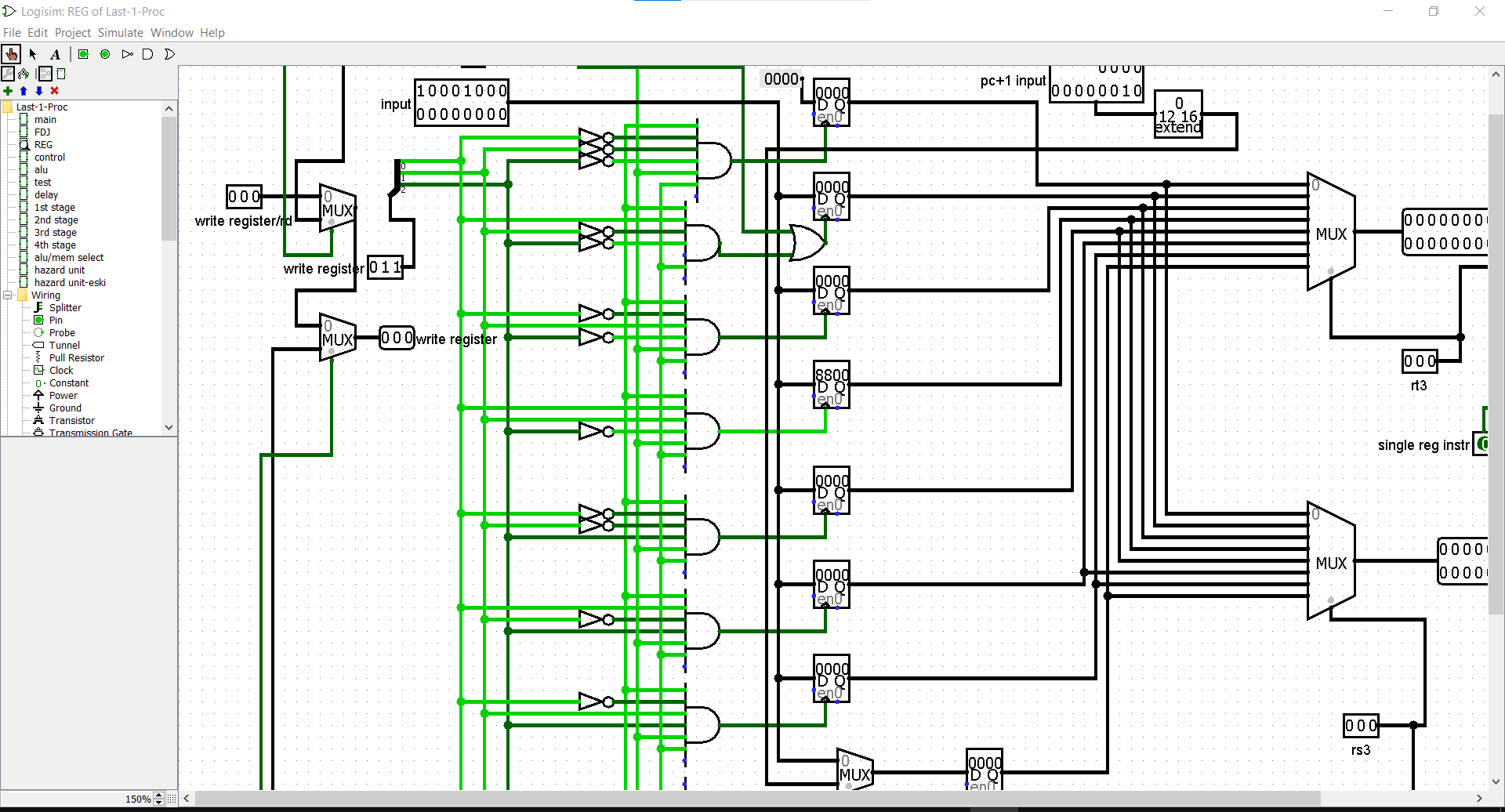
**Xori r3, r2, 12 Komutu**

****

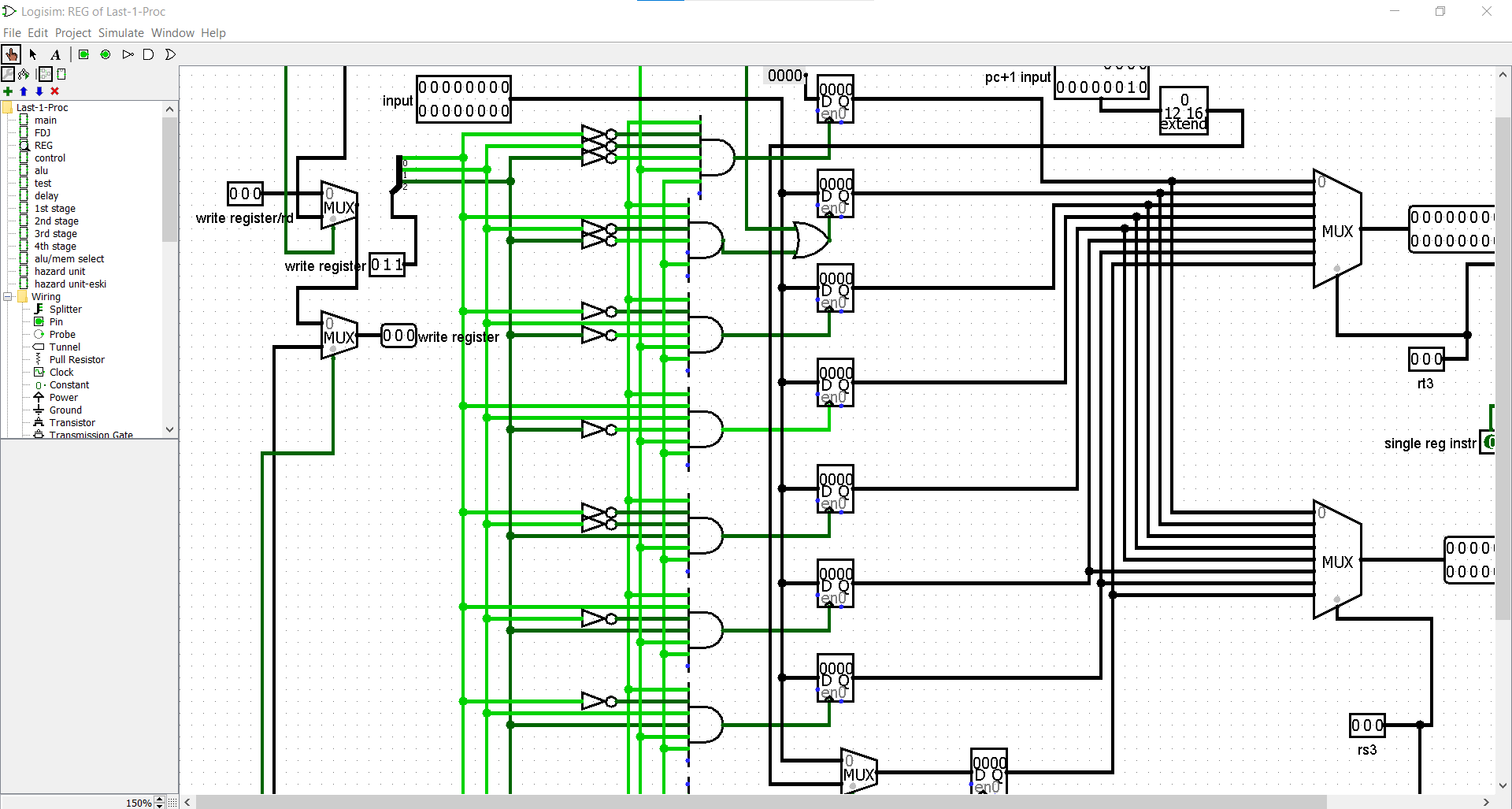
**Addi r3, r2, 12 Komutu**

****

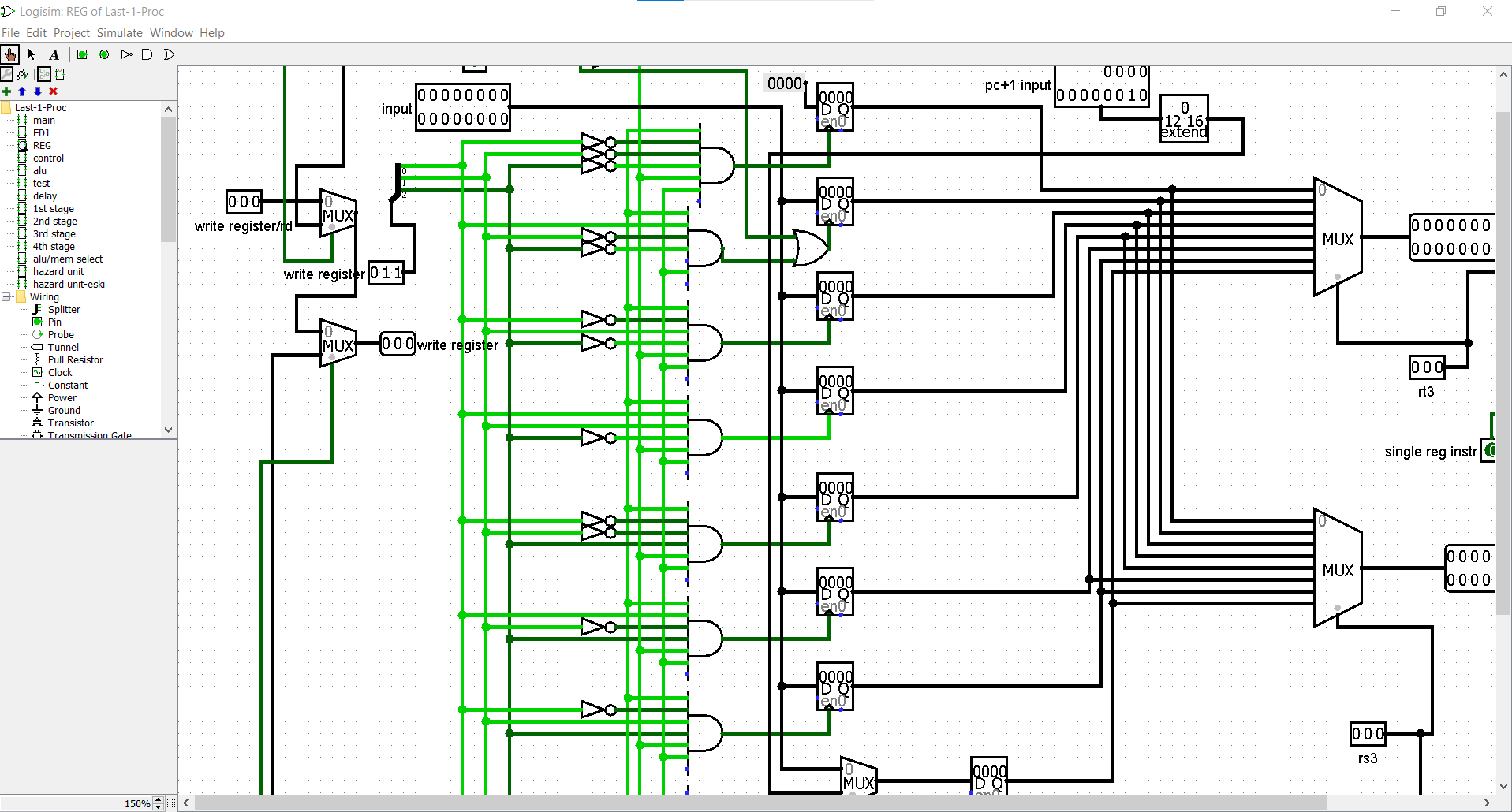
**Sll r3, r2, 12 Komutu**

****

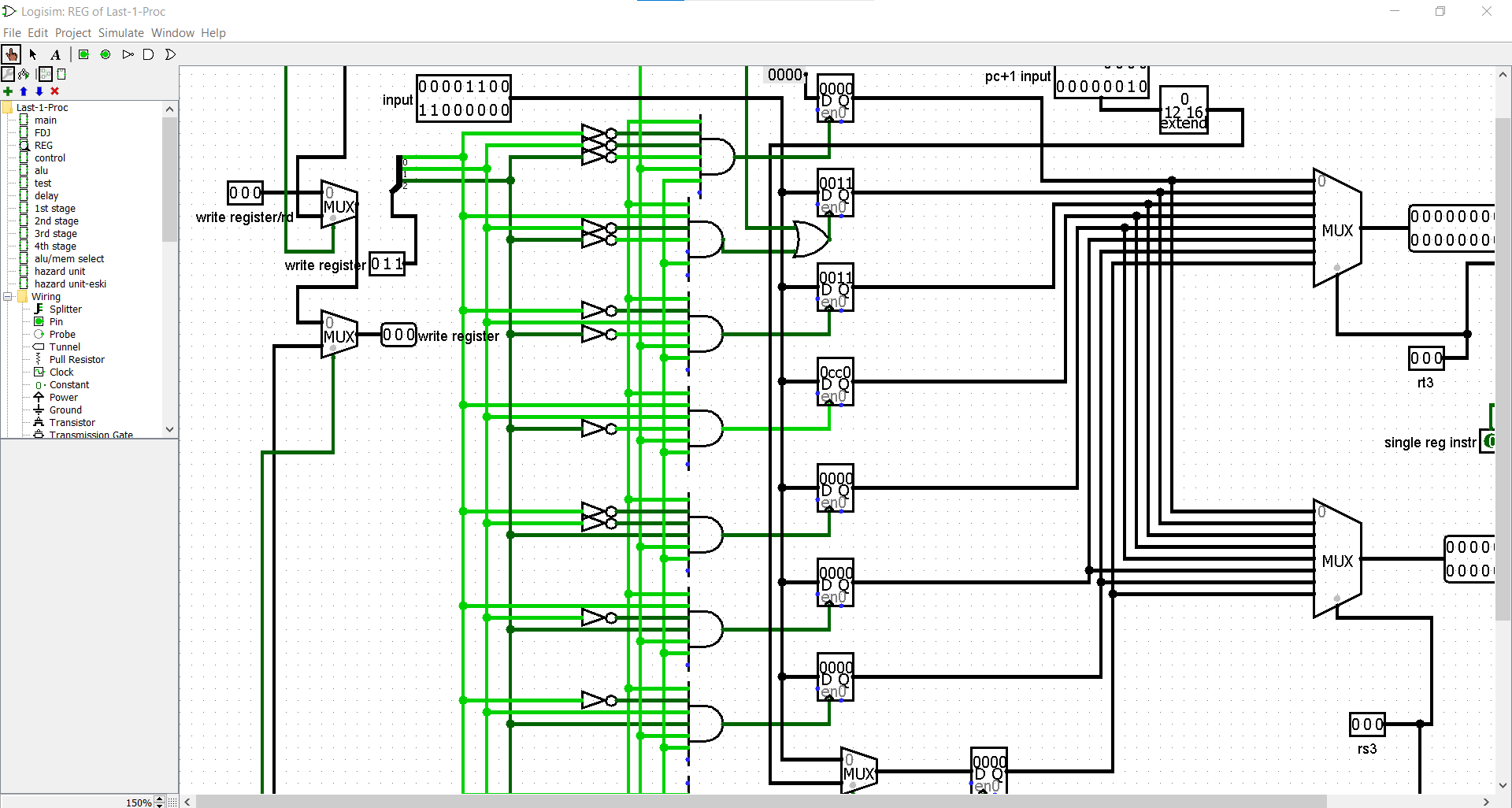
