BİLGİSAYAR DESTEKLİ EĞİTİM

Dr. Tunç Balman
Temel Bilimler Fakültesi
Boğaziçi Üniversitesi

Doçentlik Tezi
Mart 1981

ÖZET

Bu tezde Bilgisayar Destekli Eğitim'in (BDE) amaçları tanıtılmakta ve değişik araştırmacıların benimsediği yöntemler incelenmektedir.

Program yazılım maliyetini düşürücü önlemlerle BDE etkinliğini arttıracak programlama teknikleri tartışılmakta ve BDE hizmeti veren bilgisayar ünitelerinde gerekebilecek altyapı yazılım desteği incelenmektedir. Bu doğrultuda 1978 senesinde geliştirmiş olduğum, bilgisayarlar arası etkileşimli iletişim ve program erişim dizgeleri özetlenmektedir.

Bilgisayar Destekli FORTRAN Eğitimi için önerip geliştirmiş olduğum, FCN dizgesi tanıtılmakta ve bu dizgenin içerdiği etkileşimli-artımlı FORTRAN derleyicisinin öğretim programına katkısı tartışılmaktadır. FCN dizgesinin alıştırıcı nitelikteki alt sistemlerini kullanarak, FORTRAN kurslarının çeşitli safhalarında, öğrenciye sağlanabilen eğitim hizmetinden örnekler verilmektedir. Dizgedeki kaynak kütük değiştiricisinin içerdiği artımlı derleyicinin değiştirici yazılımına getirdiği yenilik ve kullanıcıya sağladığı yararlar belirtilmektedir.

FCN dizgesinde sağlanan komut alıştırıcılarını bilgisayarlar arası taşınabilir bir biçime dönüştürmek amacıyla, FORTRAN dili kullanılarak, geliştirilen FCN-F programı teknik ayrıntılarıyla tanıtılmaktadır.

İÇİNDEKİLER

BÖLÜM 1

BİLGİSAYAR DESTEKLİ EĞİTİM

Bilgisayar Destekli Eğitim (Computer Aided/Assisted/Based veya Managed Instruction/Learning/Teaching veya Education) adı altında değişik eğitim düzeylerinin çeşitli dallarında yaklaşık 1960 yılından beri birçok uygulama yapılmıştır (CAI 70, CAI 73). Bazen bu tür çalışmalar konuya ilgi duyan kişiler tarafından kişisel olarak başlatılıp yıllarca sürdürüldükten sonra kuruluşlar arası ortak bir zemin hazırlanmış ve Bilgisayar Destekli Eğitim bilinçli ve bilimsel plânlar çerçevesinde yoğun bir araştırma konusu olmuştur (NCET 69a, 69b, CET 75). Öte yandan bir çok uygulama da genellikle bağımsız ve gelişigüzel girişimlerden öteye gidemeyip oldukça verimsiz olmuştur.

Bir çok Bilgisayar Destekli Eğitim (BDE) öncüsü tarafından ileri sürülen "eğitimde yeni bir çağ", "matbaadan bu yana en önemli eğitim aracı" ve benzeri savlar BDE'in beklendiği şekilde yaygınlaşmamasından ötürü henüz gerçekleşememiştir. Bunun başlıca nedenlerini şöyle sıralayabiliriz

i) Kısa ve orta dönemli hedeflerin gerçekçi bir biçimde saptanmaması. Gerek basın ve gerekse araştırmacıların öne sürdüğü abartılmış savların kamuoyunu ve öğretmenleri olumsuz yönde etkilemesi

ii) Dağınık ve bağımsız araştırmalar. Bir çok araştırmalar ya bilgisayar bilgisi amatörce olan öğretmenler ya da eğitimle fazla ilişkisi olamayan bilgisayar uzmanları tarafından yapılmıştır . Eğitim uzmanı, öğretmen ve bilgisayar uzmanı üçlüsü sağlıklı ve ortak araştırmalara girememiştir. En başarı uygulamalar öğretmen ve bilgisayar uzmanının aynı kişi olduğu üniversite ortamında yapılmıştır.

iii) BDE'in öğrenciye ve öğretmene sağladığı ve sağlayabileceği yararların kesin ve bilimsel olarak kanıtlanmaması. Bunun başlıca sebebi eğitimde mutlak ölçüleri bulmakta karşılaşılan güçlüklerdir.

iv) Tutucu olan eğitim sektörünün değişikliğe karşı geleneksel tepkisi. Bu etmen özellikle ilk ve orta öğretim kesiminde belirgindir. BDE'in yararları kesin olarak kanıtlanmadan bu sektörde yaygın bilgisayar kullanımı gerçekleştirilemez.

Henüz sağlıklı bir yaygınlaşma sürecine girememesine rağmen halen BDE büyük bir gelişme ve yaygınlaşma potansiyeline sahiptir.

Yedi bölümden oluşan bu tezin ilk bölümünde değişik BDE yöntemleri ve bunların amaçları tanıtılmaktadır.

İkinci bölümde üniversite mühendislik eğitiminde başarılı sonuçlar vermiş olan Bilgisayar Destekli Deney tanıtılacak ve bu yaklaşımın ekonomik etkenleriyle şimdiye dek kullanılan başarı değerlendirme yöntemleri incelenecektir.

Üçüncü bölümde BDE maliyetiyle yaygınlaşmayı önemli ölçüde etkileyen program taşınabilirliği işlenecek ve taşınabilirliği güçleştiren unsurlarla kolaylaştırıcı önlemler tanıtılacaktır.

Dördüncü bölümde ise BDE hizmeti verecek sistemlerde aranan yazılım ve donanım özellikleriyle işletim sistemine yapılması gerekebilecek değişiklikler tanıtılmaktadır.

Beşinci ve altıncı bölümlerde Bilgisayar Destekli FORTRAN Eğitiminde kullanılmak üzere geliştirilen bir derleyici tanıtılmaktadır.

Son bölümde Bilgisayar Destekli FORTRAN Eğitiminde yapılan araştırmalar eleştirilmekte ve yeni bir yöntem özetlenmektedir.

1.1 BDE- amaçlar ve yöntemler

Tüm BDE araştırmacıları iki ana unsuru hedef almıştır:

Daha iyi eğitim

Daha ekonomik eğitim

Uygulamanın yapıldığı kuruluş ve uygulama dalına göre doğal olarak bu iki hedefe verilen göreceli ağırlık değişmektedir. Eğitimi ucuzlatmak amacıyla kaliteyi düşürmek söz konusu olabileceği için bu iki hedef birbirine bağımlıdır. Bu çok genel kapsamlı hedeflere ulaşmak için kullanılan yöntemlerin farklılığından başka önerilen çözümler de çeşitlidir. Örneğin bazı araştırmacılar okul, ders ve öğretmen yerine BDE programları önerirken, diğerleri mevcut eğitim sistemini zedelemeden bilgisayarı sadece eğitim aracı olarak kullanmayı amaçlamaktadır. Günümüzdeki bilgisayar teknolojisi gözönüne alındığında daha gerçekçi olan ikinci yaklaşım özellikle üniversite ortamında başarılı olurken, yinelenen profesyonel eğitimde ikinci yaklaşım tercih edilmektedir.

Daha"ekonomik" ve "iyi" eğitim sağlamak amacıyla değişik uygulamaların getirmeye çalıştığı başlıca yenilikleri şöyle sıralayabiliriz:

1. Her öğrencinin öz yeteneğine göre ders ve ödev esnekliği
2. Öğrenci/öğretmen oranının yüksek olduğu durumlarda eğitim ve denetimde öğretmene yardım
3. Sakıncalı veya pahalı makine ve alet kullanımının azaltılması
4. Profesyonal sektördeki yinelenen öğretimin bilgisayara kaydırılması

Bilgisayar Destekli Eğitimin geliştirildiği veya kullanıldığı kuruluşları şöyle sıralayabiliriz:

1. Üniversiteler ve bilimsel eğitim merkezleri
2. Orta okul ve liseler
3. İlk okullar
4. Profesyonel eleman eğiten özel veya kamu kuruluşları (örneğin sağlık personeli, teknisyen vb.)
5. Askeri eğitim kuruluşları.

Bunlar arasında en yaygın kullanıcı sektörünün üniversiteler olduğunu görürüz. Değişik araştırma ve uygulama ortamlarında birbirinden çok farklı BDE yöntemleri benimsenmiştir. Bu yöntemler çoğu kez mevcut bilim dallarından (örneğin simülasyon, karar teorisi, vs.) BDE'e aktarmalar yapılarak gerçekleştirilmiştir. Şimdiye dek kullanılan ve başarılı sonuçlar veren başlıca BDE yöntemlerini şöyle özetleyebiliriz.

1. Özdevimli ders kitabı (automated textbook)
2. Çevirim dışı (off line) ders notu ve ödev dağıtımı
3. Gerçek zamanlı (real time) aygıt benzetimi
4. Çevirim içi (on line) eğitici şans oyunları
5. Çevirim içi (on line) eğitici çizim ve hareket oyunları
6. Konuşmalı (conversational) soru-yanıt-yardım sistemleri
7. Bilgisayar Destekli Deney benzetimi

Yukarıda belirtilen ilk altı yaklaşım bu bölümde yedinci yaklaşım ise daha ayrıntılı olarak 3. bölümde incelenmektedir.

1.1.1 Özdevimli ders kitabı

Profesyonel kesimde özellikle teknik servis personelinin sürekli gelişen ve değişen teknolojiye bağımlı olarak yeniden eğitilmesi söz konusu olmuştur. Tekrarlı olan bu eğitim şekli genellikle yenileyen veya hatırlatan, veya yetenekleri denetleyen bir niteliktedir.

Bir çok kuruluşta rutin olarak gereken bu tür kurslar süratle gelişen teknoloji yüzünden bir problem oluşturmaktadır. Karşılaşılan eğitim problemlerini şöyle özetleyebiliriz

i) Kursa katılan personelin kurs süresince normal görevine devam edememesi (örneğin teknik personelin servise çıkamaması)

ii) Büyük gruplu eğitimin aynı sürede birçok personeli devre dışı bırakması

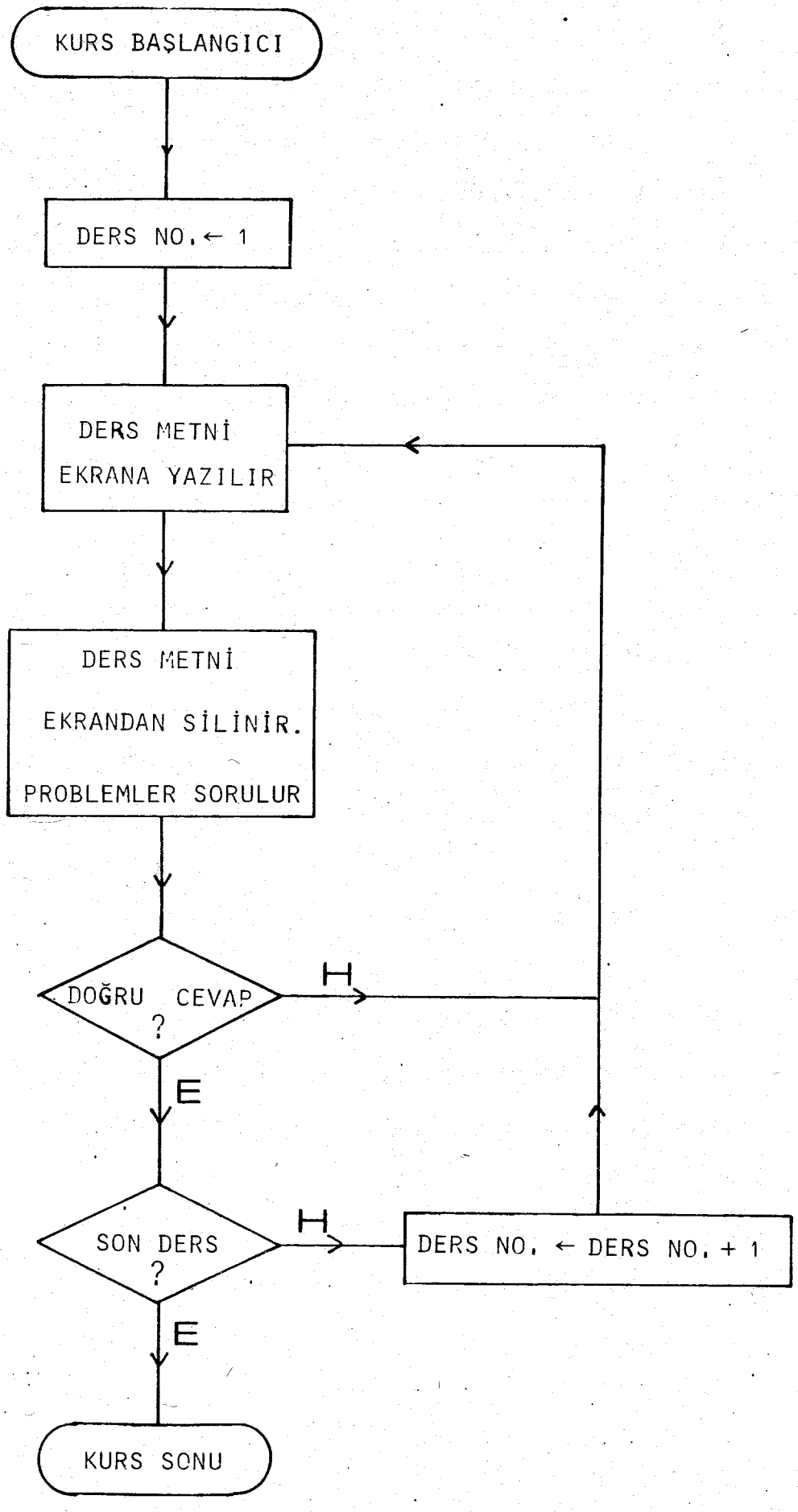
iii) Küçük gruplu eğitimin doğurduğu zaman kaybı ve ekonomik sakıncalar.

Bazı kuruluşlarda çözüm yolu olarak BDE denenmiş ve sonradan benimsenmiştir. Bu alanda BDE programlarının salt eğitici rolünü başarılı bir şekilde yüklendiği görülmektedir. Bu başarıda belki de en önemli etken eğitilenlerin deneyim ve sorumluluk sahibi kişiler olmasıdır.

Çoğunlukla profesyonel eğitim kesiminde özdevimli ders kitabı yöntemi uygulanmaktadır. En yaygın ve nisbeten en basit çevirimiçi BDE uygulama türü olan özdevimli kitabın genel akış çizeneği Şekil 1.1'de gösterilmiştir. İkincil bellekte saklanan ders metni öğrencinin kullandığı ekranlı uca gönderilir. Öğrenci devam komutunu verince ekrandaki bilgi silinir ve dersle ilgili sorular ekrana yazılır. Verilen cevaplardaki başarı derecesine göre ya aynı ders tekrarlanır ya da bir sonraki derse geçilir. Ders hızı kendiliğinden öğrencinin yeteneklerine bağımlı olduğu gibi aynı zamanda öğrenci gerektiğinde dersi bırakıp başka bir görev alabilir. Böylece personel kurs süresince normal görevine devam edebilir.

Kabataslak anlatılan bu katı yöntem bir takım ayrıntılar sayesinde daha etkin ve esnek bir eğitici olabilmektedir. Örneğin, çizim uçları (graphics terminal) kullanılıyorsa dersle ilgili açıklayıcı şekiller çizilebilir; bazı uygulamalarda öğrenci dersle ilgili daha ayrıntılı bilgi isteyebilir; öğrenci ve bilgisayar arasında nisbeten serbest bir diyalog olabilir; bilgisayar başarısız cevaplar karşısında dersin kapsamını genişletebilir, vb.

Profesyonel eğitim ortamında, daha önce de belirtildiği gibi, eğitilenlerin sorumlu kişiler oluşu BDE programlarının etkinliğini arttırmıştır. Bu yak-



Şekil 1.1 : Özdevimli ders kitabı yaklaşımının genel akış çizeneği.

laşım çerçevesinde her öğrenci, kursa katılan diğer öğrencilerden tamamıyla bağımsız olarak, öğrenmesi istenilen dersleri iş ortamının koşullarına göre en uygun zamanda izleyebiliyor. İlk bakışta özdevimli kitap alışılagelen ders kitabından farksız gibi görünüyorsa da derslerin sonunda sorulan problemlerin oluşturduğu engeller sayesinde özdevimli kitap kaydedilen ilerlemeleri sürekli olarak denetleyebilmektedir. Ayrıca, ders metninin kâğıt üzerinde değil de bilgisayarın ikincil belleğinde oluşu, derslere yapılacak değişikliklerde büyük bir kolaylık sağlamaktadır.

Öte yandan özdevimli kitapla ders kitabı arasında en büyük benzerlik de ders metninin hazırlanmasında görülen güçlüktür. Özellikle metnin tek eğitim aracı olduğu durumlarda büyük bir özen ve plânlama gerekmektedir. Normal ders içeriğine öğretmenin sağladığı esneklik bilgisayarda olmadığından her olasılık gözönüne alın bu esneklik yapay olarak ders metnine yerleştirilmelidir.

Çevirimiçi özdevimli kitap yaklaşımı yalnız profesyonel kesimde değil aynı zamanda orta ve yüksek eğitimde dersleri destekleme ve yetenekleri denetleme amacıyla kullanılmaktadır.

1.1.2 Çevirim dışı ders ve ödev dağıtımı

Tüm eğitim düzeylerinde yüksek öğrenci-öğretmen oranının eğitim kalitesin olumsuz yönde etkilediği belirgindir. Büyük sınıflarda her öğrencinin dersleri ne ölçüde izleyebildiğini erken teşhis etmek zordur. Zayıf öğrenciler çok erken saptansa bile sınıftaki uyumu zedelemeden gereken düzeltici önlemleri almak olanaksızdır. Öğretmen her öğrenciye özel özen gösteremeyeceği için aşağıdaki üç yaklaşımdan birini seçecektir

i) Ders hızını ve içeriğini vasat öğrenci kesitine göre ayarlamak veya yılların kazandırdığı deneyim sonucunda kararlaştırılan ideal tempoyu her sene aynen uygulamak.

ii) Zayıf öğrencilere yönelik ders verip parlak öğrencileri yavaşlatmak ve, en azından, parlak öğrencinin öğrenme potansiyelini değerlendirememek ; belki de dersten soğutmak.

iii) Parlak öğrencilere yönelik ders verip zayıf öğrencileri oluruna bırakmak; yani büyük bir olasılıkla ders içeriğinin belki de en önemli kısmını sebat ve özen sayesinde tam olarak öğrenebilecek birisine hiçbirşeyi öğretememek

En yaygın olan birinci yaklaşım çerçevesinde zayıflara ve parlaklara normal derslerin dışında özel özen gösterilse bile diğer iki yaklaşımda görünen sakıncalar yine de etkindir.

Bazı BDE uygulamaları, bu ve benzeri durumlarda, eğiticiye ve eğitilene iki şekilde yardım etmeyi amaçlamaktadır:

i) Öğrenci-öğretmen oranı yüksek olduğunda öğretmenin yükünü hafifletmek

ii) Her öğrencinin şahsi yeteneklerine göre bilgisini genişletmek

Bu yararların nasıl sağlandığını göstermek için İngiltere'de Hertfordshire County Council (Tagg 75) tarafından gerçekleştirilen bir uygulamayı inceleyebiliriz. 1973 yılından beri kullanılan bu BDE uygulamasına halen 12 değişik okulun 11-14 yaş grubundan yılda yaklaşık 4000 öğrenci katılmaktadır.

Hertfordshire uygulamasında ders yılına her öğrencinin aynı yetenekte olduğu varsayılarak başlanır ve ders yılı boyunca sınıftaki ders hızı ve içeriği daima vasat öğrenciye göre ayarlanır. Ödev olarak her öğrenciye bilgisayarın bastığı sorular ve verilen cevapların işaretlenebileceği ÖSS cevap kâğıdına benzer kâğıtlar dağıtılır. Her soru ve cevap kâğıdında öğrencinin ismi belirtilmektedir. Cevap kâğıtları doldurulduktan sonra çevirim dışı olarak im okuyucu (mark sensing device) tarafından okunup değerlendirilir. Her öğrencinin başarı derecesine göre bilgisayar yeni örnekler veya sorular

basar. Zayıf öğrencinin zayıflık derecesine göre açıklayıcı örnekler ve daha kolay alıştırıcı sorular verilir; parlak öğrencilerin başarılarına göre ya daha güç problemler ya da sınıfta henüz yapılmamış dersler, örnekler ve bunlara ilişkin problemler verilir. Bilgisayar her öğrenciye verdiği örnekleri ve ödevleri öğrencinin geçmiş ve güncel başarısına göre dinamik olarak ayarlar. Her ödev sonrası öğretmene sınıfının başarı derecesini gösteren çizelge, her öğrencinin kaydettiği ilerleme ve ortalama başarı derecesi BDE programı tarafından bir rapor halinde verilir. Böylece öğretmen hem vasat öğrenciye göre ders süratini ayarlayabilir hem de zayıf öğrencileri kolaylıkla ve erkenden tesbit edip gereken önlemleri almaya çalışabilir.

Bilgisayar bu örnekte, özdevimli kitaptan farklı olarak, öğretmen-öğrenci ilişkisini bozmadan ikinci bir eğitici rölündedir. Klâsik sınıf ortamına getirdiği önemli değişiklikleri şöyle sıralaybiliriz:

i) Öğrencinin dersle ilgili sınıf dışı faaliyetini yeteneklerine göre dinamik olarak belirlemek

ii) Bu faaliyeti etkin bir şekilde denetleyebilmek

iii) Öğretmene öğrencisinin kaydettiği geşmiş ve güncel ilerlemeyi bildirmek.

1.1.3 Gerçek zamanda aygıt benzetimi

Özellikle askeri ve sivil havacılıkta görülen bir BDE yöntemi pahalı ve tehlikeli araçların gerçek zamanda çevirim içi simülasyonudur. Bu yöntem askeri kesimde uçak veya tank gibi pilotlu araç personelinin saldırı ve karşı koyma eğitiminde kullanılmaktadır.

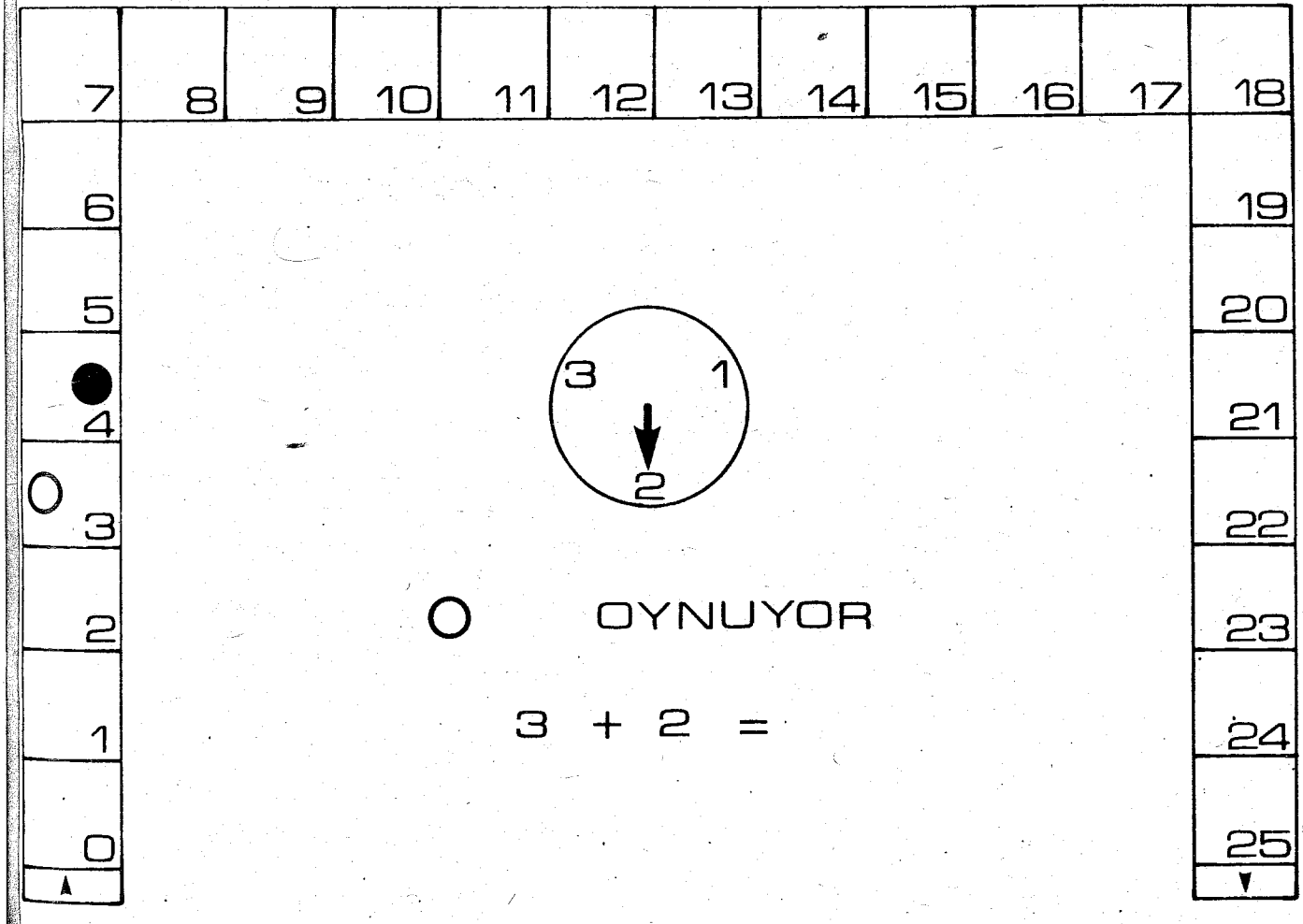
Bu tür uygulamaların bir örneği olarak BAC şirketinin (British Aircraft Corporation) geliştirdiği hava savaş benzetimini (Air Combat Simulator) gösterebiliriz (Morrison 75,77). BAC sisteminde gerçek bir savaş uçağının pilot kabini tüm kontrol ve gösterge aygıtlarıyla birlikte kullanılmaktadır. Kabindeki tüm kontrol ve gösterge aygıtları bilgisayar denetiminde olduğundan pilotun kontrol aygıtlarını kullanarak aldığı önlemler dinamik olarak gösterge aygıtlarına yansıtılmaktadır. Bir küre içerisine monte edilmiş olan kabinin etrafında yine bilgisayar kontroluyla değişik savaş görüntüleri ve senaryoları gösterilebilmektedir. Bu senaryo çerçevesinde eğitilen pilotun aldığı önlemlere duyarlı olarak gösterge aygıtları, görüntü ve karşı tarafın pilotu dinamik olarak simüle edilmektedir.

Gerçek zamanda yapılan bu simülasyon kuşkusuz gerçek teçhizatla yapılan eğitimden çok daha pratik ve ekonomiktir. BAC modelinde bilgisayar, pilotun aldığı her önlemi belleğine kaydettiği için, eğitilenlerin kazandığı tecrübeyi değerlendirebilmektedir. Gerektiğinde bilgisayardan alınabilen senaryo dökümü eğiticiyle eğitilen arasında tartışma zemini hazırlamaktadır.

BAC modeline benzer uygulamalar sivil havacılıkta da görülmektedir. Bu kesimde genellikle değişik hava koşulları, motör ve kanat arızaları, kalkış ve iniş senaryoları simüle edilmektedir.

1.1.4 Eğitici şans oyunları

Verilmiş olan diğer örneklerden farklı olarak, ilk eğitimdeki yaklaşımlarda oyun ve eğlence unsurlarına ağırlık verilip çekici eğitim programları oluşturulduğunu görürüz. Bu kesimde en yaygın yaklaşım çizim yetenekli uçlarda iki öğrencinin katılabileceği oyunlardır. Çoğunlukla basit zar oyunlarından esinlenerek bilgisayara aktarılan bu oyunlarda güdülen amaç öğrencinin sayı kavramıyla



Şekil 1.2 : Eğitici şans oyunu.

basit aritmetik yeteneklerini (toplama ve çıkarma) bir yarışma ortamında geliştirmektir (Dugdale 75).

Genellikle oyun kuralları basittir; her oyuncu atılan zara göre sayılarla işaretlenmiş olan karoların üzerinde kendi taşını hareket ettirip belirli bir hedefe hasımından önce ulaşmaya gayret eder (Şekil 1.2). Zar atma yerine sırayla her oyuncu için rasgele sayı üreten bilgisayar sürekli olarak en son oyun pozisyonunu ekranda gösterir. Öğrenci taşını ilerletmek istediği karonun sayısını ekran altındaki tuşları kullanarak bilgisayar programına bildirir. Tarafsız bir şekilde "zar atan" bilgisayar, taşını yanlış karoya hareket ettirmek isteyen oyuncuya doğru karoyu bildirdikten sonra "ceza" olarak sırasını iptal eder. Öte yandan doğru ilerlemeler ekrandaki oyun pozisyonuna yansıtılır. Ceza unsurunun yarattığı yarışma ortamında öğrencinin basit aritmetik yeteneklerini geliştirmesi amaçlanır

Kabataslak anlatılan bu yöntem daha da geliştirilebilir. Örneğin

i) Bilgisayar eşit yetenekli öğrencileri eşleştirip heves kırıcı rekabetleri ortadan kaldırabilir

ii) Karo sayıları ve üretilen " zar" değerleri sınıfta öğretilen derse göre ayarlanabilir

iii) Oyunculara kısıtlı bir yanıt süresi tanınabilir

iv) Birden fazla zar ve taş kullanılabilir

v) Birden fazla zar ve taş kullanılıp karolar arasına engeller konabilir.

iv. ve v. maddeler öğrencinin hesap yeteneklerini geliştirmenin yanısıra onu oyun stratejisi kurarak düşünmeye teşvik eder.

1.1.5 Çizim Oyunları

Logo kaplumbağası (LOGO turtle) ilk eğitimde 8 ile 12 yaş arası çocuklar için M.I.T'de geliştirilmiş olan yapay us kökenli bir uygulamadır (Papert 71a, 71b, 71c, 72). Karnının altında çizici uç (kalem) taşıyan kaplumbağa bilgisayar kontrolünde küçük bir robottur. Öğrenci LOGO programlama diliyle verdiği komutlarla robotu yönetip kalemin beyaz zemin üzerinde bıraktığı izlerle geometrik şekiller

çizmeye çalışır (Şekil 1.3). LOGO dili çocukların güçlük çekmeden öğrenebileceği beş komut türünden oluşur; bunlar

kalemi kaldır

kalemi indir

x uzunluk birimi ilerle

y açı birimi sağa dön

y açı birimi sola dön

Kalemi indirdikten sonra kaplumbağa ilerletilirse kalem beyaz zemin üzerinde iz bırakır.

Bu komutları kullanarak öğrenci kaplumbağaya "resim çizmeyi" öğretir. Görüldüğü gibi kaplumbağanın simgelediği bilgisayar öğrencinin dilediği gibi oynayabileceği bir kalemdir.

Öğrenci kendine verilen herhangibir şekli kaplumbağaya çizdirebilmek için ilk önce kendinin o şekli nasıl çizebileceğini incelemesi gerekir. Kendi çizim metodunu en ince ayrıntıya kadar düşündükten sonra çizim işlemini LOGO komutlarına dönüştürmek zorundadır. Kaplumbağanın çizdiği şekilde hata varsa, öğrenci vermiş olduğu komutlardaki hatayı ayıklarken çizilen ve çizilmesi gereken şekillerin yanı sıra komut zincirini de gözönünde tutmak zorundadır.

Tüm bu işlemleri yaparken öğrencinin problem çözümleme yeteneğinin geliştirildiği savunulabilir. LOGO kullanırken öğrenci

i) Çözülecek problemi çözülmesi daha kolay olan parçalara bölmeyi (örneğin insan şekli=baş, gövde, iki kol, iki bacak)

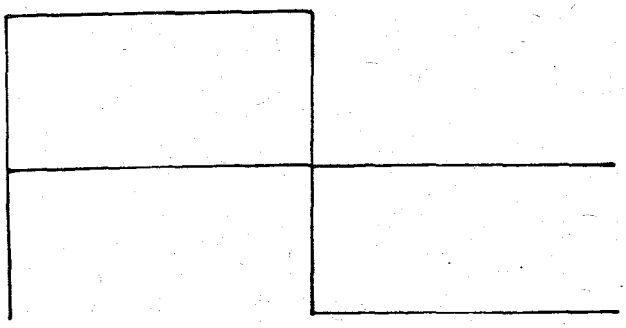
ii) Çözümü ayrıntılı basit parçalara indirgemeyi

iii) Çözümdeki hatayı sistematik olarak ayıklamayı

öğrenebilir.

Değişik yerlerde uygulanmış olan LOGO sistemlerinin pek azında gerçek robot kaplumbağa kullanılmıştır. Pratik nedenlerden ötürü çoğunlukla LOGO dili

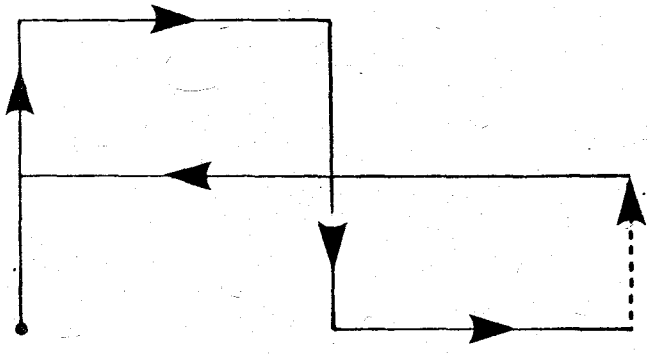
ÇİZİLECEK ŞEKİL



ÇİZİM KOMUTLARI

```
KALEM İNDİR
20 İLERİ
90 SAĞ
20 İLERİ
90 SAĞ
20 İLERİ
90 SOL
20 İLERİ
KALEM KALDIR
90 SOL
10 İLERİ
90 SOL
KALEM İNDİR
```

ÇİZİLİŞ



ÇİZEREK YAPILAN İLERLEME

ÇİZMEDEN YAPILAN İLERLEME

BAŞLANGIÇ NOKTASI

Şekil 1.3 : LOGO çizim örneği.

çizim yetenekli uçlarda (graphics terminal) resim çizmek için kullanılır.

8-12 yaş grubu için çizim yetenekli uçların kullanıldığı çeşitli hareket oyunları da geliştirilmiştir (Tenczar 75). Eğitim felsefesi bakımından LOGO'ya benzeyen bu sistemlerde öğrenci ekranda gördüğü şekilleri basit bir programlama diliyle hareket ettirmeye çalışır. Güdülen amaç aynen LOGO'daki gibi öğrencinin problem çözümleme yeteneklerini geliştirmektir.

1.1.6 Konuşmalı Bilgi Sistemleri

Şimdiye dek incelediğimiz beş BDE yaklaşımını kullanıcıya iletilen bilginin kaynağına göre üç gruba ayırabiliriz

i) Programın erişebileceği ikincil bellekten kaynaklanan bilgi (örnek 1.1.1 ve 1.1.2)

ii) Programı içinde gömülü matematiksel modelden kaynaklanan bilgi (örnek 1.1.3, 1.1.4 ve bkz. 2.bölüm)

iii) Öğrenciden kaynaklanan bilgi (örnek 1.1.5)

Bu üç gruptaki yöntemlerin sağladığı eğitim esnekliği farklıdır. Bilgi eğer matematiksel modelden kaynaklanıyorsa genellikle ders içeriği BDE programıyla bütünleşmekte ve onun ayrılmaz bir parçası olmaktadır; örneğin 1.1.3'deki uygulama yalnız belirli bir savaş uçağı türünün simülasyonudur ve başka savaş uçaklarında kullanılabilmesi için programın değiştirilmesi gerekir. Bilgi eğer öğrenciden kaynaklanıyorsa, kısıtlanmış bir ortamda (örneğin LOGO tipi çizim veya FORTRAN (Balman 80a)) öğrenci, ders içeriğini kendisi sağlayabilmektedir; ancak matematik sel modellemedeki gibi ders konusu programın ayırılmaz bir parçası olmaktadır.

Öğrenciye iletilen bilginin ikincil bellekte saklandığı yaklaşımlarda BDE programlarının işlediği ders içeriğinin esneklik kazandığı belirgindir. Anımsanılacağı gibi ilk iki örnekte (1.1.1 ve 1.1.2) öğrenciye iletilen bilginin tümü bellekte tutulmakta ve öğretmenin öngördüğü koşullar tamamlanınca bu bilgi bellekten alınıp aynen öğrenciye iletilmekteydi. Yani öğrencinin bilgisayar kullanırken karşılaşabileceği alıştırma ve açıklamaların hepsi ders konusundan bağımsız olarak önce-

den doğal dil kullanarak hazırlanıp ikincil belleğe yüklenmiştir. Bu durumda ders içeriğini değiştirmek için sadece ikincil bellekteki bilgiyi değiştirmek yeterlidir. Ancak, özdevimli kitap yaklaşımında da belirtildiği gibi, ders içeriğine verilen esneklik alıştırma ve açıklayıcı metinlerin sayısını kabartmakta ve ders yazılımını güçleştirmektedir. Ayrıca, dersin parçaları bilgisayarda bölünmez kalıplar halinde tutulduğundan, öğrenci belirli bir bilgiyi almak istediğinde kendisine o bilginin bulunduğu bölünmez parça tümüyle verilmektedir.

Özellikle yapay us kökenli BDE araştırmalarında bilgisayarı etkin ve esnek bir eğitici yapabilmek için üç ilke vurgulanmıştır:

i) Bilgi metinleri yerine bellekte salt bilgi özetinin tutulması ve gerektiğinde bilgisayarın bu özeti kullanarak bu metni üretebilmesi

ii) Özet bilginin BDE programına "öğretilebilmesi"; yani verilecek herhangibir metindeki bilgiyi bilgisayarın özetleyebilmesi

iii) Öğrenci-bilgisayar arasındaki iletişimde her iki tarafın da doğal dil kullanabilmesi.

Bu yeteneklere sahip BDE programları geliştirilince öğrencinin dilediği bilgiyi bilgisayarla konuşarak öğrenebileceği savunulabilir. İlgili araştırmalar yapay us alanındaki doğal dil ve anlam ağları (semantic nets) araştırmalarının başarısına bağımlıdır; bu alanda da, çok kısıtlı bilgi ortamlarında elde edilen birkaç başarı dışında, henüz tatmin edici bir uygulama gerçekleştirilememiştir.

Yapay us kökenli konuşmalı bir BDE sistemi Stanford Üniversitesinde gerçekleştirilmiştir (SCHOLAR sistemi (Collins 77)). Coğrafya öğretiminde kullanılan bu sistem öğrenciye değişik ülkelere ilişkin "kuru" bilgi verebilmektedir. Örneğin nüfus, başkent, konuşulan dil, vb. Konuşmalı sistemler arasında başarılı sayılabilecek bu sistem görmüş olduğumuz diğer BDE örneklerinin yanında sönük kalmaktadır.

1.2 Sonuç

Bilgisayar Destekli Eğitimin belki de en çekici özelliği yüksek öğrenci-öğretmen oranının yarattığı sakıncaları giderebilme potansiyelidir. Ne yazık ki bu problemin en belirgin olduğu ilk ve orta eğitim kesimlerinde pek az uygulama gerçekleştirilmiş ve gerçekleştirilen uygulamaların da eğitsel değerini kanıtlama çalışmaları yapılmamıştır.

Kuşkusuz ilk ve orta eğitimde yaygın bilgisayar olanaklarının bulunmaması bu kesimdeki BDE çalışmalarını kısıtlamıştır. Her okula BDE için özel bilgisayar vermek ekonomik bir yaklaşım olamaz. Ancak, 1.1.2'de incelediğimiz yaklaşımda 12 değişik okulun 4000 öğrencisinin aynı merkezi bilgisayardan yararlandığını görmüştük; bu öğrencilerin sağladığı bilgi işlem yükü merkez kapasitesinin sadece %20'sidir. Kuşkusuz bölgesel merkezi BDE'in ekonomik yükü çok daha düşük olacaktır.

İlk ve orta eğitimde BDE kullanımı için gereken büyük başlangıç yatırımına henüz yeterli bir gerekçe gösterilememiştir. BDE araştırmalarında eğitsel değerlendirme çalışmalarına daha çok önem verilmelidir.

BÖLÜM 2

BİLGİSAYAR DESTEKLİ DENEY

Özellikle üniversite mühendislik eğitiminde görülen bu BDE yöntemi matematiksel veya kalitatif davranışı bilimsel olarak saptanabilen herhangi bir laboratuar deneyinin bilgisayarda benzetimidir (Smith 76). Bazı durumlarda BDD uygulamalarının sadece gerçek laboratuar deneylerinin benzetimiyle sınırlı kalmayıp, kuramsal ve hiçbir şekilde gerçek laboratuarda öğrenciye sunulmayacak "deney"leri de içerdiğini görürüz.

Kısaca,deneyde kullanılan araçların değişken ve karakteristiklerinin bilgisayara aktarılmasıyla yapay bir laboratuar ortamı hazırlanmaktadır. Eğer uygulama çevirimiçi ve öğrenciyle deneyin bağlantısı bir bilgisayar ucuysa somut deney aparatının sağladığı dene-gör-düzelt döngüsüne yaklaşık bir ortam sağlanabilmektedir.

2.1 Amaçlar

Üç sebepten ötürü BDD benzetimlerinin gerçek deneylerin yerini alması amaçlanabilir.

i) Tehlikeli ve öğrenci tarafından kullanılması çok sakıncalı araç veya maddelerin bulunduğu deneyler-örneğin radyasyon, reaktivite deneyleri

ii) Pahalı ve sağlanması olanaksız araç gerektiren deneyler-örneğin reaktör fizik, yüksek gerilim deneyleri

iii) Aparat hazırlık veya deney gözlem süreleri çok uzun olduğundan olanaksız deneyler-örneğin dayanırlılık, uzun süreli geribildirim deneyleri.

