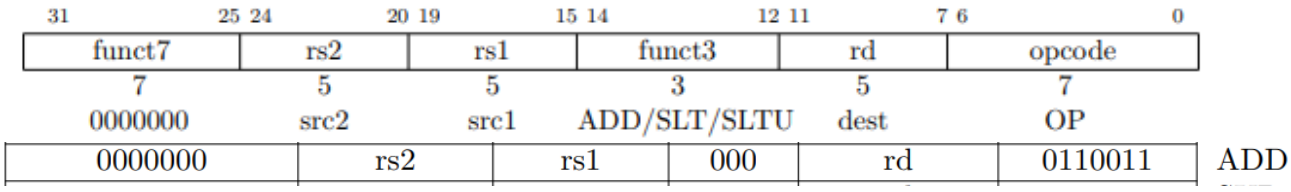
Mustafa Aydoğan 191101002 BİL361 Ödev-1 Raporu

1. Evrişim İşleminde Kullanılacak Buyruklar

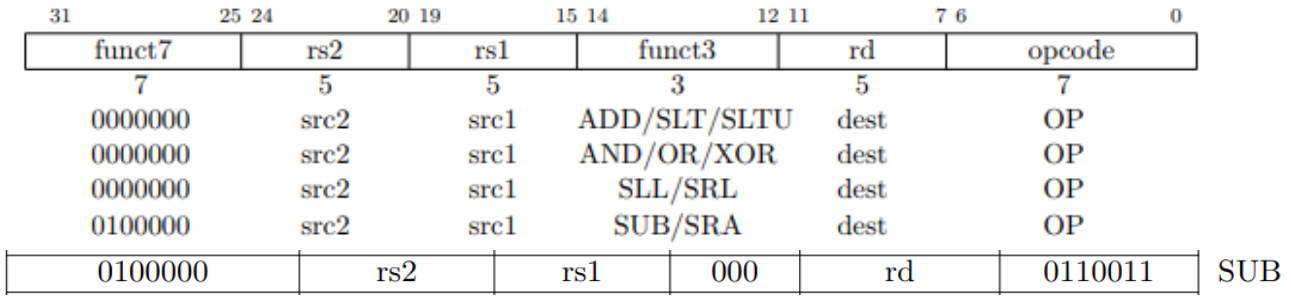
a. **ADD** add rd, rs1, rs2
b. **SUB** sub rd, rs1, rs2
c. **ADDI** addi rd, rs1, #imm
d. **BEQ** beq rs1, rs2, #imm
e. **SLL** sll rd, rs1, rs2
f. **LW** lw rd, #imm(rs1)
g. **SW** sw rs2, #imm(rs1)
h. **JALR** jalr rd, rs1, #imm

