DERLEYİCİ TASARIMI

Derleyici Tasarımı dersinde neler göreceğiz ?

- Derleyicinin Yapısı (Compiler) nedir ?
- Sözdizimi Analizi (Lexical Analysis)
- Kurallı İfadeler (Regular Expression)
- Sonlu Otomat (Finite Automata-FA)
- Sentaks Analizi (Syntax Analysis)
- Belirsizlik (Ambiguity)
- Ayrıştırma (Parsing)
- Tip Kontrolü
- Arakod üretimi (Intermediate Code)
- İyileştirme (Optimizations)
- Hedef Dil

Kaynaklar

- Dick Grune, Kees van Reeuwijk, Henri E. Bal, Ceriel J.H. Jacobs, and Koen G. Langendoen (2012). Modern Compiler Design - Second Edition
- Alfred V. Aho, Monica S. Lam, Jeffrey D. Ullman, Ravi Sethi (2011). Compilers: Principles, Techniques, and Tools. Pearson Education, Inc. ISBN 0-201-10088-6, ISBN 0-321-48681-1
- Torben Ægidius Mogensen(2008). Basics of Compiler Design. Department of Computer Science University of Copenhagen.
- Andrew W. Appel (2004). Modern Compiler Implementation in C.

Bu Haftaki Konu Başlıkları

- Derleme (Compilation) nedir ?
- Yorumlama (Interpretation) nedir ?
- Derleyici ve Yorumlayıcının artı-eksileri nelerdir ?
- Derleyicinin Kısa Tarihi
- Basit bir derleyici yapısı
- Temel Kavramlar

- Bilgisayarlar sadece düşük seviyeli dillerde yazılmış programları çalıştırabilirler. Bu yüzden yüksek seviyeli dillerde yazılmış programlar çalıştırılmadan önce bir işlemden geçmelidir. Bu ek işlem biraz zaman alır, bu da yüksek seviyeli dillerin dezavantajıdır. Ancak avantajları oldukça fazladır.
- Yüksek seviyeli dilde yazılmış programlar daha az sürede yazılır, daha kısadır, okuması daha kolay ve doğru olma ihtimalleri daha yüksektir.
- Farklı bilgisayarlarda az değişiklik veya değişiklik yapmadan çalıştırılabilirler. Düşük seviyeli programlar sadece tek bir bilgisayar çeşidinde çalışabilirler ve başka bir bilgisayarda çalışabilmesi için tekrar yazılmalıdır.

- Düşük Seviyeli Diller
  - Makina Dili: Bilgisayarların ilk dönemlerindeki programlama dilleri, kullanımı çok zor olan makine dili ve assembly dilleriydi. Makine dili, geliştirilen ilk programlama dilidir. Tüm komutlar 0 ve 1'lerden oluşur.

    Makina dilinde programlama:

    - Çok yavaştır ve hataya açıktır.
    - Kullanıcı dostu olmayan yapıları vardır.
    - Kullanılan bilgisayar mimarisi ile ilgili ayrıntılı bilgi gerektirir.
    - Başkalarının yazdıkları kodları anlamak çok güçtür.
    - Makine değiştiği takdirde kod kullanılamaz.

- Assembly Dili: Makine dili talimatları, daha kolay bir şekilde anlaşılabilecek olan sembollerle ifade edilir. Günümüzde kullanılır çünkü bu dille yazılan programlar genellikle çok hızlı çalışır ve daha az depolama alanı gerektirir. Fakat programlama yapmak sıkıcı, yorucu ve zaman alıcıdır.

Orta Seviye Diller: Oldukça esnek olan bu diller hem düşük seviye, hem yüksek seviye programlama yapabilirler. Düşük seviyeli dillere oranla biraz daha anlaşılırdır. C, C++ gibi..

Yüksek Seviye Diller: Öğrenmesi daha kolay, program yazılması daha az zaman alan, daha iyi sonuçlar sağlayan programlama dilleridir. Fortran, Java..

Çok Yüksek Seviyeli Diller: İnsan diline en yakın özellikler gösteren dillerdir. Visual Basic, Access, VB.NET …

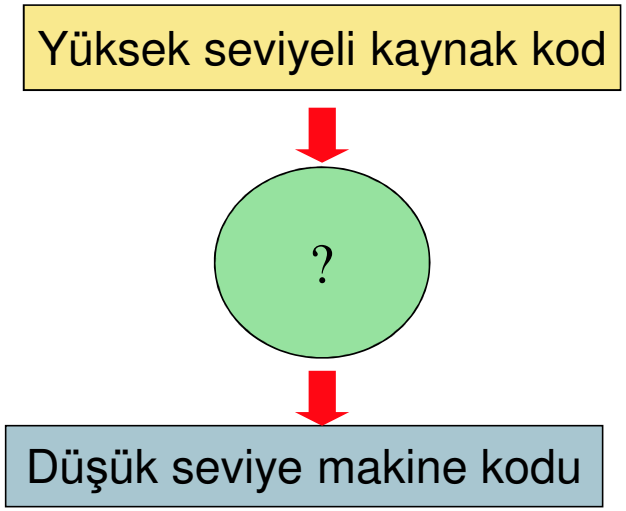
Dil çevirici (translator) nedir ?