Öte yandan bir çok deneyler de hem gerçek aparatla hem de bilgisayardaki benzetimle öğrenciye sunulabilir. Kuşkusuz laboratuvarda yapılması mümkün olan bir deneyin bilgisayara aktarılması sakıncalıdır. Bilgisayardaki deneylerin genellikle daha yoğun ve eğitici olmalarına rağmen öğrencinin alışılagelmiş laboratuvardan bu şekilde uzaklaştırılması şüphesiz onun pratik yeteneklerini olumsuz yönde etkiler. Ancak bazı durumlarda, öğrencinin deneyi her iki ortamda da yapması kaydıyla, bilgisayardaki benzetimin eğitim programına katkısı olabilir:

i) Yetersiz araç sayısından ötürü laboratuvar çalışmalarının derslerden çok geride kaldığında

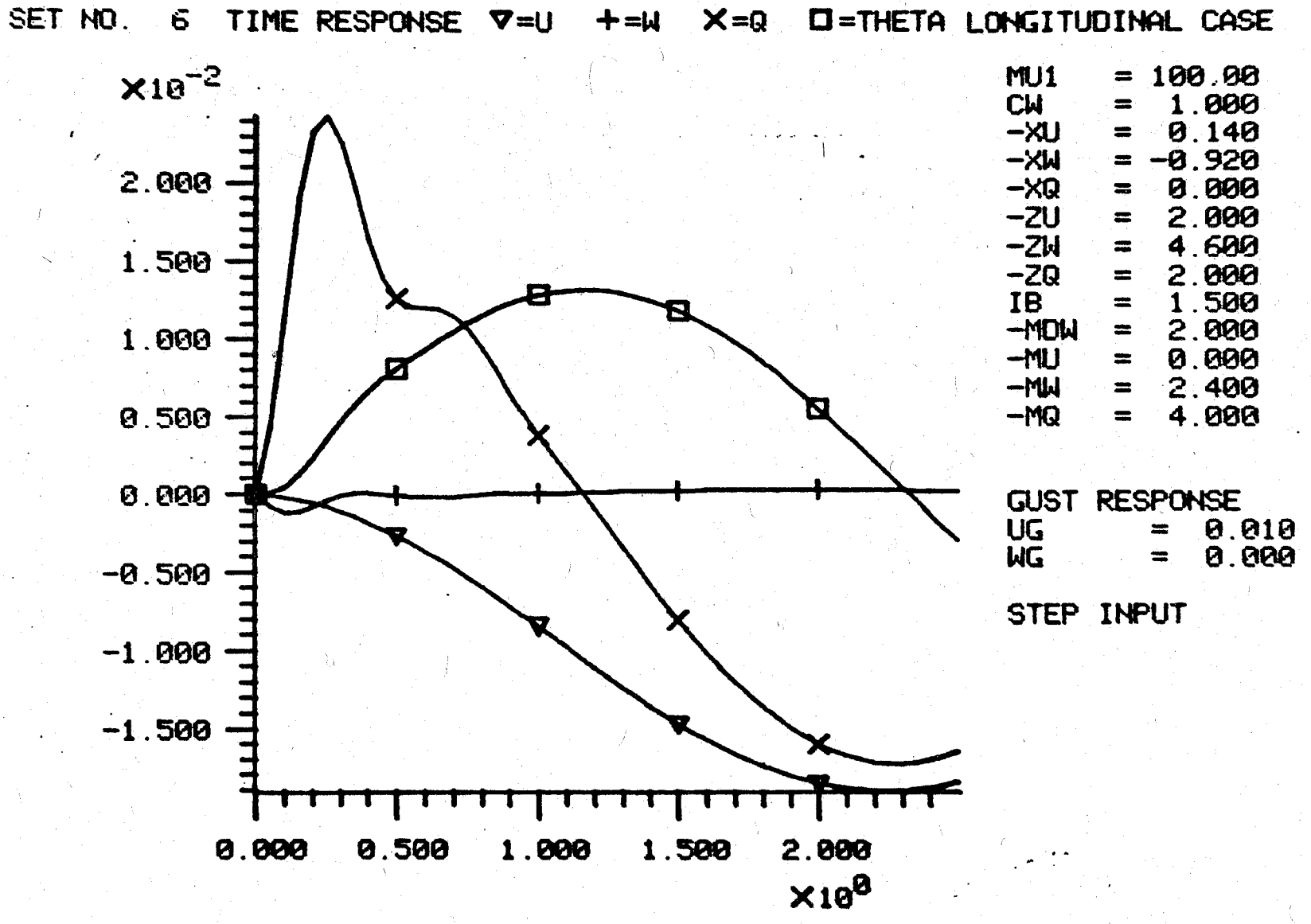
ii) Gerçek aparat deneme ve gözlem süratini önemli ölçüde düşürdüğünde

iii) Gerçek aparat yan tesirleri ve gizli değişkenleri göstermediğinde

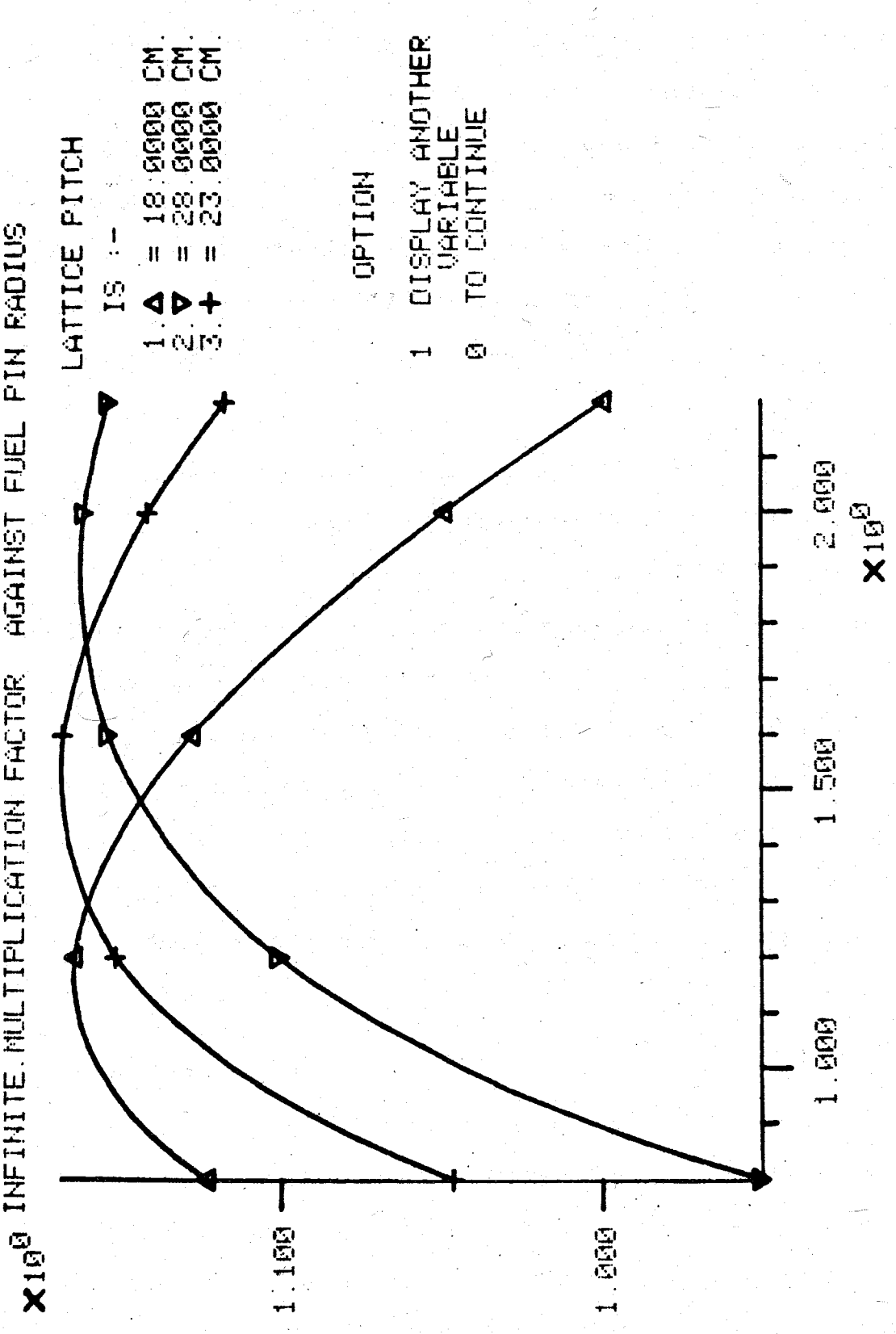
iv) Kuramsal sonuçların deneysel sonuçlarla karşılaştırılmasında

2.2 Öğrenci ve BDD

Daha önce de belirtildiği gibi BDD uygulamaları deneyde kullanılan aparatın karakteristiklerini yansıtacak şekilde tasarlanır. Modellenen sistemin giriş parametrelerini öğrenci saptar; ancak bu değerler modele bilgisayar ucundaki tuşlar kullanılarak verilir. Deney süresince öğrenci parametre değerlerini dilediği gibi değiştirebilir. İzlenmek istenen çıkış parametresi (veya parametreleri) de aynı şekilde bildirildikten sonra bilgisayar deneyin çıkış değerlerini öğrenciye bildirir. Bir çok durumlarda çizim yetenekli uçlardan ve bilgisayarın işlem süratinden yararlanılarak izlenmek istenen parametre öğrenciye graf olarak gösterilebilir (Şekil 2.1). Buna ek olarak değişik graflar, ileride kıyaslanmak üzere, bilgisayar belleğinde veya ikincil bellekte muhafaza edilebilir (Şekil 2.2). Sonuçların kesikli değerler olarak değil de graf olarak verilmesi öğrenciyi rutin değer derleme işleminden kurtarıp ona daha çok deneme ve izleme olanağı sağlar.



Şekil 2.1 : Aerodinamik Bilgisayar Destekli Deneyinden alınan titreşim çizelgeleri.



Şekil 2.2 : Bir reaktör fizik deneyinden alınan üç sonucun kıyaslandığı çizelge.

Uygulamaların çoğunda öğrenciye tanınan deney süresinin bir ile üç saat arasında değiştiğini ve genellikle öğrencilerin ikişer kişilik gruplar halinde çalışmayı tercih ettiklerini görürüz (Balman 77). Deney süresince, gerçek deneylerde olduğu gibi, öğrencilerin yanında bir laboratuvar asistanının bulunmasında yarar vardır; bu bakıcı hem deneyin kullanımına hem de sonuçların yorumuna yardımcı olur. Öğrencinin BDD çalışmalarını yönlendirmek ve denetlemek amacıyla yanıtlanacak sorular verilir ve deney sonunda bir laboratuvar gözlem raporu istenir. Ender olarak yönlendirme ve denetleme görevini BDD programının üstlendiği de görülür.

BDD'in eğitime bir katkısı da ders esnasında kullanılabilmesidir. Dersanedeki bilgisayar ucu veya kapalı devre televizyon (CCTV) ile derste incelenen sistemin karakteristikleri canlı olarak öğrenciye nakledilebilir. Özellikle karmaşık sistemlerin açıklanmasında kullanılabilen bu yöntemin etkin bir eğitim aracı olduğu kanısı yaygındır (Broadhurst 74).

Bilgisayar Destekli Deneyin bir başka özelliği de aynı laboratuvarda çok farklı deneylerin bir arada yapılabilmesi ve deney hazırlık süresinin çok kısa oluşudur. Şöyle ki, her öğrenci istediği deneyi istediği zamanda tekrarlayabilmektedir (Balman 78 b); her deneyin öngörülen süre içinde tamamlanması kaydıyla, öğrenci enterese duyduğu deneyleri daha ayrıntılı ve bağımsız olarak inceleme olanağına kavuşmaktadır.

2.3 Tasarım ve sunuluş

Tüm Bilgisayar Destekli Eğitim uygulamalarında olduğu gibi Bilgisayar Destekli Deneyde de programın eğitsel içeriğini en iyi saptayabilecek kişi uygulamanın kullanılacağı dersin hocasıdır. Fakat, sonuçta programın etkinliğinin büyük ölçüde öğrenciyle bilgisayar arasındaki iletişim ve koordinasyona bağımlı olduğunu görürüz (Balman 79). Modellenen sistemin eğitici gücünü bir yana bırakırsak uygulama safhasında öğrenci-bilgisayar ilişkisini sağlayan iki unsur vardır

i) Donanım ve özellikle çevresel donanım

ii) Yazılım

Deneyin yapılış ortamını sağlayan bu iki unsurun öğrenciyi etkileyiş şekli aşağıdaki bölümlerde özetlenmektedir.

2.3.1 Donanım

Öğrenci bilgisayardaki deney benzetimini ancak değişkenlere verdiği değerlerle etkileyebilir. Bilgisayar ise deney sonuçlarıyla ilgili bilgiyi program çıktısı olarak iletebilmektedir. Kuşkusuz çevresel donanımın bu bilgi alışverindeki etkisi ve dolayısıyla deneyin eğitim değerine katkısı küçümsenemez. Bazı araştırma merkezlerinde insan-bilgisayar bağlantısını kolaylaştıran uçlar geliştirilirken (Johnson 71, Agajanian 74) birçok BDD uygulamalarında geleneksel elektromekanik (teleprinter tipi), ekran (VDU) veya çizim yetenekli ekran (graphics terminal) kullanıldığını görürüz. Bu üç değişik tipteki bilgisayar ucu arasında en uygun iletişim ortamını seçmek güçtür. Ekonomik koşulları bir yana bırakırsak BDD açısından seçim kriterleri şunlardır

i) Sonuçların süratli bildirimi

ii) Sonuçların grafik veya şekil olarak bildirimi

iii) Sonuçların taşınabilir şekilde bildirimi

Özellikle uzun metinler veya tablolar öğrenciye sunulacaksa sürat kriteri önemlidir Böyle durumlarda elektromekanik uçlar deney süratini ve öğrenci sabrını belirgin bir şekilde etkilemektedir. Ekranlar bilgi bildirim işlemini çok daha süratli yapabildikleri halde öğrenciye ilettikleri bilgi geçici olduğundan sakıncalı görülebilir; eğer öğrenci kendisine ekran aracılığıyla iletilen bilgiyi ileride kullanacaksa bu bilgiyi kâğıt üzerine aktarmak zorundadır. Öte yandan, özellikle mühendislik deneylerinde, sonuçların en etkin ve çarpıcı biçimde bildirimi graf şeklindedir. Bir çok durumlarda adi ekran veya elektromekanik uçlarda verilebilen kesikli graflar yetersiz olduğundan en uygun ortam çizim yetenekli uçlardır; bu tür ekranda verilen bilgi ne yazık ki geçici niteliktedir.

BDD uygulamalarında bu üç tip bilgisayar ucu arasında bir seçim yapmak güçtür. Bazı araştırmacılar her deney istasyonunda hem elektromekanik hem de çizim yetenekli ekran kullanmışlarsa da (Smith 75) bu çözüm ekonomik değildir. Daha uygun bir çözüm, öğrencinin ekrandaki bilgiyi, seçenekli olarak, paylaşılan süratli yazıcı kuyruğuna göndermesidir (Balman 78d).

Önceden de belirtildiği gibi BDD uygulamalarında başarılı eğitim için önemli unsurlardan biri öğrenciye verilen servisin kalitesi ve süratidir. Bilgisayar donanımının seçiminde ve işlem yükünün tasarımında özellikle sürat faktörüne ağırlık verilmelidir. Bir çok bilgisayar kullanıcısının tahammül edebildiği tepki süresinin (response time) BDD kullanıcısı için yetersiz olduğu ve eğitim kalitesini düşürdüğü belirgindir (Balman 77, 78c). Gerçek deneyde aparattan anında tepki görmeye alışkın öğrenci aynı hizmeti bilgisayar ucundan da bekler. BDD ortamındaki izlenimlerde öğrencinin sabrını geciken neticelerden fazla tepki göstermeyen bilgisayar ucunun taşırdığı görülmüştür. "Ölü" gözüken uç aynı zamanda öğrencinin BDD uygulamasına olan güvenini sarsmaktadır.

2.3.2 BDD yazılımı

Kuşkusuz BDD yazılım yöntemlerinin (özellikle bilgisayarla öğrenci arasındaki bağlantıyı sağlamakta kullanılan yazılım tekniğinin) eğitim değerini etkilemesi doğaldır. Bu yöntemleri saptamadan önce kullanıcının yeteneklerini ve bilgisayar deneyimini incelemek gerekir.

Öğrenci programlama dili bilse bile, deney süresinde program yazmasını ve hata ayıklamasını istemek BDD felsefesine aykırıdır. Kullanıcının uygulamanın içerdiği modelden en iyi şekilde yararlanabilmesi için deneyi bilgisayardan mümkün mertebe soyutlamak gerekir. Bundan ötürü bilgisayar programında kullanılan uygulama dilinin giriş desenlerinden ve hata mesajlarından arındırılması amaçlanır. Yani programın öğrenciden alacağı sayısal veri serbest desenli alfabetik giriş olarak okunur ve tüm giriş hatalarının analizi programlama dilinin bünyesinde değil de BDD programı tarafından yapılır (Balman 78d).

Modeldeki giriş değerlerinden başka öğrenciden beklenilen bir diğer veri türü de modelin manipülasyonu için gereken komutlardır-örneğin değiştirmek istediği parametre, görmek istediği çıkış parametresi, tercih ettiği graf vb. Bazı deneylerdeki komut sayısı elliyi aşmaktadır. Deney manipülasyonu için gereken öğrenci-bilgisayar arasındaki "konuşma"nın iki şekilde sağlandığını görürüz

i) Her komut için ayrı anahtar sözcük kullanımı

ii) O anda geçerli olan komutların numaralanmış listesinin bilgisayarca belirtilmesi ve öğrencinin istediği komutu sayıyla (veya ışıklı kalemle) bildirmesi

Anahtar sözcük kullanımının iki önemli sakıncası vardır.

i) Komut sayısı arttıkça öğrencinin anahtar sözcükleri ezberlemesi güçleşir ve deney kullanımını kuşkusuz yavaşlatacak "sözlük" kullanma zorunluğu doğar.

ii) Öğrenci program yazmamış veya daktilo kullanmamışsa deney süresini tuşları aramakla geçirir.

Deney süresince kullanılabilecek geçerli komutlar dinamik olarak değiştiği için ikinci yöntemle öğrencinin seçtiği komutları program daha etkin bir şekilde kısıtlayıp denetleyebilmektedir. Rakam tuşlarının yerleri daha kolay öğrenildiğinden ve her karar noktasında öğrenciye sadece geçerli komut listesi gösterildiğinden birinci yaklaşımda belirtilen dezavantajlar bu yöntem için geçersizdir. Ancak, seçenek listesinin gönderildiği uç süratli bir ekran değilse, tekrarlanan listeler deney süratini önemli ölçüde yavaşlattığından bu yöntem "anahtar sözcük" yönteminden daha sakıncalı olabilir.

Öğrencinin uygulamaya olan güvenini sarsmamak ve deneyin sık sık yarıda kesilmesini önlemek için uygulamanın deney süresince dayanıklı olması gerekir. Kasıtlı veya kasıtsız veri kurcalamalarını program erken teşhis edebilmeli ve öğrenciye hatasını anlaşabilir mesajlarla bildirip güvenirliğini koruyabilmelidir.

2.4 Değerlendirme

BDE uygulamalarıyla daha "ekonomik" ve daha "iyi" eğitim amaçlandığını söylemiştik. Bu idealler çerçevesinde BDD uygulamalarının başarısını şu şekilde ölçebiliriz.

i) Donanım maliyetini gerçek aparat maliyetiyle karşılaştırarak

ii) Uygulama programlarının geliştirme ve bakım maliyetini gerçek aparatınkıyla karşılaştırarak

iii) BDD programlarının öğrenciyi nasıl etkilediğini inceleyerek

2.4.1 Donanım maliyeti

Genellikle Bilgisayar Destekli Eğitim uygulamalarının mevcut bir bilgisayar sistemine uç bağlayıp zaman kiralayarak gerçekleştirildiğini görürüz. Çevirimiçi uç kullanımı desteklemekte olan bir üniversitede bu yaklaşımın gerçek maliyeti önemsenemeyecek kadar düşüktür. Ne var ki eğitim açısından bu yaklaşımın sakıncaları olabilir

i) Mevcut uçlar genellikle çizim yetenekli olmadığından neticelerin öğrenciye sunuluş şekli etkili olamamaktadır

ii) Değişik kullanıcıların paylaştığı merkezi bilgisayar sistemi fark gözetmeksizin bütün kullanıcılara eşit servis vermeyi amaçlar. BDD programları bir laboratuar servisi vermeyi amaçladığından belirli zaman koşulları çerçevesinde ve gözetim altında öğrenciye sunulmalıdır

iii) Çok kullanıcılı sistemlerdeki bilgisayar tepki zaman (response time) yetersiz olmaktadır

Bu sakıncalar çizim yetenekli bilgisayar uçlarının özel bir BDD laboratuarında barındırılmasıyla kısmen bertaraf edilebilir. Fiatı ekrandaki nokta yoğunluğu ve sayısına göre değişen çizim yetenekli ucun maliyetini $ 6000 olarak alırsak, her ucun günde üçer saatlik üç deneyde ve senede 180 gün kullanıldığı varsayarsak, her deneyin donanım maliyeti beş senelik bir sürede yaklaşık $ 2 olmaktadır.

BDD'in ciddi bir eğitim aracı olarak icra edildiği kuruluşlarda bu amaçla 6-12 uçlu minibilgisayar kullanılabilir. Donanım maliyetini fazla kabartmayan bu yaklaşım sayesinde öğrencilere her an kullanılmaya hazır ve sorularına çok kısa sürede tepki gösterebilen bir BDD laboratuarı sağlanabilir. 6 çizim uçlu minibilgisayarın maliyetini $ 80.000 olarak alırsak beş senelik bir sürede yapılan her deneyin donanım maliyeti yaklaşık $ 5'dir. Böyle bir donanımın kapasitesi günde üçer saatlik olmak üzere on sekiz deneydir.

BDD maliyetini gerçek aparat maliyetiyle kesin bir şekilde kıyaslamak güçtür. Bilgisayarda simüle edilebilecek bir çok deneyin tehlikeli veya pahalı olduğu gerekçesiyle gerçek aparatla yapılması söz konusu değildir. Gerçek aparatın kullanımı mümkün olsa bile bilgisayarın sağladığı deney esnekliği olmadığından sayıca kısıtlı deneyler için geçerlidir; yani bir çizim ucunun birbirinden farklı bir çok deneyde kullanılabilmesine karşın (örneğin hem aerodinamik hem reaktör fizik) gerçek aparat sadece geçerli oduğu deneylerde kullanılıp uzun müddet boş durur.

2.4.2 Geliştirme ve bakım maliyeti

Deney geliştirme maliyetinin donanım maliyetinden daha yüksek olduğu görülür. Her deneyin geliştirme çalışması 4-5 ay sürer (ESPCAL 77, Bitzer 77, Fielden 74); yani ortalama deney maliyetini 150.000 T.L. olarak alabliriz. Deney eğer senede 20 kez kullanılıyorsa beş senelik bir sürede her deneyin maliyeti 1.500 T.L. olur. Bilgisayara aktarılan deneyin eskiyip bozulması söz konusu olmadığından normal şartlar altında geliştirilen deneylerin ömrü beş seneyi aşar. Uygulama çalışır duruma geldikten sonra bakım maliyeti önemsenemeyecek kadar düşüktür.

Deney yazılım maliyeti üniversiteler arası deney takasıyla önemli ölçüde düşürülebilir. İngiltere'de NDPCAL gelişme programı (National Development Program for Computer Assisted Learning) çerçevesinde denenen bu yöntem (CET

NDPCAL 75, Edmonds 75) bir dereceye kadar başarılı olmuşsa da takas işlemleri kolay olmamıştır. Akademik anlaşmazlıklar ve bilgisayarlar arası uyumsuzluk takas işlemini güçleştirmiştir. Bunun başlıca nedenleri

i) Uygulamada kullanılan programlama dilinin yaygın standarda uymaması

ii) Sayısal altrutinlerde yaygın bir standard olmaması

iv) Kütük yönetim altrutinlerinde yaygın bir standard olmaması

v) Her kuruluşun, doğal olarak, kendi çevresel donanımına uygun deneyler hazırlaması

Aslında bu beş maddenin hiçbiri tam bir dayanışma içinde olan kuruluşlar arasında giderilmesi imkânsız engeller değildir (bkz. Bölüm 3).

2.4.3 Eğitsel değer

Bir çok BDE uygulamasında olduğu gibi Bilgisayar Destekli Deneylerin eğitsel değeri üç şekilde ölçülmüştür (Mc. Intosh 74)

i) Öğrenci başarısına göre

ii) Öğrenciler arasında yapılan anketlere göre

iii) Öğretmenlerin izlenimlerine göre

Her üç ölçekle olumlu sonuçlar alınmışsa da başarı göstergesi deneyin konusuna kullanılan giriş ve çıkış değişken birleşimlerine, ve öğrenci-bilgisayar iletişim yöntemine göre değişmektedir. Alınan sonuçlar BDD'in başarılı bir yaklaşım olduğunu kesinlikle göstermiştir ama aynı zamanda başarı derecesinin yazılım yöntemlerine olan bağımlılığını da vurgulamıştır.

Öğrenci başarısına göre BDD değerlendirmesi sınıfın yarısına BDD kullandırmayıp uygulamanın kapsadığı model hakkında bilgilerini ölçerek elde edilmiştir. Kuşkusuz gerçek aparatın var olamdığı durumlarda BDD kullanılması çok daha başarılı olmuştur. Bu sonuç şaşırtıcı olmasa bile en azından BDD uygulamasının kullanıcıya zarar değil yarar sağladığını kanıtlamıştır.

İkinci ve üçüncü değerlendirme yöntemleriyle de benzer sonuçlar alınmıştır. Bu yöntemlerle deney bitiminde kullanıcıya verilen soru kâğıdıyla deney hakkındaki izlenimler derlenmektedir. Tipik anket soruları ve alınan cevaplar ESPCAL 77, Bitzer 77'de verilmiştir.

2.5 Sonuç

BDD tehlikeli ve pahalı deneylerde uygulandığında en az eleştiriye uğrayan BDE yöntemidir. Normal şartlar altında yapılamayacak ve dolayısıyla eğitimde boşluk yaratacak deneyleri ekonomik olarak sağladıklarından, bu tür programların yararı kaçınılmaz bir gerçektir. Ancak, BDD yöntemi Bitzer 77'deki gibi lise düzeyindeki basit deneylere uygulandığı zaman, öğrenciler pratik yeteneklerini yeterince geliştiremez.

BÖLÜM 3

PROGRAM TRANSFERİ

Bilgisayar Destekli Eğitimde program transferi yazılım maliyetini önemli ölçüde düşürebilir. Akademik uyum içinde olan iki kuruluş arasında yapılacak iş bölümü ve program takası yazılım maliyetini yarıya indirir.

Maliyet bir yana, BDE'in önemli hedeflerinin biri de yaygınlaşma olduğuna göre (Edmonds 75) program taşınabilirliği kritik bir araştırma konusudur.

Taşınabilirlik özellikle üniversiteler arası BDD takasında gündeme gelmiş ve halen tatmin edici bir çözüm bulunamamıştır . BDD uygulamalarının öncülerinden Computer Assisted Teaching Unit'te program takası için iki değişik yaklaşım kullanılmıştır:

i) Aynı deneylere gereksinme duyan üniversiteler arasında iş bölümü yapılması ve hazırlanan programların koşulsuz bire bir takası. 1974-1976 yıllarında denenen bu yöntemin başarısı tatmin edici olmamıştır. Ortalama olarak üretilen programların ancak %25'i transfer edilebilmiştir.

ii) 1976 senesinde aynı kuruluş Engineering Science Programme Exchange adıyla bir BDD bankası kurdu. Herhangi bir üniversite dilediği BDD programını hibe ederek karşılığında bir program seçebilir. Ayrıca bankaya hibe edilen programın her transferi için bir program çekmeye hak kazanılır. Değişik ülkelerden 22 üniversitenin katıldığı bu takas sistemiyle sürekli program alışverişi yapıldığı halde çoğu kez üye üniversiteler kendi programlarını geliştirmeyi yeğledikleri belirtmişlerdir.

Her iki yaklaşımdaki iyi niyet ve kooperasyon gerek teknik gerekse akademik engeller yüzünden pek başarılı olamamıştır. Karşılaşılan ve çözümlenmesi gereken engeller bu bölümde incelenmektedir.

3.1 Programlama dili seçimi

Yazılan programın taşınabilmesi için kullanılan programlama dilinin

. yaygın olması

. kabullenmiş yaygın bir standartının olması

şarttır. Nümerik uygulamalara elverişli en yaygın diller FORTRAN ve BASIC'tir. Bu dillerin en büyük avantajı sözü edilebilecek her bilgisayar firmasının her iki dili de desteklemesidir.

BASIC dili transferabilite açısından incelendiğinde iki önemli sakıncası görülür

i) Bu dilin yaygın bir standardı yoktur. Kabul edilen "Pure BASIC" çok kısıtlı bir dil olduğundan bazı bilgisayar firmaları "Basic Plus " kisvesi altında birbirinden farklı diller üretmişlerdir. Özellikle matris manipülasyon komutlarında, ve standard fonksiyonlarda uyumsuzluk vardır. Kütük düzenleme ve girdi /çıktı kanallarına ilişkin komutlar da çok farklıdır.

ii) Yapısal olarak BASIC büyük ve modüler program yazılımına elverişli değildir. Her altrutinin paylaştığı bir tanıtıcı boşluğu kullanan BASIC ile modülerleşme güçtür. Altrutinlere argüman bildirimi ancak adama komutlarıyla gerçekleştirilebildiğinden, özellikle parametrenin matris olduğu zamanlarda, gereksiz bellek harcanmaktadır. Bu durumda programlar ve kuruluşlar arası ortaklaşa kullanılacak nümerik ve girdi/çıktı modüllerinin değişken ve komut işaretlerini ayarlamak dikkatli plânlama ve sıkı bir işbirliği gerektirir.

FORTRAN dilinin BASIC'e oranla taşınabilme potansiyeli daha yüksektir. Değişik makinalarda uygulanan FORTRAN'lar birbirlerinden farklı olabilir ama asgari müşterek FORTRAN IV'de birleşir. Saf BASIC'in aksine bu ortak nüve bilimsel programlamaya elverişlidir. FORTRAN IV'ün BASIC'ten üstün olan niteliklerinden biri de parametre stratejisinin mödülerleşmeyi mümkün kılmasıdır. Bu dille gereksinme duyulan nümerik ve girdi/çıktı rutinleri hazırlanıp sistem modül repertuvarına eklenebilir ve kuruluşlar arası modül standarlaşmasına gidilebilir.

FORTRAN IV, program taşımak için en uygun dil olmasına rağmen, yine de bazı hususlarda yetersizdir.

3.1.1 Karakter manipülasyonu

FORTRAN IV'ün en zayıf noktalarından biri olan karakter kod manipülasyonu program taşınabilirliğini etkilemektedir. Karakterlerin bellekte saklanış şeklini sağlayan A-deseninin (A-format) azami karakter/kelime sayısı bilgisayarın kelime uzunluğu ve kodlama sistemine göre değişir. Burada azami müşterek karakter/kelime sayısını ancak A2 deseninin sağladığını görürüz. Standard FORTRAN kullanılarak girdi tamponunda karakter manipülasyonu yapılacaksa A1 kullanımı şarttır. Bu yöntem de bellek kullanım etkinliğini düşürdüğünden çoğu kez programcılar yüksek karakter/kelime sayısıyla standard olmayan karakter manipülasyon fonksiyonlarını kullanmayı tercih ederler.

3.1.2 Kütük kullanımı

Kütük yaratma ve erişim komutlarının bilgisayar sistemine göre değiştiği görülür. Sistemler arası ortak kütük kullanım nitelikleri tüm kayıtların kart görüntüsünde olduğu ve kayıtların sırasal olarak normal READ komutuyla erişilebileceğidir. Bu iki nitelik dışında bir çok önemli kütük düzenleme ve denetleme komutları kullanılan bilgisayar sistemine bağımlıdır; örneğin,

- kanal seçim komutları
- Kütük koruma bilgisi-Oku, oku/yaz,oku/yaz/genişlet vs.
- Kütük isimlerinin ve birikim ortamının tanımlanması
- Kütük açma parolası-kısıtlı parolalar vs.
- Kütük paylaşımı
- Aynı anda açılabilecek kütük sayısı
- Geçici kütük kullanımı
- Dolaysız erişim
- Değişken uzunluklu kayıtlar
- İkili bilgi kütükleri
- Dolaysız erişim

Genellikle Bilgisayar Destekli Eğitimde hem öğrenciyi denetlemek hem de programa gerekli olan girdi ve geçici bilgi birikim ortamını sağlayabilmek için kütük kullanmak

gerekir. Bilgisayarlar arası kütük kullanım uyumunun olmaması program transferini önemli ölçüde baltalamıştır.

3.1.3 İşlem hata mesajları

Öğrenci-program iletişimini bilgisayardan mümkün mertebe soyutlamak amaçlandığından, program kendi işlem hatalarını donanımdan bağımsız olarak farkedilebilmeli ve, yazılım dilinden bağımsız olarak, anlaşılır (ders içeriği ile ilgili) hata mesajları verebilmelidir. Uzun süre aktif kalabilmesi gereken Bilgisayar Destekli Eğitim programının işlem hatasından ötürü yarıda kesilmesi öğrenciyi negatif olarak etkiler.

İdeal çözüm hataya geçilmeden önce programın hatayı farkedip kontrolü sistemin hata işlemcisine kaptırmamasıdır. Kuşkusuz program içerisinde sürekli olarak gelebilecek hatalı işlemleri kontrol etmek işlem süratini önemli ölçüde düşürür. Daha etkin bir önlem tüm program girdilerinin işleme geçilmeden program içerisinde incelenmesidir; ancak, özellikle BDD uygulamalarında, girdi değerlerindeki hatayı işlemden önce farketmek olanak dışıdır. Birçok gerçek zamanlı işletim sisteminde görülen zaman uyumlu hata tuzakları (synchronous traps) en yaygın ve en etkin hata önleme yöntemidir. Ne yazık ki, bu yöntem standardlaşmadığından, değişik sistemlerde çok farklı kullanım komutları vardır.

3.1.4 İşlem doğruluğu ve hataya tolerans

Değişik sistemlerin sağladığı sayı doğruluğu (gerek REAL gerek DOUBLE PRECISION) ve kabul edilen sayı büyüklüğü değişmekte ve sistemler arası transferlerde beklenmedik işlem hataları görülmektedir. Bundan daha ciddi bir ayrıcalık ise değişik FORTRAN sistemlerinin aynı hataya gösterdikleri farklı duyarlılıktır; örneğin

- üst taşma hatalarında görülen farklı hassasiyet: bazı sistemlerde koşulsuz olarak program öldürülürken bazılarında cevabı sıfırlayarak veya işlemi yapmayarak hatanın örtbas edilmesi
- dizi erişiminde endekslenme hatalarına hassasiyet: bazı sistemlerde hata mesajı, bazılarında bellek dışına taşmamak koşuluyla hatalı erişim,

bazılarında da endeksi geri sarma

. hesaplanmış GOTO komutunda hesaplanan endeksin belirtilen komut işaret listesinde karşılığının bulunmaması: bazı sistemler hata mesajı verirken diğerleri bir sonraki komuta geçer.

Görüldüğü gibi bazı FORTRAN sistemleri bir kısım işlem hatalarını örtbas edip program yazıcıya iletmemektedir. Bu yüzden normal çalışır gibi görünen programdaki gizli hatalar ancak belirli girdi kombinasyonlarıyla ortaya çıkabilir. Eğer hatayı gösteren girdi kombinasyonu bulunmazsa görünürde sağlam olan program gizli hatalarıyla birlikte transfer edilebilir. Programın transfer edildiği sistem aynı hataya tolarans göstermeyebilir.

3.1.5 Leksik uyumsuzluklar

Sistemler arasında üç önemli leksik ayrıcalık vardır. Bunlar

. bazı sistemlerde komutların 72'inci sütunu aşabilmesi

. farklı devam kartı kuralları

. bazı sistemlerde "tab" karakterinin (CTRL I) 6 boşluk olarak yorumlanması

Program transferini güçleştirebilen bu leksik farklar bir ön-işlemciyle giderilebilecek niteliktedir.

3.1.6 Diğer uyumsuzluklar

FORTRAN dilinin tanımında belirtilmeyen bir takım kısıtlamalar ve varyasyonlar değişik bilgisayarlarda farklı yorumlanmıştır. Bunların arasında derleyicinin empoze ettiği kısıtlamaları şöyle sıralayabiliriz

. Derleyiciye ayrılan geçici belleğe bağımlı olarak aritmetik adama, hesaplanmış GOTO, FORMAT vs komutlarına tanınabilen karmaşıklık derecesi.

. Yine geçici belleğe bağımlı olarak, izin verilen modül büyüklüğüyle parametre sayısı.

. bazı derleyicilerin FORTRAN standard fonksiyon isimleriyle komut anahtar sözcüklerinin değişken tanıtıcı olarak kullanılmalarına izin vermemesi

Derleyiciye bağımlı olarak sistem bağlantı programında ve işlem zamanında kullanılan FORTRAN fonksiyonlarında da bir takım farklı uygulamalar program davranışını etkileyebilmektedir

. standard trigonometrik fonksiyonlarda (SIN, COS) eksi açıların bazı FORTRAN işlem zamanı modüllerinde hata olarak tanımlanması
. standard olmayan (dış) fonksiyonlarda isim farkları (ASIN, ARSIN, ARCSIN vs) veya daha önemli olarak tanımlanmayan fonksiyonlar
. hesaplanmış GOTO komutlarında, etiket endeksi birden küçük veya etiket sayısından büyük olduğu zamanlarda görülen farklı uygulamalar. Örneğin, hata, ilk veya son etiketin seçimi, bir sonraki komut.

3.2. Sayısal Çözümleme Altrutinleri .

Sayısal çözümleme rutinleri özellikle BDD uygulamalarında gerekir. Genellikle bu tür altrutinler bilgisayar sisteminin modül kitaplığında kullanılmaya hazır biçimde durduğundan programlayıcının uygulamada sistem kitaplığından yararlanmak istemesi doğaldır. Ancak her bilgisayar firması özel olarak patentli modül geliştirdiğinden ve genellikle bu modüller bağlanmaya hazır amaç kodu olduğundan başka bir bilgisayara transferleri söz konusu değildir.

Değişik bilgisayar firmalarının kendi sistemleri için hazırladıkları altrutinlerde yalnızca farklı rutin isimleri değil aynı zamanda farklı giriş/cevap parametre birleşimleri ve farklı hesaplama yöntemleri kullanılır. Herhangi bir standarttan yoksun olan bu altrutinlerde programlayıcı açısından güçlük yaratan önemli farkları şöyle sıralayabiliriz:

. işlemi yapan altrutinlerin konrolü; cevap doğruluğunun veya döngü sayısının girişte belirlenebilmesi.

. hata kontrolü; işlem süresinde altrutinin farkedip iletebileceği hata çeşitleri ve bu hataların kullanıcıya iletiliş şekli

. altrutinin istediği girdi/çıktı ve özellikle geçici işlem alanı parametreleri

. altrutin argümanlarının nasıl düzenlenmesi gerektiği; örneğin aynı giriş/cevap dizisi, farklı giriş/cevap dizileri, giriş dizisinin ters organizsyonu

Kuşkusuz, kullanılan hazır modül program tasarım şeklini etkiler. Karşılaşılan problem yalnızca sistemdeki hazır modülün transferinin imkânsızlığı değil, aynı zamanda başka kuruluşlarda bu modül yerine kullanılabilecek benzer modül olsa bile, program yapısına uymayabileceğidir. Kaldı ki bir çok mini-bilgisayar firması sayısal çözümleme paketinden yoksundur.

Program taşınabilirliğini etkileyen bu probleme iki çözüm önerilebilir:

i) Sistemden bağımsız olarak paket program geliştiren ticari kuruluşlardan sayısal çözümleme kitaplığı satın almak.

Dezavantaj: Yazılım fiatları çok yüksektir.

ii) Gereksinme duyulan sayısal rutin BDD programını hazırlayan kuruluş tarafından taşınabilir biçimde geliştirilmesi.

Dezavantaj: BDD programını hazıralayan kuruluşun sistem kitaplığında gereksinme duyulan modül varsa bu yöntemi uygulamak güçtür.

3.3. Giriş/çıkış yöntemleri

BDE programlarında eğitim kalitesini en çok etkileyen unsur programda kullanılan giriş/çıkış stratejisidir (Balman 79). Öğrenci ile bilgisayar arasındaki iletişimi sağlayan giriş/çıkış yöntemleri doğal olarak programın öğrenci üzerinde yarattığı tesirde önemli bir rol oynar.

Program transferinde karşılaşılan güçlüklerin biri de program tasarımının çevresel donanıma büyük ölçüde bağımlı olmasıdır.

Giriş/çıkış yöntemlerinin program transferinde doğurduğu iki önemli engel

görülmektedir:

i)Giriş/çıkış yöntemlerinin çevresel donanıma büyük ölçüde bağımlı oluşu

ii)Çizi altrutinlerinde standard olmayışı

Her iki engel de değişik kuruluşlar farklı çevresel donanım kullanıldığında belirgindir. Birinci engel, her kuruluşun haklı olarak kendi çevresel donanımını en iyi şekilde kullanmak istemesinden doğar. Bu durumda transfer edilen program yabancı donanımda etkinliğini kaybedebilir. İkinci engel, farklı fiziksel çizi altrutinleri ve (daha önemli olarak) farklı mantıksal çizi altrutinleridir. Sayısal altrutinlerde olduğu gibi mantıksal çizi altrutinlerinin de program yapısını etkilediği belirgindir.

3.3.1 Çevresel donanım ve giriş/çıkış yöntemleri

Çevirim içi BDE uygulamalarında kullanılan uçlar hem sürat, hem yetenek bakımından birbirlerinden çok farklıdır. Doğal olarak her kuruluş kendi çevresel donanımını en iyi şekilde kullanacak giriş/çıkış yöntemlerini benimser. Bu sebepten geliştirilen programlar farklı giriş/çıkış yöntemi benimsemiş olan kuruluşlara güçlükle transfer edilebilir.

BDE'de yaygın olarak kullanılan uçlar ve bunların karakteristikleri aşağıda verilmektedir.

Tele-daktilo tipi elektomekanik uçlar

Giriş tuşlarladır. Sürat 50 ile 1200 baud arasında değişir. Öğrenciye iletilen bilgi kesikli graf veya bar-graf olarak verilebilir ama detaylı çizimler olanak dışıdır. Özellikle 50 ve 110 baud teledaktilolarda öğrenciye verilen bilgi az ve öz olmalıdır. Uzun metin ve cevap tablolarının dökümü zaman kaybına yol açtığından bu tür program çıkışları mümkün olduğu kadar seyrek olmalıdır. Öğrenciye iletilen bilgi kâğıt üzerine kaydedildiğinden öğrenci açısından bilgi kaybı hiç yoktur.

Grafik ve uzun metin çıkışlarının kullanıldığı BDE programları tele-daktilo

kullanan kuruluşlara adapte edildiğinde program etkinliğini kaybedebilir. Ayrıca metnin geçici bir sürede öğrenciye gösterileceği uygulamalarda (örneğin özdevimli kitap) tele-daktilo kullanılmaz.

Ekran (VDU) tipi uçlar

Giriş tuşlarladır. Yazış sürati 50 ile 8.600 baud arasında değişir ama genellikle tele-daktilodan çok daha süratlidir. Bu yüzden uzun metin ve cevap tablolarının dökümüne elverişlidir. Ancak, program çıkışları kâğıt üzerine aktarılmadığından iletilen bilgi geçicidir. Ayrıca değişik ekranların satır kapasitesi farklı olabildiğinden (20-64) iki sakınca belirgindir.

i) Ekrana yazılan bilgi geçici bir süre içinde okunabilmektedir

ii) İletilen metin (veya cevap tablosu) ekrana sığmayacaksa bilgi parçalara ayrılmalıdır. Parça uzunlukları da ekran kapasitesine göre ayarlanmalıdır.

Tele-daktiloda olduğu gibi kısıtlı olarak graf çizilebilir; ancak, dikey koordinatlar ekran satır kapasitesine göre ölçeklenmelidir.

Grafik ve kalıcı metin/cevap çıkışları olan BDE programlarının bu tür uçlara aktarılması güçtür. Farklı ekran satır kapasiteleri öğrenciye iletilen tüm bilginin yeniden düzenlenmesini gerektirir. Eğer seçenekli olarak cevapların kâğıda dökümünü sağlayacak yardımcı elektromekanik aygıtlar yoksa öğrenci BDE kullanımı süresinde not tutmak zorundadır.

Çizi yetenekli birikici ekran (storage tube)

Giriş tuşlarla veya "cursor" ile olabilir. Yazış sürati 1200 ile 19.200 baud arası değişir. Ekranın çizi yeteneği adreslenebilecek nokta sayısı ve nokta yoğunluğuna göre değişir. Nokta sayısı 40.000-1.000.000 arasında, nokta yoğunluğu ise her milimetre kareye 1-16 nokta arasında değişebilir. Nokta yoğunluğu yüksek olan ekranlarda çizilebilen çok ayrıntılı şekiller, düşük yoğunluklu ekranda çizilirse kuşkusuz önemli bilgi kaybı olur.

Birikici ekranların (storage tube) önemli karakteristiklerinden birisi ekrandaki bilginin yazıldıktan sonra erişilememesidir. Bu sebepten ötürü şekil hareketleri ve modifikasyonu olanak dışıdır.

Normal ekranda olduğu gibi, çizi yetenekli ekrana yazılan tüm bilgi geçicidir; ekran satır kapasiteleri de farklı olabilir. Bu sebepten ötürü ekranlı uçlardaki transfer problemleri çizi yetenekli ekranlar için de geçerlidir. Ayrıca, nokta sayısı ve nokta yoğunluğu farklılıkları, şekil modifikasyon ve hareketleri, tuş ve imleç (cursor) dışındaki giriş yöntemleri transferleri güçleçtirici niteliktedir.

Çizi yetenekli yenilenen ekran (refresh tube)

Giriş tuşlarla, ışık kalemiyle (light-pen) veya özel fonksiyon tuşlarıyla (special function keys) olabilir. Yazış sürati, nokta sayısı, ve nokta yoğunluğu birikici ekrana benzer olarak değişir. Ekrandaki bilgi sürekli olarak bellekten yenilendiği için, çizilen şekiller hareketlendirilebilir ve değiştirilebilir. Ancak iletilen bilgi geçicidir ve bu karakteristiğin doğurduğu tüm sakıncalar yenilenen çizi yetenekli ekran için de geçerlidir.

