**BIL361**

**ÖDEV 1 RAPORU**

**Ödev Amacı:**

*RISC-V* mimarisindeki 15 buyruğa uygun, sentezlenebilir bir işlemci tasarlamak.

**Ödev Sahibi:**

*Ad-Soyad:* Hasan TUNA

1. **A Şıkkı**

İşlemci **clk**, **rst** ve **buyruk** olmak üzere 2 girişe ve **ps** adında 1 çıkışa sahiptir.

* **clk** girişi işlemcinin yapacağı işlemleri zamanlamak için kullanılmaktadır. Örneğin, **clk** yükselirken buyruğu getirmek ve alçalırken hesaplanan verileri yazmak gibi.
* **rst** girişi işlemciyi resetlemek için kullanılmaktadır. Reset işlemi sonucunda bellek ve yazmaçtaki tüm değerler sıfırlanır ve işlemci yeniden 0x0000\_0000 adresindeki buyruğu getirir.
* **buyruk** girişi işlemcinin işleyeceği buyruğu tutar.
* **ps** çıkışı, işlemcinin işleyeceği buyruğun adresini dışarıya verir. Bu sayede işlemci modülü, dışarıdaki bir buyruk belleğinden işlenecek buyrukları alabilmektedir.

Girişler ve çıkışlar dışında, modülün içinde tutulan bazı yazmaçlar ve kablolar vardır. Bunlardan en önemlileri **veri\_bellek** ve **yazmaç\_obegi**’dir.

* **veri\_bellek** dizisi 128 adet 32 bitlik yazmaç tutar. Belleğe yazma ve bellekten okuma işlemleri bu yazmaçlar üzerinde gerçekleşir. Dizideki her bir elemanın adresi, elemanın indisi çarpı 4 ile hesaplanır.
* **yazmaç\_obegi** dizisi 8 adet 32 bitlik yazmaç tutar. İşlemciye gelen yazmaçla ilgili işlemler bu dizi üzerinden yapılır. Dizideki her bir elemanın adresi, elemanın indisine eşittir.

**Buyrukların İşlenmesi**

İşlemci, **clk** girişinin yükselen kenarında sıradaki buyruğu getirir. Eğer geçerli bir buyruk geldiyse işlemi yapar ve sonraki buyruğun adresini hesaplar. Yazılacak değer varsa alçalan kenarda yazar.

1. **LUI**

****

Bu buyruk, gelen anlık değerin en üst bitler olarak kullanarak 32 bitlik bir değer oluşturur ve bu değeri gönderilen adresteki yazmaca yazar.

1. **AUIPC**

****

Bu buyruk, gelen anlık değerin en üst bitler olarak kullanarak 32 bitlik bir değer oluşturur ve bu değeri o anki ps ile toplayarak oluşan değeri gönderilen adresteki yazmaca yazar.

1. **JAL**



Bu buyruk, kendisinden sonraki buyruğun adresini ilgili yazmaca kaydeder ve ps değeri ile gelen anlık değeri toplayarak yeni ps değerini bulur.

1. **JALR**



Bu buyruk, kendisinden sonraki buyruğun adresini ilgili yazmaca kaydeder ve ilgili adresteki yazmacın değeri ile gelen anlık değeri toplayarak yeni ps değerini bulur.

1. **BEQ**



Bu buyruk, gönderilen adreslerdeki yazmaçların değerlerini karşılaştırır ve eğer bu değerler eşitse ps+anlık adresindeki buyruğu alır.

1. **BNE**



Bu buyruk, ilgili iki adresteki yazmacın değerini karşılaştırır ve eğer bu değerler eşit değilse ps+anlık adresindeki buyruğu alır.

1. **BLT**



Bu buyruk, rs1 adresindeki yazmacın değeri rs2’deki değerden küçükse ps+anlık adresindeki buyruğu alır.

1. **LW**



Bu buyruk, ilgili yazmaçtaki değer ile anlık değeri toplayarak bellek adresi hesaplar ve bu adresteki veriyi ilgili yazmaca yazar.

1. **SW**



Bu buyruk, ilgili yazmaçtaki değer ile anlık değeri toplayarak bellek adresi hesaplar ve ilgili yazmacın değerini bellekteki bu adrese yazar.

1. **ADDI**

****

Bu buyruk, anlık değer ile ilgili yazmaçtaki değeri toplayıp ilgili yazmaca yazar.

1. **ADD**



Bu buyruk, ilgili yazmaçlardaki değerleri toplayıp ilgili yazmaca yazar.

1. **SUB**

****

Bu buyruk, ilgili yazmaçlardaki değerleri çıkarıp ilgili yazmaca yazar.

1. **OR**



Bu buyruk, ilgili yazmaçlardaki değerleri *bitwise-or* işleminden geçirip sonucu ilgili yazmaca yazar.

1. **AND**



Bu buyruk, ilgili yazmaçlardaki değerleri *bitwise-and* işleminden geçirip sonucu ilgili yazmaca yazar.

1. **XOR**



Bu buyruk, ilgili yazmaçlardaki değerleri *bitwise-xor* işleminden geçirip sonucu ilgili yazmaca yazar.

1. **B Şıkkı**

Bu kısımda A şıkkına ek olarak, **veri\_bellegi** adlı yazmaç dizisini işlemci modülünün dışına çıkardık ve **buyruk\_bellegi** adında başka bir modülden buyrukları aldık. Bu 3 modülün birlikte çalışabilmesi için bir de **çevreleyici** modülü yazdık.

* **buyruk\_bellegi:** işlemcinin işleyeceği buyrukları tutan birim. **ps** girişinden gelen adresteki buyruğu dışarıya veriyor.
* **veri\_bellegi:** işlemcinin load ve store işlemleriyle erişeceği bellek modülü. Yazma işlemleri **clk** alçalan kenarında yapılırken okuma işlemleri her daim gerçekleşebilmektedir.
* **cevreleyici: islemcib, buyruk\_bellegi** ve **veri\_bellegi** modüllerinin giriş ve çıkışlarını birbirine bağlayarak birlikte kullanılmalarını sağlamak için tanımlanan bir ana modül.

Buyrukların işlenmesi A şıkkından çok farklı olmadığı için bu kısmı geçiyorum.

A’dan tek farkı **cevreleyici** sayesinde bu 3 modülün ayrı dosyalarda bulunması.

1. **C Şıkkı**

Bu aşamada ödev metninde istenilen işlemleri yapan bir assembly programı yazılmıştır.

Program, öncelikle x1, x2, x3, x4, x5 yazmaçlarına sırasıyla 2, 4, 6, 8, 10 değerlerini atamaktadır. Sonrasında bu yazmaçlarda tutulan değerleri veri belleğinin 0, 4, 8, 12 ve 16. adreslerine atamıştır.

Benzer şekilde x1, x2, x3, x4, x5 yazmaçlarına 20, 30, 40, 50, 60 değerleri yazılmıştır. Bu değerler de veri belleğinin 20, 24, 28, 32 ve 36. adreslerine atanmıştır.

Son olarak bu değerler veri belleğinden getirilip toplanmış ve en sonda toplanan değerler birbirinden çıkarılmıştır. Sonucu veri belleğinin son adresine yazarak program sonlanmıştır.

*İlgili program için assembly buyrukları ve buyrukların onaltılık sistem karşılıkları aşağıdaki sayfadadır.*





Program A ve B şıkkındaki modüllerde sorunsuz çalışmıştır.

Aşağıda A şıkkındaki çalışması için yazmaç-bellek değerlerini ve waveform’u görebilirsiniz.

Graphical user interface, application

Description automatically generated