Diller arası çeviri işleminin (translation) gerekliliği **dil çevirici** yazılımların oluşturulmasına neden olmuştur.

Bir programlama dilinde yazılmış olan programı eşdeğer olarak başka bir dile dönüştüren programa çevirici program (translator) denmektedir. Örneğin C# dilinde yazılmış bir programı VB.NET diline dönüştüren program bir çevirici programdır.

Eğer bir çevirici programın dönüştürme yaptığı hedef dil, düşük seviyeli bir dilse, bu tür çevirici programlara derleyici denmektedir.

Dil çevirici nedir ?



Şekil 1. Dil çevirici

Şekil 2. Dil çevirici

Dil çevirici nedir ?

Çeviri işleminin 2 temel yöntemi vardır.

- Derleme (compilation)
- Yorumlama (interpretation)

Derleme (Compilation)

Belli bir dille yazılmış program metnini anlamını bozmadan başka bir dildeki program metnine dönüştürme işlemine **derleme** denir.

```
program gcd(input, output);
var i, j: integer;
begin
    read(i, j);
    while i <> j do
        if i > j then i := i - j;
        else j := j - i;
    writeln(i)
end.
```

Derleme (Compilation)

```
27bdffd0 afbf0014 0c1002a8 00000000 0c1002a8 afa2001c 8fa4001c
00401825 10820008 0064082a 10200003 00000000 10000002 00832023
00641823 1483fffa 0064082a 0c1002b2 00000000 8fbf0014 27bd0020
03e00008 00001025
```

Şekil 3. Derleme

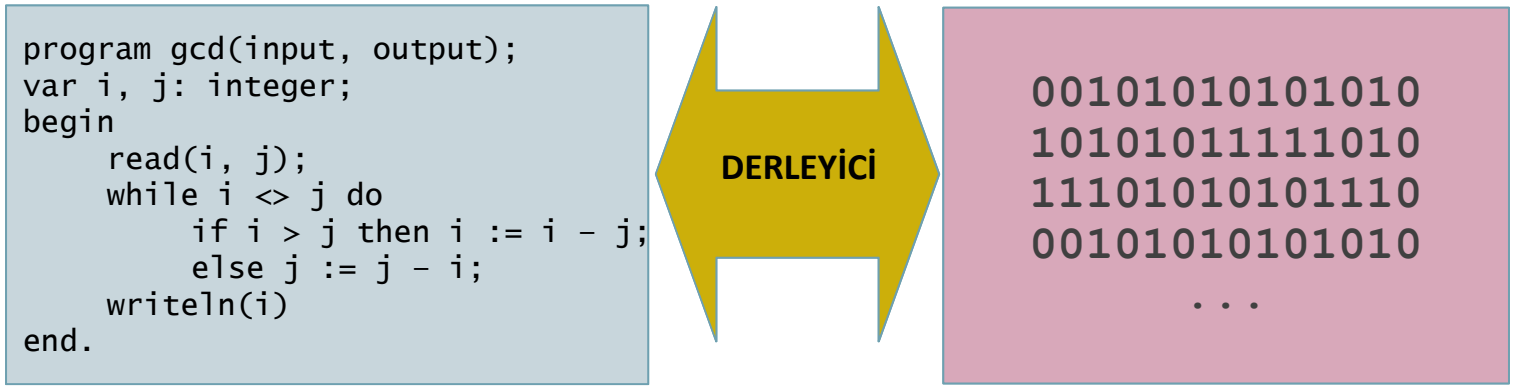
Yorumlama (Interpretation)

Bir programın her deyiminin birer birer makine diline çevrilip çalıştırılması işlemine yorumlama (interpreter) denir.

Programın derleme işlemini gerçekleştiren programa **Derleyici (Compiler)**, yorumlama işlemini gerçekleştiren programa ise **Yorumlayıcı** (**Interpreter**) adı verilir.

Derleyici (Compiler) Nedir ?

En genel tanımıyla bir **derleyici (compiler)**, belli bir dille yazılmış program metnini giriş olarak alan ve metnin anlamını bozmadan başka bir dildeki program metnine dönüştüren bir programdır. Bu süreç çeviri **(translation)** olarak adlandırılır.



Şekil 4. Derleyici

Derleyici (Compiler) Nedir ?