**Srl r3, r2, 12 Komutu**

****

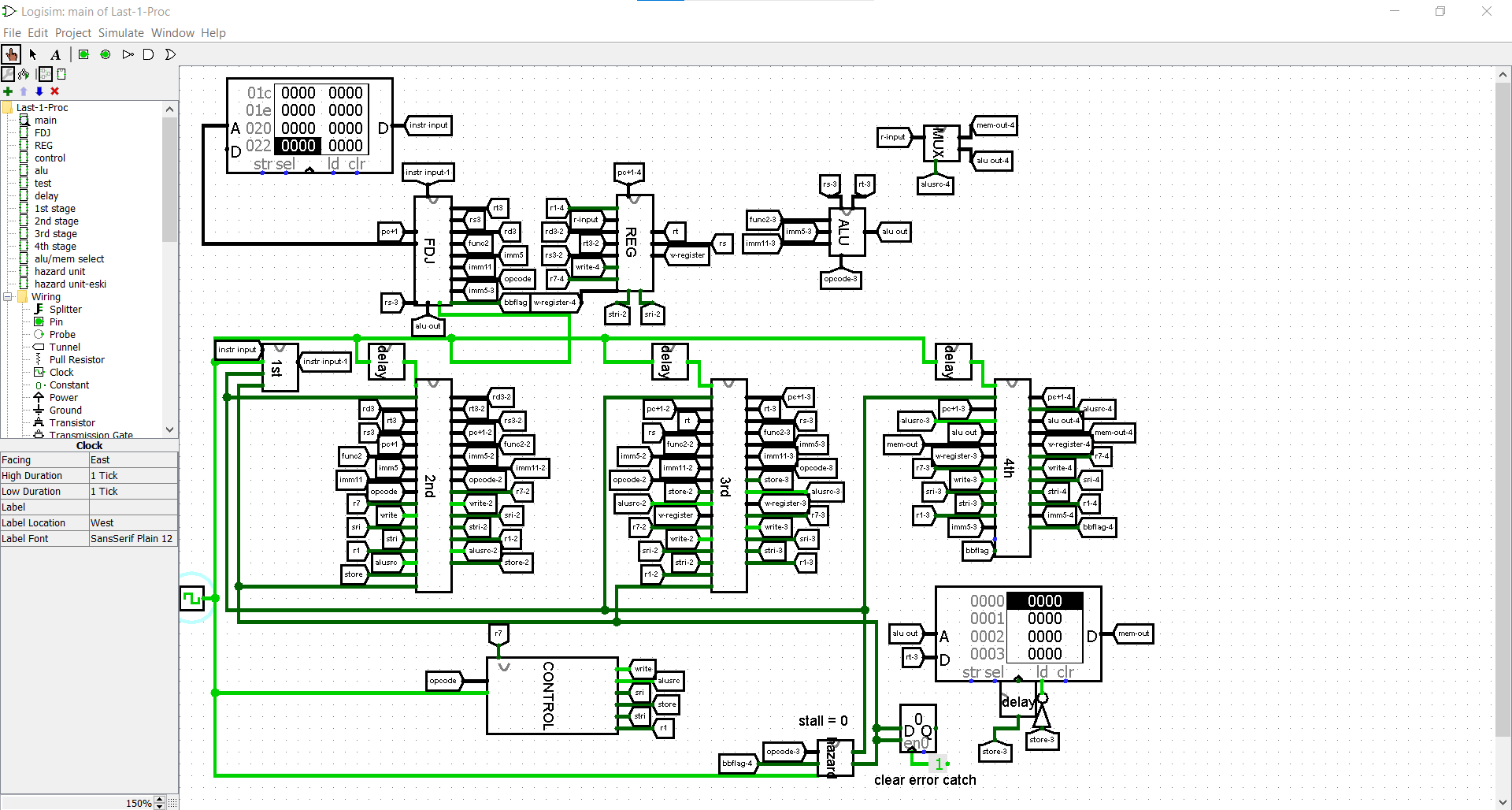
**Sra r3, r2, 12 Komutu**

****

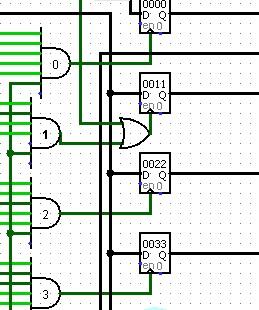
**Ror r3, r2, 12 Komutu**

****

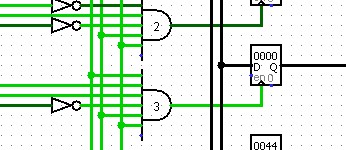
**Jr r2 Komutu**

****

**Lw r3, r2, 12 Komutu**

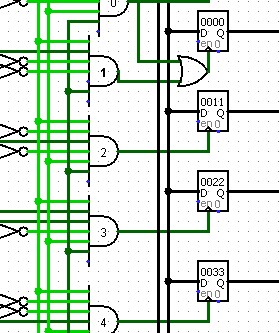
