**SCI (Superscalar Architecture) mimarisi nedir?**

İşlemci performansını artırmak için birden çok talimatı aynı anda işleyebilen bir mikroişlemci mimarisidir.

metin, diyagram, plan, teknik çizim içeren bir resim

Açıklama otomatik olarak oluşturuldu

**ALGORİTMA**

1. Başlangıç adresi, konumu, program uzunluğu 0 ile başlatıldı.
2. Program ismi için boş satır oluşturuldu.
3. Kaynak dosyası ("optab.txt") okuma modunda açıldı.

3.1. Ara dosya ("intermediate.txt") yazma modunda açıldı.

3.1.1.Sembol tablosu dosyası ("symtab.txt") yazma modunda açıldı.

3.1.1.1. Dosya konumu için başlangıç değeri belirlendi.

3.1.1.2. Dosya işlemleri sürene kadar bir döngü kur

3.1.1.2.1. Dosyadan bir satır okunarak ve dosya konumu güncellenerek "line" ve "pos" değişkenlerine atandı

3.1.1.2.2.Okunan satır boş ise **Git 4**

3.1.1.2.3. Etiket, opcode, operand ve geri kalan kısımlar ayrı değişkenlere atandı.

3.1.1.2.4. Eğer opcode “START” ise:

3.1.1.2.4.1. Başlangıç adresi ve oprand güncellen

3.1.1.2.4. 2.Ara dosyaya başlangıç verileri yazıldı.

3.1.1.2.5. Eğer opcode "END" ise: **Git 4**

3.1.1.2.6.Eğer etiket varsa ve etiket nokta ile başlamıyorsa

3.1.1.2.6.1. Eğer sembol tablosunda bulunuyor ise ,ata mesajı yaz

3.1.1.2.6.2. Etiket ve locctr değeri satır bilgileri yazıldı.

3.1.1.2.7. Ara dosyaya locctr değeri ve satır bilgileri yazıldı.

3.1.1.2.8. Eğer opcode optab dosyasında bulunuyorsa Konum 3 ileri alındı locctr 3 arttır.

3.1.1.2.9. Eğer opcode "WORD" ise: Konum 3 ileri alındı locctr 3 arttır.

3.1.1.2.10. Eğer opcode "RESW" ise: Locctr değeri, operand ile çarpılıp üç ile arttır.

3.1.1.2.11. Eğer opcode "RESB" ise: Locctr değeri, operand ile değiştir.

3.1.1.12.12 Eğer opcode " BYTE " ise:

3.1.1.12.12 1. Oprandın ilk karakteri "C" ise: Locctr değeri, operandin uzunluğundan üç çıkar.

3.1.1.12.12.2. Oprandın ilk karakteri "X" ise: Locctr değeri, operandin uzunluğundan üç çıkar ve ikiye böl.

3.1.1.12.12 3. Hata mesajı yazdır.

4.Dur

**hexadecimal\_to\_decimal(x):**

1.Girdi sıfır olana kadar döngü kur

1.1. Girdinin 10'a bölümünden kalan al.

1.2. Onaltılık tabandaki sayıya çevrildi ve sonuca eklendi

1.3.Girdi ondalık tabana çevir

1.4.Sayaç arttırıldı.

**read\_line\_from\_file(src, pos):**

1.Satırı saklayacak değişken oluşturuldu.

2.Dosya belirli konuma girildi.

3.Dosyadan bir satır okundu.

4.Dosyanın son konumu saklandı.

5.Okunan satır ve son konum döndürüldü.

**break\_line(line):**

1.Satır, sekme karakterine parçalandı

2.Etiket,opcode,operand ve geri kalan kısımlar için boş dizeler oluşturuldu.

3.Eğer satırın en az bir öğesi varsa:

3.1lk öğe, etiket olarak belirlendi ve sonundaki boşluklar temizlendi.

4.Eğer satırın en az iki öğesi varsa:

4.1İkinci öğe, opcode olarak kabul edilir ve diğer boşluklar temizlenir.