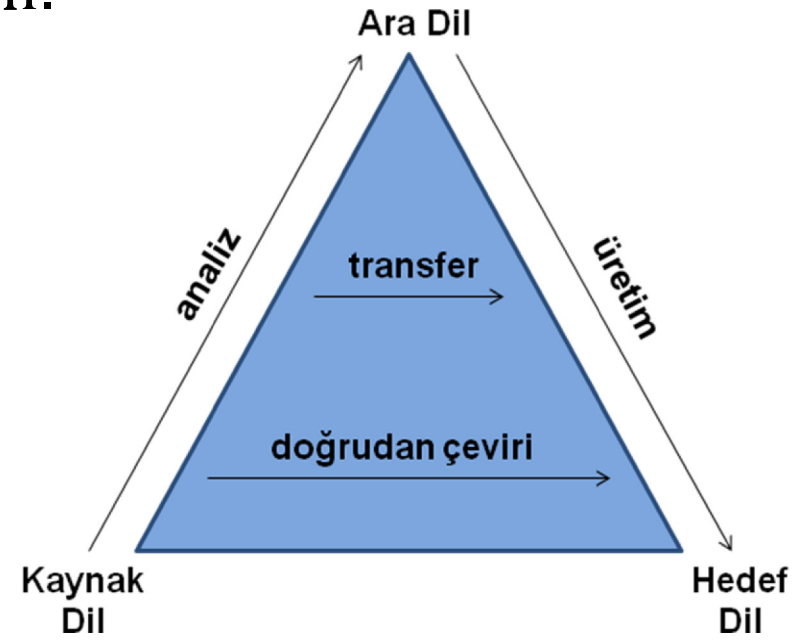
Bu üretim sırasında ya doğrudan işletim sisteminin anlayacağı ve çalıştıracağı kodları üretirler ya da işletim sisteminde bulunan veya yine dil bağımlı olarak çalışan bağlayıcı (**linker**) programların anlayacağı ara kodları üretirler.

Derleyiciler bu kod üretmesi sırasında, üretilen kodun en verimli şekilde üretilmesi için kod iyileştirmesi (optimisation) de yapmaktadırlar. Yani hedef dildeki çalışma süresi ve hafıza ihtiyacı en az olan kodu üretmek bir derleyicinin daha başarılı olma kriterlerinden birisidir.

Aynı zamanda kaynak kodda (source code) bulunan hataların yakalanması bu hataların programcıya bildirilmesi de derleyicilerin diğer görevlerinden birisidir.

Derleyici (Compiler) Nedir ?

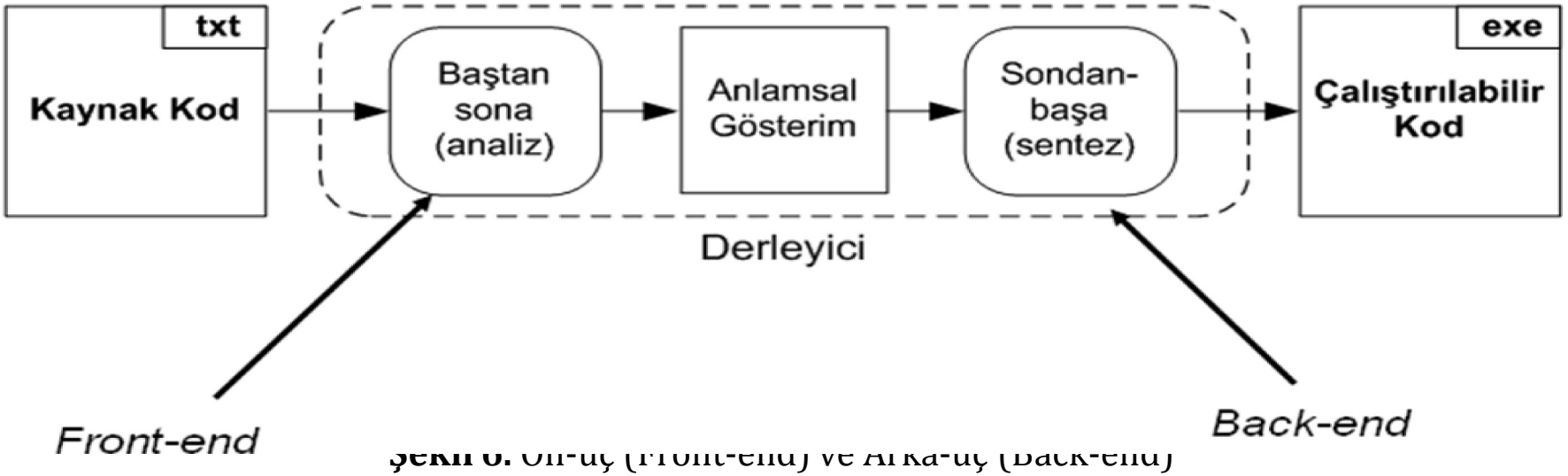
Bir derleyici elde etmek için derleyicinin kaynak kodundan meydana gelen ve derleyici için çalıştırılabilir kodlar üreten başka bir derleyici kullanılır.



Şekil 5. Otomatik çeviri paradigması

Derleyici (Compiler) Nedir ?