2. Buyrukların Türleri ve İçerikleri

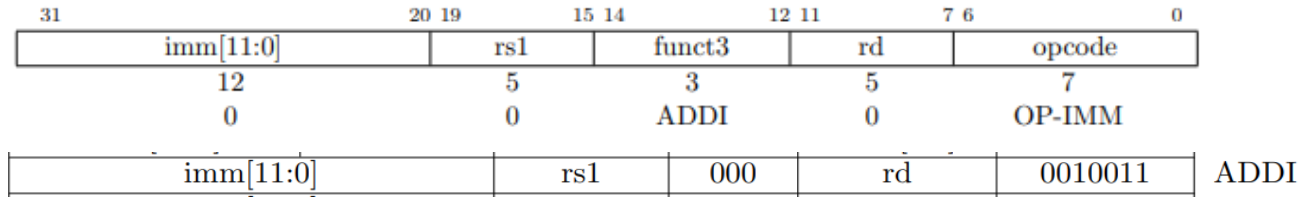
- **ADD:**



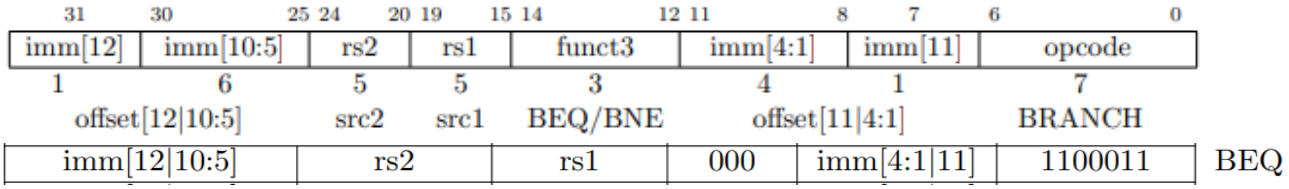
- **SUB:**



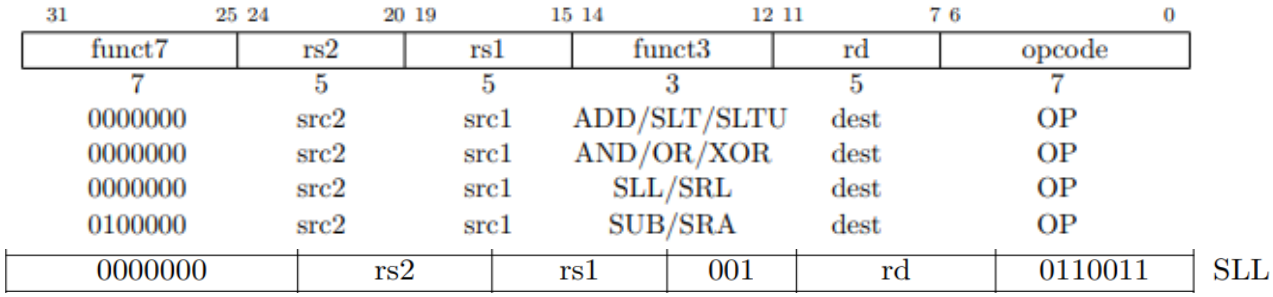
- **ADDI:**



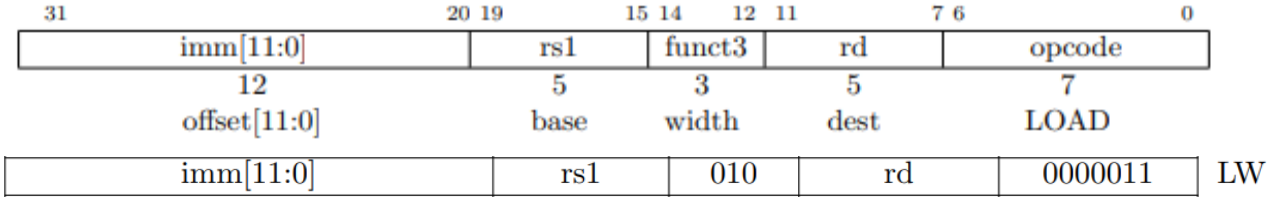
- **BEQ:**



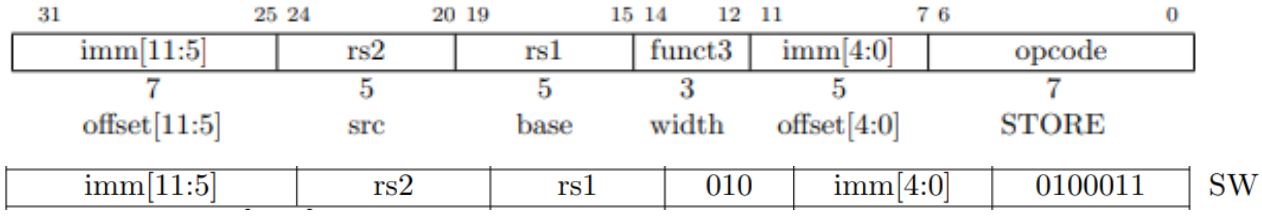
- **SLL:**



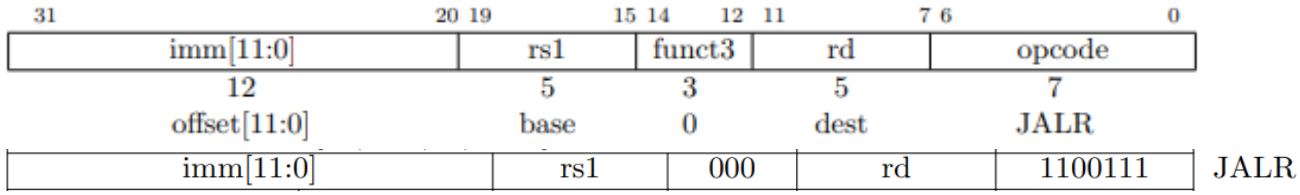
- **LW:**



- **SW:**



- **JALR:**



3. Evrişim İşleminde Kullanılan Buyrukların Genel Amaçları:

1. **ADD:** ADD buyruğu R-tipi bir buyruktur. İşlem kodu 0110011'dir. Funct3 değeri 000 iken funct7 değeri 0000000'dır. Rs1 yazmacındaki değeri ve rs2 yazmacındaki değeri okuyup toplar. Sonucu rd yazmacına yazar.
2. **SUB**: SUB buyruğu R- tipi bir buyruktur. İşlem kodu ADD buyruğunun işlem kodu (0110011) ile aynıdır. Funct3 değeri 000 iken toplamadan farklı olarak funct7 değeri 0100000'dir. Rs1 yazmacındaki değerden rs2 yazmacındaki değeri çıkarır. Sonucu rd yazmacına yazar.
3. ADDI: ADDI buyruğu I- tipi bir buyruktur. İşlem kodu 0010011'dir. Funct3 değeri 000'dır. 12 bitlik anlık değeri genişletir ve rs1 yazmacındaki değer ile toplar. Sonucu rd yazmacına yazar.