Diğer tipteki ekranların tüm yeteneklerini içerdiğinden yenilenen çizi ekranına kolayca program aktarılabilir. Öte yandan, bu tür ekranın tüm esnekliğini kullanarak hazırlanan programlar, diğer ekranlara aktarılamaz.

Özel BDE ekranları

Genellikle giriş tuşlarla, özel fonksiyon tuşlarıyla ve ekrana dokunarak olur. Yazış sürati, nokta sayısı ve nokta yoğunluğu yenilenen ekrana benzer olarak değişir. Genellikle bu tür ekranlarda hem çizim, hem hareket, hem de diapozitif projeksiyon mümkündür (Bitzer 77). Özel olarak geliştirilen bu ekran tipi diğer uçlar kadar yaygın değildir. Diğer tip ekranların kullanıldığı bir kuruluşa program transferi söz konusu değildir.

Tüm ekranların karakteristiği olan bilgi kaybı özel BDE ekranları için de geçerlidir.

Yardımcı uçlar-öğretim istasyonu

Ekran kullanımının en önemli sakıncası ekrandaki bilginin geçici oluşudur. Bu problemi gidermek amacıyla değişik yöntemler benimsenmiştir.

Birinci yaklaşım ekrana paralel bir karakter basıcıyla ekrandaki bilginin seçenekli olarak kâğıda aktarılmasıdır. Genellikle tüm ekran tarandığı için hem zaman kaybı hem de ekranda bulunan istenilmeyen bilginin de kâğıda dökülmesi söz konusudur. Bu yöntem çizi yetenekli uçlar için geçerli değildir.

Alternatif, çizi ekranına paralel nokta basıcıdır. Kopyalama işleminin süratle tamamlanabilmesi için nokta basıcının kullandığı özel kâğıt pahalı olduğundan yaygın bir yöntem değildir.

Üçüncü yaklaşım, ekranın yanısıra bir tele-printer ile bir öğretim istasyonu oluşturmaktır. Öğrenci seçenekli olarak istediği bilgiyi tele-printere gönderebilir. Her iki uç da program kontrolünde olduğundan gereksiz bilgi dökümü önlenebilir. Ancak hem döküm süresi uzundur hem de teleprinter kullanım oranı düşüktür.

Dördüncü yaklaşım ekrandaki bilginin seçenekli olarak paylaşılan bir satır basıcıya gönderilmesidir (Balman 77). Kâğıda dökülecek bilgi geçici sistem kütüklerinde tutulduğundan döküm esnasında zaman kaybı söz konusu değildir.

Bu yaklaşımlardan birini benimsemiş olan bir kuruluşla program takası yapılacaksa program modifikasyonu zorunludur.

3.3.2 Çizi altrutinleri

Standart yoksunluğunun sayısal çözümleme altrutinlerinin transferinde yarattığı tüm problemler çizi altrutinleri için de geçerlidir. Ancak, ekran özelliklerinin çizi modüllerine ve program tasarımına yansımasından ötürü transfer problemleri çok daha önemli boyutlara ulaşabilmektedir. Şöyle ki, iki değişik tip çizi ekranı kullanan kuruluşlar bile belirli bir ekran tipi için geliştirdikleri programı diğer tipteki ekranlarında (programdaki çizi komutlarını yeniden düzenlemeden) kullanamamaktadır.

Çizi altrutinlerini iki sınıfa ayırmak mümkündür:

i) Fiziksel çizi komutları

ii) Mantıksal çizi komutları

Fiziksel rutinler yazılım diliyle ekran arasındaki ilkel irtibatı sağladıklarından ekran özelliklerini programa yansıtırlar. Bu sebepten değişik fiziksel özelliklere sahip olan ekranlarda (örneğin nokta adresleme özelliği) aynı fiziksel rutinler geçerli olmayabilir. Fiziksel komutlar en ilkel düzeyde çizi ve nokta adresleme olanağı sağladıklarından salt bu tür komutlarla çizim, programlayıcı açısından, elverişli değildir.

Tipik fiziksel çizi komutlarını şöyle sıralayabiliriz:

i) Belirtilen (göreceli veya mutlak) noktaya çizmeden ilerle

ii) Belirtilen (göreceli veya mutlak) noktaya kadar çizerek ilerle

iii) Alfanümerik ve çizi opsiyonları arası statü değişimi

vs.

Gerçekte, ekranı gizlemediklerinden ötürü, en etkin ve esnek çizim yöntemi fiziksel rutinlerdir. Ancak, ekran tipinin tüm özellikleri bu komutlara yansıdığından program transferi çok güçtür.

Mantıksal çizi komutları fiziksel komutlardan çok daha güçlüdür. Tüm mantıksal komutlar sistemin modül kitaplığı aracılığıyla fiziksel komut dizisine dönüştürülür. Tipik mantıksal çizi komutları:

i) Belirtilen ekran sınırları içerisine verilen koordinatları kullanarak X-Y grafı çiz.

ii) Belirtilen ekran sınırları içerisine belirtilen ölçekle prototip şekli (örneğin kare) çiz.

Görüldüğü gibi bu komutlar da ekran adresleme sisteminden bağımsız değildir. Ancak mantıksal rutinlerde kullanılan ekran adresleri gerçek değil de mantıksal koordinatlar olabilmektedir. Yani belirtilen ekran sınırları ekran tipine bağımlı

olan gerçek sınırlar değil de mantıksal altrutinin ekran tipine göre gerçek nokta adreslerine dönüştürdüğü sınırlar olabilmektedir.

Mantıksal ekran adresleme sistemleri program taşınabilirliğini bir dereceye kadar kolaylaştırabilir. Şöyle ki, eğer değişik kuruluşlar belirli bir mantıksal komut standardı oluşturup kendi ekranlarının özelliklerine göre mantıksal modül kitaplığını geliştirirlerse, bu komutların kullanıldığı herhangi bir program daha kolay transfer edilebilir. Maalesef şimdiye dek oluşturulan standardlar ne yaygınlaşabilmiş ne de kararlaştırılan komutlar ekran özelliklerinden tamamıyla bağımsızlaştırılmıştır.

Çizi altrutinlerinin standardlaşmasına doğru ilk adım ortak çizi ölçeklerinin kabullenmesidir. Ekrandaki çizi pozisyonları nokta adresleme sistemi yerine mantıksal adresleme sistemiyle belirtilmelidir. Bu işlem ortak referans pozisyonunun (örneğin 0-0 sol alt köşe), ortak ölçü biriminin (örneğin ekran büyüklüğünün nokta yoğunluğuna olan oranı), en çok kullanma olasılığı olan üst düzey modüllerin ve bunların giriş/çıkış argümanlarının kararlaştırılmasını gerektirir.

Şimdiye dek en başarılı standardlaşma GINO sistemiyle gerçekleştirilmiştir. GINO modül kitaplığı değişik fiziki özellikleri olan ekranlarda kullanılabilen çizi komutlarından oluşmaktadır. Programlayıcı mantıksal çizi komutlarını ekran karakteristiklerinden nisbeten bağımsız olarak verebilmesine karşın mantıksal-gerçek çevirisini yapabilmek için ekran tipine göre değişik GINO modül kitaplığı kullanılmalıdır. Yani mantıksal komutlar ekran tipinden bağımsızdır ama değişik ekran tipleri için değişik çeviri modülleri gerekmektedir.

GINO mantıksal sistemiyle tüm çizi ekranlarının sol alt köşesi 0-0 koordinat olarak ve diğer ekran pozisyonları milimetrik ölçeğe göre x-y koordinatları olarak tanımlanır. bu yaklaşımın iki önemli sakıncası vardır:

i) Programlanan çizi komutları kesinlikle ekranın fiziki büyüklüğüne bağımlı olur. Popüler bir çizi ekranı büyüklüğünün (yaklaşık 150x200mm) varolması-

na karşın her ekranın aynı büyüklükte olduğu söylenemez. Bu sebepten genellikle program transferinde tüm çizi komutlarının giriş değişkenlerinin yeniden ayarlanması söz konusu olabilir.

ii) Milimetreye düşen nokta sayısı tam sayı değilse çizilen şekillerde ciddi bir hata birikimi olabilir.

Tüm sakıncalarına rağmen GINO sistemi ideale en yakın olabilen standarttır. Getirdiği en önemli kolaylık komut ve parametre değişikliklerinin program tasarımı şeklini etkilemeden gerçekleştirilebilmesidir. Ancak, sistemin sağladığı üst düzey çizi rutinleri salt graf çiziminde kullanılabilen modüller olduğundan BDE uygulamaları için yeterli değildir. Ayrıca üç-boyutlu çizim ve şekil hareket standartlarını sağlamadığından tüm çizim sorunlarına çözüm getirdiği söylenemez.

3.4. Program değişebilirliği

Program transferinde karşılaşılan teknik sorunlardan başka aşılması gereken önemli bir engel de programın eğitsel içeriğinin yabancı kuruluşlarca benimsenmesidir. Burada söz konusu olan BDE'i benimsemiş olan kuruluşlarda görülen yaklaşım uyuşmazlıkları değildir (kuşkusuz hiç bir kuruluş sakıncalı bulduğu yöntemlerin kullanıldığı programları istemeyecektir). Önemli engel programı almak isteyen kuruluşun yapmak isteyebileceği ayrıntı değişiklikleridir. Bu kategoriye

i) Veri giriş ve çıkış yöntemleri

ii) Giriş değerlerine uygulanacak farklı kısıtlamalar ve öndeğerler

iii) Çıkış değerlerinin değişik olarak düzenlenmesi (ek bilgi vb.)

girer.

Geliştirilen BDE programlarının en kritik ve değişebilir olan giriş/çıkış parametreleri sadece yabancı kuruluşlar değil aynı zamanda program geliştiren kuruluş tarafından da değiştirilmek istenebilir. Genellikle bu tür değişiklikler iki sebepten ötürü gerekir:

i) Programı kullanan öğretim elemanının değişmesi

ii) Program öğrencilerle denendikten sonra elde edilen netice ve tecrübe sonucu yapılmak istenen değişiklikler.

Çoğunlukla BDE programları uzun süre gelişme sürecinden çıkamamaktadır. Geliştirilen herhangi bir BDE programının, kullanılmaya başladıktan sonra, er geç değişikliğe uğraması doğaldır. Programın yazılımında alınacak bir takım tedbirlerle ileride yapılacak değişikliklerin program üzerindeki etkisi yumuşatılabilir. Tedbirleri şöyle sıralayabiliriz:

i) Veriye bağımlı genel amaçlı yazılım

ii) Modüler yazılım

iii) Dokümentasyon

3.4.1 Veriye bağımlı yazılım

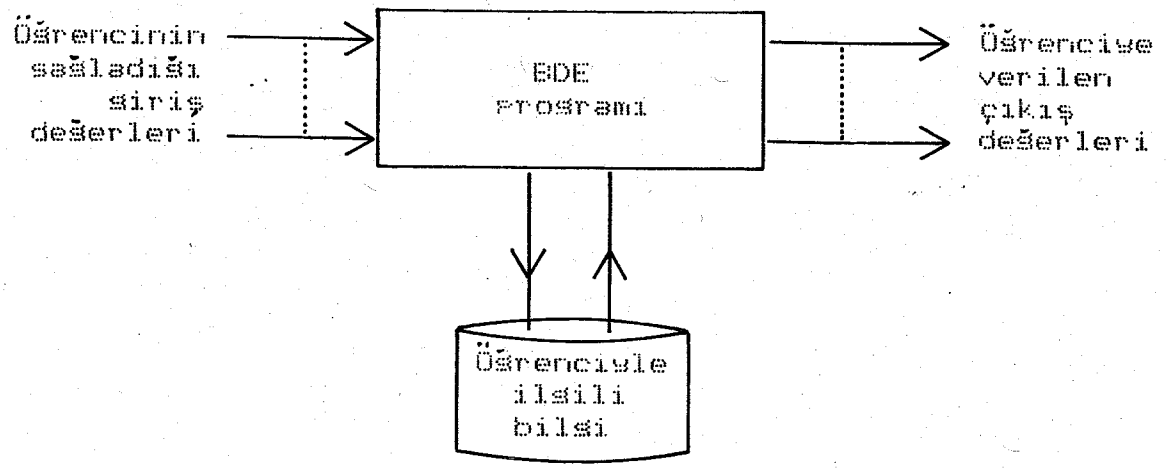
BDE sistemlerini kabataslak olarak Şekil 3.1 deki gibi özetlemek mümkündür. Şekilde gösterilen yaklaşımın önemli bir sakıncası yapılacak herhangi bir değişikliğin programdaki komutları etkilemesidir. Değişen komutlar programın bütünlüğünü etkileyen nitelikte olabilmektedir.

Daha sağlıklı bir yaklaşım programın veriye (veya gereğinde karar tablolarına, bilgi kütüklerine) bağımlı olarak geliştirilmesidir. Böylece kullanıcıya sunulan bilgi ve kullanıcının sağladığı giriş değerlerine sistemin davranışı program komutlarını kurcalamadan değiştirilebilir. Bu yaklaşım Şekil 3.2'de özetlenmektedir.

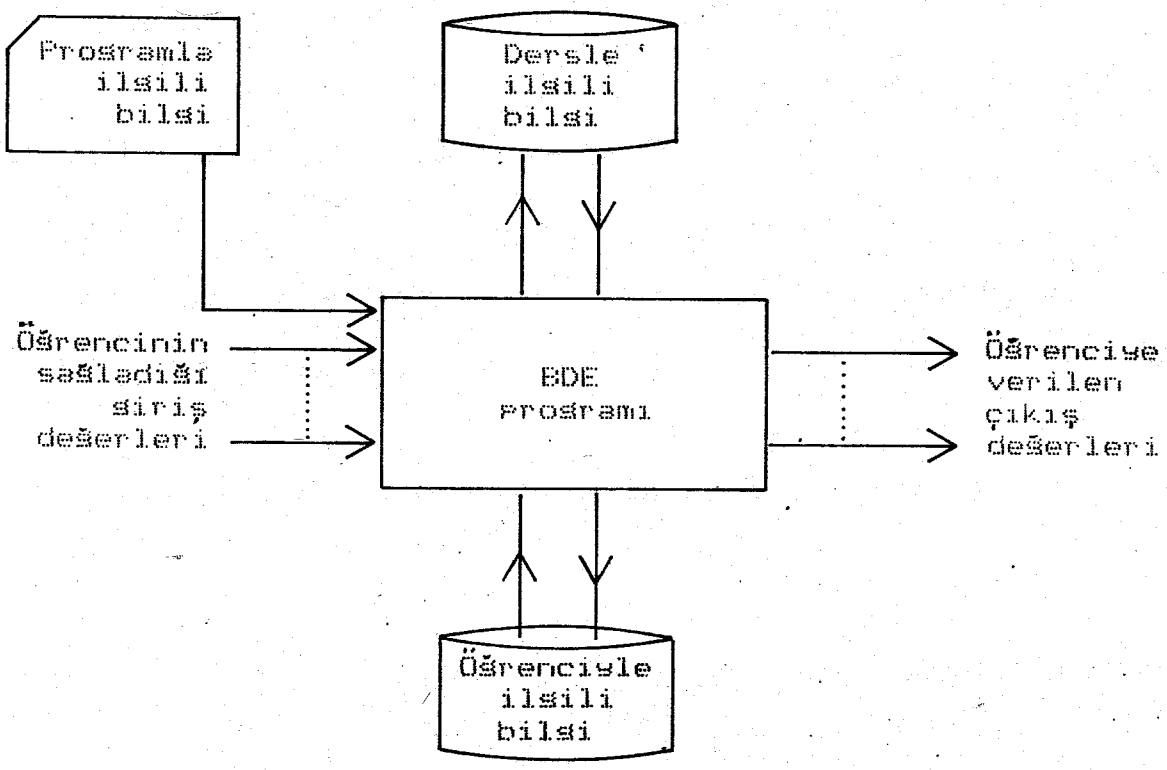
Veriye bağımlı BDE programı yazılımına iki örnek verebiliriz.

i) BDD uygulamaları

Genellikle BDD programlarında kullanılan matematiksel formüllerin geçerliliği bir çok akademisyen tarafından kabullendiği halde bu formüllerin kullanılış şekli tartışma konusu olabilmektedir. Örneğin belirli bir akademisyenin empoze etmek istediği giriş değişken değer kısıtlamaları, öğrenci adına program içerisinde seçilen ve değiştirilemeyen değerler, çizi altrutinleri için kullanılan koordinat sayısı ve ölçekleme, çıkış değerlerinin sunuluş şekli, vb.



Şekil 3.1 : Tipik BDE programı şeması



Şekil 3.2 : Veriye bağımlı olarak yazılan BDE programı

Bu tür kısıtlamalar nümerik işlem modüllerine değişmez değer olarak yansıtılırsa ileride yapılacak kısıtlama değişiklikleri hem güçleşir hem de değişiklikler program uyumunu aksatabilir. Halbuki kısıtlamalar nümerik işlem modüllerine ana modüldeki değişkenler aracılığıyla yansıtılmışsa gereken kısıtlama değişiklikleri sırf ana modüle yapılacak değişikliklerle gerçekleştirilebilir.

Bu yöntem bir dereceye kadar başarılıdır. Eğer öğrenci-bilgisayar iletişimine bir değişiklik yapılacaksa bu yöntem etkinliğini kaybeder.

ii) Özdevimli kitap

Genel amaçlı BDE yazılımına en iyi örnek özdevimli kitap yöntemidir. Bu yaklaşımla ders içeriği tümüyle programdan bağımsız olabilmektedir. Programın sağladığı ders şekli benimsendiği takdirde ders içeriği tümüyle değiştirilebilir.

Ancak,bu yöntemle de programın sağladığı ders şeklinin değişmesi söz konusu olabilir-örneğin öğrenciye ek kontrol, başarısızlığa tolerans ölçüsü, vs. Bu durumlara önlem olarak BDD'deki yaklaşım kullanılabilir.

3.4.2 Modüler yazılım

Programın parçalar halinde tasarlanması kuşkusuz yapılacak değişikliğin tüm programı etkilemesini önler. Ara birimleri ve amacı tanımlanan her modül (bu tanımlara uymak koşuluyla) gereğinde tümüyle değiştirilebilir. Her modül programın diğer modüllerinden büyük ölçüde izole edildiğinden, yapılan değişiklikler yereldir.

3.4.3 Dökümantasyon

Yapılacak program değişikliklerine en büyük yardımcı dökümantasyondur. Tüm uygulama kısıtlamaları, bu kısıtlamaların programa nasıl ve nerede yansıdığı, tüm program modülleri ve bunların ara birimlerinin tanımı kâğıt üzerinde belirtilirse ilerideki program değişiklikleri daha kolay gerçekleştirilir. Dokümentasyon hem programın geliştirildiği hem de transfer edildiği kuruluşlarda yapılacak tüm program değişiklikleri için elzemdir.

Ayrıca, programda kullanılan giriş/çıkış yöntemleri, standart olmayan komutlarla yerel altrutinler yabancı kuruluşların programcılarına ancak dokümantasyon aracılığıyla tanıtılabilir.

3.5 Sonuç

Hem insan gücünü ekonomik olarak kullanmak hem de BDE'in yaygınlaşmasını sağlamak için taşınabilir programlar geliştirilmelidir.

Özetle program transferi aşağıdaki sebeplerden ötürü kolay değildir.

i) Programlama dillerinin 100% standart olmayışı

ii) Hizmet altrutinlerinde (kütük kullanım, sayısal çözümleme) standart olmayışı

iii) Farklı çevresel donanım karakteristikleri

iv) Farklı kullanıcı-bilgisayar iletişim yöntemleri

v) Farklı program kısıtlamalarıyla farklı giriş/çıkış parametre grubu arzulanması.

Program takası yapmak isteyen kuruluşlara önerilebilecek başlıca önlemler ise

i) En yaygın ve standartlaşmış dil olan FORTRAN IV'ün BDE yazılımında kullanılması. Program yazılımında standart olmayan komutlardan kaçınılması.

ii) Ortak hizmet altrutinlerinin geliştirilmesi

iii) Benzer çevresel donanım kullanılması

iv) Ortak öğrenci-bilgisayar iletişim yöntemlerinin benimsenmesi.

v) Program değişikliklerini kolaylaştırmak için genel amaçlı ve modüler yazılım.

vi) Yabancı kuruluşlarda transfer edilen programın çalıştırılabilmesi için i, ii, iii, iv. Maddelere ilişkin bilginin ayrıntılı dokümantasyonu. Yapılacak program değişikliklerini kolaylaştırmak amacıyla program modüllerinin dokümentasyonu

BÖLÜM 4

BDE DESTEKLEYEN SİSTEMLERİN YAZILIM ORGANİZASYONU

Önceden de belirtildiği gibi eğitim için kullanılacak bilgisayar sisteminin tümüyle BDE'e adanması arzulanır. Genellikle kurulan bilgisayar sistemiyle yalnız BDE servisi ve BDE araştırmaları yapılır.

Bu kurumlaşma arzusu BDE ünitelerini 8-32 uçlu minibilgisayar kullanımına yöneltmiştir. Tipik olarak 16-bit hücrelerden oluşan 64-256K hücrelik ana bellek ve 2.4-12 milyon karakterlik yardımcı bellek kapasiteleri olan minibilgisayar sistemleri arasında en yaygın kullanılan PDP-11 serisidir.

Çok kullanıcılı minibilgisayar işletim sistemlerinin az gelişmiş olmasından doğan sakıncalar vardır. Örneğin

i) Zayıf yazılım desteği, kısıtlı sistem modül repertuvarı, nümerik çözümleme modüllerinin desteklenmemesi.

ii) Sistem kütüklerinin kullanıcıya karşı etkin bir şekilde korunmaması

iii) Sisteme erişim protokolünün yokluğu (veya etkin olmaması)

iv) Program başlatma komutlarının bilgisayar tanımayan öğrenciler için karmaşık oluşu

v) Kısıtlı çevresel donanımın program transferini güçleştirmesi

Minibilgisayarların bu sakıncaları hem program geliştirenler hem de program kullananlar için büyük bir sorun teşkil etmektedir. Bu iki kullanıcı sınıfının sistemde aradıkları ve minibilgisayarların sağlayamadığı nitelikler incelenirse program gelişticilerin kütük güvencesi (yani güçlü bir erişim protokolü) ve BDE için elverişli geniş kapsamlı modül kitaplığı arzuladıkları görülür. Öğrenci kullanıcılar ise protokolsüz veya kolay öğrenilebilen bir protokolle program başlatabilmek ister.

Her iki kullanıcı sınıfını tatmin etmek için hem BDE altyapı modülleri oluşturulmalı hem de sistem erişim protokolüne daha basit ve daha güvenilir bir şekil verilmelidir.

Bu bölümde belirli bir BDE merkezindeki bilgisayara, yukarıda belirtilen koşullar çerçevesinde, yapılan altpayı ve işletim sistemi değişiklikleri tanıtılacaktır.

4.1 BDE modül kitaplığı

BDE araştırmacılarına gerekebilecek altrutinleri dört sınıfa ayırabiliriz

i) Sayısal çözümleme mödülleri

ii) Kütük düzenleme modülleri

iii) Veri giriş modülleri

iv) Çıkış ve çizi modülleri

Her modül karşılaştığı işlem hatalarını ya kendi içinde çözümlemeli, ya da çağırıldığı üst modüle bir çıkış parametresi aracılığıyla hatanın niteliğini bildirmelidir. Bu yaklaşım hata kontrolünün sisteme geçmesini önleme açısından önemlidir. Aynı hata kodlama sisteminin geliştirilen tüm modüllerce kullanılması sağlıklı bir yöntemdir. Modülü kullanan program hatayı çözümleyici önlemi kendi içinde almak istemediği durumlarda, hata mesajlarını kullanıcıya iletmek amacıyla ayrı bir "hata mesajı" modülü geliştirilmelidir.

Yani, kitaplıktaki her modül ancak çözümleyemediği işlem hatalarını çağırıldığı programa iletmeli; modülü çağıran program da hatayı çözüleyemiyorsa hata mesajını kullanıcıya iletecek modülü çağırmalıdır.

4.1.1 Sayısal çözümleme modülleri

Program önerilmeden ne tip nümerik rutinlere gereksinme doğacağını tahmin etmek güçtür. Bu sebepten ötürü BDE programlarından önce sayısal çözümleme kitaplığını geliştirmek insan gücü israfı olabilir. Ancak, belirli bir sayısal modül gerektiğinde, bu modül en genel biçimiyle geliştirilmelidir. Sağlamlığı kanıtlandıktan sonra ayrıntılı kullanım dokümantasyonu hazırlanıp modül sistem kitaplığına alınmalıdır. Bu işlem aynı modülün başka programlarda kullanılabilmesini sağlar.

4.1.2 Kütük düzenleme modülleri

Genellikle minibilgisayar sistemlerinde sırasal erişimden başka kütük erişim yöntemi desteklenmediğinden büyük öğrenci kütüklerinin kullanımı zaman kaybına neden olur. Zaten minibilgisayarlarda yaygın olarak kullanılan BASIC ve FORTRAN dilleri de kütük kayıtlarını kart görüntüsü olarak kabullendiklerinden programda yalnızca sırasal erişim gerçekleştirilebilir.

Geliştirilecek kütük denetim, erişim ve düzenleme modüllerinde aranan başlıca nitelikleri şöyle sıralayabiliriz.

i) Kayıt arama, kayıt düzeltme ve kayıt ekleme işlemlerini kalıplaştırma

ii) Erişim süresini azaltacak şekilde kütük düzenleme

FORTRAN ve BASIC dillerinin kurallarına uyarak gelitirilebilecek en süratli sırasal erişim anahtar sözcük kütüğü kullanarak gerçekleştirilebilir. Ana kütükteki kayıtlar anahtar sözcüğe göre sıralanır. Ana kütük kayıtları her kayıt kütüğünde tutulan en büyük anahtar, kayıt sayısı ve kayıt kütüğünün ismini içerir (Şekil 4.1). Bu yaklaşımla, aranan kayıtın bulunduğu kütük, ana kütükteki anahtar sözcükler taranarak saptanır; ilgili kayıt kütüğü sırasal okunarak kayıt pozisyonu bulunur.

Şekil 4.1'den de görülebileceği gibi bu yöntem dizim sıralı kütük organizasyonunu FORTRAN ve BASIC dillerinin komutlarını kullanarak yaratmayı amaçlar. Bir avantajı kütük erişimini süratlendirirken bilgisayarın kendi kütük yönetim sistemine bağımlı kalmasıdır; sistemin birikim ortamında yaptığı kütük hareketlerinden etkilenmez.

Bu yaklaşım çerçevesinde gereken altyapı modülleri şöyledir

i) Kütük yaratma- kullanıcının sağlayacağı parametreler: anahtar uzunluğu, ana kütük kayit kapasitesi, kayıt kütüklerinin kayıt kapasitesi, kayıt uzunluğu.

ii) Kütük dengesini sağlama-kayıt kütüklerindeki taşmanın ve dengesiz kayıt dağılımının önlenmesi için kütüklerin yeniden düzenlenmesi

iii) Kayıt ekleme- ekleme sırasında kayıt kütüklerinde baş gösterebile-

NA KÜTÜK KAYITLARI:

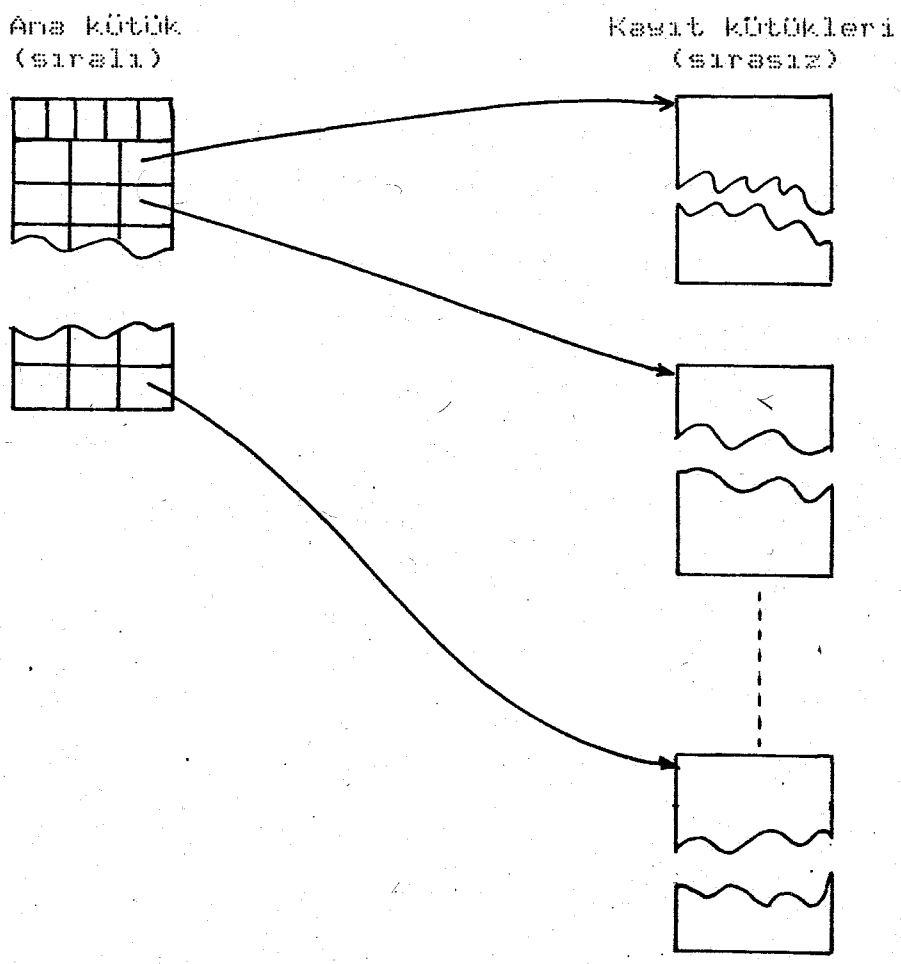
k kayıt:

anahtar uzunluğu	kayıt uzunluğu	ana kütükteki kayıt sayısı	ana kütüğün sığası	kayıt kütüğünün sığası

ser kayıtlar:

kayıt kütüğündeki en büyük anahtar	kayıt kütüğündeki kayıt sayısı	kayıt kütüğünün ismi

ÜTÜK ORGANİZASYONU:



Şekil 4.1 : Kütük organizasyonu

cek taşmanın kullanıcıya yansıtılmadan halledilmesi.

iv) Kayıt erişim

v) Kayıt değiştirme- kayıt alanları bilinmediğinden kullanıcı eski kayıtı okuyup değiştirdikten sonra tekrar yazdırmalıdır

Bu altrutinler dahili hata mesajı kodlamasına uygun olarak hata parametreleri kullanmalı ve yapay kütük denetim sisteminin gördüğü hatalar kullanıcı programa iletilmelidir.

4.1.3 Veri giriş modülleri

Belirli veri giriş yorumlama işlemleri bir çok uygulama programında kullanılacağından bu yönde yapılacak modülerleşme programcılara kolaylık sağlar Eğer her program bu modüllerden yararlanırsa öğrenci tek bir programdan edindiği veri giriş deneyimini diğer programlarda kullanabilir.

Veri giriş modüllerinde aranan nitelikler arasında en önemlisi serbest desen girişin kabul edilmesi ve veri hatalarının programa yansımadan modül içerisinde çözümlenmesidir. Bu modül kitaplığına katılabilecek tipik giriş altrutinleri şöyle olabilir (Balman 78d)

i) En alt düzeyde tek veri çizgisinden tek sayısal değer okuyan altrutin. Giriş parametresi: çağırı cümlesi; çıkış parametresi: okunan değer.

Bu altrutin veri çağırı cümlesiyle uyarı karakterini yazdıktan sonra karakter kodlu giriş çizgisini okur ve sayısal koda çevirir. Eğer veri çizgisinde hata varsa (örneğin anlamsız karakter vb.) hata mesajını yazdıktan sonra çağırı cümlesini verip tekrar okur. Bu işlem geçerli sayısal bir değer okunana dek tekrarlanır.

ii) Sınırlandırılmış değer aralığında gerçek sayı okuyan altrutin. Giriş parametreleri: değer aralığını tanımlayan en küçük ve en büyük sayılar, çağırı cümlesi; çıkış parametresi: okunan değer.

Bu modül i.de belirtilen modülü kullanarak öğrencinin sağladığı veriyi okuduktan sonra okunan sayının belirlenen aralıkta olup olmadığına bakar. Değer

aralığının dışında ise bir hata mesajıyla i.de belirtilen modülü tekrar çağırır.

iii) Belirtilen değer aralığı içinde tam sayı okuyan altrutin. ii.de tanımlanan modül gibi çalışır. Tek fark alt düzey okuma modülünün verdiği değerin tam sayıya çevirilmesidir.

iv) Okunacak değerler çok büyük veya çok küçük olabilirse bazi harfleri birimlerle bağdaştırma arzulanabilir (örneğin "K" ≡ X1000,"M"≡ X1000000, "m" ≡ X 0.001 vb). Bu olanağı sağlamak için iki modül gerekir. Modüllerin biri her harfin anlamını tanımlar, diğeri ise alt düzey serbest desenli okuyucunun bir varyasyonudur. Tanımlanan birim harflerinin birisi sayının ardından gelirse okunan sayı birim değeri ile çarpılır.

v) Aynı veri çizgisinden birden çok değer okunacaksa yukarıda tanımlanan altrutinlerin varyasyonları gerekir. Tipik olarak bu altrutinlerde giriş parametreleri arasında alanları ayıran karakter(", ", ":", boşluk vs.) ve çizgiden kaç değer okunacağı belirtilmelidir.

4.1.4 Çıkış ve çizi modülleri

Sayısal çözümleme mödüllerindeki gibi çizi modüllerini de program geliştirilirken hazırlamak daha akılcı olur. Ancak, 3.3.2'nci bölümde de belirtildiği gibi mantıksal koordinat sisteminin önceden saptanıp benimsenmesi gerekir. Eğer seçilen mantıksal yöntem uçların fiziksel özelliklerinden farklı ise ilk safhada tüm alt düzey mantıksal ve fiziksel çizi modüllerinin hazırlanması gerekir. Diğer çizi altrutinleri gerektikçe, bu alt düzey modülleri kullanarak geliştirilir. Nümerik çözümleme modüllerinde olduğu gibi geliştirilen her çizi altrutini en genel biçimde yazılmalı ve gerekli dökümantasyon hazırlandıktan sonra modül kitaplığına eklenmelidir.

Çizi altrutinleri dışında kalıplaştırılabilecek çıkış işlemi yoktur. Ancak 3.3.1'inci bölümde tanıtılan "paylaşılan süratli yazıcı" yöntemi kullanılacaksa bu olanağı sağlayıp kullanıcı programları denetleyecek sistem geliştiril-

melidir. Gerçek zamanlı programlar arası iletişim komutları sağlayan bir bilgisayarda bu sistemin nasıl hazırlanabileceği Balman 77 ve 78'de anlatılmaktadır. Özetle, bu yaklaşım iki modül ve bağımsız bir sürücü programdan oluşur. Programcının süratli yazıcıyı kullanmak için çağırdığı iki modül şunlardır:

i) Belirtilen çıkış kanalından erişilmek üzere geçici yazıcı kütüğünün açılması. Bu modül çağırıldığı programın ismine ve uç özelliklerine göre çıkış kütüğünü açar. Eğer programın henüz yazıcıya gönderilmemiş kütüğü varsa program bekletilir.

Geçici kütük açıldıktan sonra programın belirtilen çıkış kanalına gönderdiği tüm dizgeler kütüğe aktarılır.

ii) Belirtilen kanalda kullanılan yazıcı kütüğünün kapanması. Kütük kapatıldıktan sonra, süratli yazıcı sürücü programının, yazılacak kütük isimleri kuyruğuna eklenir. Eğer sürücü program aktif değilse başlatma komutu verilir.

Süratli yazıcının sürücü programı işleme geçtiği zaman kuyruğundaki ilk kütük ismini alıp o kütüğü açar. Kütükteki tüm bilgi süratli yazıcıya gönderildikten sonra kütük silinir. Bu işlem kuyruktaki tüm kütükler boşaltılana dek sürer. Kuyruk boşalınca sürücü program, yazıcı kütüğünü kapayan modül tarafından başlatılana dek bekler.

4.2. Sistem programları

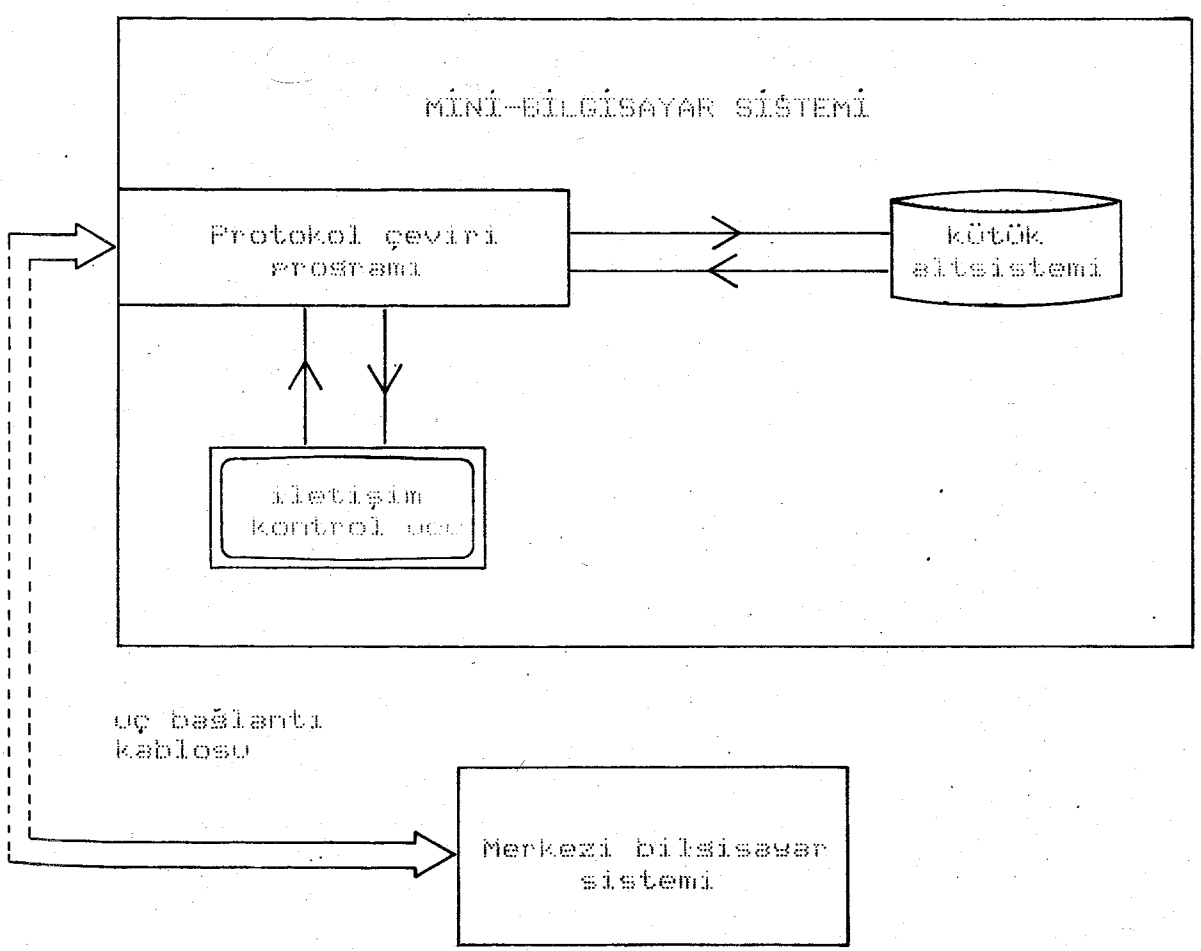
4.2.1. Bilgisayarlar arası iletişim

BDE ünitesinin kendi kendine yeterli durumda olması kullanıcılara verilen servis kalitesini arttırır. Kendi bağımsız bilgisayarı olan ünite koşulsuz ve kısıtlamasız BDE servisi verebilir. Ancak, minibilgisayarla bağımsızlaşmanın getirdiği problemler vardır. Bu problemlerin biri, bilgisayara (ekonomik ve teknik nedenlerle) bağlanan çevresel donanımda çok çeşit olmayışıdır. Çevresel donanımın kısırlığı program transferi sırasında etkindir. Örneğin, kart veya manyetik şerit oku-

yucusu olmayan ünitelere bu taşıma ortamlarında program getirilemez. Aynı şekilde,bu alt sistemlerin bulunmadığı bir ünite manyetik şerit ve kart taşıma ortamlarından yararlanıp diğer ünitelere program ihraç edemez.

En ekonomik çözüm BDE ünitesinin merkezi bir bilgisayar sistemine bağlanmasıdır. Bu yaklaşım ünitenin bağımsızlığını etkilemez; ünite,yalnız gerektiğinde,merkezi sistemin olanaklarından yararlanır.

Merkezi sistem çok-uçlu düzenle çalışıyorsa iki bilgisayar arasında kütük transferlerini gerçekleştirmek için gereken sadece merkezin izni, bağlayıcı kablo (uç iletişim kablosu veya telefon hattı) ve bir protokol çeviri programıdır (Balman 78e). İdeal yaklaşım, protokol çeviri programıyla, minibilgisayarı merkezi sisteme normal bir uç olarak göstermektir (Şekil 4.2). Şöyle ki,iletişim kontrol ucunda yazılan tüm komutlar gerekli protokol çevirisi yapıldıktan sonra merkezi sisteme gönderilir. Aynı biçimde merkezi sistemden gelen mesajlar iletişim kontrol ucuna gönderilirse bu uçtaki kullanıcı merkezi sistemin tüm olanaklarından yararlanabilir.



Şekil 4.2 : Çevirim içi bilgisayarlar arası iletişim

Kütük transferlerini gerçekleştirmek için iki özel komut geliştirilmelidir. Komutların biri merkezi sisteme gönderilecek mesajların iletişim kontrol ucundan değil de minibilgisayarın belirtilen kütüğünden alınacağını bildirir; kütük boşalana dek tüm mesajlar kütükten alınır. Diğer komut merkezi sistemden gelen mesajların iletişim kontrol ucu yerine belirtilen minibilgisayar kütüğüne gönderilmesini sağlar; bu işlem merkezi bilgisayar yeni komut giriş çağırısını gönderene dek sürer.

Özetle kütük gönderme işlemi şöyledir.

i) İletişim kontrol ucundan merkezi sisteme gir

ii) Merkezi sistemde yeni kütük aç

iii) Minibilgisayardaki kütüğü gönder

iv) Merkezi sistemdeki kütüğü kapa

Basamak iii'de kullanılan özel komut dışındaki komutlar, merkezi çok uçlu sistemin komutlarıdır.

Kütük getirme işlemi ise

i) İletişim kontrol ucundan sisteme giriş

ii) Getirilecek kütüğün ismiyle yerleştirileceği kütüğün isimlerinin belirtilmesi.

Basamak ii'de, protokol çeviri sistemi kütükleri açtıktan sonra merkezi sisteme "listele" (LIST) komutunu gönderip gelen tüm mesajları minibilgisayar kütüğüne yöneltir.

4.2.2. Kolay ve güvenilir erişim

Minibilgisayar denetim sistemleri genellikle kullanıcıya etkin kütük güvencesi sağlayamaz. Sistemdeki erişim açıklığı hem deneyimsiz kullanıcıyı şaşırtır hem de az deneyimli (veya kasıtlı) kullanıcıyı denetleyemez.

Tipik çok-uçlu minibilgisayar sisteminde kullanıcının program başlatmak için bilmesi gereken komut zinciri çok karmaşık olabilmektedir:

i) İstenen programın işlenebilir amaç kodu kütüğünün kopyalanıp isminin

değiştirilmesi. Bu işlem program birden çok uçta kullanılacaksa ve işletim sistemi ana bellek dağıtımında amaç kod kütüğünü takas alanı olarak kullanıyorsa gerekir. Kullanıcı kopyalama işlemini yapacaksa sistem kütük yönetim komutlarını öğrenmelidir. Kütük güvencesi yoksa deneyimsiz ellerde çok sakıncalı bir işlemdir.

ii) Kütükteki işlenebilir kodun bellekteki kullanıma hazır programlar listesine eklenmesi.

iii) Bellekte kullanıma hazır programın kanallarının kullanılacak olan uca atanması. Programın hangi kanalları ne amaçla kullandığını bilmek gerekir.

iv) Programın başlatılması

v) Program bitişinde bellekteki kullanıma hazır programlar listesinden çıkarılması

vi) Kopyalanan işlenebilir amaç kodu kütüğünün silinmesi.