Bir derleyici, tipik olarak ele aldığı kaynak kodu önce analiz eder. Sonra onun üzerinde sentez işlemini yaparak kod üretir. Derleyicilerin analiz işlemini yapan bölümlerine ön yüz (front-end), sentez işlemini yapan bölümlerine de arka yüz (back-end) denmektedir. Önyüz kaynak dille, arka yüz hedef dille ilgilidir. Önyüzle arka yüz arasında bir orta yüz bulunabilir.



Şekil 6. Ön-uç (Front-end) ve Arka-uç (Back-end)

Derleyici (Compiler) Nedir ?

Çalışma mantığına göre derleyiciler ikiye ayrılır:

- Tek geçişli (one pass) derleyiciler
- Çok geçişli (multi pass) derleyiciler

Burada geçiş ile kastedilen kavram, bir derleyicinin kaynak kodu baştan sona kadar okumasıdır. Yani tek geçişli derleyicilerde kaynak kod baştan başlanıp sona kadar bir kere okunmakta buna mukabil çok geçişli derleyicilerde (örneğin iki geçişli bir derleyicide) birden çok kereler (örneğin iki kere) kaynak kod baştan sona kadar taranmaktadır.

Tek geçişli derleyiciler tahmin edileceği üzere çok geçişlilere göre çok daha hızlı çalışmaktadırlar. Ancak bazı durumlarda dilin tasarımı tek geçişli derleyicilere izin vermemektedir.

Örneğin kodun sonlarına doğru kodun başında yapılan bir tanımı etkileyecek bir işlem yapıldığını düşünün bu durumda tek geçişle bu olayın algılanması ve kodun doğru şekilde derlenmesinin yapılması mümkün olamaz.

Tek geçişli derleyicilerin diğer bir eksik yanı ise kod iyileştirmesi sırasında kodun üzerinden sadece bir kere geçtiği için kodun önceki satırlarında bulunan ve daha sonradan anlaşılan iyileştirmelerin yapılamamasıdır.

Bağlayıcı (Linker) nedir ?

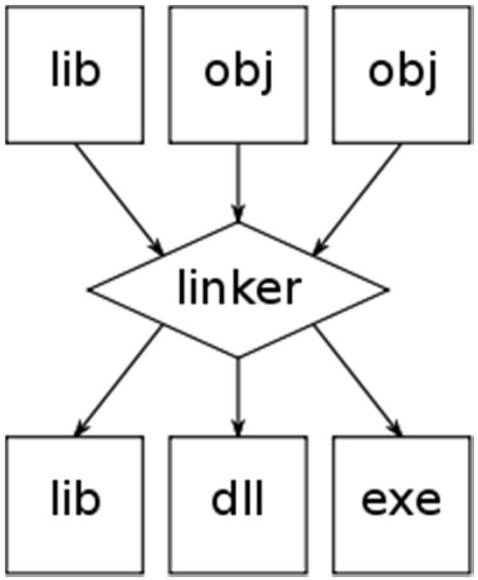
Bir derleyici tarafından üretilmiş olan kodları bağlayarak işletim sisteminin çalıştırabileceği tek bir kod üreten programdır.

Günümüzde hızla gelişen programlama ihtiyaçları sonucunda programlamada modüler yaklaşıma geçilmiştir. Buna göre büyük bir yazılım küçük alt parçalara bölünmekte ve her parça ayrı ayrı işlenerek büyük program elde edilmektedir. Yapısal programlamanın da çıkış sebeplerinden birisi olan bu yaklaşıma göre dillerde fonksiyon desteği gelmiş ve değişik parametrelere göre aynı kodun farklı sonuçlar üretmesi sağlanmıştır. Daha sonradan gelişen nesne yönelimli programlama bu konuda bir sonraki nesil olarak kabul edilebilir. Nesne yönelimli programlamada, programlar nesnelere bölünerek farklı bir yaklaşım izlenmiştir.

Bu yaklaşımların derleyicilere yansıması da uzun sürmemiş, daha yapısal programlamanın ilk geliştiği günlerde derleyiciler de farklı kütüphaneler ve bu kütüphaneleri birleştirmeye yarayan harici programlar kullanmaya başlamışlardır.

Bağlayıcı (Linker) nedir ?

Kodun birden fazla parçaya bölünmesi ve her parçanın ayrı ayrı üretilmesi durumunda bu parçaların birleştirilmesi ve tek bir program halinde üretilmesinden sorumlu olan programlara **bağlayıcı (linker)** adı verilmektedir



Şekil 7. Bağlayıcı

Yorumlayıcı nedir ?

Eğer Şekil-6'da gösterilen kod üreteci sürecinde sentez bloğu ile bir yorumlayıcı bloğu yer değiştirirse bütün program **yorumlayıcı (interpreter)** olarak adlandırılır.