****

Register Önceki Durum

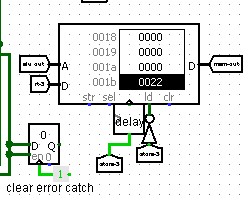


Son Durum Görüntüsü

**Sw r3, r2, 12 Komutu**

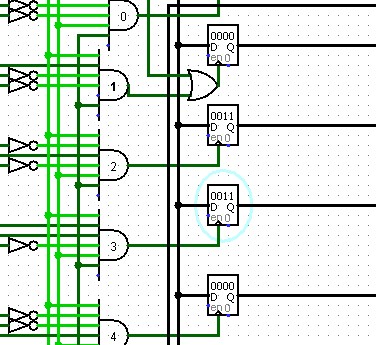
****

Durum Çıktısı

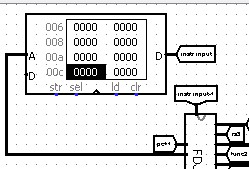
****

Son Durum Çıktısı

**Beq r3, r2, 12 Komutu**

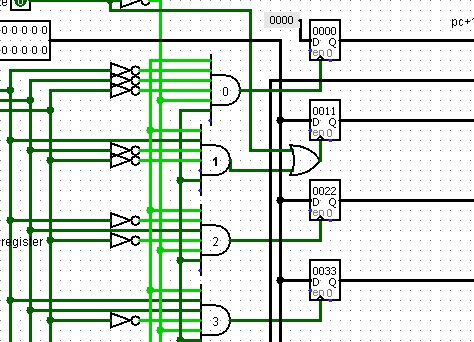
****

Durum Çıktısı

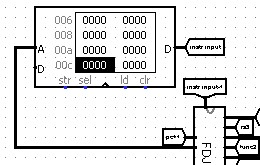
****

Durum Çıktısı

**Bne r3, r2, 12 Komutu**

****

Durum Çıktısı



Jump Yaptığına Dair Çıktı