Minibilgisayar sistemini sırf ders öğrenmek için kullanacak ve belki de o gün yapacağı ders dışında bilgisayarla başka hiç bir çalışması olmayacak birisine sistem komut felsefesiyle sistem komutlarını öğretmek sakıncalıdır. Komutların öğretilmesine şu itirazlar haklı olarak yöneltilebilir

i) BDE'in amacı bilgisayar kullanımını öğretmek değil bilgisayar kullanarak eğitmektir.

ii) BDE kullanan bazı öğrenciler (örneğin Beşeri Bilimler) bilgisayar deneyimleri olmadığından sistem komutlarını öğrenmekte güçlük çekebilirler. Bu komutların öğrenilmesi için gereken çaba onları bilgisayarlardan soğutabilir.

iii) BDE kullanımı için belirli bir süresi olan öğrenci bu sürenin bir kısmını sistem komutlarıyla uğraşarak harcamaya zorlanmamalıdır

iv) Öğretilmesi gereken bazı komutlar (özellikle i. ve vi.) sistem kütükleri için tehlike yaratır.

Uç sayısının az olduğu ünitelerde program başlatma işlemini personel yürütebilir. Ancak, rezervasyon sistemiyle servis veren bir ünitede tüm kullanıcıların çok kısa bir sürede değişmesi söz konusu olabilir. Bundan ötürü, 5-6 uçtan faz-

lası kullanılıyorsa (veya tüm uçlar aynı laboratuvarda değilse), program başlatma işlemini personelin zamanında gerçekleştirmesi olanak dışıdır.

İdari açıdan minibilgisayar sistemlerinin bir sakıncası da kullanım istatistiklerinin sistem tarafından tutulmamasıdır. Doğal olarak, sisteme erişim protokolü olmayan sistemlerden kullanım istatistikleri beklenemez. Ünite yönetimi, en azından, her araştırma konusunun ne kadar bilgisayar zamanı aldığını ve geliştirilen her BDE programının kaç saat ve kaç kişi tarafından kullanıldığını bilmek ister.

Tipik minibilgisayar işletim sisteminin bu eksiklikleri giderilmediği takdirde düzenli ve randımanlı bir BDE hizmeti sağlanamaz. Tek çözüm işletim sisteminin BDE kullanımına uygun bir biçimde geliştirilmesidir.

Erişim ve istatistik problemlerini gidermek için kullanılan bir yöntem Balman 78 b ve 78 c'de ayrıntılı olarak anlatılmıştır; bu yöntem aşağıda kabataslak tanıtılmaktadır.

Kullanım istatistiği kütükleri

Sistem beş değişik kütükte günlük ve aylık kullanım istatistiklerini tutar. Kütükler:

i) Günlük araştırma amaçlı kullanım istatistikleri

ii) Günlük eğitim amaçlı kullanım istatistikleri

iii) Aylık araştırma amaçlı kullanım özeti

iv) Aylık eğitim amaçlı kullanım özeti

v) Sistemin günlük ve aylık çalışma istatistikleri

Sistem açılışı

İşletim sistemine yapılan bir değişiklikle sistem açılınca saat ve tarih okuyan program başlatılır. Sistem çalışma istatistikleri kütüğü incelenerek:

i) Anormal sistem kapanışı olmuşsa (arıza veya yanlış kapama) kapanış saati sorulup günlük ve aylık kullanım istatistikleri düzeltilir. Günlük kullanım istatistiklerinin dökümü verilir.

ii) Yeni bir aya geçilmişse ve girilen tarih hem sistem çalışma kütüğü hem de operatörce doğrulanmışsa aylık kullanım raporu verilir.

Sistem kapanışı

Sistem kapanış programını operatör başlatır. Operatör sistemin kaç dakika sonra kapatılacağını ve sisteme bağlı olan kullanıcılara kaç dakikalık arayla uyarı mesajları gönderileceğini programa bildirir. Kullanıcılara tanınan süre dolunca kapanış programı

i) Kullanılmakta olan tüm programları durdurup günlük ve aylık istatistik kütüklerindeki bilgiyi yeniler

ii) Günlük kullanım raporu verir

iii) Sistem çalışma istatistiklerinin bulunduğu kütüğe bağlantı saatini ekledikten sonra sistemin normal kapandığını gösteren bilgiyi yerleştirir

iv) Sistemi kapatır.

Sisteme giriş

Programcılar ve öğrenciler sisteme aynı program aracılığıyla girer. İşletim sistemine yapılan değişiklikle uç sürücüsü veri girişini iki sınıfa ayırır: çağırılı giriş (işlemde olan bir programın veri çağırısı), çağırısız giriş (kullanılmıyan bir uçta herhangibir tuşa basılması).

Çağırısız girişlerde, eğer uç bağlı değilse, uç sürücüsü sistem erişim programını o uçtan başlatma komutu gelmiş gibi başlatır; yani kullanılmayan bir uçta rasgele bir tuşa basılırsa erişim programı devreye girer.

Sisteme giriş programı kullanıcıya hangi eğitim programını istediğini sorar. Bu safhada öğrenciden istenen tek veri izlemek istediği programın ismidir. Program ismi okunduktan sonra eğitim programları kütüğünden kanal kullanım bilgisi alınır. Kullanılan ucun özelliklerine göre programa sistemin tanıyabileceği geçici bir isim verilip 4.2.2'de belirtilen işlemler yapılarak eğitim programı başlatılır. Öğrencinin eğitim programını başlatmak için yapması gereken işlemler böylece iki basamağa indirgenmiştir.

i) Uçtaki herhangi bir tuşa basılması

ii) Sistem erişim programının çağırısı üzerine program isminin yazılması.

Sistemi program geliştirme amacıyla kullanmak isteyenler daha sıkı bir

denetimden geçer. Erişim programının çağırısı üzerine, eğitim program ismi yerine, rakam dizgisi verilirse ve verilen sayı geçerli bir kullanıcı tanımlayıcısıysa kullanıcıdan parola sözcüğü istenir. Parolanın geçerliliği kullanıcı numaralarına karşılık parolaların tutulduğu sistem kütüğünü tarayarak öğrenilir. Doğru parola verilmişse o uç sisteme bağlı sayılır.

Her iki kullanıcı sınıfının sisteme giriş zamanları günlük istatistik kütüklerine işlenir.

Sistemden çıkış

Eğitim programı kullananların sistemden çıkışı programın bitmesiyle gerçekleşir. İşletim sistemi, program bitiş işlemcisine yapılan bir değişiklikle, biten her programın ismine bakar. Eğer program ismi sisteme giriş programının ürettiği geçici isimlerdense biten programın eğitim amacıyla kullanıldığı anlaşılır Bağlı olduğu uç program denetim öbeğinden bulunduktan sonra sistemden çıkış programı, o uçtan başlatma komutu gelmiş gibi, başlatılır.

Uç eğer programı geliştirmek için kullanılıyorsa sistemden çıkış komutunun verilmesiyle gerçekleştirilir.

Her iki tür çıkışta günlük ve aylık kullanım istatistikleri kullanım zamanına göre değiştirilir. Uç "kullanılmıyor" statüsüne geçirilir.

4.3. Sonuç

Koşulsuz ve kesintisiz hizmet verme arzusu BDE ünitelerini paylaşılan çok amaçlı merkezi bilgisayar sistemlerinden uzaklaştırmaktadır. Öte yandan ekonomik koşullar hem maliyeti hem de destek giderleri düşük olan minibilgisayar sistemlerinin kullanımını zorunlu kılmaktadır. Bu sistemler için gereken altyapı modülleri geliştirilmeden program üretimine geçmek insan gücü israfı yaratır. Yapımcı firmaların geliştirdiği işletim sistemleri, bilgisayarı kullanmak için yeterli olduğu halde, etkin bir BDE kullanımı için (en azından) kusursuz değildir.

İşletim sisteminin BDE etkinliğini arttırmak için gereken değişikliklerin bir kısmı bu bölümde belirtildi; yapılması elzem görülen diğer sistem değişiklikleri Balman 78c'de ayrıntılı olarak tanıtılmaktadır.

BÖLÜM 5

BİLGİSAYAR DESTEKLİ FORTRAN EĞİTİMİ

FCN (Balman 78f, 80a) birinci sınıftaki mühendislik öğrencilerine FORTRAN öğretiminde yardımcı olmak üzere geliştirilmiş etkileşimli bir programlama sistemidir. Londra Üniversitesi Queen Mary Koleji'nin Computer Assisted Teaching Unit'inde geliştirilen FCN, PDP11/40 bilgisayarının RSX-11M işletim sisteminde kullanılmaktadır (Balman 77).

FCN dört altsistemden oluşur

i) FORTRAN derleyicisi

ii) Kütük denetim sistemi

iii) Kütük değiştirme sistemi (editör)

iv) İşlem zamanı program deneticisi

Derleyici sistem adama ve desenli giriş/çıkış komutları için alıştırıcı nitelikte iki altsistemden oluşan bir artımlı (incremental) FORTRAN derleyicisidir.

CATU'da geliştirilen diğer eğitici programların aksine (Smith 76) FCN'in yazılımında programlanmış bir ders konusu olmadığından sistem öğrenciye tamamıyla açıktır. Yani, tüm programlama kavramlarının denenip hızla yorumlanmasını sağlayan genelleştirilmiş bir FORTRAN programlama sistemidir. Bu açıdan ele alınınca FCN öğrenciye bilgi verdikten sonra ona soru yönelterek başarısını ölçen PLATO programlarından (Montanelli 77) tamamıyla değişik bir yöntem izler.

İki alıştırıcı altsistem ve tam FORTRAN derleyicisi kullanıcıya gittikçe zorlaşan programlama tekniklerini tanıtır. Bu bakımdan öğrenciye PL/1 dilinde yeni teknikleri tanıtan 8 altsistemin oluşturduğu SP/k sistemine (Holt 77) benzemektedir. İki sistem arasındaki en önemli fark FCN'in etkileşimli kullanılmasıdır.

5.1. Motivasyon ve amaçlar

Tanıtıcı bir kursta çoğunlukla ilk ders ve ilk denenebilir FORTRAN programı arasında soyut programlama ile geçen bir zaman olur. Bu zaman sürecinde öğrenciye değişkenler, değişmezler, deyimler, atama komutları ve basit giriş/çıkış komutları gibi temel programlama araçlarıyle ilgili bilgi verilir. Bu süre sonunda öğrencinin algoritma ve programlarında kullanabileceği kuramsal nitelikte bilgisi olur. Fakat, öğrencilerin çoğu, program yazıp bilgisayarda deneyene dek bu araçların nasıl ve niye kullanıldığını anlayamaz. Bir çok programlama tekniği denenmeden tam olarak kavranamaz. Bu nedenle dersler pratik çalışmalarla birlikte yürütülürse öğrenci bazı fikirleri daha çabuk benimser. Öğrenciyi soyut örnek ve açıklamalarla doldurmaktansa bilgisayarla en kısa zamanda bir araya getirmek çok daha heveslendirici ve eğitici bir yöntemdir.

Öğrencinin bilgisayar kullanmaya geç başlamasının başlıca nedeni geleneksel olarak derleyicilerin sadece tamamlanmış programları kabul etmesidir. Yani öğrenci, en basit atama komutunu öğrendiği dersten hemen sonra bu komutu bilgisayarda deneyemez. Bu engel ancak basit komutları tek tek kabul edip işleyebilecek bir derleyicinin geliştirilmesiyle kaldırılabilir. Böyle bir sistemin tasarımında en büyük güçlük, öğretilen dilin özelliklerini bozmadan her komutun ayrı ayrı derlenmesi, işlenmesi ve sonuçların bildirilmesidir.

Tanıtıcı kursun her safhasında öğrencilere bilgisayar kullandırabilmek için kolaydan güçe doğru en az üç derleyici altsistemin gerekli olduğu görülmüştür En önce, öğrenciyi değişken, değişmez, deyim ve standart fonksiyon kullanmaya alıştırmak için her zorluktaki atama komutlarının denenmesini sağlayan bir altsistem gerekir. Bu altsistemdeki sonuç çıkışları kendiliğinden halledilmelidir. İkinci altsistem desenli giriş/çıkış kullanımını alıştıracak daha üst düzeyde bir sistemdir. Bu iki alıştırıcı altsistem öğrenciye yeterli deneyimi kazandıracak özellikleri içerdiğinden bir sonraki aşamada tamamlanmış program yazımına geçilebilir. Bu üçüncü derleyici altsistem diziler, DO-döngüleri ve altrutinler gibi orta düzeydeki kavramları kapsar.

Etkileşimli kullanımın öğrenciye daha iyi eğitim hizmeti verdiği bilinmektedir. Çevirim dışı kullanıma oranla, öğrenci hatalarını çok daha çabuk farkedip düzeltebilir ve daha fazla program yazma olanağı kazanır. Özellikle atama komutlarıyla desenli giriş/çıkış alıştırıcılarının kullanımı sırasında hızlı ve ayrıntılı beceri kazanmak bakımından etkileşimli yaklaşım daha etkindir.

Etkileşimli ortamda FORTRAN programlarının artımlı (incremental) çevirisi sağlanabilir (Gros 78). Bu yolla leksik ve sözdizim hataları hemen farkedilir ve kullanıcı yanlışlığı anında düzeltme olanağını elde eder.

FCN yaklaşımının önemli bir avantajı ders esnasında sınıfta kullanmaya elverişli olmasıdır. Etkileşimli programlama sistemi kapalı devre televizyonla sınıftaki ekranlara bağlanabilir ve yeni kavramlar ders esnasında somut örneklerle tanıtılabilir. Gereğinde sınıfa bir bilgisayar ucu getirerek gerçek örnekler gösterilebilir.

Etkileşimli sistemin en önemli sakıncası kullanıcının öğrenmesi gereken program yaratma, değiştirme, derleme, bağlama, başlatma ve kütük kullanma için gereken işletim komutlarının karmaşıklığıdır. Bu nedenle, öğrencinin gerçek etkileşimli işletim komutlarını günün birinde öğrenmesi gerektiği halde, programlama sisteminin anlaşılması ve kullanımı kolay olan bir işletim ortamına alınması uygun görülmüştür. Böylece öğrenci işletim komutlarını öğrenerek geşireceği zamanı programlama tekniklerini öğrenerek daha çok değerlendirebilir.

5.2 İşletim ortamı ve komutları

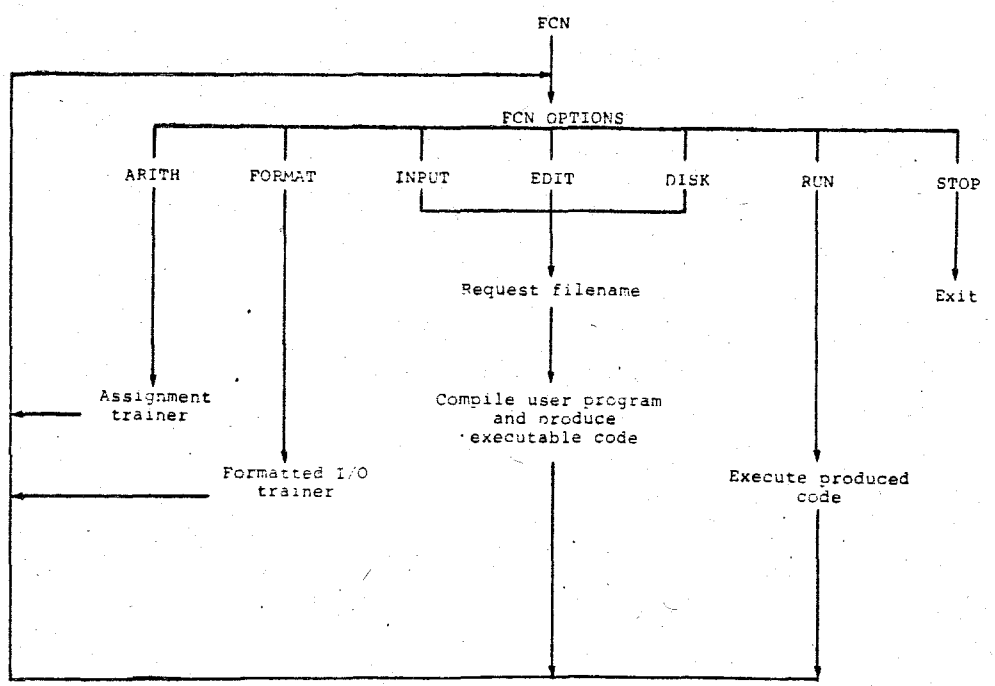
Daha önce de belirtildiği gibi FCN'in geliştirilmesinde göz önüne alınan kriterlerden biri işletim komutlarının deneyimsiz programcı için en kolay düzeyde tutulmasıydı. Öğrenci FCN sistemine yerel olarak geliştirilmiş sistem erişim programı (Balman 78 b, Bölüm 4.2.2) aracılığıyla kolaylıkla erişebilir (Şekil 5-1). Erişim için kullanıcı tanıtıcısı ve kullanım parolası belirtmeye gerek yoktur. FCN sistemine girmek için kullanılmayan herhangi bir uçta "FCN"

```
Q.M.C. C.A.T.U. PDP11/40 WITH RSX-11M V3.1

GOOD AFTERNOON!
PACKAGE NAME ? FCN
>
>
>

Q.M.C. C.A.T.U. CONVERSATIONAL FORTRAN SYSTEM

FCN VERSION 1.0, JUL-78
```

Şekil 5.1 : FCN sistemine erişim

yazılması yeterlidir. FCN başlatılınca kullanıcı gerekli anahtar sözcüğü belirterek herhangi bir sistem olanağını seçebilir. Seçilen sistem hizmetinin bitişinde kullanıcı tekrar seçme düzeyine döner.

FCN'in anahtar sözcükleriyle seçenek yapısı Şekil 5.2'de verilmiştir. Öğrencinin kullanabileceği altı esas işletim komutu vardır



Şekil 5.2 : FCN seçenek yapısı

1. Atama komutu alıştırıcısı - ARITH sözcüğü
2. Desenli giriş/çıkış alıştırıcısı - FORMAT sözcüğü
3. Kullanıcı programın ilk çevirimi ve girilen FORTRAN kaynak komutlarının kütüğe yerleştirilmesi INPUT sözcüğü
4. Program kütüğündeki kaynak komutların değiştirilerek derlenmesi-EDIT sözcüğü
5. Program kütüğündeki kaynak komutların çevirisi DISK sözcüğü
6. 3, 4, 5. maddelerin ürettiği işlenebilir kodun işleme alınması-RUN sözcüğü.

İlk beş FCN olanağından çıkış "FINISH" komutunun verilmesiyle gerçekleşir. RUN olanağından çıkış ise FORTRAN "STOP" komutunun işlenmesiyle veya bir işlem hatasının görülmesiyle gerçekleşir. Her iki durumda da kullanıcı seçim düzeyine getirilir. Seçim düzeyinde "STOP" sözcüğünün yazılmasıyla FCN durdurulur.

Hata mesajlarının biçimini ve kütükten derlerken kaynak komutların dökümünü değiştiren, nisbeten önemsiz, dört anahtar sözcük daha vardır. Bu seçenekler öğrenciye ancak sistemde yeterli deneyim edindikten sonra öğretilir.

Dikkate değer bir husus, öğrencinin atama komutu alıştırıcısını kullanmak üzere bilgisayarla yapacağı ilk temasta öğrenmesi gereken FORTRAN-dışı işletim komutları yalnız "FCN" ve "ARITH" sözcükleri olmasıdır. Kullanıcı tam bir program yazmayı öğrenene kadar diğer sistem olanaklarını tanımak zorunda değildir.

5.3 Atama komutu alıştırıcısı

Öğrenci ilk programlama deneyimini atama komutu alıştırıcısını kullanarak elde eder. Kullanıcı değişken, değişmez, standart fonksiyon ve işlem işaretlerinden türetilmiş gerçek veya tam sayı değerli herhangi bir deyimi kullanarak atama komutu yaratabilir.

Alıştırıcı, verilen komutları tek tek alarak çözümledikten sonra işlenebilir kodunu üretir ve üretilen kod hemen işlenir. Atama komutunun solundaki değişkene atanan değer uca yazılarak öğrenciye bildirilir. Böylece çıkış komutlarını ayrıca belirtmeye gerek kalmaz. Elde edilen geçerli tüm değişken değerleri bellekte

tutulduğundan öğrenci oldukça karmaşık ve gerçekçi sayısal problemler çözen komut dizileri yaratabilir. Tutulmakta olan değişken değerlerini silip bir sonraki problemin değişken değerlerini hazırlamak için FORTRAN'a benzer bir "RESET" komutu sağlanmıştır.

Alıştırıcının işlem döngüsünde dört basamak vardır:

1. Kaynak komutun okunması
2. Komutun çözümlemip sözdizim ve anlam hatalarının saptanması
3. İşlenebilir kodun üretilip işlenmesi
4. Atanan değerin yazılması

İkinci basamakta normal sözdizim hataları yanısıra şu anlam hataları denetlenir:

1. Henüz değer almamış olan değişkenlerin sağ taraftaki deyimde kullanılması
2. Deyimlerde karma aritmetiğin (tam sayı ve gerçek sayı karışık) kullanılması
3. Standard fonksiyonlara yanlış tipte ve yanlış sayıda argüman verilmesi

Komutların tek tek çözümlenmesi ve bekletilmeden işleme alınması bir sonraki komuta geçilmeden hatalı komutun düzeltilebilmesini sağlar. Öğrenciye mümkün olan her yerde, yanlışın açıklanarak ve hataya neden olan kısım gösterilerek yardım edilir. Bütün işlem hataları yazılan en son komutla ilgili olduğundan hata ayıklamak kolaydır. Bu alıştırıcıda yapılabilecek işlem hataları ancak üst taşma, sıfırla bölme ve fonksiyonlara geçersiz argüman değeri verme olabilir.

Bu yaklaşım öğrenciye derste yapılan örneklere ilişkin deneme yapabileceği rahat bir ortam verir. Deyim hazırlanması, gerçek ve tam sayı aritmetik arasındaki farkı görmesi, değişkenlere atanan değerlerin daha sonraki komutlara nasıl yansıdığını görmesi ve standart fonksiyonların kullanımında tecrübe kazanması için öğrenci deneyleri bu alıştırıcıyla teşvik edilir. Alıştırıcıdan hem geçerli değişken ve değişmezlerle deney yapan öğrenci, hem de gerçekçi sayısal problemleri çözmek için komut zincirleri hazırlayan öğrenciler yararlanabilir.

Atama komutu alıştırıcısından seçilmiş örnekler Şekil 5.3'de verilmiştir.

```
OPTION - ARIT

FCN ARITHMETIC EXERCISER MODULE

-       CENTIGR=25.E-1.5
*ERROR*
"CENTIGR" IS AN ILLEGAL VARIABLE NAME
MAX.NO OF CHARS ALLOWED IS 6
RETYPE THE LINE
-       CENT=25.E-1.5
*ERROR*
"25.E-1." HAS A "." IN THE EXPONENT
RETYPE THE LINE
-       CENT=25.0*(10.**-1.5)
VALUE =  0.790569
-       FAHR=CENT*9/5+32.0
*ERROR*
"*" IS APPLIED TO INCOMPATIBLE TYPES
RETYPE THE LINE
-       FAHR=CENT*9.0/5.0+32.0
VALUE =   33.4230
-



        X=0.51
VALUE =  0.510000
-       PHI=-1.6
VALUE =  -1.60000
-       X1=EXP(X)
VALUE =   1.66529
-       A=COS(PHI)*X1+SIN(PH)/X1
*ERROR*
"PH" DOES NOT HAVE A VALUE AT THIS POINT
ITS USE IS NOT ADVISED
RETYPE THE LINE
-       A=COS(PHI)*X1+SIN(PHI)/X1
VALUE =-0.648865
-       Y=ALOG(A/(1.0+A*A*(1.0+A))
*ERROR*
MISSING ")"
RETYPE THE LINE
-       Y=ALOG(A/(1.0+A*A*(1.0+A)))

**EXECUTION ERROR**
LOGARITHM OF NEGATIVE VALUE
-       A=ABS(A)
VALUE =  0.648865
-       Y=ALOG(A/(1.0+A*A*(1.0+A)))
VALUE =-0.959750
-
```

Şekil 5.3 : FCN atama komutu alıştırıcısından örnekler.

5.4 Desenli giriş/çıkış alıştırıcısı

Yeni programcılar atama komutu alıştırıcısı ve onun tek tip çıkışından sonra desen alıştırıcısını kullanmaya teşvik edilir. Atama komutlarının yanısıra bu altsistemde READ, WRITE ve FORMAT komutları da kullanılabilir. Bir önceki alıştırıcıdaki gibi bütün komutların karşılığı olan işlem kodu yaratılınca hemen işlenir. Fakat, değişkenlere değer atandığında bu değerler kendiliğinden yazılmaz. Tanımlanan tüm desen ve değişken değerleri "RESET" komutu yazılana dek tutulur.

Bu alıştırıcının tasarımında karşılaşılan bir problem iki komuttan oluşan desenli READ/WRITE komutunun hemen işleme sokulmasıydı. Komutta kullanılan FORMAT daha önceden tanımlanmışsa komut derlendikten hemen sonra işleme geçilebilir. Desen daha önceden tanımlanmamışsa öğrenci kullanmak istediği deseni tanımlayana dek kod işlemi bekletilir; komut karışıklığını önlemek amacıyla, tanımlayıcı komutu girmesi için, öğrenci uyarılır ve desen tanımlanana dek başka komut kabul edilmez. Desen özelliklerinin belirtilmesi ile onu kullanan giriş/çıkış komutunun kodu işlenir.

Genellikle öğrenciler giriş/çıkışta kullanılan değişken listeleri ile desen öğeleri arasındaki ilişkiyi öğrenmekte güçlük çeker. Komutların tek tek okunup çözümlenmesi ve işlenmesiyle hazırlamış olduğu desenin etkisi öğrenciye hemen gösterilir. Öğrenci kısa zamanda çeşitli desenler deneyerek değişkenle desen öğesi arasındaki bağlantıyı görebilir. Bu alıştırıcıyı bir saat kullanan öğrenci bir dönem boyu çevirim dışı olarak kullanacağı giriş/çıkış komutlarından daha fazla desen deneme fırsatı elde eder .

Aynı alıştırıcı içerisinde atama komutlarının da kabul edilmesi kullanıcılara sonucunu yazdırabilecekleri kısa programlar hazırlama olanağını sağlar. Fakat, her komut işlendikten sonra yok edildiğinden bu kısa programlar yalnız bir defa işlenebilir. Komut yalnız bir defa işlendiğinden ve READ komutu kullanıcıdan hemen veri istediğinden, bu safhada öğrenciyi giriş komutunun yararlarına inandırmak güçtür. READ'in ne kadar gerekli bir komut olduğunu kullanıcı daha sonraki bir safhada tam program yazmaya başlayınca takdir eder. Bu gecikme öğrencinin desenli giriş ve verinin desene göre ayarlanması ile ilgili

da uygulanabilecek yararlı bir komut dizisi okuma komutunun ardından gelen ve kullanıcıya hatalı veriyi gösterebilen yazma komutudur.

Bu alıştırıcı ilkinin aksine tanıtıcı kursun bitiminde bile önemini ve geçerliliğini kaybetmez. Öğrencinin daha yeni ve daha zor giriş/çıkış tekniklerini deneyebileceği, her zaman için geçerli olan bir ortam sağlar.

Desenli giriş/çıkış alıştırıcısından alınan bir takım örnekler Şekil 5.4'de verilmiştir.

5.5 Artımlı FORTRAN derleyicisi

Etkileşimli-artımlı derleyici etkileşimli-yorumlayıcı çeviri ile derleyici arasında bir çözümleme programıdır. Komutlar alındıkları sırayla çözümlenir ve her komutun karşılığı olan işlenebilir kod öbeği üretilir (Berthand 73). İşlenilenlerin adresleribütün komutlar çevirildikten sonra koda eklenir. Artımlı derleyicinin yorumlayıcı çeviriye oranla üstünlüğü hızıdır-çünkü işlem için gerçek kod üretilir. Derleyici ile karşılaştırılırsa etkileşimli ortamda hem derleyici hem de ürettiği kod aynı anda bellekte tutulduğundan daha çok bellek kullanır.

Artımlı derleyici her komutu bulunduğu programlama ortamından bağımsız olarak çözümleyip tüm sözdizim hatalarını bir sonraki komutu beklemeden denetleyebilir. Böylece öğrencinin yazdığı her komutun çözümü bir sonraki komut yazılmadan önce tamamıyla çözümlenmiş olur. Yani, etkileşimli bir ortamda, öğrenci komutu yazar yazmaz sözdizim hatalarını görebilir ve bu hataları bir sonraki komutu yazmadan önce giderebilir.

FORTRAN etkileşimli-artımlı derleme yönteminin uygulanabileceği bir dildir. Tek istisna FORTRAN komut devam kartıdır-çevirici bir sonraki satır yazılana dek varlığını ferkedemez. Derleyici öğrenciye hatalı olan her komutu hemen değiştirme fırsatı tanıyacaksa komutun devam edip etmeyeceğini görmek için bir sonraki satırı okuyamaz; gelen satırda devam işareti yoksa bir önceki satırı değiştirme olanak dışıdır. FORTRAN dilinin bu özelliğini FCN'e uyacak şekilde değiştirmektense devam kartı kullanımını artımlı derleyiciden tamamen çıkarma tercih edilmiştir.

```
OPTION - FORM

FCN INPUT/OUTPUT AND FORMAT EXERCISER MODULE

-       F=25.5**4
-       WRITE(2,10)F
PLEASE DEFINE FORMAT 10
-10     FORMAT(9H RESULT IS,F8.2)
*ERROR*
"(9H RESULT I" IS AN ILLEGAL HOLLERITH STRING SPECIFIER
WRONG CHARACTER COUNT BEFORE "H" ?
RETYPE THE LINE
-10     FORMAT(10H RESULT IS,F8.2)
RESULT IS422825.0
-15     FORMAT(8H RESULT=,E9.4)
*ERROR*
",E9.4)" -- WIDTH IS TOO NARROW FOR REQUIRED ACCURACY
RETYPE THE LINE
-15     FORMAT(8H RESULT=,E11.4)
-       WRITE(2,15)F
RESULT= 0.4228E+06
-       A=SQRT(F)/2
*ERROR*
"/" IS APPLIED TO INCOMPATIBLE TYPES
RETYPE THE LINE
-       A=SQRT(F)/2.0
-       WRITE(2,15)A,F
RESULT= 0.3251E+03
RESULT= 0.4228E+06
-
-
-

-20     FORMAT(3I5)
-       READ(1,20)IJ,IK,IL
  25    29   12
-25     FORMAT(1X,3('NEXT=',I5)
*ERROR*
MISSING ")"
RETYPE THE LINE
-25     FORMAT(1X,3('NEXT=',I5))
-       WRITE(2,25)IJ,IK,IL
NEXT=   250NEXT=  290NEXT= 1200
-30     FORMAT(1X,3('NEXT=',I5,1H,,2X))
-       WRITE(2,30)IJ,IK,IL
NEXT=   250,  NEXT=  290,  NEXT= 1200,
-       READ(1,20)IJ,IK,IL
   25    29    12
-       WRITE(2,30)IJ,IK,IL
NEXT=    25,  NEXT=    29,  NEXT=    12,
-
-
-
-       FINISH
```

Şekil 5.4 : FCN desenli giriş/çıkış alıştırıcısından örnekler.

FCN'in artımlı derleyicisi sözdizim hatalarını belirtmenin yanısıra kullanıcı komutları hakkında çeviri sırasında biriken bilgiden yararlanarak komut uyumunu denetlemeye çalışır. Bu denetimler aşağıda belirtilen hataları program komutları girilirken farkedip kullanıcıya düzeltme olanağını sağlar:

1. Aynı komut işaretinin hem desen hem de işlenebilir komut tanımlaması; desen olarak ima edilen işaretlerin işlenebilir komut işareti olarak kullanılması
2. İçiçe olan DO-döngülerinde komut işaret hataları
3. Altrutinlere geçirilen argümanların tip ve sayılarındaki uyuşmazlıklar
4. Değişkenlerin değer almadan önce deyimlerde kullanılması
5. Tanımlanmamış dizilere değer adanması

Bu denetimlerin avantajı programın yaratmış olduğu işlem ortamına uymayan komutların da düzeltilebilmesidir. Bu kolaylığı sağlamak için derleyici sistem tutarsızlığın kullanıcının son yazdığı komuttan kaynaklandığını varsaymak zorundadır; hatanın farkedildiği an kullanıcı yazmış olduğu en son komutu değiştirmeye zorlanır. Tutarsızlık, hemen düzeltme olanağı olmayan daha önceki bir komutun veya onun yokluğunun bir sonucu ise, kullanıcı son komutunu değiştirerek önceden yapmış olduğu hatanın yarattığı hatalı ortama uymaya zorlanır. Böyle durumlarda yukarıda belirtilen ilk iki gruba giren yanlışları düzeltmek kolaydır; kullanıcı hatadan kurtulmak için komut işaretleme yöntemini biraz değiştirebilir. Diğer tutarsızlıklar daha önceki bir hatadan kaynaklanıyorsa değiştirici altsistemin yardımı olmadan hatadan kurtulmak genellikle olanak dışıdır.

Bazı FORTRAN hataları ancak bölüm sonu (END) ve program sonu (FINISH) komutlarından sonra ortaya çıkar. Bunların uçta hemen düzeltilmeleri değiştirici altsistemin yardımı olmadan olanaksızdır. Bu hatalar kullanıcı bölüm veya programı bitirince fark edilir ve iletilir.

END komutu kullanıcı programda üç denetimi başlatır. İlk önce bölümde tanımlanmadan kullanılan işaretler saptanır. Sonra, bölüm bir altrutinse, bölüm başlığında tanımlanan parametrelerin tip ve sayısı daha önceki kullanımlarıyla

```
OPTION - INPUT

FILE NAME - PRS89

CREATING NEW PROGRAM

-C READ N NUMBERS AND SORT
-       MASTER SORT
-       DIMENSION NOS(10)
-C READ N AND THE NUMBERS
-       WRITE(2,5)
-       READ(1,10)N
-       WRITE(2,15)N
-       READ(1,20)(NOS(I),I=1,N)
-C NOW START SORT
-       NN=N-1
-       DO 10 I=1,NN
*ERROR*
LABEL PREVIOUSLY USED FOR A FORMAT
RETYPE THE LINE
-       DO 100 I=1,NN
-C ASSUME NOS(I) IS MIN
-       MINIM=NOS(I)
-       MINSB=I
-       J=I+1
-C NOW CHECK ASSUMPY\Y\TION AND ALTER IF NECESSARY
-       DO 200 K=J,M
*ERROR*
"M" DOES NOT HAVE A VALUE AT THIS POINT
ITS USE IS NOT ADVISED
RETYPE THE LINE
-       DO 200 [\[\K=J,N
-       IF(NO(K).GE.MINIM)GO TO 200
-       MINIM=NOS(K)
-       MINSB=K
-100    CONTINUE
*ERROR*
ILLEGAL NESTING OF DO LOOPS
RETYPE THE LINE
-200    CONTINUE
-C NOW EXCHANGE
-       MM=NOS(I)
-       NOS(I)=NOS(MINSB)
-       NOS(MINSB)=MM
-C WRITE THE SORTED NUMBERS AND STOP
-       WRITE(2,30)(NOS(I),I=1,N)
-       STOP
-C FORMATS
-5      FORMAT(' HOW MANY NUMBERS')
-10     FORMAT(I3)
-15     FORMAT(' TYPE IN',I3,' NUMBERS')
-30     FORMAT(' THE SORTED LIST:',/,1X,2014)
-       END
UNDEFINED LABEL(S) :
20
100



-       FINISH
UNDEFINED SEGMENT(S) :
NO


OPTION -
```

Şekil 5.5 : FCN'de program yaratılışı

tutarlı olup olmadığı saptanır. Son denetim ise COMMON alanlardaki tip, genişlik ve sınır tutarlılığı ile ilgilidir. Öğrenci parametre ve COMMON alandaki dizilerin tip ve uzunluğunu bölüm başındaki ilk tanımlamadan sonra tanımlama komutlarıyla değiştirebileceğinden bu denetimler END komutuna kadar bekletilir.

FINISH komutu ile adı geçen tüm altrutinlerin varlığı denetlenir. Tanımlanmayan altrutin ve fonksiyonlar kullanıcıya bildirilir.

Programların etkileşimli-artımlı derlenmesi öğrencinin bütün sözdizim hatalarını yapar yapmaz terminalde giderebilmesini sağlar. Hata eğer çözümlenmekte olan komuttaki anlam bozukluğundan doğuyorsa komut uyumsuzlukları da hemen düzeltilebilir. Hata, daha önce verilen ve sözdizimi kusursuz olan bir komuttan kaynaklanıyorsa genel olarak değiştirici altsisteme başvurmadan düzeltmek olanaksızdır.

Artımlı derlemeyle ilgili bir örnek Şekil 5.5'de verilmiştir.

5.6 Program kütükleri

İşlem hatalarının ayıklanıp düzeltilmesi ve yaratılan programların geliştirilmesi kullanıcının yarattığı kaynak komutlarının ikincil bellekte tutulmasını zorunlu kılar. FCN tüm yeni programları kendiliğinden ikincil bellekte saklar. FCN kullanıcıları yeni programcılar olduğundan yazdıkları programların çoğu 100 satırı aşmaz; sınırsız kütük kullanma kolaylığının istismar edilebileceğini de gözönüne alırsak dönem boyunca, çok sayıda kullanıcı olmadığı takdirde, bütün yeni programların ikincil bellekte saklanması ciddi bir yer problemi yaratmaz.

FCN, INPUT işletim komutuyla yaratılan her yeni program için, tanıtıcı bir isim ister. Bu isim kullanıcının komutlarıyla doldurulacak kütüğe adanır. Sistemde kütük yönetimiyle ilgili ve öğrencinin kullanabileceği hiçbir komut yoktur. Kullanıcının anahtar sözcüklerle seçtiği altsistemler gereken kütük işlemlerini gizli olarak yürütür.

Yeni programların giriş ve derlenmesi sırasında (INPUT opsiyonu) kullanıcının yazdığı her komut çözümlenir ve hatalı değilse, kullanıcının program kütü-

ğüne alınır. FINISH komutunun yazılmasıyla kütük kapanır. Bu şekilde uçtan girilen her program derlendikten sonra kullanılan tüm kaynak komutlarının ikincil bellekte saklanması sağlanır.

Program kütüğünün kendiliğinden yaratılması öğrencinin program saklama yükümlülüğünü kaldırdığından avantajlıdır. Ancak kullanıcıların sayısı çok ise seçimsiz her programın saklanması ikincil bellekte taşma yaratabilir.

Kütükteki kaynak komutlarını tekrardan derlemek için DISK opsiyonu kullanılır. Bu şekilde komut giriş ortamı terminal yerine program kütüğü olur. Kütükten okunan her komut satırı uca yazılıp artımlı derleyiciye gönderilir. Bütün komutların doğruluğu daha önce kütüğün yaratılması veya değiştirilmesi sırasında sağlandığından derleme kütük sonuna kadar devam eder.

5.7 Program kütüklerinin değiştirilmesi

EDIT opsiyonu program kütüğündeki komutların derlenişi sırasında komut ekleme, komut çıkarma ve komut değiştirme hizmeti sağlar. Bitiminde bu değiştirici düzeltilmiş program kütüğüyle programın işlenebilir kodunu hazırlar.

Kütük değiştirici sadece en gerekli satır biçimlendirme olanağını sağlar. Daha güçlü bir karakter değiştirici beraberinde getireceği çok sayıda kullanım komutları, öğrenim güçlüğü ve geniş bellek gereksinimi göz önünde tutularak sağlanmamıştır. Kütük değiştirici kolayca kullanabilen ve kütük değiştirirken aynı anda komut derleyebilen bir altsistem olarak tasarlanmıştır. Bu, alışılagelen iki ayrı aşamada değiştirme ve derleme işlemlerinden, çok daha güçlü ve yepyeni bir yaklaşımdır. Yaklaşımdaki amaç program komutlarını değiştirirken bile öğrenciye hatasını hemen bildirip hatalı değişiklikleri önlemektir. İki aşamanın birleştirilmesinden doğan çok önemli bir yan etki program değiştirilirken işlenebilir kodun artımlı olarak üretilmesidir. Böylece değiştiriciden çıkar çıkmaz kullanıcı programını hemen başlatabilir.

Değiştiricide kullanılan işletim yöntemi BASIC'den esinlenmiştir. FORTRAN'da bulunmayan ve BASIC dilinin bir özelliği olan satır numaraları değiştiricinin ürettiği komut sayılarıyla sağlanmaktadır. Değiştirici değiştirilecek kütükteki

FORTRAN komutlarının numaralanmış dökümünü paylaşılan süratli yazıcıya gönderir (Balman 78d). Sayılar gittikçe büyüyen 10 sayısının katlarıdır. Kullanıcı 10'un katı olan bir sayıyı belirterek değiştirilmesini arzuladığı komutu belirtir. Komut silme işlemi satırın boş bir satırla değiştirilmesiyle gerçekleştirilir. Yeni bir komut ekleme istemi bir önceki ve bir sonraki satırların sayıları arasında bir sayı ile bildirilir. Kullanılan numaralama yöntemi öğrenciye orijinal program kütüğünde herhangi iki komut arasına en fazla dokuz komut ekleme olanağı sağlar.

Değiştirici kullanıcının verdiği satır numarasıyla yönetilir. Verilen sayının belirlediği komuta doğru, aradaki satırlar uca yazılıp derleyiciye gönderilerek, ilerlenir. Değiştirici belirtilen program kütüğünün belirtilen yerinde durunca, üzerinden geçilip derlenmiş olan komutların yarattığı bilgi ortamı hazırdır. Bu suretle kullanıcı tarafından yazılan yeni satırların doğruluğu ve geçmişteki komutlara uyumu denetlenbilir; eğer gerekirse yeni komutun düzeltilmesi istenir. Yeni komutun kabulünden sonra değiştirici başka bir satır numarası ister ve işlem aynı şekilde sürdürülür.

Değiştirici eksi bir satır numarasıyazılarak öldürülebilir. Normal bitiş ise program kütüğündeki en son komuttan sonra FINISH eklenerek gerçekleştirilir. Bunun sonucunda değiştirilen program kütüğü kapanır ve değiştiricinin yarattığı işlenebilir kod işleme hazır duruma getirilir.

Değiştirici içinde artımlı derleyici kullanımı öğrenciye yeni program yaratılırken yapılan yardımın, program değiştirilirken de devam ettirilmesini sağlar. Editör içindeyken programların aynı zamanda derlenmesi ayrı bir derleme istemini gereksiz kılar. Öğrenci değiştiriciden çıktığında programı hemen işleme geçirebilir.

Bir değiştirme örneği Şekil 5.6'da verilmiştir.

```
OPTION - EDIT

FILE NAME - PRS89

FCN EDITOR
LINE-NUMBERED LISTING OF PROGRAM SENT TO FAST PRINTER

NO. - 180
C READ N NUMBERS AND SORT
      MASTER SORT
      DIMENSION NOS(10)
C READ N AND THE NUMBERS
      WRITE(2,5)
      READ(1,10)N
      WRITE(2,15)N
      READ(1,20)(NOS(I),I=1,N)
C NOW START SORT
      NN=N-1
      DO 100 I=1,NN
C ASSUME NOS(I) IS MIN
      MINIM=NOS(I)
      MINSB=I
      J=I+1
C NOW CHECK ASSUMPTION AND ALTER IF NECESSARY
      DO 200 K=J,N
-       IF(NOS(K).GE.MINM)GO TO 200
*ERROR*
"MINM" DOES NOT HAVE A VALUE AT THIS POINT
ITS USE IS NOT ADVISED
RETYPE THE LINE
-       IF(NOS(K).GE.MINIM)GO TO 20
*ERROR*
"20" IS THE LABEL OF A FORMAT
RETYPE THE LINE
-       IF(NOS(K).GE.MINIM)GO TO 200
NO. - 255
      MINIM=NOS(K)
      MINSB=K
200   CONTINUE
C NOW EXCHANGE
      MM=NOS(I)
      NOS(I)=NOS(MINSB)
      NOS(MINSB)=MM
-100   CONTINUE
NO. - 325
C WRITE THE SORTED NUMBERS AND STOP
      WRITE(2,30)(NOS(I),I=1,N)
      STOP
C FORMATS
5     FORMAT(' HOW MANY NUMBERS')
10    FORMAT(I3)
15    FORMAT(' TYPE IN',I3,' NUMBERS')
-20    FORMAT(15I5)
NO. - 900
30    FORMAT(' THE SORTED LIST:',/,1X,2014)
      END


-       FINISH
```

Şekil 5.6 : FCN'de program kütüğünün değiştirilirken derlenmesi.