Genel anlamda, bir kaynak kodu hedef koda çevirdikten sonra çalıştıran ve dolayısıyla koddaki hataları yakalama işlemini ve kodun iyileştirilmesini daha kod çalıştırılmadan yapan çeviricilere **derleyici,** kodu satır satır veya bloklar halinde çalıştırıp sırası gelmeyen satırları hiç çalıştırmayan bu satırlardaki hataları hiçbir zaman göremeyen ve kodun bütününe ait iyileştirmeleri yapamayan çeviricilere de **yorumlayıcı** adı verilmektedir

Yorumlayıcı nedir ?

Genel kanının tersine bir dilin derleyici veya yorumlayıcı özelliği yoktur. Yani C dili için sadece derleyicisi bulunan bir dildir demek yanlış olur. Bu durum bütün diller için geçerlidir. Her dil için bir derleyici veya yorumlayıcı tasarlanabilir. Ama daha genel bir bakışla, her dilin aslında yorumlayıcı (interpreter) yapısında bir çalışması olduğunu söylemek yanlış olmaz. Sonuçta bilgisayarın işlemcisinde anlık olarak tek bir işlem yapılabilmektedir ve çalışması istenen kod, işlemciye sırayla verilecek ve satır satır çalıştırılacaktır.

Derlemenin Avantajı/Dezavantajı

Derlemenin avantajı zaman harcayan bir işlem olduğu halde programı sadece bir kere analiz ederek çevirme işlemini yapması ve daha sonraki çalıştırma işlemlerinin ise derlenmiş program kullanılarak hızlıca yapılabilir olmasıdır.

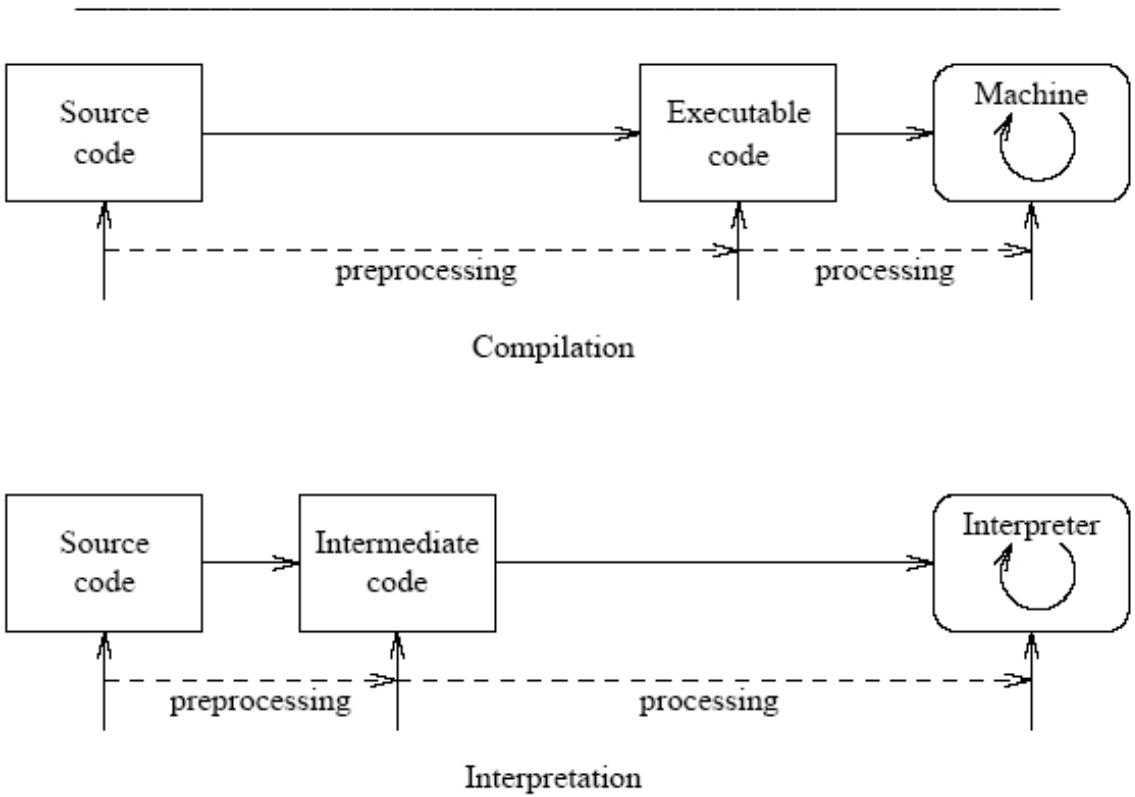
Dezavantajı ise makine koduna çevrilmiş programda eğer bir hatayla karşılaşılırsa bu hatayı düzeltmek için programın yazıldığı dile dönüp düzeltme yapma gerekliliğidir.

Yorumlamanın Avantajı/Dezavantajı

Avantajı; Eğer bir hatayla karşılaşılırsa henüz kaynak program üzerinde çalışılıyor olduğundan hatanın yeri kolayca tespit edilebilir. Bu da program geliştirmede çok büyük bir kolaylık sağlamaktadır.