4. **BEQ:** BEQ buyruğu B-tipi bir buyruktur. İşlem kodu 1100011'dir. Funct3 değeri 000'dır. Rs1 yazmacındaki değer ile rs2 yazmacındaki değeri karşılaştırır. Eşit olması durumunda program sayacını (PS) anlık değer kadar artırır.
5. **SLL**: SLL buyruğu R-tipi bir buyruktur. İşlem kodu 0110011'dir. Funct3 değeri 001 iken funct7 değeri 0000000'dır. Rs1 yazmacındaki değeri rs2 yazmacındaki değer kadar mantıksal olarak sola kaydırır. Sonucu rd yazmacına yazar.
6. **LW**: LW buyruğu I-tipi bir buyruktur. İşlem kodu 0000011'dir. Funct3 değeri 010'dır. Rs2 yazmacındaki değer ile anlık değeri toplayarak bellek adresi elde eder. Veri belleğine giderek elde edilen adresteki veriyi rd yazmacına yazar.
7. **SW**: SW buyruğu S-tipi bir buyruktur. İşlem kodu 0100011'dir. Funct3 değeri 010'dır. Rs1 yazmacındaki değer ile anlık değeri toplayarak bellek adresi elde eder. Veri belleğine giderek elde edilen adrese rs2 yazmacındaki veriyi yazar.
8. **JALR**: JALR buyruğu I-tipi bir buyruktur. İşlem kodu 1100111'dir. Funct3 değeri 000'dır. Program sayacının değerini rd yazmacına yazar. Rs1 yazmacındaki değer ile anlık değeri toplayarak program sayacını bu değer ile günceller. Buyruk belleğine giderek elde edilen adresteki veriyi rd yazmacına yazar.