5.8 Derlenen programların başlatılması

İşlem kodları kullanıcının kaynak komutlarının yaratılması, değiştirilmesi ve okunması (INPUT, EDIT ve DISK opsiyonları) sırasında hazırlanır. Hazırlanan kod FCN'in kendi programlama alanı içinde saklanır ve bu üç hizmet altsistemlerinin herhangi birinden çıkıldığı anda işleme alınabilir. END ve FINISH komutlarının yorumu sırasında herhangi bir program eksikliği farkedilirse kod işlenemez olarak işaretlenir. Böylece hatalı programların başlatılması önlenir.

Artımlı derleyicinin hazırladığı kodun işlenmesi RUN sözcüğüyle başlatılır. Kullanıcı, bir işlem hatasına bağımlı olarak veya FORTRAN STOP komutunun işlenmesiyle opsiyon düzeyine getirilir. Program değişkenlerine ayrılan alan her RUN'dan önce temizlendiğinden,öğrenci arzu ettiği kadar programı tekrarlatabilir. Bu şekilde program tekrar derlenmeden değişik veri takımları ile işletilebilir.

Program hatalarının ayıklanmasında yardımcı olmak amacıyla derleyicinin ürettiği her kod öbeğinde hangi komutun çevirisi olduğunu belirten bir sayı yerleştirilir. FCN işlemde olan komutun sayısıyla yapılan altrutin çağrı ve dönüşlerini denetlediğinden işlem hatasına neden olan komutun yerini bulabilir. Program kaynağı devamlı olarak ikincil bellekte tutulduğu için hataya neden olan komut kütükten okunarak öğrenciye iletilebilir. Anormal bir durumu belirtirken hatanın niteliği, hatayı yaratan komut ve bulunduğu bölüm öğrenciye bildirilir.

INPUT-RUN ve EDIT-RUN opsiyon zincirleri öğrenciye, programı hazırlayıp değiştirdikten sonra hemen başlatma kolaylığını sağlar. Böylece alışılagelmiş program yaratma, program derleme, kod bağlama ve program başlatma aşamaları, ve bunlara ilişkin işletim komutları gereksiz kılınmıştır. Kullanıcıya programındaki işlem hatalarını ayıklayabilmesi için de yardım edilir. Bellek kısıtlığı nedeni ve sistemi acemi kullanıcıya gereksiz bir şekilde güçleştirme endişesiyle programın işlem-zamanı tarihçesinin etkileşimli soruşturma olanağı sağlanmamıştır. İşlem hatasına neden olan komutun gösterilmesi basit programlarda yardımcı olurken karmaşık programlarda genellikle yetersiz olur.

Kullanıcı programlarının işletiminden alınan bir kaç örnek Şekil 5.7'de verilmiştir.

```
OPTION - RUN

RUN STARTED

HOW MANY NUMBERS
5
TYPE IN5ØØ NUMBERS
   25   -5  265    6  -26

**EXECUTION ERROR**
WHEN USING ARRAY  NOS
THE SUBSCRIPT WAS TOO LARGE
IN SEGMENT MASTER, AT LINE NO.   8 :
      READ(1,20)(NOS(I),I=1,N)


OPTION - RUN

RUN STARTED

HOW MANY NUMBERS
  5
TYPE IN  5 NUMBERS
   25   -5  265    6  -26
THE SORTED LIST:
 -26  -5   6  25 265


OPTION - RUN

RUN STARTED

HOW MANY NUMBERS
  8
TYPE IN  8 NUMBERS
-2654   -4  -25 -264  368  669  -99  100
THE SORTED LIST:

**EXECUTION ERROR**
INSUFFICIENT FIELD LENGTH IN OUTPUT FORMAT
IN SEGMENT MASTER, AT LINE NO.  28 :
      WRITE(2,30)(NOS(I),I=1,N)


OPTION - RUN

RUN STARTED

HOW MANY NUMBERS
  8
TYPE IN  8 NUMBERS
 -265   -4  -25 -264  368  669  -99  100
THE SORTED LIST:
-265-264 -99 -25  -4 100 368 669


OPTION -
```

Şekil 5.7 : FCN'de derlenen programın işlenmesi.

5.9 Öğretim yöntemi

FCN derslerin yerini almayı değil de derslere destek olmayı amaçlar. Tanıtıcı bir kursun her safhasında yoğun alıştırmalar için kullanılacak bir programlama sistemi yaratma gereksiniminden doğmuştur. Tasarımında hedef alınan unsurların biri öğrenmesi ve kullanması kolay olan bir işletim ortamında bu tip bir hizmetin öğrenciye sağlanmasıydı.

Sistem önceden kararlaştırılmış öğretici bir metni kapsamaz ve öğrencinin yetenek ve becerisini sınamaz. Mümkün olduğu kadar esnek ve genel yazılmıştır. FCN kullanan öğrenci,danışarak veya kendi başına, öğrenmek istediği komuları dener. Kursun başlangıcında sistem öğrenciye temel programlama tekniklerini geliştirme ve somut örneklerle tatmin olma fırsatını vererek yardımcı olur. Sözdizim hatalarının ve sonuçların hemen verilmesi, deyim kurmada daha çabuk beceri kazandırır. Problemli bir konu olan desenli READ/WRITE komutlarında, öğrenciye deneyip izleyerek öğrenebileceği bir alıştırıcı sağlanır. Daha sonraki safhalarda öğrenciye sözdizim hataları erkenden açıklanarak yardımcı olunur; böylece çevirim dışı kullanımın heves kırıcı bir özelliği yok edilir. FCN, problem analizi ve algoritma geliştirme tekniklerini öğretmez; sadece fikirlerin hızla denenip düzeltilebileceği bir ortam sağlar.

Sistem,değişik akademisyenlerin gereksinimlerini karşılayacak ve FORTRAN öğretimine yaklaşımlarını yansıtacak kadar geneldir. Bu esneklik kullanıcıların FORTRAN komutlarını bilinçsiz bir şekilde denemelerini de kolaylaştırır. Deneme kolaylığı öğrencileri bazen dikkatle tasarlanmış programlama yerine acele programlamaya yöneltir. Bu sebepten öğrenciye, FCN'den yararlanabilmesi için, önemli noktaları vurgulayan örneklerden oluşan bir rehber verilmelidir (Balman 78a).

Bu yol gösterici yaklaşım özellikle alıştırıcı modüllerin kullanımında yararlı olur (Şekil 5.8). Rehber beceri sınayan alıştırmaları da içerirse öğrencinin FCN ile yaptığı çalışmalar bir dereceye kadar denetlenebilir. Bu rehberin derslerde yeni tekniklerin tanıtılması sırasında çok yararlı olmasına karşın öğrencinin bağımsız araştırma hevesini kıracak derecede katı olmaması gerekir.

when the computer is ready to read a value for ILG it will sound the bell at your terminal and wait for input. Type in

▽▽▽28 (▽=blank)

Note that the format I5 means that the input value should be pushed as far to the right as possible in a field of width 5. To check that the correct value has been read try

```
*1001   FORMAT(1H ,I5)
*       WRITE(2,1001)ILG
```

It is important to place the input values as far to the right within the specified field as possible. Try the two examples below to see the potential dangers of not conforming to the specified format.

```
*1010   FORMAT(2I4)
*1011   FORMAT(1H ,2I4)
*       READ(1,1010)I1,I2
```

and as input

▽24▽▽215

Also try

```
*       READ(1,1010)I3,I4
```

with input

▽▽▽26▽28

To see the result of these two READ statements do a

```
*       WRITE(2,1011)I1,I2,I3,I4
```

Şekil 5.8 : FCN kullanıcı rehberinden bir sayfa (Balman 78a)

5.10 Sonuç

FCN Eylül 1978'den beri CATU'da kullanılmaktadır. 1978'in Kış döneminde deneysel olarak küçük bir öğrenci grubuna sunulmuştur. Ertesi sene Nükleer ve İnşaat Mühendisliği bölümlerinden 45 birinci sınıf öğrencisi FCN'i toplam olarak yaklaşık 1000 saat kullanmıştır. Kursun ilk safhalarında kapalı devre televizyonla FCN örnekleri bilgisayar odasından sınıfa naklen yayınlandı. Her öğrenci 12 hafta içinde yaklaşık 24'er saat FCN sisteminden yararlandı. Öğretim üyeleriyle öğrencilerin olumlu izlenimleri üzerine bu olanağın tüm fakülteye açılması kararlaştırıldı.

1980 senesinin Kış döneminde fakültenin yaklaşık 120 birinci sınıf öğrencisi FCN sistemini yaklaşık olarak 3000 saat kullandı. Bu dönemde kapalı devre televizyon yayım yerine bilgisayar ekranı sınıfa getirilerek ders esnasında FCN'den örnekler sunuldu. Bu yaklaşım daha başarılı oldu.

FCN'in iki alıştırıcı modülü hem öğrenciler hem de öğretmenler arasında en çok benimsenen yönü oldu. Sistemi derslerinde kullanan öğretmenlerin bir kısmı alıştırıcıları kullandıktan sonra öğrencilerin büyük çoğunluğunun çevirim dışı program yazmaya hazır bir duruma geldiklerini öne sürdü.

BÖLÜM 6

FORTRAN ALIŞTIRICI SİSTEMİ

FCN-F programının yazılım nedeni FCN programının uygulandığı fakültede başarılı olmasıdır. Bu programlama sistemi, kuruluşdaki öğretim elemanlarıyla kullanıcı öğrenciler tarafından benimsenmiş ve dış kuruluşlardan transfer istemleri gelmiştir. Ancak, yazılım dili sadece PDP-11 serisinde kullanılan MACRO-11 dili olduğundan ve tüm kütük giriş/çıkış komutları yaygın olmayan RSX-11M işletim sistemine bağımlı olduğundan, FCN sisteminin transferi imkânsızdır.

FCN sisteminin yaygınlaşması ancak yaygın bir üst-düzey programlama diliyle ve donanımdan bağımsız olarak geliştirilmesiyle mümkün olabilir. Tüm olanaklarıyla (değiştirici, derleyici, kütük yönetim ve işlem zamanı denetim), bu sistemin yüksek düzeyli bir programlama diliyle etkin ve kullanışlı bir şekilde geliştirilmesi söz konusu değildir. Böyle bir yaklaşımın pratik olmayan yönleri şunlardır:

i) Çevirilen FORTRAN komutları er geç donanıma iletilip işleme geçirilmelidir. FCN sistemi bilgisayar donanımından bağımsız olacaksa özel bir kodlama yöntemiyle bu kodu donanıma iletecek üst-düzey komutlar (emülatör) kullanılmalıdır. Bunun kaçınılmaz bir sonucu FCN bellek gereksiniminin artması ve işlem zamanının uzamasıdır.

ii) İşlem sırasında kullanıcının verdiği programın "işlenebilir" kodunun tümü ile, bu kodu yorumlayacak emülatörün yanısıra üst-düzey dille yazılmış FCN sistemi bellekte tutulacaktır. Üst-düzey dille, yaygın dil kurallarına uyarak yazılan FCN sistemi, etkin bellek kullanmayacağından programı çok büyük boyutlara ulaşacaktır.

Bu sakıncaların yanı sıra FCN'deki en popüler olanakların atama ve desenli giriş/çıkış komut alıştırıcıları olması göz önünde tutularak, FCN-F yalnızca bu olanakların üst-düzey bir dil kullanarak sağlandığı bir sistem olarak kuruldu. Seçilen üst-düzeyli yazılım dili en yaygın olarak kullanılan FORTRAN IV oldu. Yazılım sırasında taşınabilirlik unsuru göz önünde tutularak, FCN-F'yi donanımdan bağımsızlaştırmak amacıyla, birkaç istisna dışında standard FORTRAN-IV komutları kullanıldı.İstisna komutlar şunlardır:

i) Programın geliştirildiği bilgisayarda giriş ve çıkış kanalları 100 ile 101 olduğundan zorunlu olarak bu kanallar kullanıldı

ii) Devam kartları zorunlu olarak birinci sütunda ve " & " işaretiyle belirtildi

iii) Hollerith dizgileri yalnızca tırnak işaretleriyle belirtilebildiğinden ve " H " kullanımı desteklenmediğinden zorunlu olarak " ' " işaretinin ASCII kodu kullanıldı.

iv) Uyarı düdüğünün ASCII kodu kullanıldı.

Kullanılan sistemdeki FORTRAN derleyicisinin bir takım kısıtlamaları zorunlu olarak programa yansımıştır. Bu kısıtlamalar, program yapısını etkilemelerine karşın, FORTRAN IV kurallarına uygun biçimde çözümlenmiştir:

i) IMPLICIT, INTEGER ve REAL tanımlama komutları olmadığından tüm tamsayı değişken tanıtıcıları I-N harfleriyle başlatılmıştır

ii) İsimlendirilmiş COMMON komutu olmadığından tüm ortak veri alanları isimsizdir. Bunun dezavantajı ortak alan kullanan altrutinlerde tüm COMMON komutlarının gerekmesidir. Ortak değişken sayısını azaltmak amacıyla kullanılmayan bölgeler dizilerle doldurulmuştur.

iii) Fonksiyon ve dizi erişim gerçek argümanları aritmetik deyim olamadığından bu komutlar sadeleştirilmiştir.

Kullanılan bilgisayarın hücre uzunluğu, doğal olarak, üst taşma denetimine yansımıştır. Ayrıca, uygulamada derleyici işlem ve bilgi alanları zorunlu olarak sınırlandırılmıştır (örneğin aktif değişken sayısı, tutulan desen sayısı). Fakat, yazılım ve donanımın tüm bağlayıcı özellikleri veriye bağımlı olarak uygulandığından (Bkz. Bölüm 3.4.1) gereğinde, FCN-F işlem felsefesini etkilemeden, değiştirilebilir. Bu bağlayıcı nitelikleri tanımlayan komutlar ve ne tür değişiklikler gerekebileceği Bölüm 6.6'da incelenmektedir.

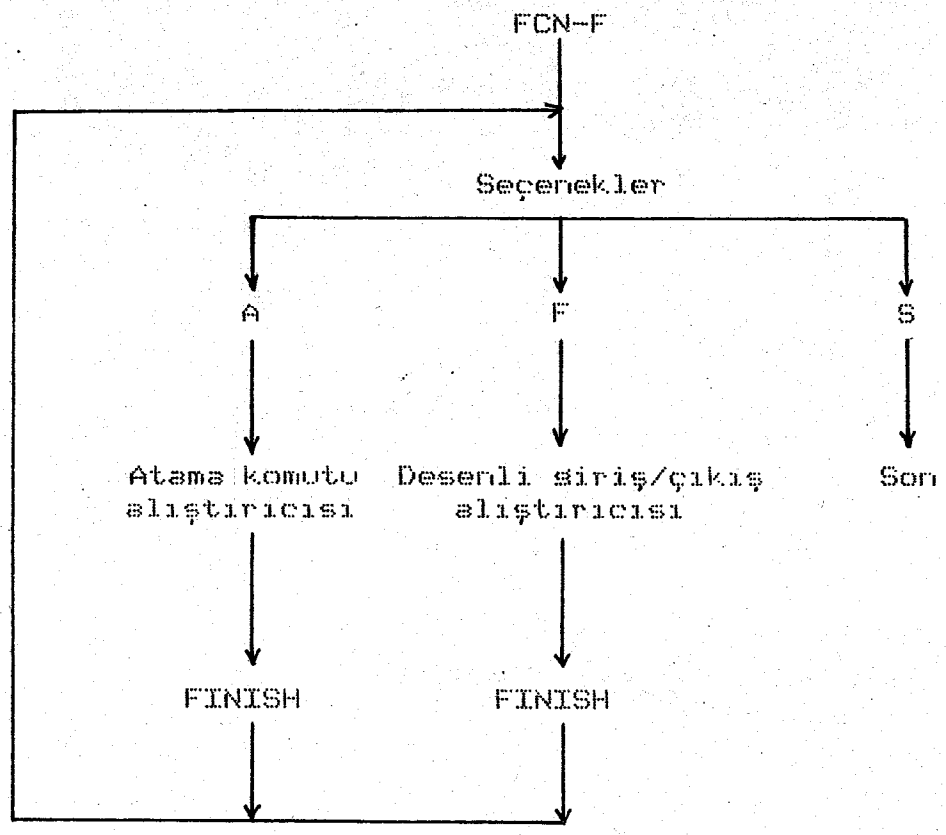
FORTRAN dilinin bir takım özellikleri etkin bellek kullanımını önlemiştir. Standard karakter kod işlem komutlarıyla fonksiyonları olmadığından tüm dizgi ve karakter kodları A1 deseniyle işlenmektedir. Bu yöntemle çalışmanın tek avantajı, FCN-F'nin bellek hücre uzunluğu ve karakter kodlama sisteminden tamamıyla bağımsızlaşmasıdır.

6.1 Kullanıcı açısından FCN-F

FCN-F, FCN sisteminin yalnızca atama ve desenli giriş/çıkış alıştırıcılarını içerir. Program başlatılınca kullanıcıya bu alıştırıcıların hangisini istediği sorulur; seçilen alıştırıcıdan çıkış FINISH komutuyla gerçekleştirilir. İki alıştırıcı sistem bir çok altrutini paylaştıkları halde kullanıcı bunları iki ayrı sistem olarak görür (Şekil 6.1). Bilgi alanlarında biriken veri RESET komutuyla temizlenir. Bölüm 5'de izlenen devam kartı kısıtlaması FCN-F için de geçerlidir. Alıştırıcılarda FORTRAN dizileri kullanılamaz.

6.1.1 Atama komutu alıştırıcısı

Bu alıştırıcıda uçtan alınan komutun leksik ve sözdizim çözümü yapıldıktan sonra işlem kodu üretilir. Herhangibir çözümleme hatası görülürse kullanıcıya bildirilir; hatalar hiç bir şekilde kullanıcıya ayırılan işlem alanını etkilemez. Eğer komutta hata yoksa ve işlem kodu başarılı olarak üretilmişse, FCN-F emülatörü kodu işleme alıp kullanıcıya ayırılan çalışma alanında gerekli değişikliği yapar ve sol taraftaki değişkenin aldığı yeni değeri ekranda gösterir. İşlem sırasında tüm fonksiyon ve üst taşma hataları bilgisayar donanım ve işletim sistemine yansıtılmadan yakalanır; hata sebebi kullanıcıya bildirilir. Ya-



Şekil 6.1 : FCN-F seçenek yapısı.

kalanan işlem hataları kullanıcıya ayırılan çalışma alanını etkilemez. Kullanıcının verdiği komut işlendikten sonra yeni bir komut uçtan okunur.

Atama komutlarında aritmetik devim hazırlarken, kullanıcı tüm FORTRAN aritmetik işlem işaretlerinin yanısıra SIN, COS, ALOG, EXP, SQRT, ALOG10, FLOAT, ABS, INT ve IABS fonksiyonlarını kullanabilir.

Şekil 6.2'de atama komutu alıştırıcısından birkaç örnek verilmiştir.

```
FCN-F ALISTIRICI SISTEMI

SECENEKLER:

1: ATAMA KOMUTU ALISTIRICISI
2: DESENLI READ/WRITE ALISTIRICISI
3: BITIS
1,2 VEYA 3? 1

FCN-F ATAMA KOMUTU ALISTIRICISI
```

Şekil 6.2 : FCN-F atama komutu alıştırıcısından örnekler.

```
?         X=5.0/2.0*6.0
X DEGERI: 15.00
?         Y=5.0*6.0/2.0
Y DEGERI: 15.00
?         K=5*6/2
K DEGERI: 15
?         L=5/2*6
L DEGERI: 12
?         RESET

KULLANICI ALANI TEMIZLENDI

?         A=-6.5
A DEGERI: -6.50
?         B=3.1
B DEGERI: 3.10
?         C=-6
C DEGERI: -6.00
?         KAREKOK=SQRT(B**2-4*A*C)
                     A
DEGISKEN TANITICI ALTI KARAKTERI ASAMAZ
?         KOK=SQRT(B**2-4*A*C)
ISLEM HATASI
EKSI SAYILARIN KARE KOKU ALINAMAZ
?         C=6
C DEGERI: 6.00
?         KOK=SQRT(B**2-4*A*C)
KOK DEGERI: 12
?         XKOK=SQRT(B*B-4A*C)
                           A
ISLENENDEN SONRA ISLEM ISARETI VERILMELI
?         XKOK=SQRT(B*B-4*A*C)
XKOK DEGERI: 12.87
?         X1=(-B-XKOK)/2*A
X1 DEGERI: 51.90
?         X2=(-B+XKOK)/2*A
X2 DEGERI: -31.75
?         Y1=A*X1*X1+B*X1+C
Y1 DEGERI: -0.17341E 05
?         Y2=X2*(A*X2+B)+C
Y2 DEGERI: -6644.46
?         X1=(-B-XKOK)/(2*A)
X1 DEGERI: 1.23
?         X2=(-B+XKOK)/(2*A)
X2 DEGERI: -0.75
?         Y1=A*X1*X1+B*X1+C
Y1 DEGERI: -0.19073E-05
?         Y2=A*X2*X2+X2*B+C
Y2 DEGERI: -0.95367E-06
?         FINISH

FCN-F ALISTIRICI SISTEMI

SECENEKLER:
```

Şekil 6.2 : (Devam)

6.1.2 Desenli giriş/çıkış alıştırıcısı

Atama komutu alıştırıcısında geçerli olan tüm komut biçimleri bu alıştırıcıda da geçerlidir. Önemli bir fark atama sonucunun kullanıcıya gösterilmeden çalışma alanına kaydedilmesidir; kullanıcı kendi giriş/çıkış komutlarını sağlamalıdır.

Desenli giriş/çıkış alıştırıcısı, önceden tanımlanmamış olan bir desen işareti kullanıldığında, bu desenin hemen tanımlanmasını ister. Desen tanımlanana dek başka komut kabul edilemez; desenin tanımlanması giriş/çıkış işlemini başlatır.

Kod üretimi sırasında görülen hatalar kullanıcıya ayırılan işlem alanını etkilemez. Ancak, READ komutu işlenirken hata olursa, hatalı veriden önce gelen değişkenlerin değerleri çalışma alanına işlenmiş olur. Örneğin,

READ (5,10) A,B,C,K

komutu işlenirken C değişkenine verilecek değerde bir hata görülmüşse A ve B değişkenlerinin değerleri sağlanan veriye göre değişmiş olur; C ve K değişkenleri yeni değer almaz.

WRITE komutu işlenirken bir işlem hatası varsa, çıkış tamponuna hata görülene dek yerleştirilmiş olan bilgi ekrana yazıldıktan sonra hata mesajı iletilir.

Şekil 6.3'de desenli giriş/çıkış alıştırıcısından bir kaç tanıtıcı örnek verilmiştir.

```
FCN-F GIRIS/CIKIS ALISTIRICISI

?        READ(5,10)IL1,ALX,IL2
TANIMLANACAK FORMAT:    10
? 10     FORMAT(I5,F5.1,I3)
VERI SATIRI  1
?   3 .5      6
      A
ISLEM HATASI
"IL1" ISLENIRKEN
I-DESENI ILE OKUNACAK ALANDA GECERSIZ KARAKTER BULUNDU
?        READ(5,10)IL1,ALX,IL2
VERI SATIRI  1
? 3        .5   6
```

Şekil 6.3 : FCN-F desenli giriş/çıkış alıştırıcısından örnekler.

```
? X=ALX**(IL1/IL2)
ISLEM HATASI
BOLERKEN UST TASMA
? 15 FORMAT(1X,2I4)
? WRITE(6,15)IL1,IL2
0000
  ^
ISLEM HATASI
"IL1" ISLENIRKEN
"WRITE" SIRASINDA YETERSIZ ALAN GENISLIGI
? 15 FORMAT(1X,2I5)
  ^
KOMUT ISARETI DAHA ONCE TANIMLANMISTI
? 20 FORMAT(1X,2I5)
? WRITE(6,20)IL$1,IL2
                  ^
GECERSIZ KARAKTER
? WRITE(6,20)IL1,IL2
30000 0
?
? RESET

KULLANICI ALANI TEMIZLENDI

? 10 FORMAT(F10.2)
? READ(5,10)A,B,C
VERI SATIRI 1
?-6.5
VERI SATIRI 2
? 3.1
VERI SATIRI 3
? 6.0
? X1=(-B-SQRT(B*B-4*A*C))/(2*A)
? X2=(-B+SQRT(B*B-4*A*C))/(2*A)
? 15 FORMAT(' KOKLER BULUNDU',F10.2)
? WRITE(6,15)X1,X2
KOKLER BULUNDU 1.23
KOKLER BULUNDU -0.75
? 20 FORMAT(' KOKLER BULUNDU',/,' KOK1:',F10.2,/,' KOK2:',F10.2)
? WRITE(6,20)X1,X2
KOKLER BULUNDU
KOK1: 1.23
KOK2: -0.75
? I=1
? J=2
? 25 FORMAT(' KOKLER:',/,2(1X,'KOK',I1,1H:,F5.2,2X))
? WRITE(6,25)I,X1,X2
KOKLER:
KOK1: 1.23  KOK
             ^
ISLEM HATASI
"X2" ISLENIRKEN
"READ/WRITE" ESNASINDA UYUSMAYAN DEGISKEN/OGE TIPLERI
? WRITE(6,25)I,X1,J,X2
KOKLER:
KOK1: 1.23  KOK2:-0.75
?
```

Şekil 6.3 : (Devam)

6.2. FCN-F'nin yapısı

FCN-F aynı sürücü program altında çalışan üç mantıksal parçadan oluşur. Parçaların görevleri şöyledir:

i) Komut çözümü ve kod üretimi

ii) Üretilen kodun yorumlanması

iii) Hata mesajlarının kullanıcıya iletilmesi

Parçalar arası iletişim ortak (COMMON) alanlar aracılığıyladır. Sürücü programın üslendiği görev kullanıcının hangi alıştırıcıyı istediğini öğrendikten sonra mantıksal parçaların koordinasyonunu sağlamaktır (Şekil 6.4).

Komut çözümüyle kod üretimi dört parçadan oluşur. Bu parçaların hiyerarşik şeması Şekil 6.5'de belirtilmiştir. Özetle parçaların görevleri

i) Komut tipinin saptanması ve komut tipine göre diğer üç parçadan birinin çalıştırılması

ii) Aritmetik deyimlerin çözümlenmesi ve atama komutları için kod üretimi

iii) READ/WRITE komutlarının çözümlenmesi ve kod üretimi

iv) FORMAT komutlarının çözümlenmesi

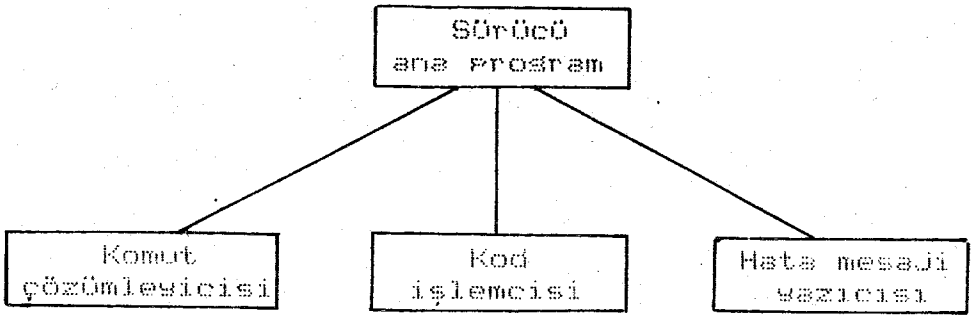
Üretilen kodları yorumalayan parçalar ise Şekil 6.6'da gösterilmiştir. Buradaki parçaların görevleri:

i) Kod tipinin saptanması ve diğer iki parçadan birinin çalıştırılması

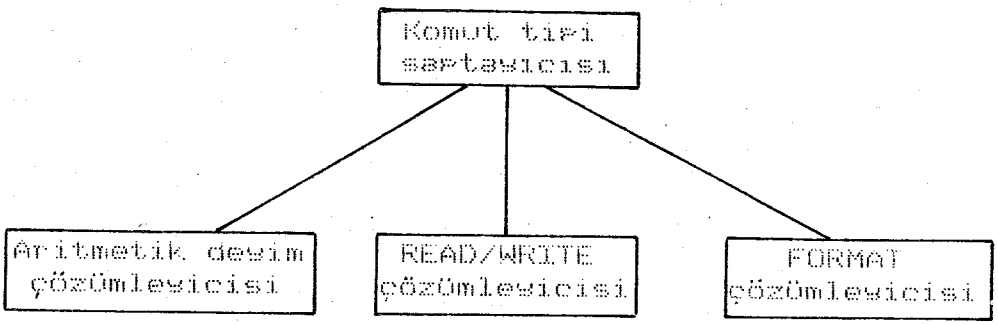
ii) Aritmetik işlemlerin yapılması

iii) Giriş/çıkış işlemlerinin verilen desene göre yapılması.

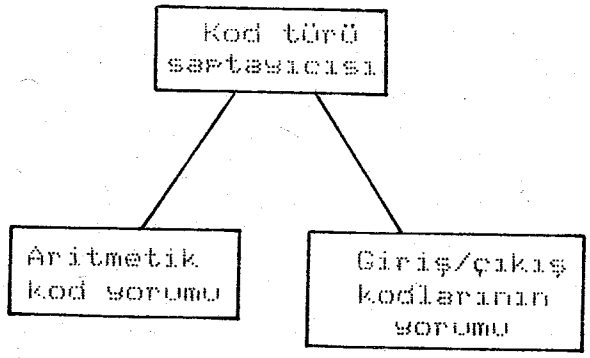
Komut çözümleyicisiyle kod işlemcisi çakışan hata mesajı kodlama sistemiyle çalışır. Hatanın hangi mantıksal parçadan geldiğini saptayabilen FCN-F sürücü programı, hata numarasını ya çözümleme hataları yazıcısına, ya da işlem hataları yazıcısına gönderir (Şekil 6.7).



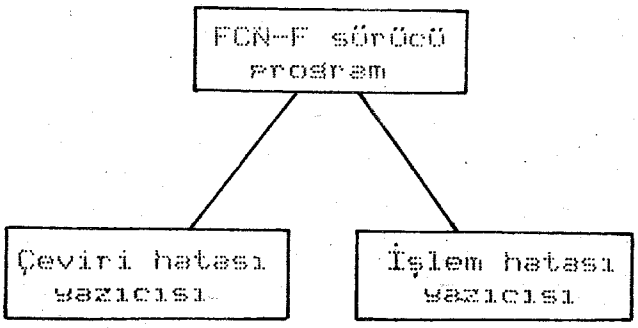
Şekil 6.4 : FCN-F'nin mantıksal modül yapısı.



Şekil 6.5 : Komut çevirisinin mantıksal modül yapısı.



Şekil 6.6 : İşlem kodu yorumlayıcısının mantıksal modül yapısı.



Şekil 6.7 : Mesaj yazıcısının mantıksal modül yapısı.

6.2.1 Ortak alanlar ve kullanıcı alanları

Kullanıcı çalışma alanlarıyla FCN-F bilgi ve işlem alanlarını yedi gruba ayırabiliriz.

i) Kullanıcı değişkenleri ve bunların değerleri

ii) Komut işaretleri ve desen kodları

iii) Üretilen işlem kodları

iv) Giriş/çıkış tampon alanları

v) Aritmetik deyim çeviri torbalama alanı

vi) İşlem zamanı giriş/çıkış çalışma alanı

vii) Diğer ortak bilgi alanları

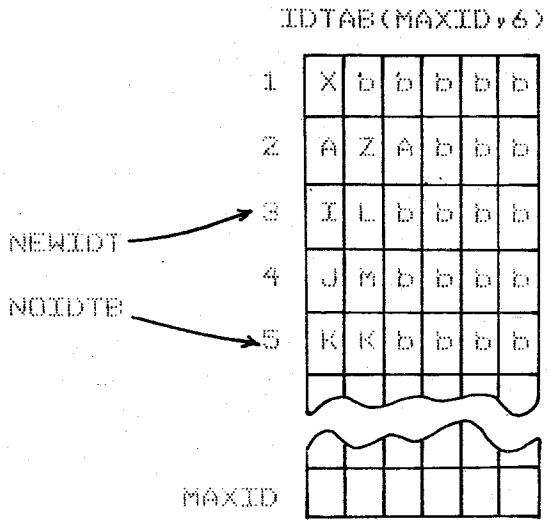
Bellekte aynı yeri paylaşan v. ve vi. alanların nasıl kullanıldığını 6.3.2 ve 6.4.3'de tanıtılacaktır.

Kullanıcı değişkenleriyle değerleri

Kullanıcı değişkenleri 6A1 deseniyle IDTAB matrisinde tutulur. Genişliği 6 sütun olan bu matrisin her sırasına bir değişken tanıtıcısı yerleştirilebilir. Azami değişken sayısı, bu matrisin sıra sayısını tanımlayan, MAXID değeriyle sınırlandırılır (Şekil 6.8). MAXID'nin değeri FCN-F yaratılırken saptanmalıdır.

İki değişkenle (NOIDTB ve NEWIDT) değişken tablosunda kullanılan en son sıra tanımlanır. NEWIDT kesin pozisyonu gösterirken NOIDTB işlenmekte olan komutun değişken tablosuna yapmış olduğu katkıyı gösterir. Böylece hatalı komutların değişken tablosuna yaptığı değişiklikler önlenir. Kullanıcının verdiği "RESET" komutu NEWIDT değerini 1 yapar.

Şekil 6.8'de bir READ komutu işlenirken elde edilen NOIDTB değeri yansıtılmaktadır. Eğer komut hata ile sonuçlanmazsa NEWIDT'nin değeri değişecektir. Bu örnekte görüldüğü gibi değişken tablosu sıralı değildir ve dolayısıyla FCN-F sırasal erişim kullanır. FCN-F'nin yazılım diliyle kullanım ortamı göz önünde tutulduğunda sırasal tarama en etkin yöntemdir. Alternatif



Şekil 6.8 : Değişken tablosu (IDTAB) kullanımı.

olarak alabileceğimiz ikili arama üç sebepten ötürü FCN-F ortamında sakıncalıdır

i) FORTRAN'da kaydırma operasyonları olmadığından pahalı bölme operasyonları kullanılacaktır

ii) Tipik tablo uzunluğu 20 civarındadır ve bu uzunluk kullanıcıya fazlasıyla yetmektedir.

iii) Değişken tanımlamanın değişken kullanımına oranı muhtemelen yüksek olacağından tabloya girilen ve az kullanılan değişken sayısı da yüksek olacaktır. Her tablo girişi sıralama gerektirebildiğinden zaman kaybı olacaktır.

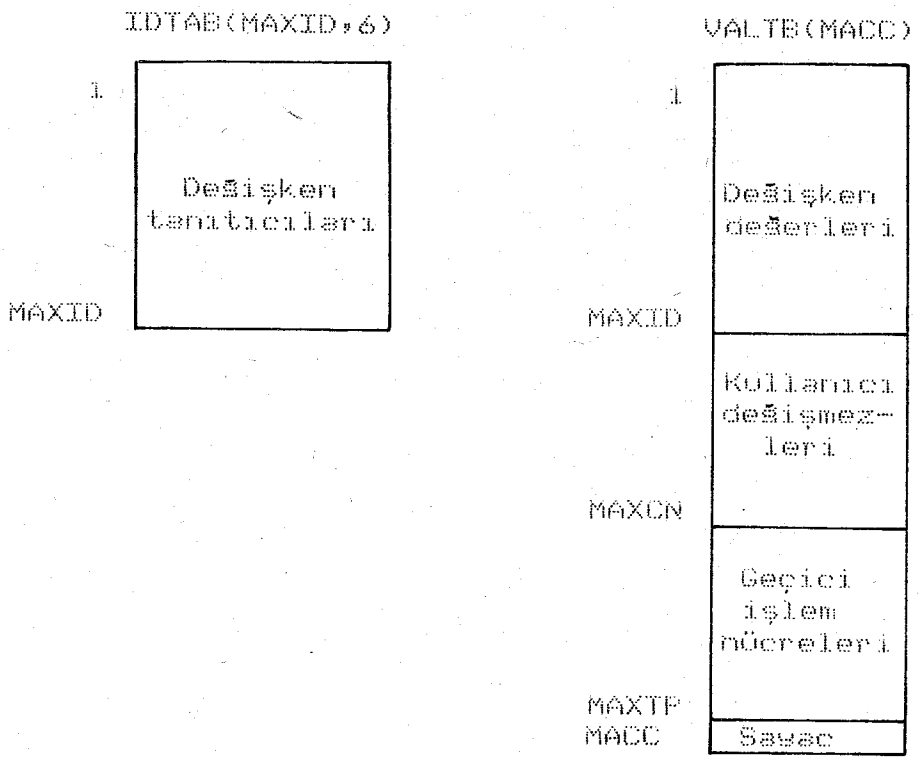
Değişkenlerin aldığı değerler kullanıcının bellek veri alanını simüle eden VALTB dizisinde tutulur. VALTB içinde tutulan tüm değerler gerçek sayıdır; üretilen işlem koduna göre buradaki bilgi gereğinde tam sayıya dönüştürülür. Bu dizide dört değişik alan vardır (Şekil 6.9).

VALTB (1-MAXID): Kullanıcı değişkenlerinin değerleri

VALTB (MAXID+1-MAXCN): Kullanıcı değişmezlerinin değerleri.

VALTB (MAXCN+1-MAXTP): Aritmetik deyimler işlenirken kullanılacak geçici işlem alanı

VALTB (MACC): Aritmetik işlemlerde kullanılan sayaç hücresi.



Şekil 6.9 : Değer tablosu (VALTB) kullanımı.

Değişmez değer alanı dışında tüm alanlardaki bilgi yalnız üretilen kod yorumlanırken kullanılır. Üretilen kod VALTB dizisindeki endeksleri "işlenen adres boşluğu" olarak kullanır.

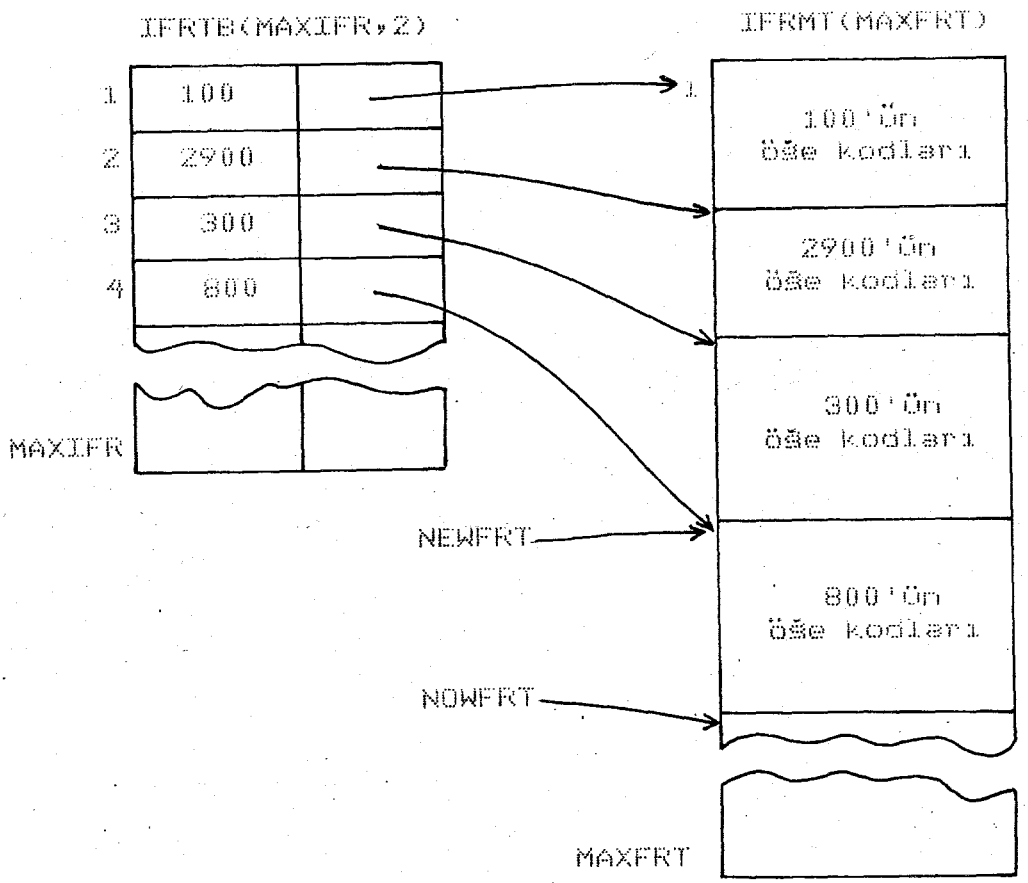
Kullanılan en son değişmez pozisyonuyla geçici hücre pozisyonu NEWCN ve NEWTP değişkenlerinde tutulur. Her iki değişken, yeni kullanıcı komutu okunurken, geçerli oldukları alanın başına alınır.

VALTB dizisinin bu şekilde düzenlenmesinin başlıca nedeni kod üretimini kolaylaştırıp gerçekçi bellek kullanımına benzer bir adresleme yöntemi kurmaktır. Böylece işlem kodları yorumlanırken tüm değişken, değişmez ve geçici hücreler aynı adresleme yöntemiyle erişilebilmektedir.

Komut işaretleri ve desen kodları

Kullanıcının tanımladığı komut işaretleri IFRTB matrisinde tutulur. İki sütunu olan bu matrise yerleştirilebilecek işaret sayısını MAXIFR değişkeni sınırlar; NEWIFR'da doğruluğu kesinleşmiş en son işaretin endeksi,NOWIFR'da ise çözümlenmekte olan komutun işaret endeksi tutulur. Geçici endeks NOWIFR, hatalı komutların kullanıcı işaret alanındaki etkisini yok etmek amacıyla kullanılır.

IFRTB matrisinin ilk sütununda, tanımlanan işaretlerin savısal değerleri tutulur. Eğer işaret bir desen tanımlıyorsa, desen öğelerinin muhafaza edildiği IFRMT dizisindeki başlangıç pozisyonu ikinci sütunda tutulur (Şekil 6.10).



Şekil 6.10: İşaret ve desen öğe tablolarının kullanımı.

Tanımlanan giriş/çıkış desenleri IFRMT dizisinde (uzunluk MAXFRT) tutulur. Giriş/çıkış komutlarının kod işlemlerini kolaylaştırmak amacıyla desen öğeleri FCN-F öğe kodlarına dönüştürülerek muhafaza edilir. NEWFRT ve NOWFRT değişkenleri IFRMT dizisinde kullanılan en son kesin pozisyonla geçici pozisyonu gösterir.

Şekil 6.10'da 100, 2900 ve 300 sayılı FORMAT'lar tanımlandıktan sonra, 800 sayılı FORMAT komutu çözümlenirken, IFRTB ve IFRMT dizilerinin kullanım şekli gösteriliyor.

" RESET " komutu NEWIFR ve NEWFRT değişkenlerini IFRTB ve IFRMT dizilerinin başına alarak kullanıcı desen alanını temizler.

FORMAT komutu çözümlenirken, işaret alanında görülen sayı NGIVEN değişkeninde tutulur; eğer tanımlanmayan bir desen giriş/çıkış komutunda kullanılmışsa, tanımlanması gereken işaret NEXPCT değişkeninde tutulur. NGIVEN ve NEXPCT değişkenleri giriş/çıkış komutundan sonra tanımlanacak olan desenin koordinasyonunu sağlar.

Üretilen kod alanı

VALTB ve IFRMT dizilerine kullanıcıya ayırılan bellek veri alanı olarak bakarsak, ICODEX dizisi de kullanıcının işlenebilir kod alanıdır. Uzunluğu MAXCD ile tanımlanan bu alanda kullanıcının yazmış olduğu son işlenebilir komutun işlem kodu tutulur. Alana yerleştirilen en yeni kodun endeksini gösteren NEWCD değişkeni her komuttan önce alan başına alınır.

Aynı ortak veri alanındaki iki değişkenle donanımın özellikleri tanımlanır. XMXINT, bilgisayarın kullanabileceği en büyük tam sayıyı gerçek sayı olarak gösterir. MXPOW, en büyük gerçek sayının on tabanına göre tamsayı logaritmasıdır.