5.Eğe satırın en az üç öğesi varsa:

4.2.Üçüncü öğe, operand olarak kabul edilir ve baş ve sondaki boşluklar temizlenir.

6.Eğer satırın en az dört öğesi varsa

6.1.Dördüncüğ öğe ve sonrası silinir.

**search\_symtab(label):**

**1.**Etikeitn bulunup bulunmadığını belirten bayrak oluşturuldu.

2.symtab.txt dosyası okuma modunda açıldı

3.Dosyanın her satırı için döngü oluşturuldu.

3.1.Satırdaki sembol ve değer, boşluk karakterine göre ayrıldı.

3.2.Eğer satırdaki sembol, aranan etiketle eşleşiyorsa

3.2.1.Bulundu bayrağı olarak işaretlendi.

**search\_optab(opcode):**

**1.**verilen bir opcode'un optab.txt dosyasında bulunup bulunmadığını kontrol etmektir.

2.Opcode var mı bayrağı oluşturuldu.

3. "optab.txt" dosyası okuma modunda açıldı ve "symtab" adı altında kullanıldı

3.1.Dosyanın her satırı için döngü oluşturuldu.

3.1.1Satırdaki sembol ve değer, boşluk karakterine göre ayrıldı.

3.1.2.Eğer satırdaki sembol, aranan opcode ile eşleşiyorsa

3.1.3.Bulundu

**Proje Linki:**

https://github.com/hafizeogut/SCI\_Pass1\_Python

**Çalışma videosu:**



**Proje ne işe yarar?**

Bu proje, assembly dilinde yazılmış kodları analiz ederek ara dosyalar oluşturan bir araç sunar. Bu ara dosyalarda, her bir komutun yerini, sembol tablosunu ve programın uzunluğunu belirten bilgiler bulunur. Bu şekilde, assembly kodlarının derlenmesi, çevrilmesi veya analiz edilmesi gibi işlemler kolaylıkla yapılabilir.

metin, ekran görüntüsü, diyagram, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**Proje içerği:**

**Proje neden faydalıdır?**

Proje, assembly dilinde yazılmış kodların analizini sağlayarak yazılım geliştiricilere ve sistem tasarımcılarına büyük bir kolaylık sunar. Assembly kodlarının anlaşılması ve işlenmesi karmaşık olabilir, bu araç sayesinde kodun anlaşılması ve yönetilmesi daha basit hale gelir. Ayrıca, proje sayesinde kodun hatalarını bulmak ve geliştirmek de daha kolay hale gelir.

**Nasıl Kullanılır**

1. src.txt dosyasına analiz edilecek assembly kodu eklenir.
2. optab.txt dosyasına opcode tablosu eklenir.
3. Gerekirse sembol tablosu (symtab.txt) oluşturulur.
4. Betik çalıştırılır ve analiz işlemi gerçekleştirilir.
5. Sonuçlar, intermediate.txt ve Details.txt dosyalarında bulunabilir.

**Fonksiyonlar**

metin, çizgi, yazı tipi, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

**hexadecimal\_to\_decimal(x)**

Verilen bir onaltılık sayıyı onluk sayıya dönüştürür.

**read\_line\_from\_file(src, pos)**

Belirtilen dosyadan belirli bir konumdan itibaren bir satır okur.

**break\_line(line)**

Verilen bir satırı parçalar ve etiket, opcode, operand ve geri kalan kısımları döndürür.

**search\_symtab(label)**

Verilen bir etiketin sembol tablosunda bulunup bulunmadığını kontrol eder.

**search\_optab(opcode)**

Verilen bir opcode'un opcode tablosunda bulunup bulunmadığını kontrol eder.

**main()**

Ana işlemlerin gerçekleştirildiği fonksiyon. Kaynak dosyayı analiz eder, sembol tablosunu ve geçici dosyayı oluşturur ve sonuçları raporlar.