4. Buyrukların Evrişim İşleminde Kullanılma Amaçları:

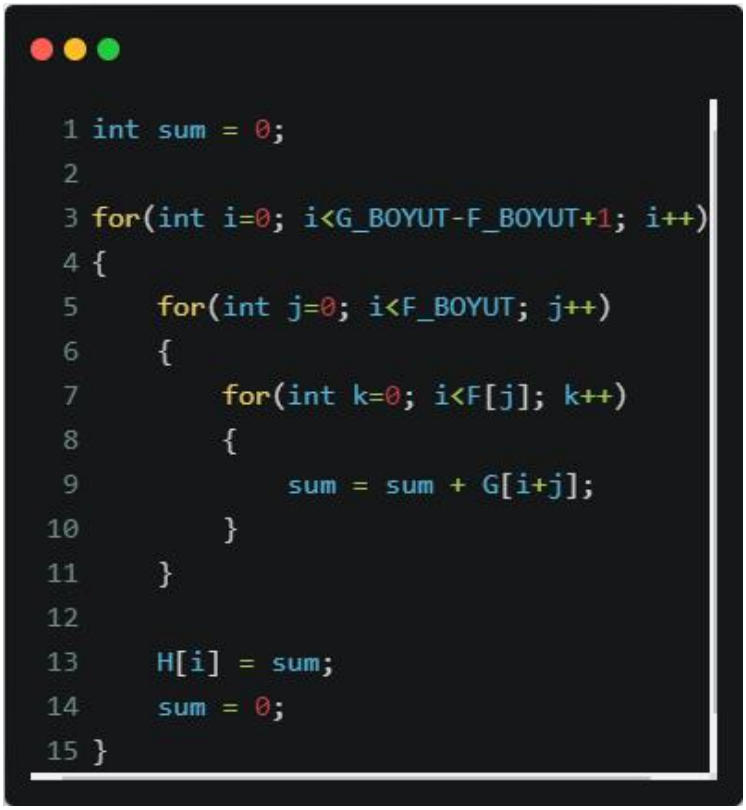
1. **ADD** buyruğu evrişim işleminde yazmaçlardaki değerleri toplamak için kullanılmıştır. Ayrıca yazmaçlar sıfırlanmak istendiğinde **ADD** buyruğu kullanılmıştır. Sıfırıncı yazmacın kendi ile toplanması ve hedef yazmaca yazılması esnasında kullanılmıştır. Bir başka kullanım amacı da çarpım işlemidir. İki sayı çarpılmak istendiğinde **BEQ** buyruğu ve **JALR** buyrukları kullanılarak bir döngü elde edilmiştir. Bu döngü esnasında **ADD** buyruğu kullanılarak çarpma işlemi gerçekleştirilmiştir.
2. **SUB** buyruğu evrişim işleminde en az kullanılan buyruklardan biridir. Yazmaçların farkı alınıp başka bir yazmaca yazılması istendiğinde kullanılmıştır.
3. **ADDI** buyruğu evrişim işleminde anlık değerleri yazmaçlara ve belleğe aktarmak için kullanılmıştır.
4. **BEQ** buyruğu evrişim işleminde gerekli olan döngüyü elde etme esnasında kullanılmıştır. **BEQ** buyruğu ve **JALR** buyruğu program sayacını günceller ve kontrol eder. Bu sayede bir dallanmayı kontrol eder ve döngü elde edilir.
5. **SLL** buyruğu evrişim işleminde az sayıda kullanılan bir buyruktur. Kullanılma amacı veri belleği ile işlem yapılması için gerekli olan bellek adresinin elde edilmesidir. Veri belleğinden yükleme yapmak için bellek adresine ihtiyacımız vardır. Bellek adresleri 4'ün katı şeklinde olduğu için değişkenlerimizi **SLL** buyruğu sayesinde 4'ün katı haline getirebiliriz.
6. **LW** buyruğu veri belleğinden değer okuyup belirtilen yazmaca yazma esnasında kullanılmıştır.
7. **SW** buyruğu veri belleğinde değer saklamak amacıyla kullanılmıştır. Belirtilen bellek adresine giderek verileri belleğe yazar.
8. **JALR** buyruğu evrişim işleminde gerekli olan döngüyü sağlamak için kullanılmıştır. Aynı zamanda çarpma işlemi yapılacağı zaman gerekli olan döngü bu buyruk sayesinde gerçekleşir.

5. Ödev Kapsamında Assembly Dilinde Yazılan Programın Açıklaması:

```
 0| addi x1,  x0,  1
 4| addi x8,  x0,  #G_BOYUT
 8| addi x9,  x0,  #F_BOYUT
12| addi x11, x0,  #G_BASLANGIC
16| addi x12, x0,  #F_BASLANGIC
20| addi x13, x0,  #H_BASLANGIC
24| addi x18, x0,  2
28| sub  x10, x8,  x9
32| add  x10, x10, x1
36| beq  x6,  x10, 92
40| beq  x7,  x9,  60
44| add  x14, x6,  x7
48| sll  x14, x14, x18
52| add  x14, x14, x11
56| sll  x15, x7,  x18
60| add  x15, x12, x15
64| lw   x3,  0(x14)
68| lw   x4,  0(x15)
72| beq  x17, x4,  16
76| add  x2,  x2,  x3
80| add  x17, x17, x1
84| jalr x19, x0,  72
88| add  x17, x0,  x0
92| add  x7,  x7,  x1
96| jalr x19, x0,  40
100| sll  x16, x6,  x18
104| add  x16, x16, x13
108| sw   x2,  0(x16)
112| add  x2,  x0,  x0
116| add  x6,  x6,  x1
120| add  x7,  x0,  x0
124| jalr x19, x0,  36
```