Giriş/çıkış tampon alanı

Kullanıcının yazdığı komutlar çözümlenene kadar 80 karakterlik ISTRNG dizisinde tutulur. ICH, komut çözümlenirken işlenen en son karakteri, ICHCNT ise bu karakterin ISTRNG içerisindeki endeksini gösterir.

Kullanıcı değişkenleri tampondan okunurken geçici olarak 6 karakterlik IDTEMP dizisinde tutulur. ISSIX, bu geçici alandaki karakter sayısını gösterir.

ISTRNG dizisiyle ICHCNT, aynı zamanda READ/WRITE komutları işlemdeyken, giriş/çıkış tamponu ve tampon pozisyon göstergesi olarak kullanılır.

Diğer ortak değişkenler

FCN-F yaratılırken tanımlanması gereken MDCHK, karma deyimlerin (tam sayı gerçek sayı karışık) hata sayılıp sayılmayacağını saptar. MDCHK sıfırsa karma ifadeler kullanılabilir, eğer sıfır değilse karma değişken ve değişmezler aynı deyimde kullanıldığında hata mesajı verilir.

ICTYP kullanıcının seçtiği alıştırıcı altsistemi gösterir. Kullanıcı atama komutu alıştırıcısını seçmişse ICTYP değeri bir olur; desenli giriş/çıkış alıştırıcısında ICTYP değeri sıfırdır.

NCLASS ve NVALUE, kullanıcı komutundan alınan en yeni komut öğesinin özet kodunu gösterir.

IEROR sıfır dışında bir değer taşıyorsa, komut çözümlenirken veya üretilen kod işlenirken hatalı bir durum farkedilmiştir. Bu değişkenin değeri hata numarasını gösterir.

6.2.2 FCN-F kodlama sistemi

Modüller arası bilgi alış verişinde üretilen ve iletilen bilgi kodlarını beş sınıfa ayırabiliriz.

i) Üretilen işlenebilir komut kodları

ii) Giriş/çıkış desen kodları

iii) FORTRAN komut öğelerini tanımlayan kodlar

iv) Derleyici hata kodları

v) İşlemci hata kodları

Hata kodlarının IEROR ortak değişkenine yerleştirildiğini daha önce izlemiştik. Derleyicinin ürettiği hata kodları 1 ile 80 arasında değişir; işlemci ise 1 ile 25 arası hata kodları üretebilir. Hata numaralarının karşılığı olan mesajlar Ek.B'de verilmiştir.

6.2.2.1 İşlem kodları

Tek sayaçlı ve tek adres komutlu soyut bir bilgisayara göre düzenlenmiş olan işlenebilir komut kodları 0 ile 99 arasında değişen 36 tamsayıdan oluşur. ICODEX dizisine yerleştirilen her FCN-F işlem kodu bir veya iki hücrelik yer tutar. İki hücrelik komutlarda birinci hücreye işlem kodu ikinci hücreye ise işlenenin bellek adresi (yani VALTB veya IFRTB endeksi) yerleştirilir. Komutta tek işlenen sayaç ise komut tek hücrelik yer tutar.

İşlem kodlarını üç sınıfa ayırabiliriz:

i) Temel işlem kodları

ii) Fonksiyon kodları

iii) Giriş/çıkış kodları

Kod üretimiyle işlemini kolaylaştırmak amacıyla FORTRAN fonksiyon ve giriş/çıkış komutları yaratılan yapay bilgisayarda tek kodla belirtilir. FCN-F'nin kullandığı bu yüksek düzeyli kodlarla gerçekçi ortamda kullanılabilen "sözde kodlar" (pseudo code) arasında bir paralel kurabiliriz.

Temel işlem kodları

Bu sınıftaki 17 kod, işlemi durdurmak (kod 0) ve sayaç manipülasyonu (kod 1-kod 8, kod 11-kod 18) için kullanılır. 1 ile 8 arasındaki kodlar gerçek sayı cevaplı işlemler için, 11 ile 18 arasındaki kodlar ise tam sayı işlemler için geçerlidir. Tam sayı işlem kodları gerçek sayı işlem kodlarına 10 eklenerek elde edilir.

Kullanılan işlem kodları, isimleri ve anlamları Şekil 6.11'deki tebloda verilmektedir. Bu tabloda aşağıdaki kısıtlamalar kullanılmıştır

S: Sayaç hücresinin endeksi

İ: İşlenen hücrenin endeksi

//: Tam sayı sonuçlu bölme işlemi

Tüm tamsayılı işlemlerde sonuç gerçek sayı olarak FCN-F "belleğinde" tutulur; örneğin, bölme işleminin sonucu 8.68 ise bu sonuç 8.0 olarak muhafaza edilir. Tam sayı işlem yapılmadan ve yapıldıktan sonra işlenecek olan sayılar tamsayıya dönüştürülür.

Örnek: Kullanıcının sağladığı atama komutu

```
A=B*(-(D+E)+I/J)-5.05
```

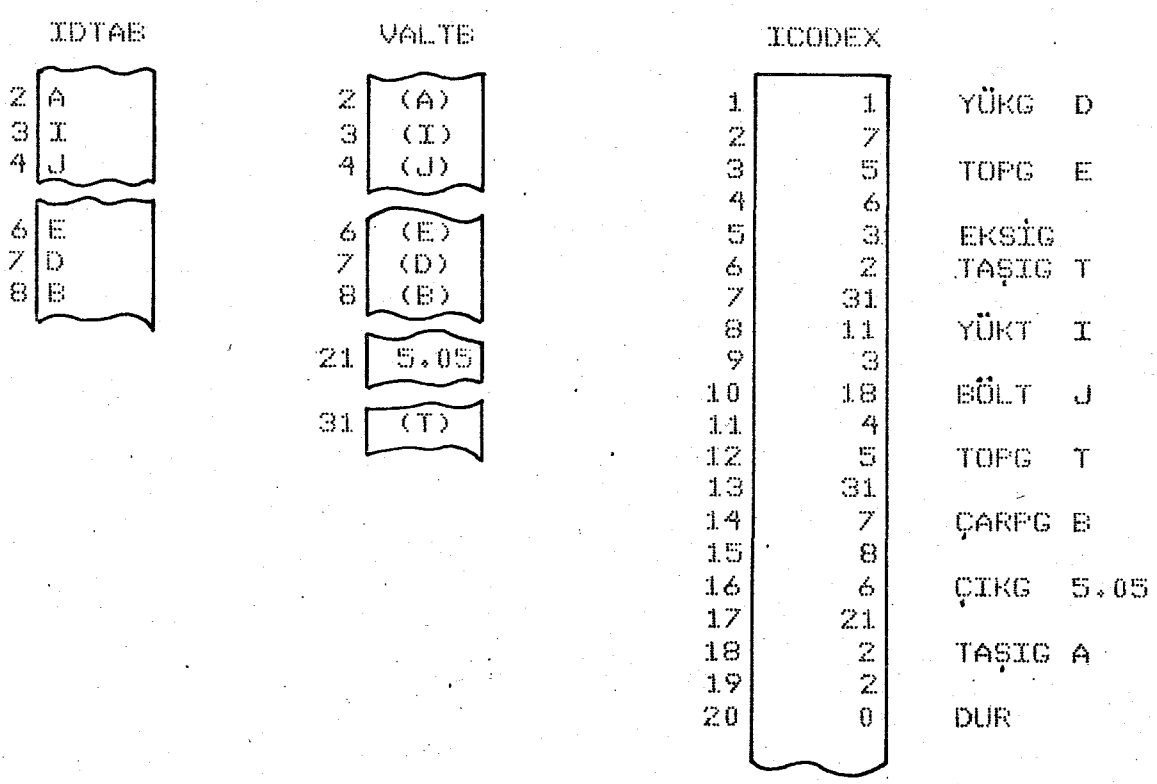
olduğunda üretilen işlem komutları şöyle olur

```
YÜKG     D      Sayaç:=D
TOPG     E      Sayaç:=D+E
EKSİG           Sayaç:=-(D+E)
TAŞIG    T      T:=-(D+E)
YÜKT     I      Sayaç:=I
BÖLT     J      Sayaç:=I//J
TOPG     T      Sayaç:=-(D+E)+I//J
ÇARPG    B      Sayaç:=B*(-(D+E)+I//J)
ÇIKG     5.05   Sayaç:=B*(-(D+E)+I//J)-5.05
TAŞIG    A      A:=B*(-(D+E)+I//J)-5.05
DUR             İşlem sonu
```

A, B, D, E, I, J değişkenlerine sırasıyla VALTB'deki 2, 8, 7, 3 ve 4'üncü hücrelerin atandığını, geçici hücre (T) olarak 31'in kullanıldığını, ve 5.05 değişmezinin VALTB'da 21'inci hücrede tutulduğunu varsayarsak, bu örnekte üretilip ICODEX dizisine yerleştirilen işlem kodu Şekil 6.12'de gösterildiği gibi olur.

Kod	Komut	İşlenen	Sayaçtaki sonuç tipi	İşlem
0	DUR	Yok	Yok	İşlem sonu
1	YÜKG	Var	Gerçek	VALTB(S) <- VALTB(I)
2	TAŞIG	Var	Gerçek	VALTB(I) <- VALTB(S)
3	EKSİG	Yok	Gerçek	VALTB(S) <- -VALTB(S)
4	ÜSG	Var	Gerçek	VALTB(S) <- VALTB(S)**VALTB(I)
5	TOPG	Var	Gerçek	VALTB(S) <- VALTB(S)+VALTB(I)
6	ÇIKG	Var	Gerçek	VALTB(S) <- VALTB(S)-VALTB(I)
7	ÇARPG	Var	Gerçek	VALTB(S) <- VALTB(S)*VALTB(I)
8	BÖLG	Var	Gerçek	VALTB(S) <- VALTB(S)/VALTB(I)
11	YÜKT	Var	Tam	VALTB(S) <- VALTB(I)
12	TAŞIT	Var	Tam	VALTB(I) <- VALTB(S)
13	EKİST	Yok	Tam	VALTB(S) <- -VALTB(S)
14	ÜST	Var	Tam	VALTB(S) <- VALTB(S)**VALTB(I)
15	TOPT	Var	Tam	VALTB(S) <- VALTB(S)+VALTB(I)
16	ÇIKT	Var	Tam	VALTB(S) <- VALTB(S)-VALTB(I)
17	ÇARPT	Var	Tam	VALTB(S) <- VALTB(S)*VALTB(I)
18	BÖLT	Var	Tam	VALTB(S) <- VALTB(S)/VALTB(I)

Şekil 6.11: FCN-F Temel İşlem kodları



Şekil 6.12: Temel İşlem kodlarının kullanımı

Fonksiyon kodları

41 ile 60 arasındaki işlem kodları standart FORTRAN fonksiyonlarına ayırılmıştır; bu alandaki kodların yalnız on tanesi kullanılmaktadır. Bu tür kodlar ICODEX dizisinde tek hücre tutar ve fonksiyon argümanının değeri her zaman sayaçtan alınıp sonucu yine sayaca bırakılır. FCN-F'nin kullandığı fonksiyonlar ve üretilen fonksiyon kodları Şekil 6.13'de gösterilmektedir.

Örnek: Kullanıcının yazdığı atama komutu

CDX=SIN(-ALOG(A))/COS(ALOG(SQRT(X+Y)))

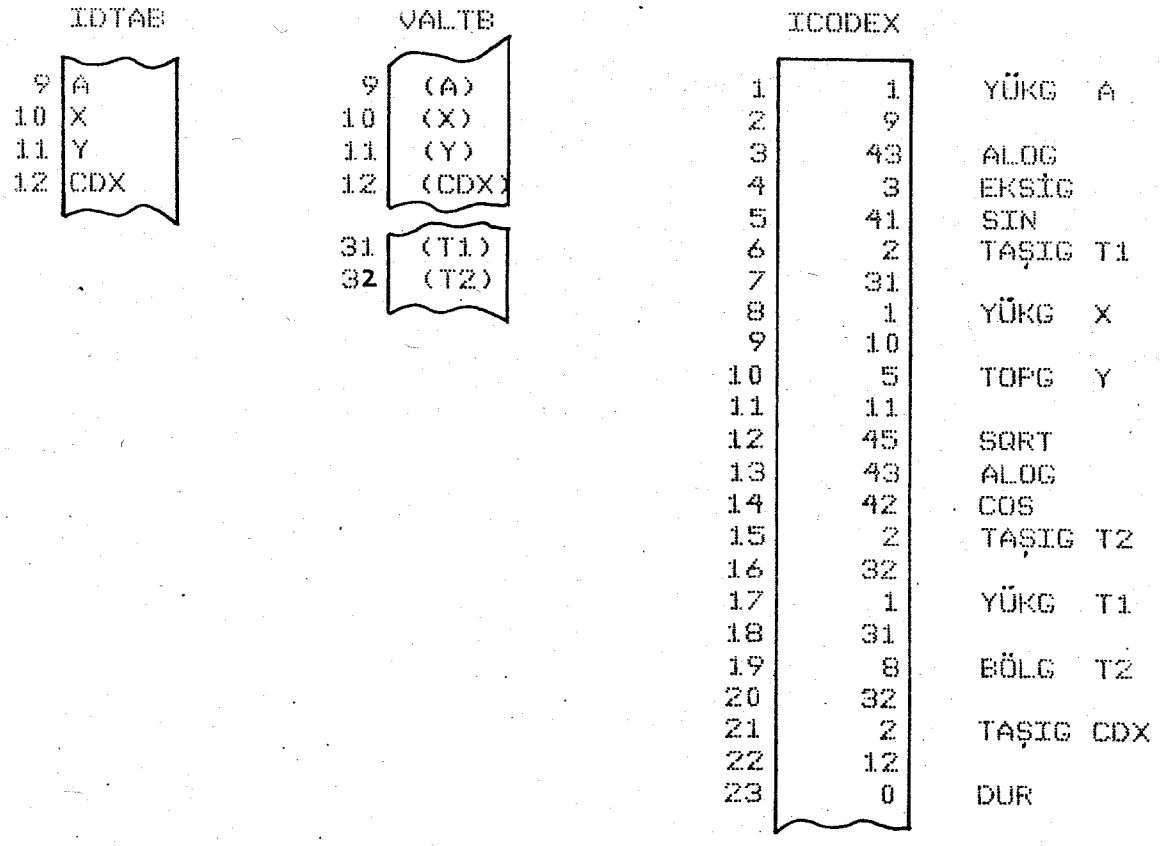
olduğunda FCN-F'nin ürettiği işlem komutları şöyle olur

YÜKG	A	Sayaç:=A
ALOG		Sayaç:=ALOG(A)
EKSIG		Sayaç:=-ALOG(A)
SIN		Sayaç:=SIN(-ALOG(A))
TAŞIG	T1	T1:=SIN(-ALOG(A))
YÜKG	X	Sayaç:=X
TOPG	Y	Sayaç:=X+Y
SQRT		Sayaç:=SQRT(X+Y)
ALOG		Sayaç:=ALOG(SQRT(X+Y))
COS		Sayaç:=COS(ALOG(SQRT(X+Y)))
TAŞIG	T2	T2:=COS(ALOG(SQRT(X+Y)))
YÜKG	T1	Sayaç:=SIN(-ALOG(A))
BÖLG	T2	Sayaç:=SIN(-ALOG(A))/COS(ALOG(SQRT(X+Y)))
TAŞIG	CDX	CDX:=SIN(-ALOG(A))/COS(ALOG(SQRT(X+Y)))
DUR		İşlem sonu

CDX, A, X ve Y değişkenlerine VALTB'deki 12,9,10 ve 11'inci hücrelerin atandığını ve T1, T2 geçici işlem hücrelerine 31 ile 32'inci pozisyonların verildiğini varsayarsak üretilen işlem kodu Şekil 6.14'de gösterildiği gibi olur.

Kod	Komut	Islem
41	SIN	VALTB(S) <- SIN(VALTB(S))
42	COS	VALTB(S) <- COS(VALTB(S))
43	ALOG	VALTB(S) <- ALOG(VALTB(S))
44	ALOG10	VALTB(S) <- ALOG10(VALTB(S))
45	SQRT	VALTB(S) <- SQRT(VALTB(S))
46	EXP	VALTB(S) <- EXP(VALTB(S))
47	ABS	VALTB(S) <- ABS(VALTB(S))
48	FLOAT	VALTB(S) <- FLOAT(VALTB(S))
57	IABS	VALTB(S) <- IABS(VALTB(S))
58	INT	VALTB(I) <- INT(VALTB(S))

Şekil 6.13: FCN-F Fonksiyon kodları



Şekil 6.14: Fonksiyon kodlarının kullanımı

Giriş/çıkış kodları

FCN-F 71 ile 99 arasındaki sayıları giriş/çıkış işlem kodları için kullanır. Bu kodlar Şekil 6.15'de özetlenmektedir.

SERG ve SERT komutlarını sadece adama komutu alıştırıcısı üretir. Bu kodlar işlenirken yazılacak değere uygun olan deseni kod yorumlayıcısı seçer. Desenli giriş/çıkış komutlarında üretilen ilk kod giriş/çıkış işleminde kullanılacak olan desenin işaretini tanımlar.

YAZG ve YAZT komutları çıkış değişken listesinden alınan tek bir değişkenin kullanıcın tanımlamış olduğu ve YDESEN komutuyla yorumlayıcıya tanıtılan desene göre yazılmasını sağlar. OKUG ve OKUT komutlarının kullanılış şekli YAZG ve YAZT komutlarına benzer. Tek fark kullanılacak olan desenin ODESEN komutuyla tanımlanması ve işlemin desene göre okuma oluşudur.

DURY komutu desenli çıkış işleminin sonunu belirtir; bu kod işlendiğinde çıkış tamponu boşaltıldıktan sonra işlemler durdurulur. Giriş komutlarının sonu normal DUR komutuyla (kod 0) belirtilir.

Kod	Komut	İşlenen	İşlem
71	SERG	Değişken	Gerçek sayının uygun desenle yazılması
72	YAZG	Değişken	Gerçek sayının yazılması
73	OKUG	Değişken	Gerçek sayının okunması
81	SERT	Değişken	Tam sayının uygun desenle yazılması
82	YAZT	Değişken	Tam sayının yazılması
83	OKUT	Değişken	Tam sayının okunması
91	YDESEN	Desen	Yazma deseninin tanımlanması
92	ODESEN	Desen	Okuma deseninin tanımlanması
93	DURY	Yok	Yazma işleminin sonu

Şekil 6.15 : FCN-F giriş/çıkış kodları.

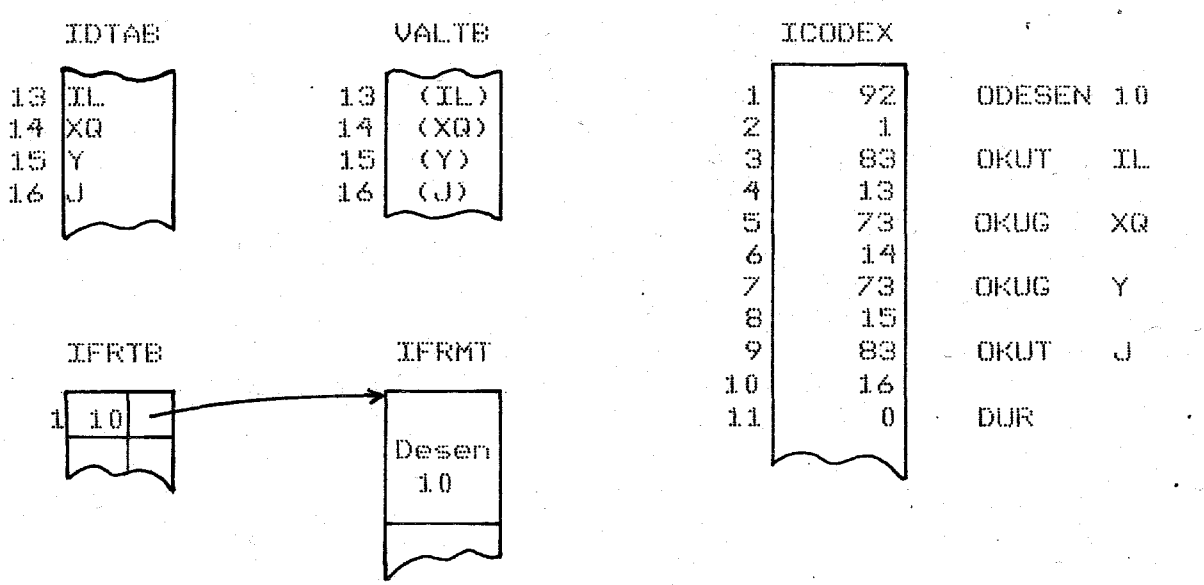
Örnek: Verilen FORTRAN okuma komutu

```
READ(5,10)IL,XQ,Y,J
```

olursa üretilen komutlar şunlardır

```
ODESEN   10      Desen 10 ile okunacak
OKUT     IL      IL:=Okunan tam sayı
OKUG     XQ      XQ:=Okunan gerçek sayı
OKUG     Y       Y:=Okunan gerçek sayı
OKUT     J       J:=Okunan tam sayı
DUR              İşlem sonu
```

10 numaralı desen IFRTB matrisinin ilk hücresindeyse ve IL,XQ,Y,J değiskenleri VALTB içerisindeki 13,14,15 ve 16 savılı hücreleri kullanıvorsa üretilecek olan işlenebilir kod Şekil 6.16'da gösterilmektedir.



Şekil 6.16 : Giriş kodlarının kullanımı.

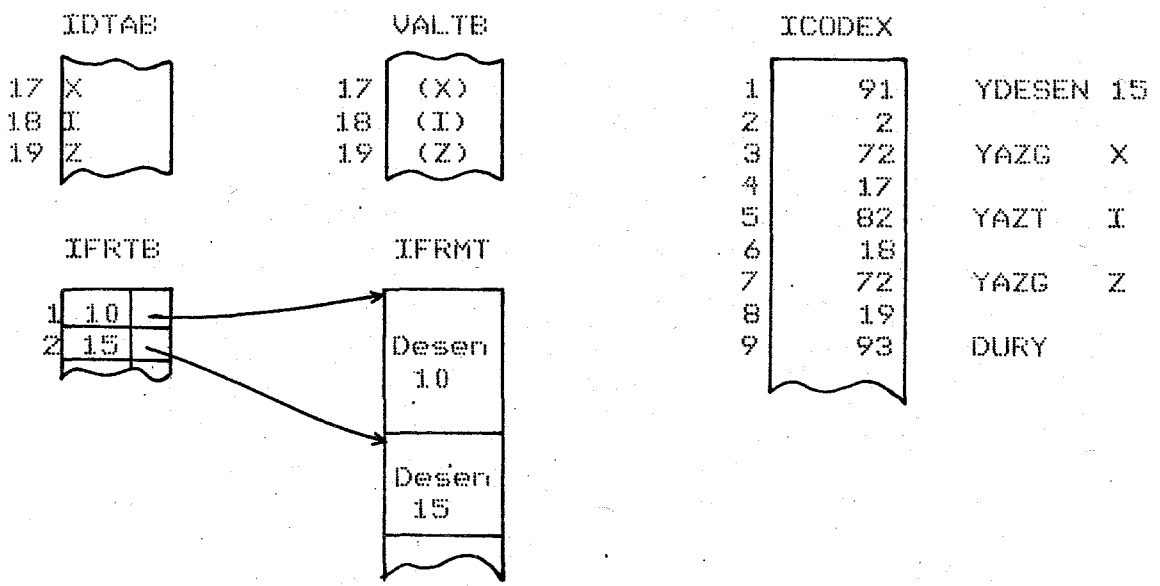
Örnek: Verilen FORTRAN yazma komutu

```
WRITE(6,15)X,I,Z
```

olduğunda üretilen komutlar şunlardır

YDESEN	15	Desen 15 ile yazılacak
YAZG	X	Gerçek sayı (X) yazılacak
YAZT	I	Tam sayı (I) yazılacak
YAZG	Z	Gerçek sayı (Z) yazılacak
DURY		İşlem sonu

15 işaretli desen IFRTB tablosundaki ikinci hücredeyse ve X,I,Z değişkenlerinin değerleri 17, 18 ve 19 'uncu VALTB hücrelerindeyse, FCN-F'nin üreteceği işlenebilir kod Şekil 6.17'de gösterilmektedir.



Şekil 6.17 : Çıkış kodlarının kullanımı.

6.2.2.2 Giriş/çıkış desen kodları

Önceden de belirtildiği gibi, kullanıcının FORMAT komutuyla tanımladığı desenler çözümlenip FCN-F desen kodlarına dönüştürüldükten sonra, üretilen desen kodları, IFRMT dizisinde tutulur. FCN-F, bu kodlara , IFRTB tablosu aracılığıyla erişir (bkz. Şekil 6.10, 6.16, 6.17). Kullanılan desen kodları Şekil 6.18'deki tabloda verilmiştir. Bu tabloda şu kısaltmalar kullanılmıştır

n: Desen parçasının kaş kez tekrarlanacağı

+1: Desen öğesini tanımlayan koddan sonra gelen hücre

+2: Desen kodundan sonraki ikinci hücre

Örnek: Verilen

```
100    FORMAT(5X,3F10.2,E9.1,3(I5,/))
```

ve

```
200    FORMAT(' CEVAP:',3I5,/,1X,7(3(1H-)))
```

FORTRAN desen tanımlayıcı komutlarının, IFRTB ve IFRMT dizilerindeki etkisi Şekil 6,19'da gösterilmektedir.

Öğe	Öğe Kodu	+1	+2
I	10*n+1	Alan genişliği	-
F	10*n+2	Alan genişliği	Kesir genişliği
E	10*n+3	Alan genişliği	Mantis genişliği
X	10*n+4	-	-
H	10*n+5	-	-
','	10*n+5	-	-
/	16	-	-
(	-(10*n+7)	-	-
)	-8	-	-
İlk (	-5	-	-
Son)	-9	-	-

Şekil 6.18 : FCN-F Desen Öğe kodları.

IFRTB

1	100	1
2	200	15

IFRMT

1	-5	İlk (
2	54	5X
3	32	3F10.2
4	10	
5	2	
6	13	E9.1
7	9	
8	1	
9	-37	3(
10	11	I5
11	5	
12	16	/
13	-8	)
14	-9	Son)
15	-5	İlk (
16	75	7H CEVAP:
17	" "	
18	"C"	
19	"E"	
20	"V"	
21	"A"	
22	"P"	
23	":"	
24	31	3I5
25	5	
26	16	/
27	14	1X
28	-77	7(
29	-37	3(
30	15	1H-
31	"_"	
32	-8	)
33	-8	)
34	-9	Son)

Şekil 6.19 : Desen Öğe kodlarının kullanımı.

6.2.2.3 Komut öğe kodları

Aritmetik deyim çözümleyicisiyle leksik çözüleyici arasındaki iletişim kod çiftleriyle (tokens, doublets) sağlanmıştır. Her komut öğe türü için ayrı bir kod çifti üreten leksik çözümleyici, ilettiği bilgiyle deyim çözümleyicisinin görevlerini azaltıp yaptığı işlemleri kolaylaştırır. Komut öğe kodları iki parçadan oluşur

i) Öğe türü - NCLASS

ii) Öğe bilgisi - NVALUE

Aritmetik deyim çözümleyicisi NCLASS'ın taşıdığı bilgiyi deyimi postfix notasyonuna dönüştürürken, NVALUE'nun taşıdığı bilgiyi ise işlem kodu üretirken kullanır.

FORTRAN aritmetik deyimlerinde kullanılabilen öğeleri üç genel sınıfa ayırmak mümkündür

i) İşlenecek değişken veya değişmezler

ii) Fonksiyonlar

iii) Aritmetik işlem veya ayırıcı işaretler

Bu üç sınıf içerisinde kullanılan öğeler ve bunlara verilen NCLASS, NVALUE çiftleri Şekil 6.20'de belirtilmiştir.

Öğe bir değişken veya değişmez olduğunda NVALUE'nun aldığı değer VALTB içerisinde işlenene ayırılan hücrenin endeksidir. Fonksiyon, işlem ve ayırıcı işaretlere verilen NCLASS değerleri hem işlem önceliğini hem de işlem tipini tanımlar "+,-" ve "*,/" işlemlerine verilen çakışan NCLASS değerleri deyiminin sözdizim çözümü sırasında zararsızdır. Fonksiyonlara verilen NVALUE değerleri, işlem kodu üretimi sırasında kullanılacak komut kodlarıdır (bkz. Bölüm 6.2.2.1). "(" , ")" ve "deyim sonu" işaretleri dışındaki tüm işlem işaretlerinin NVALUE değerleri, o işaretin temel işlem kodunu yansıtır. Buradaki işlem kodları sadece gerçek sayılı işlemler için geçerlidir; işlem kodları üretilirken işlem sonucu tam sayı olacaksa NVALUE'daki koda 10 eklenir (bkz. Bölüm 6.2.2.1).

Örnek: Çözümlenecek olan atama komutu

```
KL=EXP(-K)+B*C/5.99
```

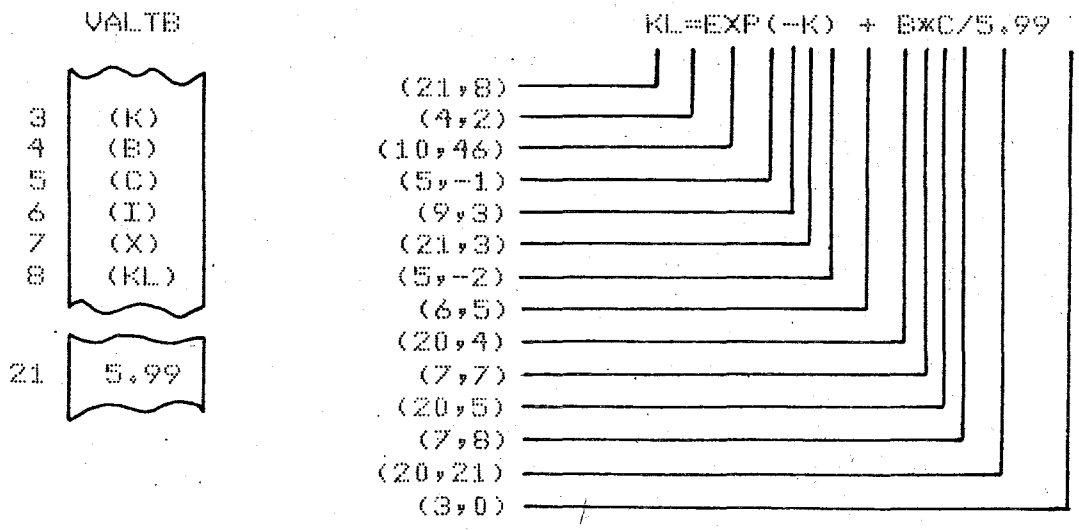
olduğunda leksik çözümleyicinin deyim çözümleyicisine gönderdiği kod çiftleri Şekil 6.21''de gösterilmektedir. Bu örnekte, KL,K,B,C,I ve X değişkenlerine 8,3,4,5,6 ve 7 numaralı hücrelerin değeri 5.99 olan değişmeze ise 21 numaralı hücrenin atandığı varsayılmaktadır.

Deyim öğesi	NCLASS	NVALUE
Tam sayı işlenen	21	VALTB endeksi
Gerçek sayı işlenen	20	VALTB endeksi
SIN	10	41
COS	10	42
ALOG	10	43
ALOG10	10	44
SQRT	10	45
EXP	10	46
ABS	10	47
FLOAT	10	48
IABS	10	57
INT	10	58
Tekli çıkarma	9	3
**	8	4
*	7	7
/	7	8
+	6	5
-	6	6
(	5	-1
)	5	-2
=	4	2
deyim sonu	3	0

Fonksiyon işlem kodları

Temel işlem kodları

Şekil 6.20: FCN-F Komut öğe kodları.



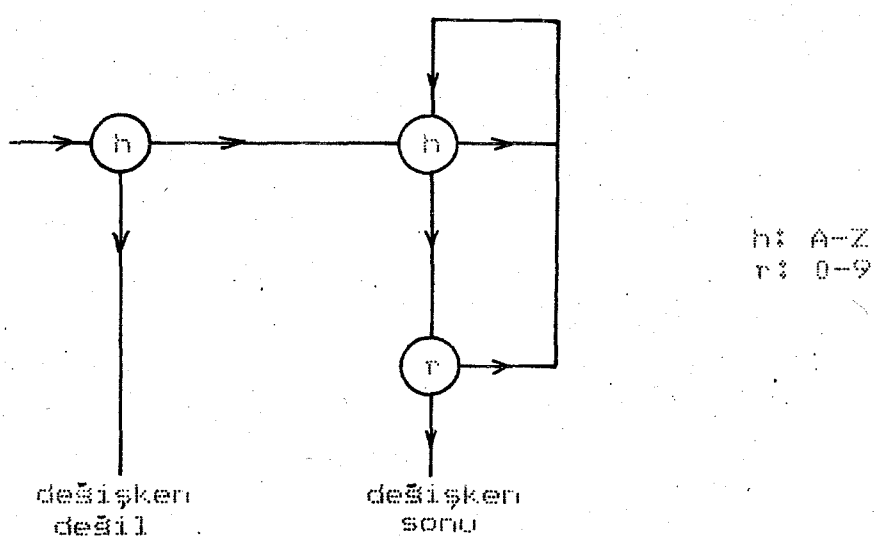
Şekil 6.21: Komut öğe kodlarının kullanımı.

6.3 Komut çözümleme modülleri

Şekil 6.4'de özetlendiği gibi sürücü (ana) program, komut çözümleyicisi ile kod işlemcisini koordine eder. Çözüm-işlem döngüsünün başlangıç noktası kullanıcının sağladığı komut türünün saptanmasıdır.

Tüm sözdizim çözümleri leksik çözümleyicinin ilettiği bilgiye göre yapılır. FCN-F'de sözdizim çözümü okunan komutun okuyucu tamponundaki görünümüne değil de leksik çözümleyicinin tamponu yorumlayış şekline bağımlıdır. Tek leksik çözümleyici yerine çözümlenmekte olan komutun özelliklerine göre dört değişik çözümleyici tercih edilmitir. Veri tablolarına bağımlı olarak çalışan bu rutinler GSM'nin (Glennie Syntax Machine) bir uyarlamasıdır (Glennie 60, Balman 80c).

GSM, iki çıkışlı düğümlerden oluşan sonlu bir otomattır. Her düğüm bir karakter kodunu veya karakter sınıfını temsil eder. Okuyucu tamponundan çekilen karakter, düğümün taşıdığı karakterle kıyaslanarak düğüm çıkışlarından biri seçilir ve yeni bir düğüme geçilir. Düğümlerdeki iki çıkış "doğru" (düğümdeki karakterle okunan karakter aynı) ve "yanlış" (düğümdeki karakterle okunan karakter değişik) olarak adlandırılır. "Doğru" geçişler tampondan yeni bir karakter çekerek düğüm değiştirirken, "yanlış" çıkışlar aynı ka-



Şekil 6.22: FORTRAN değişkenlerini tanıyan GSM modeli.

rakteri koruyarak yeni bir düğüme geçer. FORTRAN değişken tanıtıcılarının GSM ile nasıl tanınabileceği Şekil 6.22'de gösterilmektedir. Düğümlerden sağa doğru ilerleyen çıkışlar "doğru", aşağıya inen çıkışlar ise "yanlış"tır.

GSM geçerli tüm komut öğelerini tanıyabilecek nitelikte olduğu halde sözdizim çözümleme ve kod üretim için düğümler arası geçişleri denetleyip geçiş tipine göre, tanınma sürecinde olan öğe hakkındaki, bilgiyi yenilemesi gerekir. Şekil 6.22'de gösterilen GSM bir veri tablosuna dönüştürülebilir. Beş sütundan oluşan bu tablonun her sırası bir düğümü tanımlar. Sütunlar şu bilgileri yansıtır

i) Düğümdeki karakter veya karakter tipi

ii) "Doğru" çıkışın bağlandığı düğüm

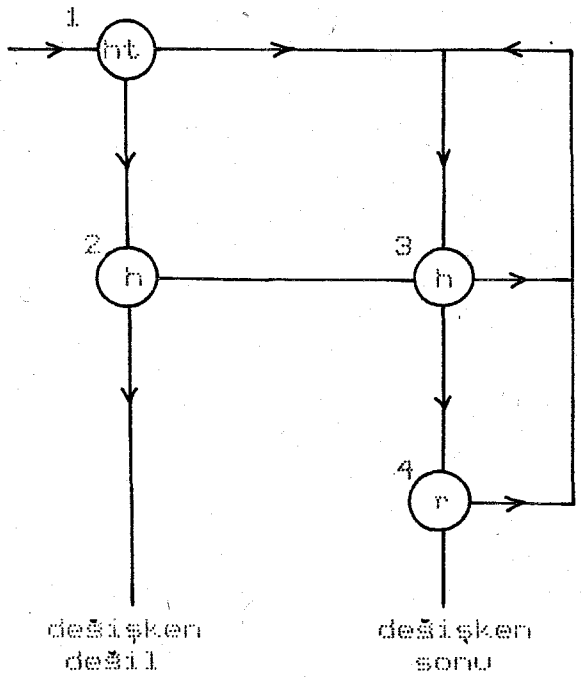
iii) "Doğru " geçişlerde yapılacak işlem

iv) "Yanlış" çıkışın bağlandığı düğüm

v) "Yanlış" geçişlerde yapılacak işlem

Tanıma işlemini durdurmak amacıyla ii. ve iv. sütunlarda eksi sayılar kullanmak mümkündür; hatta bu eksi sayıların tanınan öğenin NCLASS değeri olması uygundur. Hatalı duruşları -100'den küçük sayılarla belirtebiliriz; örneğin, bir sonraki düğüm "-21" olarak verilirse NCLASS değeri "21" olan komut öğesi tanınmıştır; düğüm "-108" olarak verilirse "8" numaralı hata görülmüştür.

Bu yöntemi kullanarak, FORTRAN gerçek ve tamsayı değişken tanıtıcılarını saptayan GSM ve karşılığı olan veri tablosu, Şekil 6.23'de gösterilmektedir.



Düğüm	Karakter	"Doğru"	"Doğru" işlem	"Yanlış"	"Yanlış" işlem
1	ht	3	İşlem1	2	-
2	h	3	İşlem2	?	-
3	h	3	İşlem3	4	-
4	r	3	İşlem3	-20	İşlem4

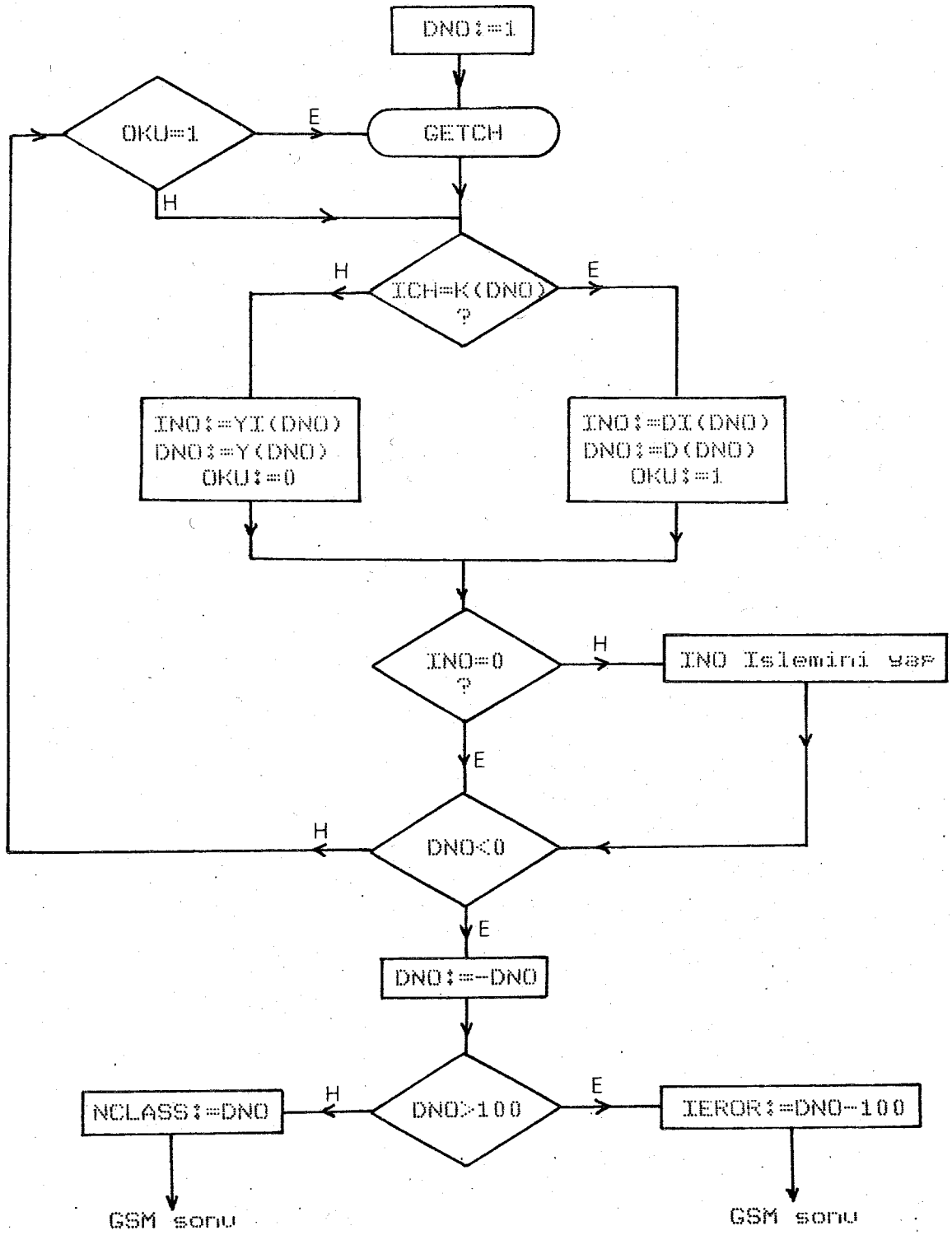
İşlem1: TS:=1;
Tam sayı değişkenin ilk harfi.

İşlem2: TS:=0;
Gerçek sayı değişkenin ilk harfi.

İşlem3: Karakteri derlenmekte olan değişkene ekle;
Karakter sayısını arttır ve gerekirse hata mesajı ver.

İşlem4: NCLASS:=TS+20;
Değişken tablosundaki pozisyonu saptayıp NVALUE bilgisini derle.

Şekil 6.23: Tam sayı ve gerçek sayı değişkenleri tanıyan GSM modeli.



Şekil 6.24: GSM yönteminin akış çizelgesi.

Şekil 6.23'de gösterilen tablodaki, veya aynı yöntemle geliştirilen herhangibir GSM tablosundaki, bilgiyi kullanarak öğe tanıyacak programın akış çizelgesi Şekil 6.24'te verilmektedir. Çizelgede şu kısaltmalar kullanılıyor

K: Düğüm karakterleri dizisi

D: "Doğru" çıkış dizisi

Y: "Yanlış" çıkış dizisi

Dİ: "Doğru" çıkışlarda yapılacak işlemlerin dizisi

Yİ: "Yanlış" çıkışlarda yapılacak işlemlerin dizisi

GETCH: Okuyucu tamponundan bir sonraki (boş olmayan) karakteri okuyan altrutin

ICH: Okuyucu tamponundan alınan karakter kodu veya karakter tipi

DNO: Düğüm endeksi

İNO: İşlem endeksi

IEROR: Hata endeksi

NCLASS: Tanınan öğe türü

6.3.1 Komut tipinin saptanması

Tüm kullanıcı komutlarının geçtiği ilk leksik süzgeç, komut tipini saptayıp işaret alanındaki bilgiyi derleyen GSM'dir.

BU GSM'de, komut okunduktan sonra işaret alanındaki bilgi derlenir ve gerekirse tüm işaret bilgilerinin tutulduğu IFRTB tablosuna işlenir. Daha sonra, komut anahtar sözcüğü saptanılır, ve bulunan komut türünü çözümleyecek olan altrutin çağırılır. Komut türünün saptanmasında kullanılan yöntem şöyledir.

i) "FORMAT (" : Format çözümleyicisi

ii) "READ (" : Read çözümleyicisi

iii) "WRITE (" : Write çözümleyicisi

iv) "FINISH" : Alıştırıcı sonu

v) "RESET" : Çalışma alanlarının temizlenmesi

vi) değişken" " : Aritmetik deyim çözümleyicisi

Bu komutların saptanış şekli ile düğüm değişirken yapılacak işlemlerin endeksleri (İ1-İ9) Şekil 6.25'deki GSM çizelgesinde verilmiştir. Anahtar sözcükler (FORMAT, READ, WRITE, FINISH ve RESET) tanınırken, sonucun başarısı bilinmediğinden, tanıma sürecinde İ1 ve İ3 işlemleriyle adama komutuna hazırlık yapılır; okunan sözcükte bir sapma olursa anahtar sözcük tanıyan düğümlerden çıkıp değişken tanıma düğümlerine geçilir. Kullanılan işlem endeksleri aşağıda özetlenmektedir.