Dezavantajı; Yorumlama esnasında karşılaşılan ifadenin her defasında tekrardan bütün dil yapısı içinden bulunup analiz edilmesi gerektiğinden derlemeye nazaran çok daha yavaştır.

Derleme ve Yorumlama Süreçleri



Derlemede;

Program süreci önemlidir ve zaman alabilir.

Sonuçlanan ara form düşük seviyelidir (makine kodu)

Yorumlama mekanizması donanım işlemcisidir.

Program uygulaması kıyasla hızlıdır.

Yorumlamada;

Program uygulaması minimumdur.

Sonuçlanan ara form yüksek seviyelidir

Yorumlama mekanizması programdır

Program uygulaması kıyasla yavaştır.

Şekil 8. Derleme ve Yorumlama Süreçleri

İyi bir derleyicinin özellikleri

- İyi bir derleyicinin en önemli özelliği doğru kod üretebilmesidir
- Derleyicinin dilin özelliklerine tamamen uyması da önemlidir
- Derleme hızı çok önemli değildir. Küçük programlar hızlı makinelerde birkaç saniyede derlenebilirler. Büyük projeler ise genellikle alt programlarda organize edilirler (modüller, kütüphane yordamları vb.). Bunlar derleyici birimleri olarak adlandırılır ve her biri ayrı ayrı derlenebilir.
- Günümüzde birçok bilgisayarda bellek problemi olmadığından derleyici büyüklüğü önemini yitirmiş bir konudur.
- Bir derleyicinin kullanıcı dostu olması hata raporlama kalitesi ile ölçülebilir.

Derleyicinin kısa tarihi

Üzerinde assembly dili ile program yazılabilen ilk bilgisayarlar 1950'lerde üretildiler. Ancak assembly dili kullanılarak geliştirilen programlar oldukça hızlı çalışmasına rağmen o programları geliştirmek çok uzun zaman alıyordu.

1953'te John Backus tarafından üretilen ilk yorumlayıcı, assembly kodundan daha yavaş çalışıyordu.

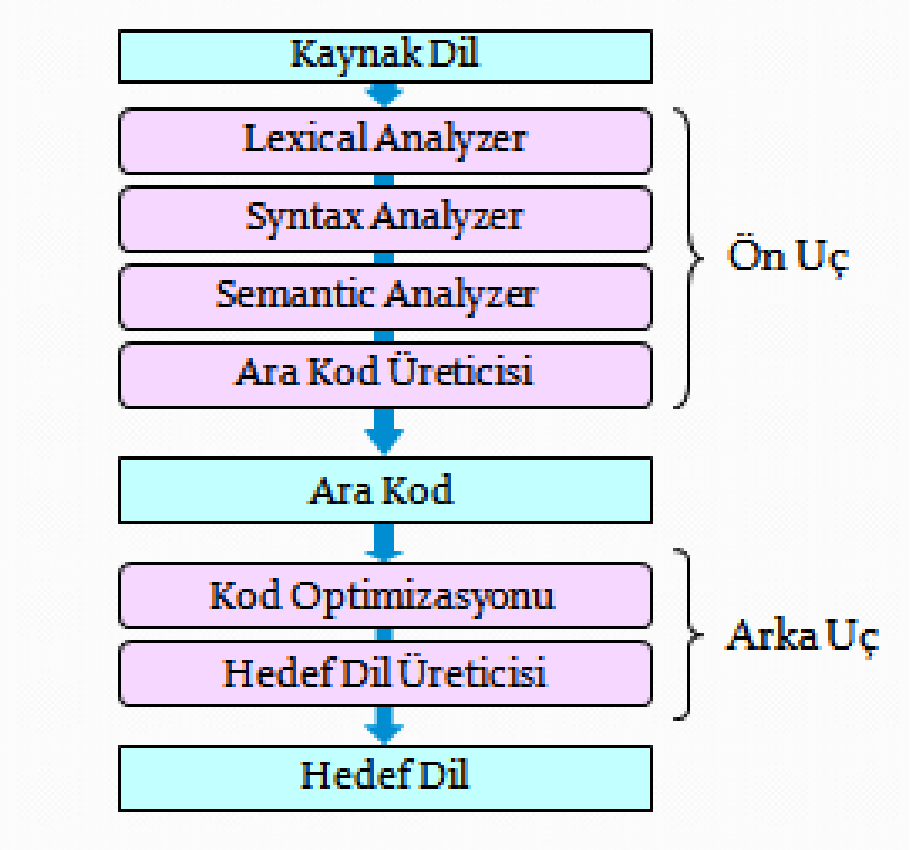
1956'da Backus ve arkadaşları, FORTRAN dilinin ilk derleyicisini ürettiğinde ise hız sorunu aşılmış oldu.

Basit Bir Derleyici Yapısı

Basit bir derleyici;

- en az bir sözcük analizi (lexical),
- bir sözdizimi analizi (syntax)
- bir içerik yönetimi (context handler) bloğundan oluşur.