- x1 yazmacına daha sonra kullanmak için 1 değeri atandı.
- x8, x9, x11, x12 ve x13 yazmaçlarına sırası ile anlık değerler olan #G_BOYUT, #F_BOYUT, #G_BASLANGIC, #F_BASLANGIC ve #H_BASLANGIC değerleri atandı.
- x18 yazmacına daha sonra kullanmak üzere 2 değeri anlık olarak atandı.
- x8 yazmacındaki değerden x9 yazmacındaki değer çıkarıldı ve x10 yazmacına yazıldı.
- x10 yazmacındaki değer 1 artırıldı.
- Döngü oluşturmak için dallanma buyruğu kullanıldı. x6 ile x10 yazmacındaki değerler karşılaştırıldı. Eğer eşitse program sayacını 92 artırdı. Eşit değilse 4 artırarak sonraki buyruğa geçti.
- İkinci bir döngü oluşturmak için BEQ buyruğu kullanıldı. x7 ve x9 değerleri karşılaştırıldı. Eşit olması durumunda program sayacı 60 artırıldı.
- SLL buyruğu kullanılarak x17 yazmacındaki değer 2 bit sola kaydırıldı. Böylece sayı 4 ile çarpılmış oldu.
- ADD komutları kullanılarak bellekten veri okumak ve yazmak için gerekli adresler elde edildi. Adresler x14 ve x15 yazmaçlarında saklanmaktadır.
- LW buyruğu ile bellekten okuma yapıldı.
- Bir döngüye daha ihtiyacımız olduğu için yine dallanma buyruğu kullanıldı.
- Bu döngüde evrişim işleminin en temel işlemi olan çarpma işlemi sağlanmış oldu.
- x17 değeri bir artırıldı.
- JALR buyruğu kullanılarak program sayacındaki değer x19 yazmacında yazıldı ve PC 72 olarak güncellendi.
- Döngü bittikten sonra x17 yazmacı sıfırlandı. x7 yazmacındaki değer 1 artırıldı.
- JALR buyruğu kullanılarak program sayacındaki değer x19 yazmacında yazıldı ve PC 40 olarak güncellendi.
- SLL buyruğu ve ADD buyruğu kullanılarak adres değeri hesaplandı.
- x2 yazmacındaki hesaplanmış değer, SW buyruğu ile x16 yazmacındaki adrese yazıldı.
- x2 ve x7 numaralı yazmaçlar sıfırlandı.
- x6 yazmacındaki değer 1 artırıldı.
- Döngü oluşması için JALR buyruğu kullanıldı ve PC 36 olarak güncellendi.

6. Programın Java Dilinde Yazılmış Hali:



```java
int sum = 0;

for(int i=0; i<G_BOYUT-F_BOYUT+1; i++)
{
    for(int j=0; i<F_BOYUT; j++)
    {
        for(int k=0; i<F[j]; k++)
        {
            sum = sum + G[i+j];
        }
    }

    H[i] = sum;
    sum = 0;
}
```

7. Değişkenlerin Saklandığı Yazmaçlar:

x0 --> 0

x1 --> 1

x2 --> sum

x3 --> G[i+j]

x4 --> F[j]

x5 --> H[i]

x6 --> i

x7 --> j

x8 --> #G_BOYUT

x9 --> #F_ BOYUT

x10 --> #G_ BOYUT - #F_ BOYUT + 1

x11 --> #G_BASLANGIC

x12 --> #F_BASLANGIC

x13 --> #H_BASLANGIC

x14 --> G[i+j] bellek adresi

x15 --> F[j] bellek adresi

x16 --> H[i] bellek adresi

x17 --> k

x18 --> 2

x19 --> JALR buyruğu sonrasında Program Counter

x20 --> 0

x21 --> 0

x22 --> 0

x23 --> 0

x24 --> 0

x25 --> 0

x26 --> 0

x27 --> 0

x28 --> 0

x29 --> 0

x30 --> 0

x31 --> 0