İ1: Gerçek sayı değişkenin başlangıcı. Değişkendeki karakter sayısı birlendikten sonra, karakter geçici değişken alanına yerleştirilir.

İ2: Tam sayı değişkenin başlangıcı. Değişkenin tipi dışında, yapılan işlemler İ1'deki gibidir.

İ3: Değişken devamı. Karakter sayısı bir arttırılır ve altıyı aşıp aşmadığı kontrol edilir. Karakter geçici değişken alanına işlenir.

İ4: Atama komutu. Geçici değişken alanındaki karakter dizisini kullanarak " =" işaretinin sol tarafındaki değişken IDTAB'da (değişken tablosu) aranır; bulunmadığı takdirde tabloya işlenir. Tabloda değişkene verilen pozisyon NVALUE'a alındıktan sonra aritmetik deyimleri çözümleyen modül başlatılır.

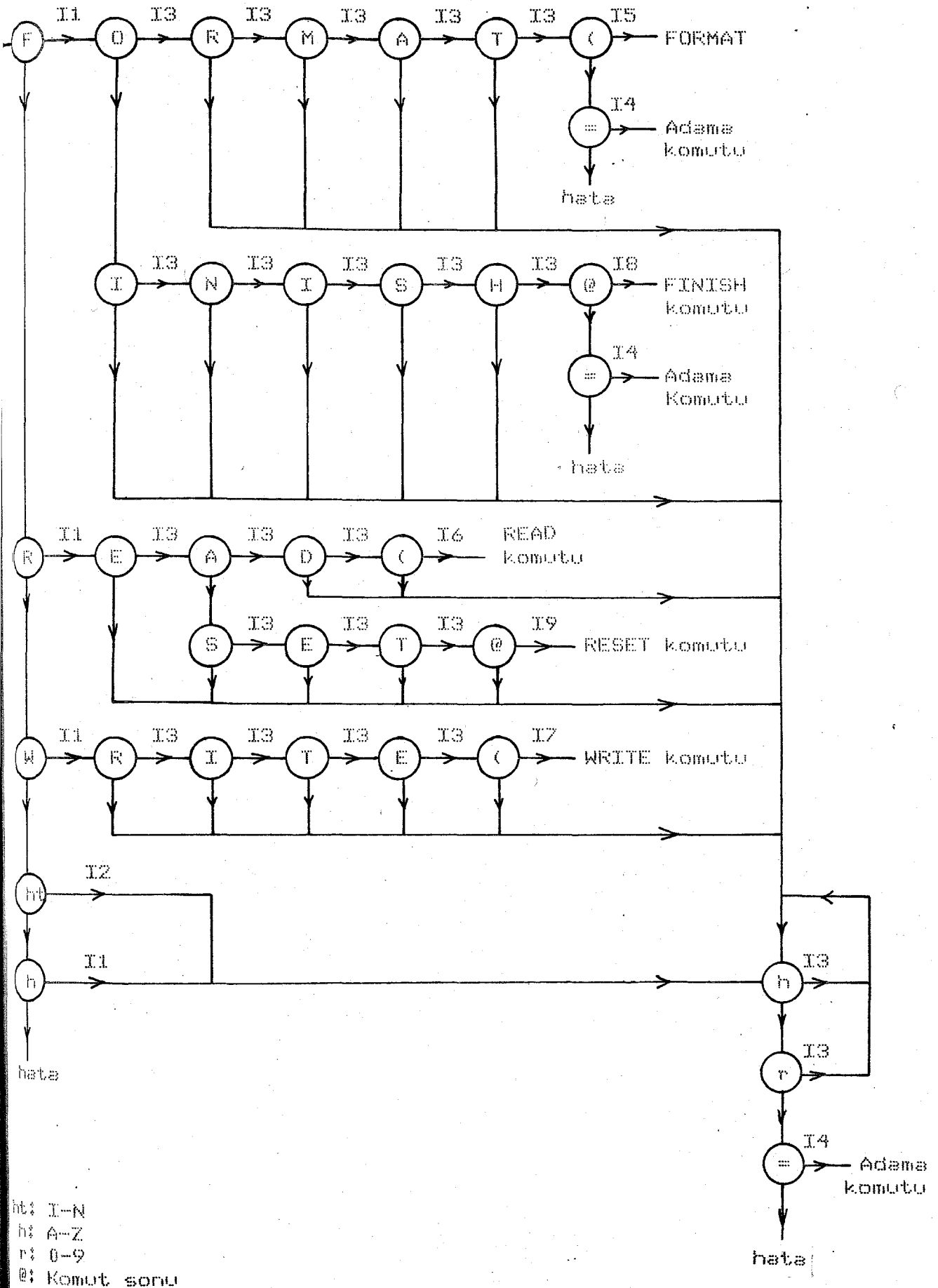
İ5: FORMAT komutu . İşaret alanı boş bırakılmamışsa FORMAT komutlarını çözümleyen modül başlatılır.

İ6: READ komutu. Komutun READ olduğunu belirtip READ/WRITE komutlarını çözümleyen modül başlatılır.

İ7: WRITE komutu. Komutun WRITE olduğunu belirtip READ/WRITE komutlarını çözümleyen modül başlatılır.

İ8: FINISH komutu. Alıştırıcıdan çıkılır

İ9: RESET komutu. Kullanıcının değişken, komut işareti ve FORMAT alan endeksleri geriye alınarak bu alanlar temizlenir.



Şekil 6.25: Komut türünü saptayan GSM modeli.

6.3.2 Aritmetik deyim çözümü ve kod üretimi

Bu modül başlatıldığında atama komutunun "=" işaretine kadar olan kısmı çözümlenmiş ve sol taraftaki değişkenin özellikleri saptanmış olur. Sağ taraftaki deyimin çözümleme sürücülüğünü infix-postfix (iç yazılım-son yazılım) çeviri programı yapar. Komut öğelerini tanıyan bir GSM modülünden (GSM2) NCLASS ve NVALUE çiftlerini alan sürücü, öğenin türüne göre, işlem ve işlenen torbalarını (IPSTAK, IDSTAK) kullanarak kod üretim modülünü (TRNOPS) çalıştırır.

Sürücü modülün genel akış çizelgesi Şekil 6.26'da verilmiştir; çizelgede kullanılan semboller aşağıda açıklanmaktadır.

APS: Açık parantez sayısı

TRNOPS: Kod üretim sürücüsü (Şekil 6.27)

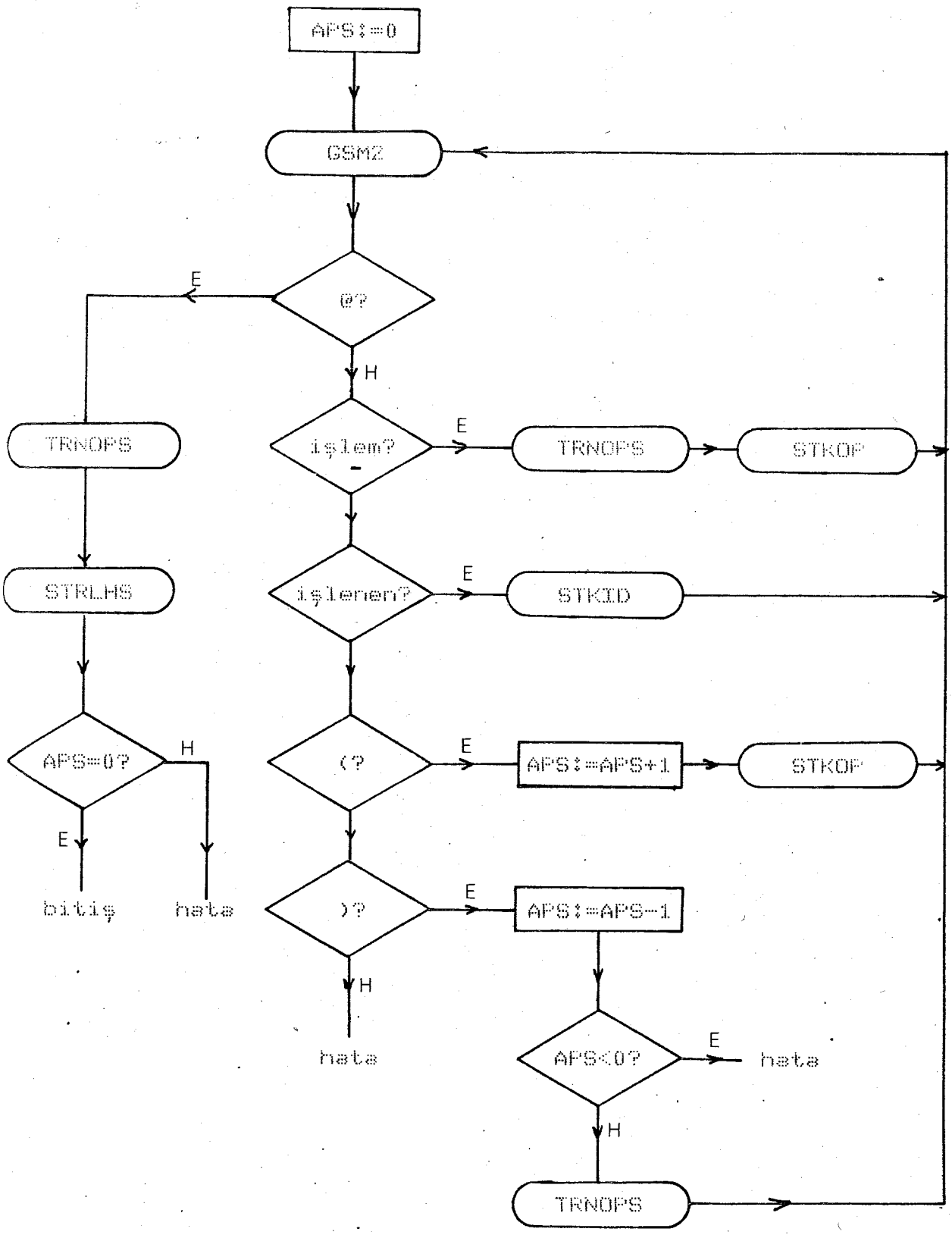
STKOP: İşlem torbasına giriş modülü. NCLASS ve NVALUE'yla tanımlanan işlemi torbaya yerleştirir.

STKID: İşlenen torbasına giriş modülü. NCLASS ve NVALUE'yla tanımlanan işleneni torbaya yerleştirir. Eğer torbada iki işlenen sayacı bastırıyorsa, sayaçtaki değeri korumak için bir taşıma koduyla, değerin tutulacağı geçici hücrenin endeksi üretilir.

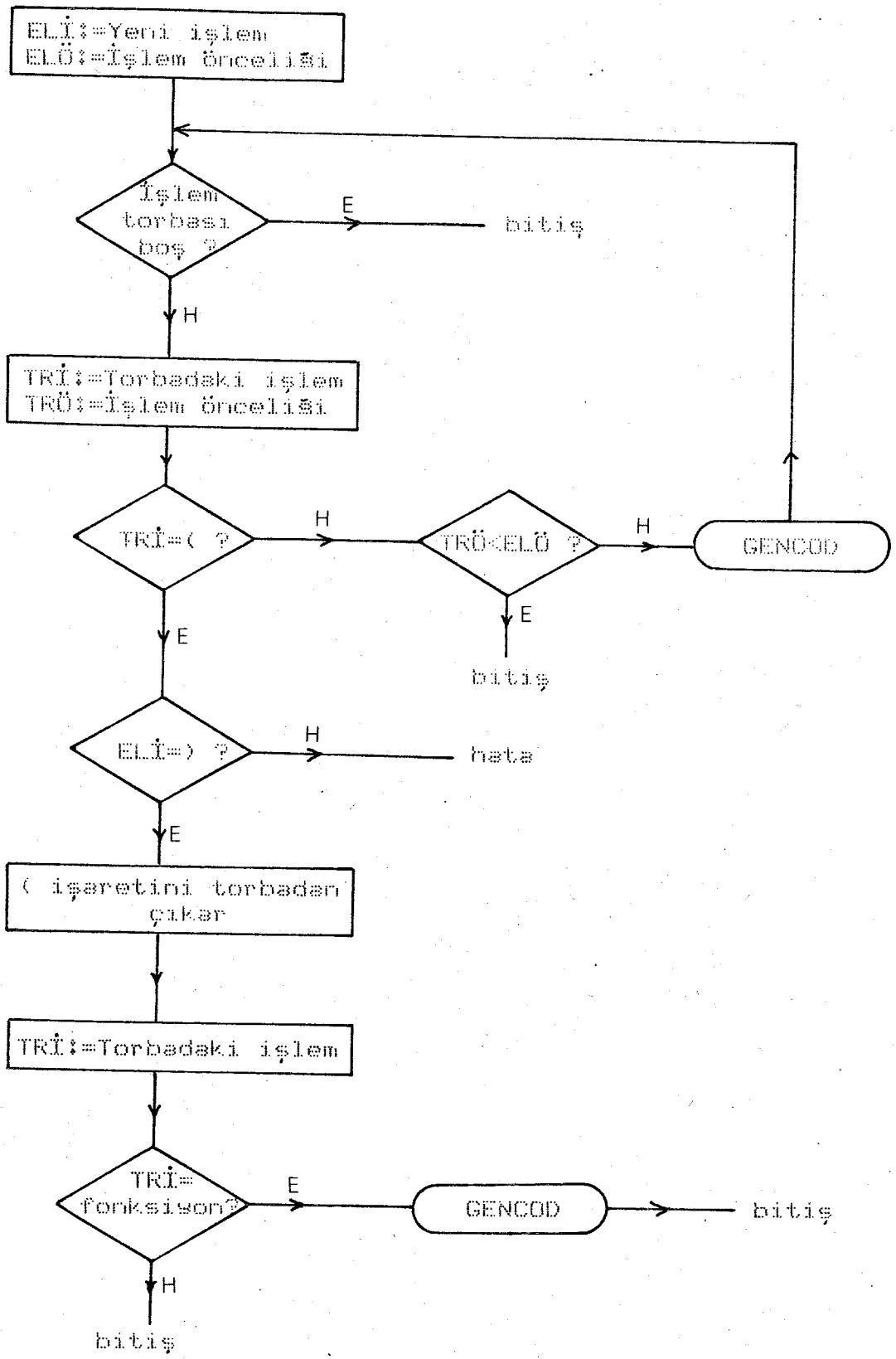
GSM2: Leksik çözümleyici. Okuyucu tamponundan toparlanan bilgi NCLASS ve NVALUE değeri olarak iletilir (Bkz. Şekil 6.29).

STRLHS: Sayaçtaki değeri sol taraftaki değişkene taşıyacak işlem kodunu üretir. Eğer sağ taraftaki deyimde hiç işlem yoksa deyim yerine kullanılan değişkeni sayaca yükleyecek kodu üretir. Eğer adama alıştırıcısı kullanılıyorsa sonucu kullanıcıya iletecek çıkış kodunu üretir.

GENCOD: Kod üreticisi. İşlem torbasından çıkış olacaksa TRNOPS tarafından çalıştırılan bu altrutin işlem ve işlenen torbalarındaki bilgiye göre kod üretip, işlem sonucunun sayaçta olduğunu belirtmek için, işlenen torbasına "-1" işaretiyle sonuç tipini (gerçek veya tamsayı) koyar.



Şekil 6.26: Aritmetik deyim çeviri sürücüsü.



Şekil 6.27: Aritmetik kod üretim sürücüsü (TRNOPS).

İşlem	İşlenen 1	İşlenen 2	Üretilen işlem komutları
Tekli çıkarma	X	—	YÜK (G/T) X EKSİ(G/T)
	Savaç	—	EKSİ(G/T)
Fonksiyon (Kod=F)	X	—	YÜK (G/T) X F
	Savaç	—	F
+	X	Y	YÜK (G/T) X TOP (G/T) Y
	Savaç	Y	TOP (G/T) Y
	X	Savaç	TOP (G/T) X
*	X	Y	YÜK (G/T) X ÇARP(G/T) Y
	Savaç	Y	ÇARP(G/T) Y
	X	Savaç	ÇARP(G/T) X
—	X	Y	YÜK (G/T) X ÇIK (G/T) Y
	Savaç	Y	ÇIK (G/T) Y
	X	Savaç	TAŞI(G/T) Z YÜK (G/T) X ÇIK (G/T) Z
/	X	Y	YÜK (G/T) X BÖL (G/T) Y
	Savaç	Y	BÖL (G/T) Y
	X	Savaç	TAŞI(G/T) Z YÜK (G/T) X BÖL (G/T) Z
**	X	Y	YÜK (G/T) X ÜS (G/T) Y
	Savaç	Y	ÜS (G/T) Y
	X	Savaç	TAŞI(G/T) Z YÜK (G/T) X ÜS (G/T) Z

Şekil 6.28: Aritmetik kod üretimi (GENCOD).

GENCOD'un değişik durumlarda üretebileceği işlem kodları Şekil 6.28'de verilmiştir; iki işlenen sütunlarındaki "X" ve "Y" işlenen torbasının en üstteki elemanlarıdır. İşlenenlerin biri sayaçsa bu durum torbada "-1" ile belirtilir. Tabloda kullanılan "Z", işlem zamanında geçici olarak kullanabilen bir hücrenin endeksidir.

Tabloda bir çok işlem komutunu izleyen "(G/T)" harfleri işlenenlerin tipine göre GENCOD tarafından saptanır. İşlenenler aynı tipteyse (her ikisi de gerçek sayı veya her ikisi de tamsayı) üretilen işlem aynı tipi korur. Eğer iki tip farklı ise üretilen işlem gerçek sayı sonuçludur. Ancak, FCN-F karma deyimlerin kabul edilmediği opsiyonla yaratıldığında (MDCHK=1, bkz.6.2.1), eğer iki tip farklı ise hata mesajı verilir; tek istisna, üs alma işlemidir.

Şekil 6.26'da izleneceği gibi, aritmetik deyim çeviri sürücüsü deyimin hatasız olduğunu varsayarak çalışır. Sürücünün bulmakla yükümlü olduğu tek sözdizim hatası dengesiz parantezlerdir. Bunun dışındaki tüm hataları leksik çözümleyici (GSM2) bulur. GSM2'de kullanılan GSM modelinin sürücüye verdiği hizmetler şunlardır.

i) Değişkenleri saptayıp değişken tablosundaki pozisyonları bildirmek

ii) Fonksiyonları saptayıp karşılıkları olan işlem kodunu bildirmek

iii) Değişmezlerin değerini saptayıp bunları değer tablosuna yerleştirmek ve tablodaki pozisyonunu bildirmek

iv) Tekil işlemleri (+ veya -) bulmak

v) "*" ve "**" ayırımını sağlamak

vi) Şekil 6.20'de verilen komut öğe kodlarını üretmek

vii) Dengesiz parantez hataları dışında tüm sözdizim hatalarını bulmak.

Komut öğelerini saptayan GSM Şekil 6.29'da verilmiştir. Değişken ve fonksiyonların tesbit edildiği en üstteki dört düğüm bağlantılarında yapılan işlemler Bölüm 6.3.1'de tanıtılan işlemler gibidir. Modelin en alttaki sekiz düğümü özel işlem gerektirmeyen işlem işareti tanıma düğümleridir. Gerçek sayı ve tamsayı değişmezlerin saptandığı orta bölümde kullanılan beş işlem (D1-D5) geçerli oldukları bağlantılarda işaretlenmiştir. Bu işlemlerde kullanılan değişkenler

DD: Değişmez değeri (başlarken 0.0)

NS: Noktadan sonra gelen rakam sayısı (başta 0)

ÜS: Kayan noktalı sayılarda üs değeri (başta 0)

Üİ: ÜS işaret katsayısı (başta 1)

D1-D5 işlemlerinde x, okunan karakter kodlu rakamın sayısal değeri olarak kabul edilmiştir.

D1: $DD = DD.10 + x$

Klâsik soldan sağa okuyarak değer tesbiti . Bu işlem noktadan önceki kısımda kullanılır.

D2: $DD = DD.10 + x$

$NS = NS + 1$

Noktadan sonra gelen rakamlarda kullanılır. Soldan sağa değer tesbiti devam ederken ondalık kesirdeki rakam sayısı arttırılır.

D3: $Üİ = -1$

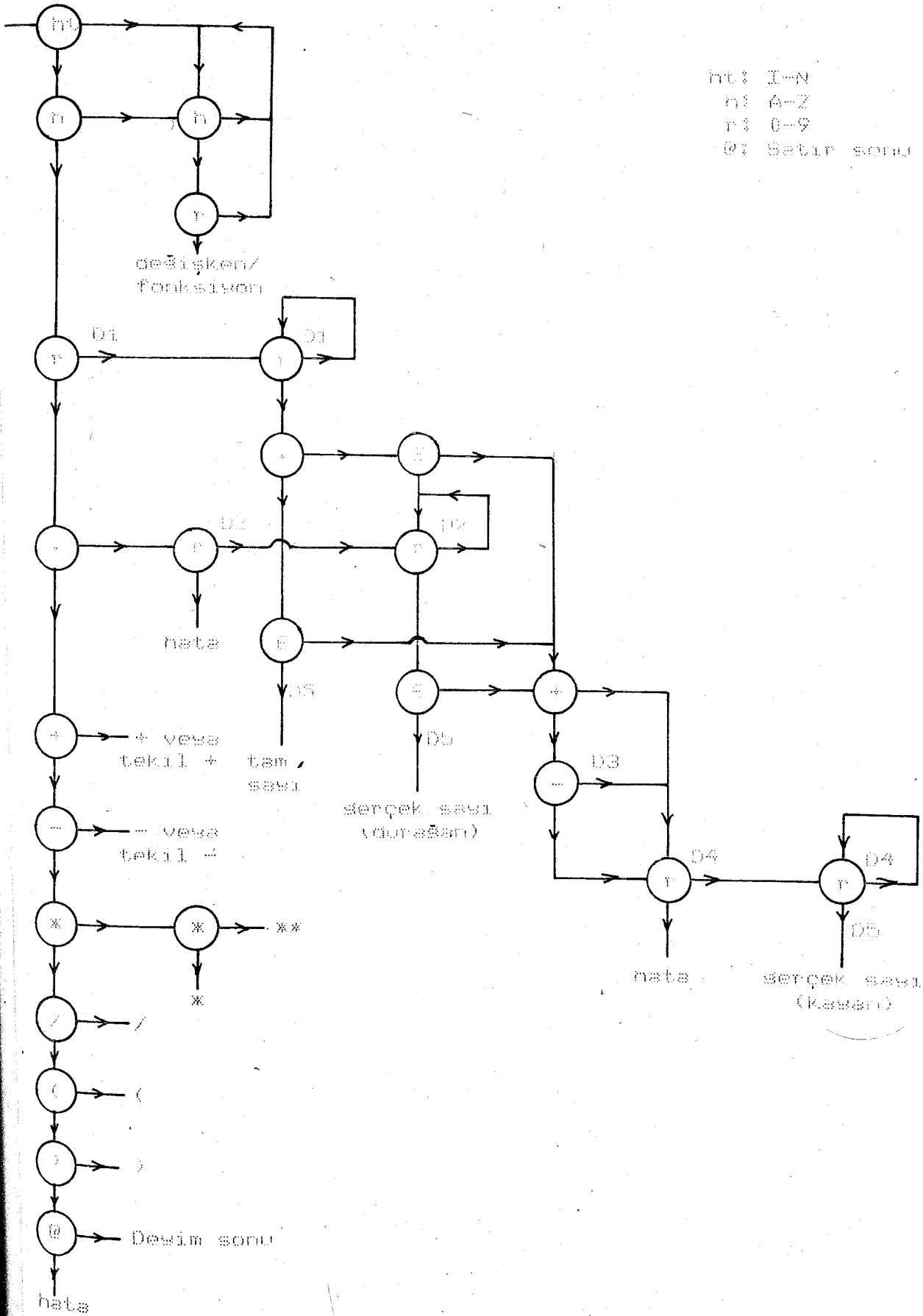
Bu işlem sadece "E" işaretinden sonra gelen "-" işareti görüldüğünde yapılır

D4: $ÜS = ÜS.10 + x$

Soldan sağa okurken "E"den sonra gelen 10'un üssü hesaplanır.

D5: değişmez $= DD.10^{Üİ.ÜS-NS}$

Tamsayı veya gerçek sayı oluşturan rakam dizgesinin sonunda yapılan işlemdir; sayının değeri bu işlemle saptanır. VALTB içinde değişmezlere ayırılan alandaki bir hücreye, bulunan değer yerleştirilir ve hücrenin endeksi NVALUE aracılığıyla deyim çözümleme sürücüsüne bil-


ht: I-N
h: A-Z
r: 0-9
@: Satır sonu
değişken/
fonksiyon
D1
D2
hata
tam
sayı
D3
D4
D5
gerçek sayı
(ondalıklı)
gerçek sayı
(kesirli)
+ veya
tekil +
- veya
tekil -
**
Devim sonu
hata

Şekil 6.20

Sayı değeri D5 ile hesaplanırken bilgisayar sisteminin sağladığı en büyük tamsayı (XMXINT) ve en büyük 10 üssü (MXPOW) gözönünde tutularak üsttaşma hataları donanıma yansıtılmadan saptanır.

Şekil 6.29'da verilen GSM modeli yalnız leksik hataları bulabilir; sözdizim hatalarıyla tekil işaretleri saptamak amacıyla komut öğeleri ikinci bir süzgeçten geçirilir. Bu işlem Şekil 6.30'da özetlenen matrisin aracılığıyla gerçekleştirilmiştir. Matris elemanlarına, bulunan öğe (sütun) ve önceki öğe (sıra) ile erişilir. Şekil 6.30'da boş bırakılan elemanlar süzgeçten etkilenmeden geçen öğeleri tanımlar. "Tekil" olarak işaretlenen elemanlar bulunan öğenin ("+" veya"-") tekil işlem olduğunu gösterir; örneğin, "("veya"="işaretinden hemen sonra gelen"-" tekildir. "Hata" olarak işaretlenen kutularda hata tipine bağımlı olarak mesaj numarası işlenmiştir; böylece hata farkedilirken gereken mesaj da bulunmuş olur.

	Deyim sonu	=	(	)	-,+	/,*,**	Fonksiyon	İşlenen
Deyim sonu								
=	Hata	Hata		Hata	Tekil	Hata		
(	Hata	Hata		Hata	Tekil	Hata		
)		Hata	Hata				Hata	Hata
-,+	Hata	Hata		Hata	Hata	Hata		
Tekil,/,*,**	Hata	Hata		Hata	Hata	Hata		
Fonksiyon	Hata	Hata		Hata	Hata	Hata	Hata	Hata
İşlenen		Hata	Hata				Hata	Hata

Şekil 6.30: Aritmetik deyim sözdizim tablosu.

6.3.3 READ/WRITE çözümü ve kod üretimi

Kullanılan üçüncü GSM modeli READ/WRITE çözümleyicisidir. Bu modülde, düğümler arası geçiş işlemlerinden yararlanılarak, kod üretimi de gerçekleştirilir.

GSM modeli Şekil 6.31'de verilmiştir. En üstteki dört düğüm, kanal numarasıyla desen işaret sayısını okumak için kullanılır. Çözümlenecek olan komut türüne göre (READ veya WRITE) verilen kanal numarası gereken kanal numarasıyla karşılaştırılır. Desen işaret sayısı okunurken karşılığı olan nümerik değere dönüştürülür (bkz. işlem D1, D3, Şekil 6.29) ve elde edilen işaretin IFRTB endeksi saptanır. Eğer giriş/çıkış işleminde kullanılacak olan desen daha önceden tanımlanmışsa, işaret sayısı IFRTB'ye yerleştirilir, ve NEXPCT değişkenine adanarak bu desenin tanımlanması gerektiği belirtilir.

Kod üretimini gerçekleştiren işlemler (R1-R3) Şekil 6.31'deki çizelgede işaretlenmiştir. Bu işlemlerin görevleri

R1: Kullanılacak desenin IFRTB endeksi bulunmuştur. Çözümlenmekte olan komut türüne göre, hangi desenin kullanılacağını tanımlayan işlem kodu üretilir:

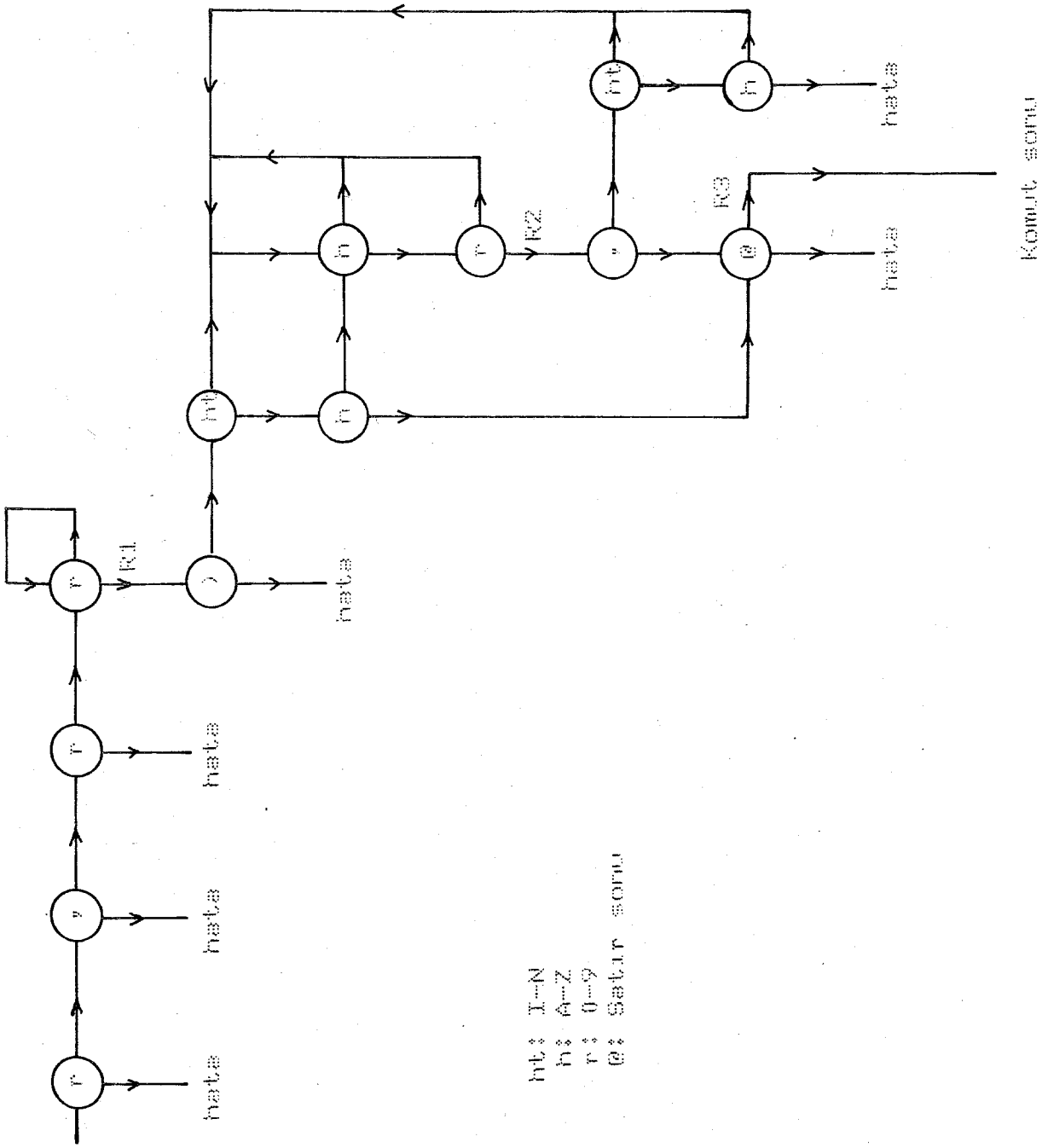
ODESEN	işaret endeksi	(READ komutlarında)
YDESEN	işaret endeksi	(WRİTE komutlarında)

R2: Giriş/çıkış listesinde yeni bir değişken görülmüş (gerçek veya tamsayı değerli) ve bunun VALTB'deki hücre endeksi saptanmıştır. Üretilen işlem kod aşağıdakilerin biri olur

OKUG	değişken endeksi	(READ, gerçek sayı)
OKUT	değişken endeksi	(READ, tamsayı)
YAZG	değişken endeksi	(WRITE, gerçek sayı)
YAZT	değişken endeksi	(WRITE, tam sayı)

R3: READ/WRITE komutunun sonu gelmiştir. İşlem sonunu belirtmek için iki koddan biri üretilir

DUR	(READ komutlarında)
DURY	(WRITE komutlarında)



Şekil 6.31: READ/WRITE çözümleyen GSM modeli.

6.3.4 FORMAT çözümüyle öğe kodu üretimi

Çözüm ve çeviri işlemlerini dördüncü GSM modeli yapar. READ/WRITE çözümleyicisindeki gibi girişte, anahtar sözcük ile birlikte ilk "(" , komut türünü saptayan GSM (bkz. 6.3.1) tarafından bulunmuş olur. IFRMT'deki kodları etkileyen işlemler düğümler arası geçişlerde yapılır.

Kullanılan GSM modeliyle bazı geçiş işlemleri (F1-F9) Şekil 6.32'de verilmiştir. Bu modelde yalnızca desen öğeleri tanındığından, peş peşe gelen öğelerin sözdizim hataları aritmetik deyim çözümleyicisindeki gibi bir tablo aracılığıyla denetlenir.

Şekil 6.32'de işaretlenen önemli işlemler şunlardır

F1: Bir rakam dizgesi görülüp sayısal değeri hesaplanmıştır. Elde edilen sayı okunacak desen öğesinin kaç kez tekrarlanacağını bildirir. Eğer tekrar sayısı belirtilmemişse, tekrar sayısı bir olarak alınır.

F2: Tekrar sayısı (yoksa, 1), "I" harfi ve alan genişliği saptanmıştır. Üretilen desen kodu iki hücre tutar

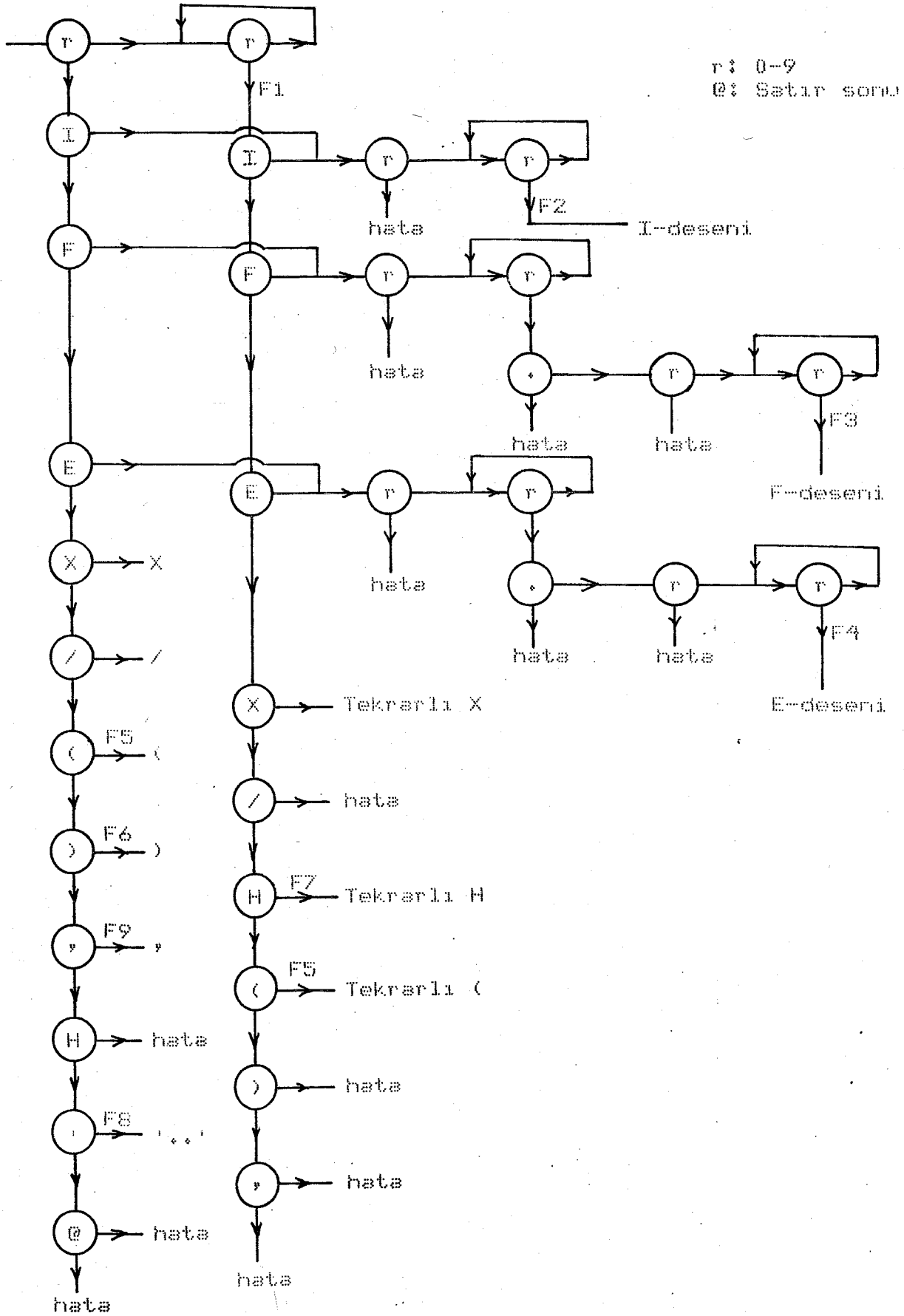
10. (tekrar sayısı) + (I-kodu)
Alan genişliği

F3: Tekrar sayısı (yoksa,1), "F" harfi, alan genişliği ve ondalık kısmın uzunluğu saptanmıştır. Alan genişliği ile ondalık kesir için avırılan verin geçerliliği saptandıktan sonra IFRMT'ye üç hücrelik desen kodu yerleştirilir

10. (tekrar savısı) + (F-kodu)
Alan genişliği
Ondalık kısmın genişliği

F4: F3'de belirtilen işlemler "E"-öğesi için tekrarlanır.

F5: Tekrarlı (örneğin "5(") veya takrarsız bir aç parantez bulunmuştur. Parantez kodu IFRMT'ye işlenir ve kapanmamış parantez sayısı arttırılır.



Şekil 6.32: Desen öğelerini saptayan GSM modeli.

F6: Parantez kapayan işaret bulunmuştur. Karşılığı olan kod üretildikten sonra kapanmamış parantez sayısı azaltılır. Eğer tüm parantezler kapanmışsa komut sonu olduğu anlaşılır. Bu safhada, okuyucu tamponunda işlenmemiş ve boş olmayan karakter varsa komut hatalıdır. Gerçekten komut sonu gelmişse desen sonunu belirten kod IFRMT'ye işlenir.

F7: Tekrar sayısının ardından "H" harfi okunmuştur. Bulunan sayı geçerli ise üretilen desen kodu

10. (Tekrar sayısı) + (H-kodu)

olur. Tekrar sayısında belirtilen karakter sayısı, okuyucu tamponundan alınıp üretilen kodun peşinden IFRMT'ye işlenir.

F8: Tırnak içine alınmış dizginin ilk tırnak işareti görülmüştür. Desen kodlarının tutulduğu dizide bir hücre boş bırakılır ve okuyucu tamponundaki karakterler, ikinci tırnak işareti görülene dek, IFRMT'ye işlenir; kaç karakter okunduğu hesaplanır ve boş bırakılan IFRMT hücresine

10. (karakter sayısı) + (H-kodu)

yazılır.

F9: "," işareti için kod üretilmez.

Diğer hatasız düğüm çıkışlarında sadece gerekli desen kodu üretilip IFRMT'ye işlenir. Tüm hatasız çıkışlarda Şekil 6.33'de gösterilen tablo kullanılarak öğe dizimleri denetlenir. Bu tablodaki sütunlar bulunan öğeleri, sıralar ise bir önceki öğeleri gösterir. Boş bırakılan kutular geçerli öğe dizimlerini belirtir; "hata" olarak işaretlenen kutularda hata mesajı numaraları vardır.

	F,E,I H,X	/	(	)	,
F,E,I,H,X	Hata		Hata		
/					
(				Hata	Hata
)	Hata		Hata		
,				Hata	Hata

6.4 Kod yorumlayan modüller

Kullanıcının sağladığı komutlar 6.3'üncü bölümde tanıtılan modüllerle çevirilirken etkilenebilen işlem alanları şunlardır

IDTAB Kullanıcı değişken tablosu

VALTB Kullanıcı değişmez değerleri

IFRTB, IFRMT Kullanıcı komut işaretleriyle desen kodları

ICODEX Yorumlanacak işlem kodları

Yazılan komut hatasız olduğu takdirde, eğer FORMAT değilse, kod yorumlayıcısı çalıştırılır. İşlem sonunda ICODEX alanı temizlenip bir sonraki komut okunur.

Kod yorumlayıcının kalıcı olarak etkilediği tek işlem alanı kullanıcı değişken değerlerinin tutulduğu VALTB dizisidir. Yorumlanan kodları dört sınıfa ayırabiliriz.

i) Temel işlem kodları (ICODEX)

ii) Fonksiyon işlem kodları (ICODEX)

iii) Giriş/çıkış işlem kodları (ICODEX)

iv) Desen kodları (IFRTB, IFRMT)

Daha önceki bölümlerde tanıtılan bu kodların yorumunda FORTRAN dili kullanılmıştır. Ancak, özellikle temel işlem ve fonksiyon kodlarına, hatalı işlemlerin bilgisayara yansımaması için özel yöntemler gerekmektedir. Hata önleyici yöntemler de FORTRAN dilinin sağladığı olanaklar çerçevesinde geliştirilmiştir. İşlem esnasında hata önlemek amacıyla kullanılan değişkenler şunlardır

XMXINT Bilgisayarın sağladığı en büyük tamsayıyı gerçek sayı olarak tanımlayan değişken

XMRL10 En büyük gerçek sayı bölü 10.

ALXMRL En büyük gerçek sayının doğal logaritması

ALALXM ALXMRL'in logaritması

6.4.1 Temel kodların yorumu

Sayaç yükleme: VALTB'ın belirtilen hücresinden alınan değer sayaca yüklenir. Yorum sırasında hata olasılığı yoktur.

Sayaç taşıma: Sayaçtaki değer belirtilen VALTB hücresine taşınır. Tek hata olasılığı tamsayı değişkene gerçek değerli deyim adandığındadır. Sayaçtaki değer XMXINT'le kıyaslanır; daha büyükse, işlem hatası verilir.

Tekil çıkarma: Sayaçtaki değerin işareti değiştirilir. Hata olasılığı yoktur.

Üs alma : Yapılan işlem , sayaçtaki sayının belirtilen VALTB hücresindeki değerle üssünü alıp sonucu sayaca koymaktır. Bu işlem yapılmadan önce gereken hata denetimleri şunlardır

i) Üst taşma

ii) Eksi sayının gerçek üssünü (gerekirse) önleme

Sayaçtaki veya hücredeki değer, 0 veya 1 ise hata denetimine gerek yoktur. Hücredeki değeri x, ve sayaçtaki değeri s olarak tanımlarsak , üst taşma denetimini

$$|\ln|x| + \ln|\ln|s|| \leq \text{ALALXM}$$

sağlar. Eğer sayaçtaki değer sıfırdan küçük bir gerçek sayıysa ve hücredeki değer tamsayı değilse işlem yapılamaz. Üs almanın sakıncasız olduğu görülürse bu işlem yapılır. Eğer tamsayı sonuçlu üs alma istenmişse, sayaçtaki sonuç XMXINT'le kıyaslanır.