Bu süreç ön-uç (front-end) olarak adlandırılır.



Şekil 9. Derleyici işlem basamakları

Basit Bir Derleyici Yapısı

Lexical Analiz: "scanner" da denir. Derlemenin ilk evresidir ve tek başına metin içerisinde anlamı olmayan karakterleri mantıksal olarak gruplar. Örneğin 'W', 'H', 'I', 'L' ve 'E' harflerinin "WHILE" kelimesini oluşturabileceği açıkça görülmektedir. Bir lexical analiz aracı şu üç şeyi yapabilmelidir:

1. Bütün boşlukları ve açıklamaları temizlemeli
2. Karakter katarı içindeki tüm tokenlar bulunmalı
3. Bulunan token için lexeme ve bulunduğu satır numarası gibi özellikler döndürülmelidir.

Syntax Analiz: Bütün programı analiz eder ve lexical analizde yapılanın bir üst evresini gerçekleştirir. Hedef, tokenlardan oluşan listenin, sözdizimine uygun olup olmadığını belirlemektir. Bu evrede scanner'ın oluşturduğu küçük grupları (token) daha büyük gruplara dönüştürür. Mesela ifadeler, döngüler, altprogramlar gibi...

Basit Bir Derleyici Yapısı

Yüksek seviyeli bir dille yazılan kodun çalıştırılabilmesi için en düşük seviyede bilgisayarın işleyebileceği komutlar dizisi haline getirilmesi gerekmektedir. Bu sebeple kaynak kodu oluşturan karakter katarı anlamlı alt parçalara ayrıştırılmalıdır. Alt parçalara token , bunları ayrıştırma işlemine de lexical analiz denir. Lexical analiz işleminin çıktısı olan tokenlar bir sonraki aşama olan syntax analiz bileşenine aktarılır.

Token sözdizimsel bir kategori belirtir. Bu kategoriler doğal diller için “isim”, “sıfat”, “fiil” vb. olabilirken, programlama dilleri için “matematiksel sembol” veya “anahtar kelime” gibi alt ifadeler olurlar. Token belirten bir alt karakter katarı **lexeme** olarak adlandırılır.

Basit Bir Derleyici Yapısı

Lexeme: Bir dilin en düşük seviyeli söz dizimsel birimidir.

Token: Lexeme'ların kategorisidir. Token'ları kendimiz isimlendiririz.

Örneğin; `int a=b+c;` ifadesinin lexeme ve tokenlarını belirleyelim.
Anlamlı herbir söz dizimsel birim lexeme'dır("=","a","+" vb.)

lexeme	token
`int`	type
`a`	variable
`=`	assign
`b`	variable
`+`	add
`c`	variable
`;`	semicolon

Basit Bir Derleyici Yapısı

Regular Expression (Kurallı İfadeler)

Nedir?
Programlama dili içinde kullanılması mümkün olan tüm lexemeler kurallı ifadeler (regular expressions) kullanılarak tanımlanır.
Regular expression, birçok programlama dilinde kullanılabilen, sayısal ve dizgisel içeriklerde belirli kurallara uyan bölümleri bulan ifadelerdir. “Düzenli ifade” ile Türkçe karşılığını kullananlar olsa da bazı sözlüklerde karşılığı “kurallı ifadeler” olarak belirtilmektedir.

Nerelerde Kullanılır?

Birçok programlama dili (Perl, PHP, Java, Javascript, .Net …), işletim sistemi komutları, veri tabanı sorgularında kullanılır.

Genelde ortak standart yazımı olmasına karşın, kullanıldığı yerlere göre farklılıklar gösterebilmektedir.

Temel Kavramlar;

Editör (Editor): Kaynak kodu oluşturmak ve gerektiğinde değişiklik yapmak için kullanılan araçtır. Editörde yazanlar seçilen dilin komutlarından oluşan metinlerdir.

Derleyici (Compiler): Editör tarafından bir bilgisayar dilinde yazılmış olan kaynak kodu makine koduna çeviren bir bilgisayar yazılımıdır. Yazılan kodun kullanılan dile uygunluğunu denetler.

Kütüphane (Library): Nesne dosyalarından oluşan kütüphanedir.

Bağlayıcı (Linker): Programın içerdiği tüm nesne dosyalarını birleştirerek tek bir yürütülebilir dosya haline getirir.

Yükleyici (Loader): Yürütülebilir dosyayı diskten belleğe kopyalar.

Hata Ayıklayıcı (Debugger): Programcının hatalarını anlayabilmesi için programın yürütülmesini adım adım kontrol edebilmesini sağlar.

Yorumlayıcı (Interpreter): Bir programın kaynak kodunu doğrudan satır satır yürüten bir programdır.