**Lui 659 Komutu**

****

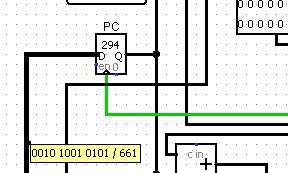
Register’ların Önceki Durumu

**metin, saat içeren bir resim

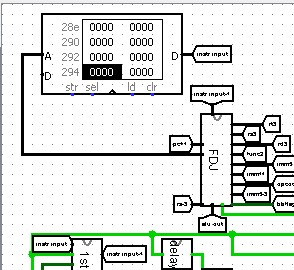
Açıklama otomatik olarak oluşturuldu**

Register’ların Simülasyondan Sonraki Durumu

**J 659 Komutu**

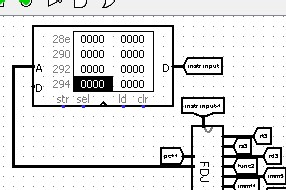
****

Program Counter’daki Çıktı

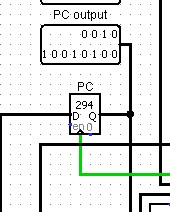


Jump Yaptığını Gösteren Çıktı

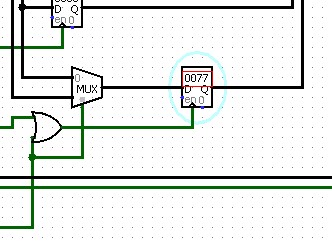
**Jal 659 Komutu**

****

Jump Yaptığını Gösteren Çıktı

****

Program Counter’daki Çıktı



Register’daki Çıktı