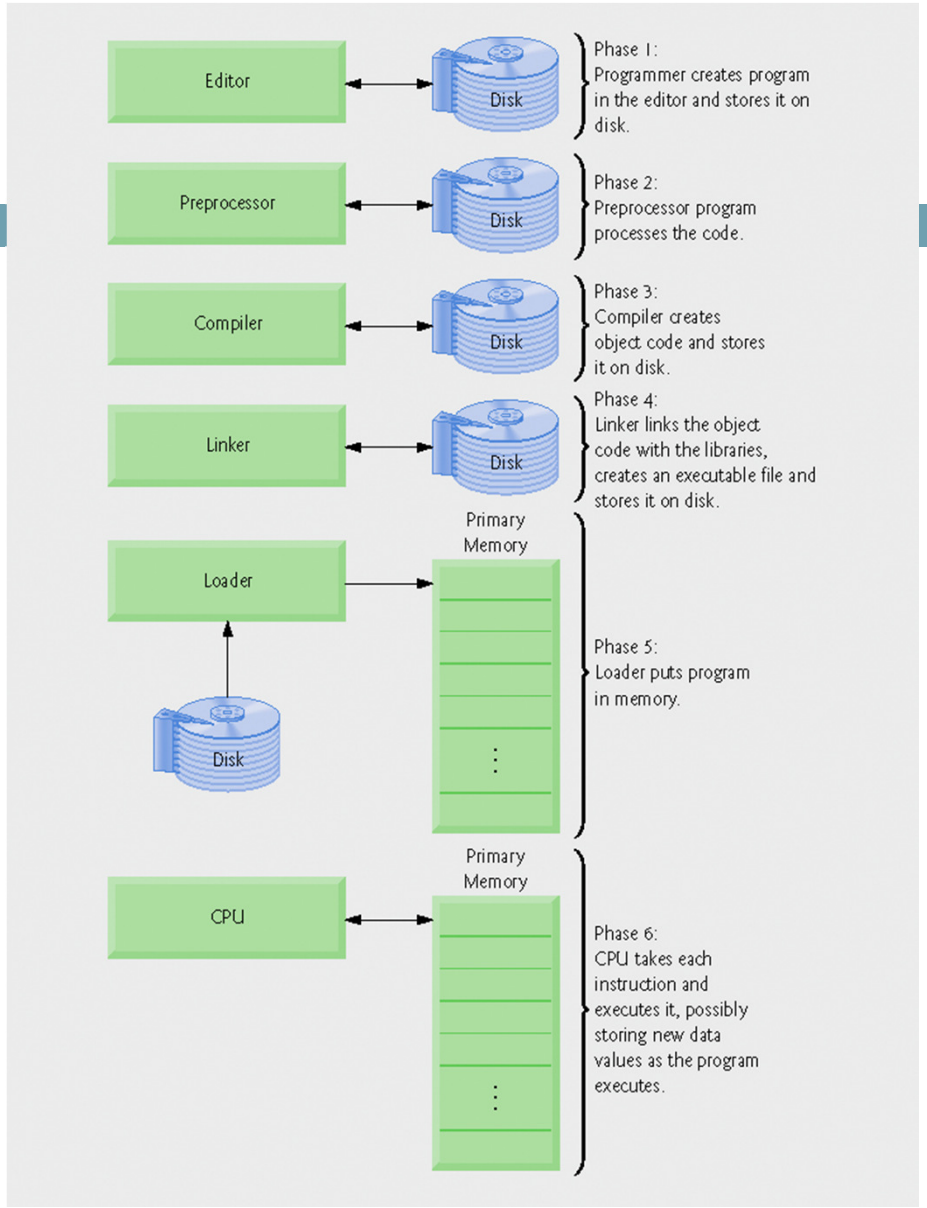
Temel Kavramlar;

Söz Dizimi (Syntax) : Sıradan dillerde olduğu gibi, programlama dillerinin de bir söz dizimi vardır. Söz dizimi, simgelerin geçerli olarak kabul edilebilmesi için hangi düzende yazılması gerektiğini belirleyen kurallar dizisidir. Sözdizimsel hataların çoğu derleyici tarafından yakalanıp raporlanacaktır. Ancak bazı hatalar derleyicilerin yakalayamayacağı türden olup çalışma esnasında hatalara neden olabilir.

Anlambilim (Semantics) : Bir programlama dilindeki bir ifadenin ne anlama geldiğidir.

Lexeme: Bir dilin en düşük seviyeli söz dizimsel birimidir.

Token: Lexeme'ların kategorisidir.


Editor
Disk
Phase 1:
Programmer creates program in the editor and stores it on disk.
Preprocessor
Disk
Phase 2:
Preprocessor program processes the code.
Compiler
Disk
Phase 3:
Compiler creates object code and stores it on disk.
Linker
Disk
Phase 4:
Linker links the object code with the libraries, creates an executable file and stores it on disk.
Primary Memory
Loader
Disk
Phase 5:
Loader puts program in memory.
Primary Memory
CPU
Phase 6:
CPU takes each instruction and executes it, possibly storing new data values as the program executes.