Toplama: Yapılan işlem, sayaçtaki sayıyla belirtilen VALTB hücresindeki sayıyı toplayıp sonucu sayaca koymaktır. Üsttaşma olasılığının düşük olmasına rağmen işlemden önce

$$\left|\frac{s}{10} + \frac{x}{10}\right| \leq \text{XMRL10}$$

denetimi yapılır. Tam sayı sonuçlu toplama işlemlerinden sonra sayaçtaki değer XMXINT ile kıyaslanır

Çıkarma: Bu işlemde, sayaçtaki sayıdan belirtilen VALTB hücresindeki değer çıkarılarak sonuç sayaca alınır. Yapılan denetim toplama işlemindekine benzer

$$\left| \frac{s}{10} - \frac{x}{10} \right| < \text{XMRL\_10}$$

Çarpma: Eğer x veya s sıfır değilse, yapılan üst taşma denetimi şudur

$$|\ln|s| + \ln|x|| < \text{AL\_XMRL}$$

İşlem tamsayı çarpma ise, sonuç XMXINT'le kıyaslanır.

Bölme: Yapılan denetim çarpma işlemine benzer

$$|\ln|s| - \ln|x|| < \text{AL\_XMRL}$$

6.4.2 Fonksiyon işlem kodları

Fonksiyon kodları yorumlanırken, kullanılan bilgisayar sisteminin SIN, COS, ALOG, SQRT ve EXP fonksiyonlarını desteklediği varsayılmıştır. Tüm işlemlerde, sayaçtaki değer fonksiyonun argümanı olarak alınıp sonuç yine sayaca bırakılır. Ancak, argüman hataları bilgisayardaki fonksiyonlara yansıtılmadan işlem durdurulmalıdır.

SIN: İşlem sırasında hata olasılığı yoktur. Ancak, sistemin sağladığı SIN fonksiyonunun eksi açıları kapsamadığı durumlar gözönüne alınarak yapılan işlem şöyledir

eğer $s \geq 0$, $s := \sin s$

eğer $s < 0$, $s := -\sin|s|$

COS: Hatalı argüman olasılığı yoktur. Yapılan işlem

$s := \cos|s|$

ALOG: Tek hata sayaçtaki değer sıfır veya daha küçük olduğundadır.

ALOG10: İşlemden önce ALOG'daki denetim yapılır. ALOG10 fonksiyonunun bilgisayar sisteminde sağlanmadığı durumlar gözönüne alınarak yapılan işlem

$$s := \frac{\ln s}{\ln 10}$$

SQRT: Tek hata argümanın sıfırdan küçük olmasıdır.

EXP: Üsttaşma hatalarına karşı yapılan denetim, sayaçtaki değerin ALXMRL'den küçük olup olmadığını saptar.

ABS: Hata olasılığı yoktur.

FLOAT: Hata olasılığı yoktur.

IABS: İşlemden sonra tamsayı üsttaşma denetimi yapılır.

INT: İşlemden sonra tamsayı üsttaşma denetimi yapılır.

6.4.3 Giriş/çıkış işlem kodları

Giriş/çıkış kodlarının yorumunda kullanılan başlıca mOdülleri şöyle özetleyebiliriz.

EXACFD: Bir sonraki aktif desen öğesini (E,F veya I) bulan modül

IFORR: Belirtilen okuyucu tampon alanından bir tamsayı okuyan modül

IFORW: Belirtilen tampon alanına bir tamsayı yazan modül

FFORR: Belirtilen tampon alanından F-deseniyle bir gerçek sayı okuyan modül

FFORW: Belirtilen tampon alanına F-deseniyle bir gerçek sayı yazan modül

EFORR: Belirtilen tampon alanından E-deseniyle bir gerçek sayı okuyan modül

EFORW: Belirtilen tampon alanına E-deseniyle bir gerçek sayı yazan modül

Bu modüller arasında en önemli rolü, desen öğeleriyle yazılacak değişkenler arasındaki koordinasyonu sağlayan EXACFD altrutini oynar. Desen kodları taranırken görülen pasif kodlar (X,/,H) EXACFD içinde işlenir. Eğer aktif kod bulunamadan desen sonu erişilirse, desen geri sarılır ve bu bilgi işlem kodu yorumlayıcısına iletilir.

6.4.4'üncü bölümde tanıtılacak hizmet altrutinleriyle işlem kodu yorumlayıcısı arasındaki iletişim şu ortak işlem alanlarıyla sağlanır

ISTRNG 80 hücrelik giriş/çıkış tamponudur

ICHCNT Tampon pozisyon endeksidir

IREW Desen tarayıcı (EXACFD) desen sonuna aktif öğe bulamadan erişmişse bu değişken 1 değerini alır

IFDP Desen tarayıcısının bulduğu aktif öğe kodudur (E,F veya I)

IWIDTH Bulunan desen öğesinde kullanılacak alan genişliğidir.

IDCP Bulunan öğenin (varsa) ondalık alan genişliğidir

IFSTAK Desen tarayıcısının kullandığı iki sütunluk desen tekrar torbasıdır. İkinci sütunda en son bulunan desen öğesinin desen tablosundaki (IFRMT) endeksi, birinci sütunda ise bu öğenin kaç kez tekrarlandığı tutulur.

ITFS IFSTAK torbasının en üst elemanının endeksini tutar.

ICODEX'den alınan giriş/çıkış işlem kodlarının, yukarıda özetlenen hizmet altrutinleri ve çalışma alanları kullanılarak, yapılan yorumları şöyledir

ODESEN: Okuma komut zincirinin ilk işlem komutudur ve kullanılacak olan desenin endeksini tanıtır. Tekrar torbasına (IFSTAK) desenin başlangıç parantezinin endeksi yerleştirilir ve ilk veri çizgisi tampona okunur. Desendeki ilk aktif desen öğesini bulmak için EXACFD çağırılır.

OKUG, OKUT: Tampondan bir gerçek sayı veya bir tamsayı okumak için kullanılır Bu komut işlenmeden önce ya elde aktif bir öğe vardır (IFDP) veya desen geri sarılmıştır (IREW=1). Eğer desen geri sarılmışsa şu işlemler yapılır

1. EXACFD aracılığıyla yeni bir aktif öğe aranır
2. Yeniden geri sarma olmuşsa, desende aktif öğelerin bulunmadığı , sonsuz bir pasif öğe döngüsü vardır. İşlem hatası verilir.
3. Aktif öğe bulunmuşsa, daha önceden geri sarma olayı gerçekleştiğinden, yeni bir veri çizgisi tampona okunur.

Sonuçta, IFDP içinde kullanılacak desen öğesinin kodu olur. Bulunan öğenin tipi okuma komutunun tipiyle kıyaslanır; OKUG komutunda desen öğesi "E" veya "F" olabilir; OKUT komutunda kullanılan öğe "I" olmalıdır. Desen/değişken uyuşmazlığı varsa hata mesajı verilir. IFORR, FFORR veya EFORR altrutini çağırılarak tampondan bir değer okutulur. Bu değer OKUT veya OKUG komutunda be-

lirtilen VALTB'deki işlenen hücresine yerleştirilir. Yapılanen son işlem, bir sonraki okuma komutuna hazırlık olarak, kullanılacak desen öğesinin EXACFD ile saptanmasıdır.

YDESEN: Yazma komutu zincirinin ilk işlem komutudur ve kullanılacak olan deseni tanıtır. Tekrar torbasına desenin ilk parantezinin endeksi yerleştirilir, ve desendeki ilk aktif öğeyi bulmak için EXACFD çağırılır. Eğer aktif eleman bulunmadan geri sarma olursa, çıkış tamponu yazılır.

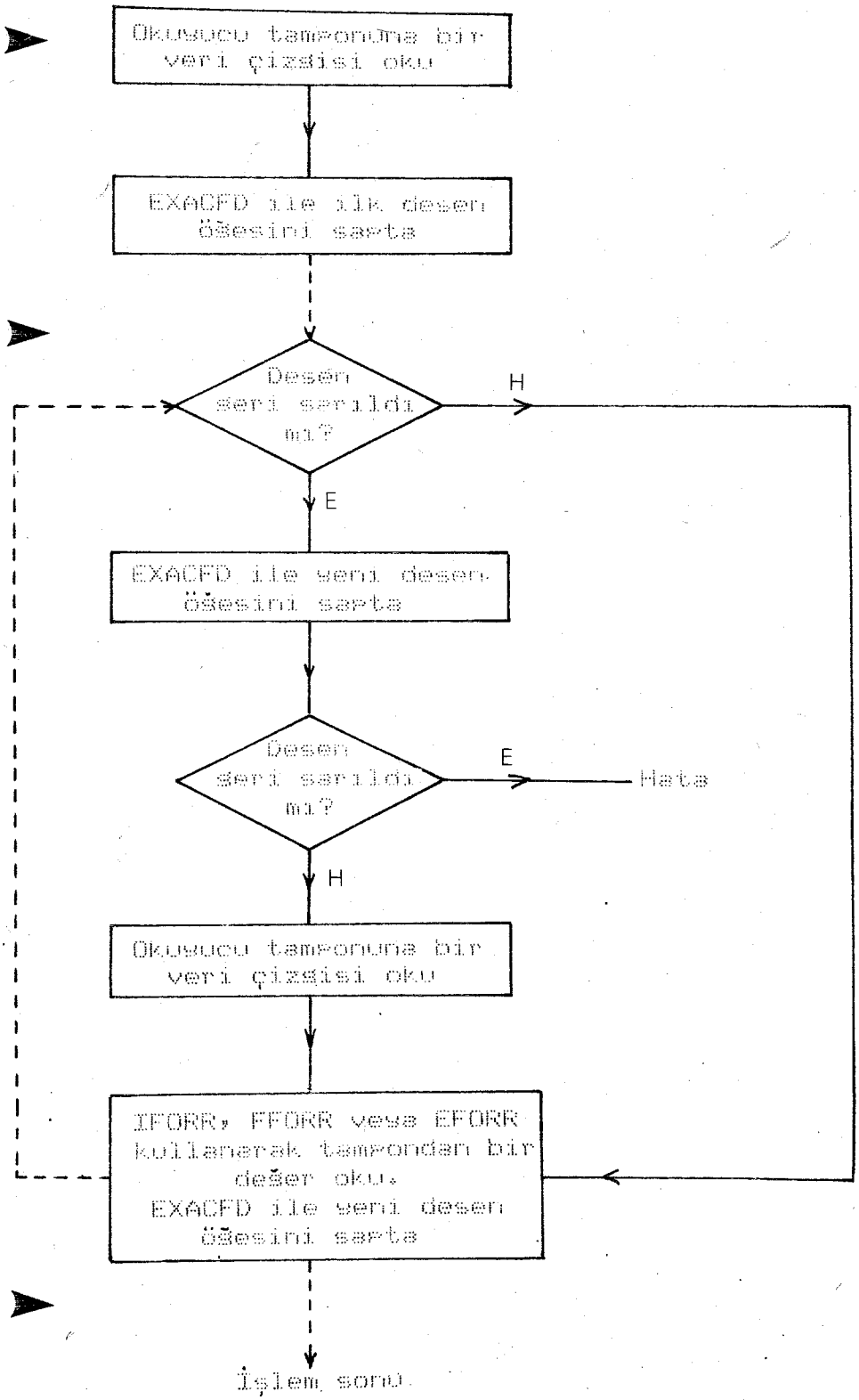
YAZG, YAZT: Yazıcı tamponuna bir gerçek sayı veya bir tamsayı yerleştirmek için kullanılır. Bu komut işlenmeden önce ya aktif bir desen öğesi bulunmuştur, ya da desen geri sarılmıştır. Geri sarma olmuşsa yapılan işlemler

1. EXACFD çalıştırılarak yeni bir aktif öğe aranır
2. Desen yeniden geri sarılmışsa desende sonsuz bir pasif öğe döngüsü vardır

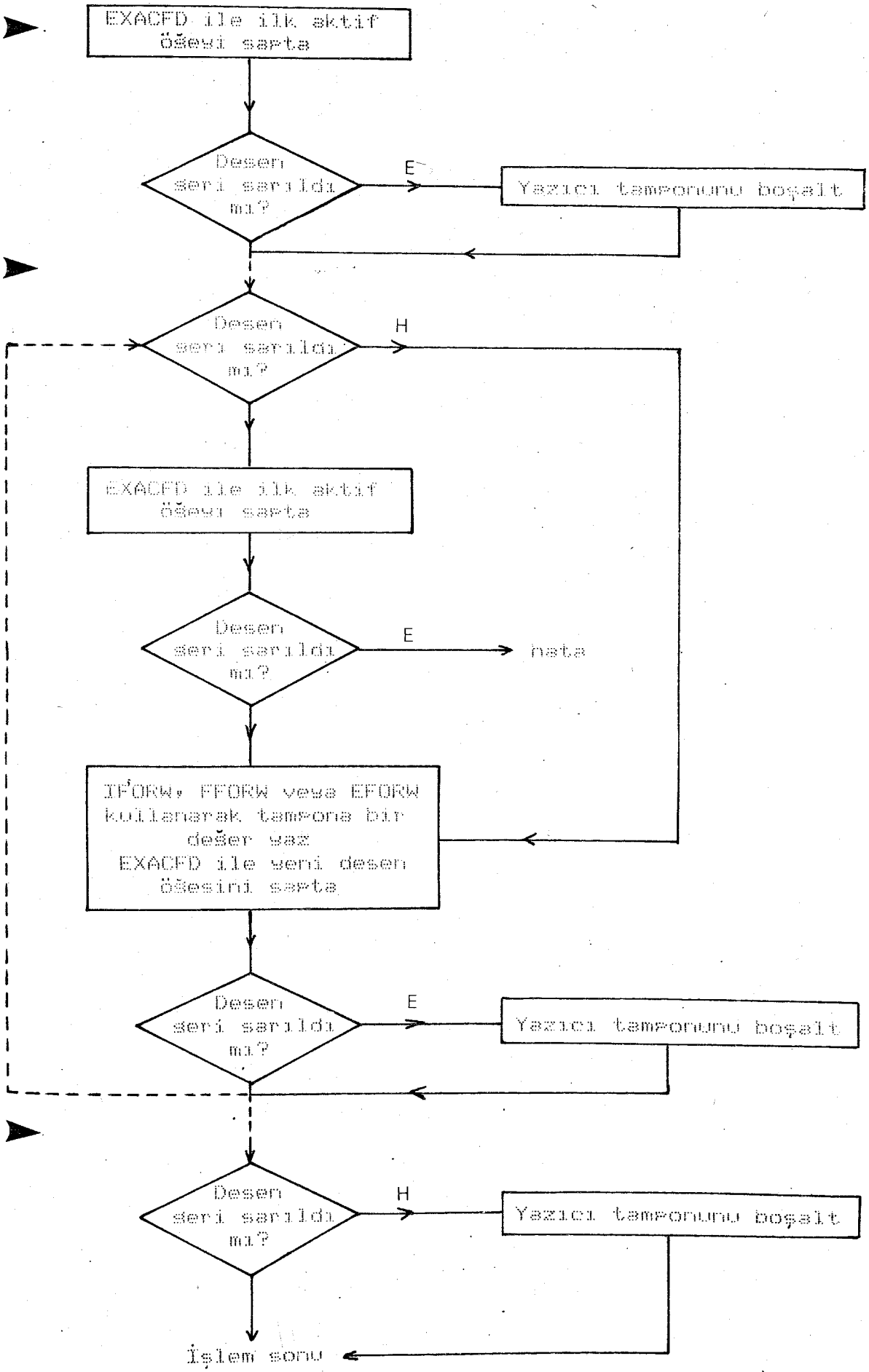
Bu işlemler sonunda IFDP değişkeninde kullanım sırası gelen aktif bir öğe kodu vardır. Bulunan öğenin tipi komut tipiyle kıyaslanır; YAZG komutunda "I" öğesi, YAZT komutunda ise "E" veya "F" kodu kullanılıyorsa hata mesajı verilir. Desen koduna göre IFORW, FFORW veya EFORW altrutini çalıştırılarak değişken değeri yazıcı tamponuna işlenir. Bir sonraki yazma komutuna hazırlık olarak EXACFD çalıştırılarak kullanım sırası gelen aktif öğe saptanır; eğer bu altrutin deseni geri sararsa, yazıcı tamponu boşaltılır.

DURY: Yazma komut zincirinin son komutudur. Eğer desen geri sarılmamışsa (yani yazma komutları desendeki tüm aktif öğeleri kullanmamışsa) yazıcı tamponu boşaltılır.

Okuma ve yazma komutlarının işlem zamanındaki koordinasyonu Şekil 6.34 ve 6.35'deki çizelgelerde özetlenmektedir.



Şekil 6.34: ODESEN, OKUG/OKUT ve DUR komutlarının işlem zamanında koordinasyonu.



Şekil 6.25 ... PIPY komutlarının işlem ...

Örnek:

```
      WRITE(6,25)IJ,X1,JI
25    FORMAT(1X,'CEVAP',I5,1X,'VE',F9.2,'DIR')
```

FORTRAN komutları için çözümleyicinin ürettiği işlem komutları şöyle olur

i) YDESEN 25

ii) YAZT IJ

iii) YAZG X1

iv) YAZT JI

v) DURY

Bu işlem komutlarıyla çıkış koordinasyonu aşağıda açıklanmaktadır

i) EXACFD kullanılır. Pasif X ve H öğeleri doğrudan çıkış tamponuna işlenir. Elde edilen aktif öğe I5'dir.

ii) Geri sarma gerçekleşmediğinden IJ değeri I5 deseniyle çıkış tamponuna aktarılır; bu işlem için IFORW kullanılır. Yeni öğeyi saptamak için EXACFD çağırılır. Bu altrutin pasif öğe 1X ile 'VE'yi çıkış tamponuna işleyip aktif öğeyi F9.2 olarak saptar.

iii) X1 değeri FFORW ve F9.2 deseni kullanılarak çıkış tamponuna işlenir. Yeni öğeyi saptamak için EXACFD kullanılır. Bu atrutin, pasif öğe 'DIR'ı tampona aktardıktan sonra, desen sonu geldiğinden, deseni geri sarar. Elde henüz aktif bir desen öğesi yoktur. Desen geri sarıldığından tampon boşaltılır.

iv) Desen geri sarıldığı için yeni aktif öğeyi saptamak üzere EXACFD çalıştırılır. Bu modül, pasif 1X ve 'CEVAP' öğelerini yeni çıkış tamponuna aktarır ve aktif öğe olarak I5'i bulur. JI değişkeninin değeri IFORW aracılığıyla tampona işlenir. Yeni aktif öğeyi saptamak için EXACFD çağırıldığında, pasif 1X ile 'VE' tampona aktarılır; bulunan aktif öğe F9.2'dir.

v) Geri sarma gerçekleşmediğinden tamponda bekletilen bilgi yazılır.

Örnek:

```
      WRITE(6,30)K
30    FORMAT(6H CEVAP)
```

komutları için üretilen komut zinciri

i) YDESEN 30

ii) YAZT K

iii) DURY

olur. Bu komutlar yorumlanırken

i) İlk aktif öğeyi bulmak için EXACFD çağırılır; pasif H desenini çıkış tamponuna alan EXACFD aktif öğe bulamadan deseni geri sarar . Geri sarma gerçekleştiği için tampon boşaltılır.

ii) Elde kullanılabilecek aktif öğe olmadığından EXACFD yeniden çağırılır pasif H yine tampona alındıktan sonra desen geri sarılır. Sonsuz pasif öğe döngüsü farkedilir; hata mesajı verilerek komut yorumu durdurulur.

Örnek:

```
      READ(5,10)IJ,X1
10    FORMAT(I5)
```

komutlarında işlenecek komut zinciri şöyledir.

i) ODESEN 10

ii) OKUT IJ

iii) OKUG X1

iv) DUR

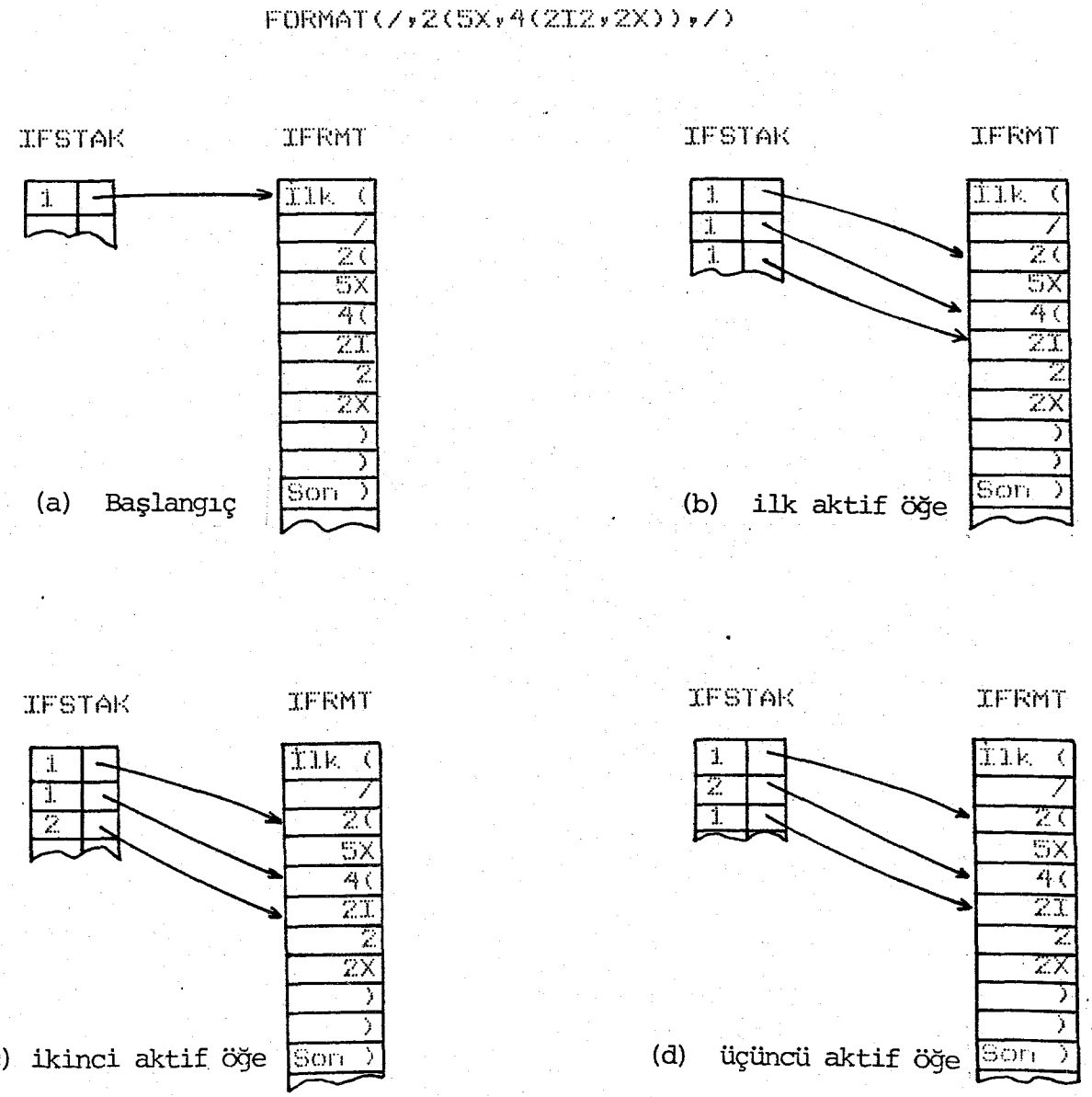
i) İlk veri çizgisi tampona okunur. EXACFD ile ilk aktif öğenin I5 olduğu saptanır

ii) Desen geri sarılmadığından aktif öğe I5 ile tampondan, IFORR kullanılarak IJ değişkenine bir değer okunur. EXACFD yeni bir öğe bulmak için çalıştırılır; öğe bulunamadığından desen geri sarılır.

iii) Elde kullanılabilir bir öğe olmadığından EXACFD yeniden çalıştırılır ve ikinci veri çizgisi okunur. Bulunan öğe tamsayı (I5) ve yapılacak işlem gerçek (OKUG) olduğundan desen/değişken uyuşmazlığı vardır. Hata mesajı verilir.

6.4.4 Desen kodlarının yorumu

Giriş/çıkış işlemleri esnasında, desen kodlarının yorumlanmasında en önemli rolü EXACFD modülüyle desen tekrar torbası olan IFSTAK oynar. Tekrar torbasının nasıl kullanıldığı Şekil 6.36'da bir örnekle gösterilmektedir; bu örnekteki (b), (c) ve (d) şekilleri peşpeşe yapılan üç EXACFD çağırımından sonra tekrar torbasının içinde tutulan bilgiyi gösterir. Tekrarlı gruplarla



Şekil 6.36: İşlem zamanında peş peşe yapılan üç EXACFD çağırımının tekrar torbasındaki (IFSTAK) etkisi.

tekrarlı öğelerin, sırayla ve teker teker kullanma zorunluğu olduğundan, ve iç içe gruplar olabileceğinden, torbalama en uygun yaklaşımdır.

Tekrar torbasındaki bilgiyi kullanıp yenileyerek aktif desen öğesi bulan EXACFD modülünün genel algoritması şöyledir

1. Torbadaki en üst elemana bakılarak en son kullanılan desen öğesinin endeksiyle bu öğenin kaç kez kullanıldığı bulunur.
2. Eğer desen öğesi IFRMT'de belirtilen tekrar sayısı kadar tekrarlanmamışsa ve öğe grup tekrarı belirlemiyorsa, yeni aktif öğe bulunmuştur. IFSTAK'daki en üst elemanın tekrar sayısı bir arttırılır ve aktif öğe, çağıran rutine iletilir.
3. Eğer desen öğesi IFRMT'de belirtilen tekrar sayısı kadar tekrarlanmamışsa ve öğe grup tekrarı ise, 5'inci basamağa geçilir.
4. Eğer desen öğesi IFRMT'de belirtilen tekrar sayısı kadar tekrarlanmışsa, IFSTAK torbasından alınır. Eğer torbadan çıkarılan grup tekrarıysa, desen endeksi grubun son parantezine kadar iletilir.
5. IFRMT dizisindeki pozisyonu gösteren endeks öğe tipine göre arttırılarak bir sonraki öğeye geçilir. Bu işlemle

   I: Endeks 2 arttırılır

   E,F: Endeks 3 arttırılır

   H: Endeks tekrar sayısı kadar arttırılır

   diğer: Endeks 1 arttırılır

6. Yeni endeks pasif öğelerden "X","H; veya "/" gösteriyorsa

   X: Tampon endeksi tekrar sayısı kadar arttırılır

   H: Tekrar sayısının belirlediği karakterler IFRMT'den tampona taşınır ve tampon endeksi ilerletilir.

   /: Okuma komutlarında, yeni veri çizgisi okunup tampon endeksi geri alınır; yazma komutlarında, tampon boşaltılıp tampon endeksi geri alınır.

Aktif bir öğe bulmak için 5'inci basamağa dönülür.

7. Yeni endeks grup tekrarını gösteriyorsa bu bilgi tekrar torbasına işlenir ve aktif öğe bulmak için 5'inci basamağa dönülür.
8. Yeni endeks son ")" işaretini gösteriyorsa, aktif öğe bulunamamıştır. Desen ilk paranteze veya dengelenen ilk grup tekrar parantezine kadar geri sarılır. Yeni IFRMT pozisyonu tekrar torbasına işlendikten sonra, desenin geri sarıldığı bildirilerek EXACFD'den çıkılır.
9. Yeni endeks grup sonunu belirten ")" ise, grup başını bulmak üzere 1'inci basamağa dönülür.
10. 6, 7, 8 ve 9'uncu basamaklardaki koşullar karşılanmamışsa, bulunan desen öğesi "I", "F" veya "E"dir. Desen endeksiyle bu öğenin bir kez kullanıldığını belirten bilgi tekrar torbasına işlenir. Desen öğesinin alan genişliği ve (varsa) ondalık alan genişliği saptandıktan sonra EXACFD sona erer.

EXACFD modülünün sağladığı aktif öğelere ilişkin bilgiye dayanarak tampon işlemlerini gerçekleştiren modüller şunlardır

IFORR: Tampon pozisyon endeksiyle alan genişliğini kullanarak I-desenine göre tampondan bir değer okur. Belirtilen alan içerisine karakter koduyla tutulan verinin nümerik değerini hesaplar. Tampon endeksini ilerletir.

FFORR: Aktif F-öğesine ilişkin bilgiyi kullanarak tampondan bir gerçek sayı okuyarak nümerik değerini hesaplar.Alanda ondalık noktası yoksa, desende tanımlanan pozisyonda nokta olduğunu varsayar. Tampon endeksini alan sonuna kadar ilerletir.

EFORR: E-öğesine ilişkin bilgiyi kullanarak tampondan bir gerçek sayı okuyup değerini hesaplar. Mantis kısmında ondalık noktası yoksa desenin ima ettiği pozisyonda nokta olduğu varsayılır. Tampon endeksi alan sonuna getirilir.

IFORW: Tampon pozisyon endeksiyle alan genişliğini kullanarak tampona bir tamsayı yerleştirilir. Sayısal değer yerleştirilirken karakter koduna dönüştürülü

FFORW: Verilen değeri karakter koduna dönüştürerek F-desenine göre tampona yerleştirilir. Ondalık kısım, belirtilen alan genişliğine göre yuvarlanır.

EFORW: Verilen değeri karakter koduna dönüştürerek E-desenine göre tampona yerleştirilir. Değer karakter koduna dönüştürülürken normalize edilir ve belirtilen mantis uzunluğuna göre yuvarlanır.

EFORW ve FFORW modüllerinde tampona işlenen rakam dizgileri en büyük tamsayı değerini aşabileceğinden, en az 16-bitlik tamsayıları destekleyen bir bilgisayar varsayılarak, ondalık ve tamsayı kısımlar dörder rakamlık parçalar halinde işlenir. Örneğin "F20.6" ile "265432.365438" değeri yazılacaksa, FFORW rakamları tampona şu sırayla yerleştirilir

ondalık kısmın son dört rakamı "5438"
ondalık kısmın ilk iki rakamı "36"
ondalık nokta "."
tamsayı kısmın son dört rakamı "5432"
tamsayı kısmın ilk iki rakamı "26"

6.5 Çeviri ve işlem hataları

FCN-F'nin her modülü, hatalı bir durumla karşılaşınca, hata kod numarasını bir üst modüle iletir. Bir hata farkedilince tüm modüller yapmakta oldukları kontrolu üst modüle devreder. Bu şekilde, hata kodu FCN-F ana modülüne kadar iletilir.

Komut çözümleme hataları 1 ile 75 arası kodlarla tanımlanırken, işlem hataları 1 ile 25 arasındadır. Ana modül, hatanın geldiği mantıksak modülü (çeviri veya işlem) saptayabildiğinden, gerekli hata yorumlama modülünü çalıştırıp hata kodunu mesaja dönüştürür.

Komut çevirisi sırasında kullanıcı hataları yanısıra FCN-F'nin işlem alanlarında başgösteren taşmalar da denetlenir. Bu tür hatalara karşılık olan mesajlar Ek.B'de "*" ile işaretlenmiştir. Komut çözümü ve çevirisi sırasında kar-

şılaşılan tüm hatalarda, hatalı komut unsuru bir okla kullanıcıya gösterilir.

İşlem hatası eğer veri okunurken yapılmışsa, kullanıcının sağladığı çizgideki hatalı unsur okla gösterilerek etkilenen değişken bildirilir. Yazma esnasında görülen işlem hatalarında, kullanıcının hazırladığı tampon boşaltılarak hatalı unsur okla gösterilir ve o anda yazılmakta olan değişken bildirilir.

6.6 Sonuç

FCN-F programı FCN programının ARIT ve FORMAT opsiyonları ile karşılaştırıldığında şu avantajları belirgindir

1. FCN-F taşınabilen programdır
2. Daha kolay değiştirilebilen bir programdır
3. Hata mesajı sistemi kullanıcıya daha çok yardım sağlar

FCN-F'nin taşınabilirliği FORTRAN programlama diliyle, mümkün mertebe donanımdan bağımsız olarak, geliştirilmesinden kaynaklanır. Bir takım basit leksik istisnalar dışında, standart FORTRAN IV'ün kullanıldığı herhangi bir bilgisayara taşınabilir. Programın transfer edildiği kuruluş FCN-F'nin bazı özelliklerini kendi gereksinimlerine uygun olarak şöyle değiştirilebilir:

Donanım özellikleri

XMXINT - Donanımın kabul edebileceği en büyük tamsayı

XMPOW - Donanımın kabul edebileceği en büyük gerçek sayının ondalık tabana göre tamsayı logaritması

ALXMRL - Donanımın kabul edebileceği en büyük sayının doğal logaritması

ALALXM - ALXMRL'in doğal logaritması

XMRL10 - Donanımın kabul edebileceği en büyük sayı bölü on.

İşlem alanlarının genişlikleri

MAXID - Kullanıcıya tanınacak olan değişken tanıtıcı sayısı (değişken alan uzunluğu)

MAXCN - Kullanıcının her atama komutunda kullanabileceği azami değişme sayısı

MAXIFR- Kullanıcıya tanınacak olan FORMAT komut sayısı

MAXFRT- Kullanıcının bellekte tutabileceği azami desen öğe sayısı

MAXCD- Üretilen işlem kodunun azami uzunluğu

Diğer:

MDCHK - Aritmetik deyimlerde karma işlenen tiplerinin hata sayılıp sayılmayacağı.

BÖLÜM 7

SONUÇ

Bilgisayar Destekli Eğitimde şimdiye dek denenen FORTRAN eğitici sistemleri özdevimli kitap yaklaşımının çeşitli varyasyonları olmuştur.Bu tür BDE sistemleri kullanıcıya bilgi vererek onu sınarken öğrencinin kaydettiği gelişmeyi izleyerek ders güçlülüğünü kişiye göre ayarlamayı amaçlar; önemli bir dezavantajı dersle ilgili metin, soru ve cevapları katı bir şekilde yorumlamasıdır. Eğitici sistem FORTRAN dilini bilmediğinden kullanıcının sorulara verdiği cevapları sadece doğru veya yanlış olarak tanıyabilir- öğrencinin verdiği cevabın neresinde ve niye hata olduğunu söyleyemez. Bu tür sistemlerde BDE programı öğrenciyi yönetir ve denetler- kullanıcı kendine yönetilen soruları cevaplamak zorundadır. Programın FORTRAN "bilgisi" yoktur; sadece sorduğu soruların cevabını bilir.

FCN ve FCN-F sistemleri Bilgisayar Destekli FORTRAN eğitimine farklı bir yaklaşım getirmektedir. Diğer sistemlerle kıyaslandığında FCN denetleyen ve yönlendiren bir eğitici değildir ama içinde derleyici olduğundan FORTRAN bilgisi kusursuzdur.

Diğer FORTRAN öğretici sistemlerinde öğrenci pasif bir rol alıp kalıplaşmış soruları cevaplandırırken FCN'de öğrenmek istediği komutları denemek zorundadır. FCN tümüyle açık ve esnek bir sistem olduğundan öğretilen konunun öğrenciye göre ayarlanması sözkonusu değildir-kullanıcı kendi çapında komut hazırlayıp denemekte serbesttir.

Öğrencinin FORTRAN dilini okuyarak değil de kullanarak daha kolaylıkla öğreneceği tartışmasız kabul edilir. Ancak bu dili çeviren derleyiciler tek komutlu

programları kabul etmediğinden öğrenimin ilk safhalarında öğrenciye program yazıp deneme olanağı verilemez. FCN'deki alıştırıcı modüller öğrenciye bu olanağı sağlar. Sistemin etkileşimli (interactive) oluşu öğrencinin verdiği FORTRAN komutlarının hemen çözümlenip değerlendirilmesini sağlar. FCN'de derleyici olduğundan öğrenciye yaptığı hatalar ve hata nedeni kesin bir biçimde bildirilebilir.

FCN sistemi taşınabilir bir program olmadığından FCN-F geliştirilmiştir. CODEX-68 mini-bilgisayarında FORTRAN IV'e mümkün mertebe sadık kalarak geliştirilen FCN-F disket üzerinde başarıyla UNIVAC 1106 sistemine transfer edilmiştir. 1980-81 senesinin yaz döneminde etkileşimli kullanım gerçekleştirildiğinde Boğaziçi Üniversitesi Ön Lisans öğrencilerine FCN-F'yi kullanma olanağı sağlanacaktır.

Tüm BDE uygulamalarında gerekli olan ve genellikle ihmale uğrayan başarı değerlendirme araştırmaları henüz yapılmamıştır. FCN sisteminin 1978 yılından beri kullanılmasına karşın uygulanan tek değerlendirme yöntemi öğrenci ve öğretmen anketleridir. FCN-F programının başarı değerlendirmesi ancak geniş bir kullanıcı kitlesi (senede yaklaşık 100) bulunduğu takdirde gerçekleştirilebilir.

FCN yöntemi özdevimli kitap yöntemiyle kıyaslandığında avantajları yanısıra bir takım dezavantajları da belirgindir. Bunların en önemlisi sistem açıklığından doğan denetim ve yönlendirme kısırlığıdır. Önceden de belirtildiği gibi öğrenme insiyatifi tamamiyle öğrenciye bırakılır; FCN öğrenciyi yöneltmediğinden zayıf ve arzusuz olanlar için etkin olmayabilir; öğrencinin ne tür ve hangi amaçla komut yazdığını bilmediğinden başarısını denetleyemez. Bu dezavantaj kullanıcıyı yönlendiren bir kitapçıkla (Balman 78e) etkisiz kılınabilir.
Öğrenciye ödev olarak verilen alıştırmalarla FCN kullanırken yapacağı çalışmalar denetlenebilir.

FCN yöntemi atama ve giriş/çıkış komutlarında etkin bir eğitici olduğu halde diğer FORTRAN komutlarında normal etkileşimli derleyiciden (üstün hata mesajları dışında) pek farklı değildir. Özellikle dizi ve DO-döngüsü kullanımında döngü kapanmadan işlem yapılamayacağından öğrenciye anında yanıt vermek olanak dışıdır. Kullanıcının 4-5 komut kullanarak bir DO-döngüsü hazırladığını varsayarsak işlem sırasında hata olduğunda tüm komutlar tekrardan girilmelidir. Bu sebepten ötürü DO-döngü alıştırıcısı FCN-F bünyesinde sağlandığında diğer alıştırıcıların verdiği süratli dene-gör-düzelt hizmetiyle aynı etkinlikte olamaz. FCN-F bünyesinde etkin olamayacak alıştırıcıları şöyle sıralayabiliriz

i) Diziler
ii) Do-döngüleri
iii) Mantıksal deyimler ve IF-komutu
iv) Altrutinler ve ortak alanlar

FCN ve özdevimli kitap uygulamalarının en iyi özelliklerini biraraya alırsak aranan ideal FORTRAN eğiticisinin şu nitelikleri içerdiğini görürüz

i) Yönlendirme ve yönlendirmeyi gerçekleştirebilmek için başarı denetimi
ii) Kullanıcının kaydettiği ilerlemenin değerlendirilmesi
iii) Sistemin FORTRAN dilini ayrıntılarıyla bilmesi
iv) Soru/yanıt sistemi yanısıra serbest komut giriş olanağı
v) Anında komut çözümleme ve değerlendirme olanağı
vi) Dene-gör-düzelt hizmetinin yanısıra katı bilginin verilmesi
vii) Öğrenimin her safhasında sistemin kullanıcıya yararlı bilgi verebilmesi.

Yukarıdaki nitelikleri taşıyan bir sistem ancak FCN yöntemiyle özdevimli kitap yönteminin birleştirilmesiyle sağlanabilir. Bu karma sistemin etüdü FCN-F'nin geliştirildiği CODEX-68 bilgisayarında yapılmaktadır.

EK.-A

FCN-F PROGRAMI

```
C     +-------------------------------------------------------------+
C     |                                                             |
C     |      FCN-F FORTRAN EGITICI PROGRAMI                         |
C     |                                                             |
C     |      T. BALMAN - BOGAZICI UNIVERSITESI                      |
C     |                                                             |
C     |      BITIS TARIHI - SUBAT 1981                              |
C     |                                                             |
C     |      SON DEGISIKLIK TARIHI - 4 MART 1981                    |
C     |                                                             |
C     +-------------------------------------------------------------+
C
      COMMON IDTAB(20,6),NOIDTB,MAXID,NEWIDT
      COMMON IFRTB(10,2),MAXIFR,NEWIFR,NGIVEN,NEXPCT,NOWIFR
      COMMON IFRMT(200),MAXFRT,NEWFRT,NOWFRT
      COMMON VALTB(40),MAXCN,MAXTP,MACC,NEWTP,NEWCN
      COMMON ICODEX(60),NEWCD,MAXCD,XMXINT,MXPOW
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      COMMON IC8(80)
      DATA IS/' '/,IU/'A'/
      MDCHK=0
   10 WRITE(101,100)
      READ(100,110)II
      IF(II.LE.0.OR.II.GT.5)GO TO 10
      GO TO(11,12,13,14,15),II
   11 ICTYP=1
      WRITE(101,111)
      GO TO 16
   12 ICTYP=0
      WRITE(101,112)
      GO TO 16
   13 STOP
   14 MDCHK=0
      WRITE(101,114)
      GO TO 10
   15 MDCHK=1
      WRITE(101,115)
      GO TO 10
   16 NVALUE=0
      NEWIDT=0
      XMXINT=32767.0
      MXPOW=74
      MAXID=20
      MAXCN=30
      MAXTP=39
```

```
      MAXCD=60
      MAXFRT=200
      NEWFRT=1
      MAXIFR=10
      NEWIFR=0
      NEXPCT=0
   20 NEWCN=MAXID+1
      NOIDTB=NEWIDT
      NOWIFR=NEWIFR
      NEWCD=1
      CALL GSM1
      IF(IEROR.NE.0)GO TO 30
      GO TO(40,20,10),NCLASS
   30 ICHCNT=ICHCNT-1
      WRITE(101,150)(IS,I=1,ICHCNT),IU
      CALL ERRCOM
      GO TO 20
   40 CALL ERREXC
      IF(IEROR.NE.0)GO TO 50
      NEWIDT=NOIDTB
      NEWIFR=NOWIFR
      GO TO 20
   50 IF(IEROR.GT.17)ICHCNT=ICHCNT+2
      IF(IEROR.GT.13)WRITE(101,170)(IS,J=1,ICHCNT),IU
      WRITE(101,140)
      IF(IEROR.GT.13)WRITE(101,160)(IDTEMP(J),J=1,6)
      CALL LOADEX('EE',1)
      GO TO 20
  100 FORMAT(//,' FCN-F ALISTIRICI SISTEMI',//,
     &       ' SECENEKLER:',//,
     &       ' 1: ATAMA KOMUTU ALISTIRICISI',/,
     &       ' 2: DESENLI READ/WRITE KOMUTU ALISTIRICISI',/,
     &       ' 3: BITIS',/,' 1,2, VEYA 3')
  110 FORMAT(I1)
  111 FORMAT(/,' FCN-F ATAMA KOMUTU ALISTIRICISI',/)
  112 FORMAT(/,' FCN-F GIRIS/CIKIS ALISTIRICISI',/)
  114 FORMAT(' KARMA DEYIMLER KULLANILABILIR',/)
  115 FORMAT(' KARMA DEYIMLER KULLANILAMAZ',/)
  140 FORMAT(' ISLEM HATASI')
  150 FORMAT(3X,72A1)
  160 FORMAT(' "',6A1,'" ISLENIRKEN')
  170 FORMAT(1X,80A1)
      END
C
CCC  KOMUT TURUNU SAPTAYAN GSM MODELI
C
      SUBROUTINE GSM1
      COMMON IDTAB(20,6),NOIDTB,MAXID,NEWIDT
      COMMON IFRTB(10,2),MAXIFR,NEWIFR,NGIVEN,NEXPCT,NOWIFR
      COMMON IFRMT(200),MAXFRT,NEWFRT,NOWFRT
      COMMON VALTB(40),MAXCN,MAXTP,MACC,NEWTP,NEWCN
      COMMON ICODEX(60),NEWCD,MAXCD,XMXINT,MXPOW
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      DIMENSION ICHH(37),ICHL(37),NESUC(37),NACTS(37),NEFAL(37)
      DATA ICHL/'F','O','R','M','A','T','(','R','E','A','D','(','W',
     &'R','I','T','E','(','I','N','I','S','H','S','E','T','I','A','A'
     &,'0','=','A','0','=','A','0','='/
      DATA ICHH/'F','O','R','M','A','T','(','R','E','A','D','(','W',
     &'R','I','T','E','(','I','N','I','S','H','S','E','T','N','Z','Z'
     &,'9','=','Z','9','=','Z','9','='/
```

```
      DATA NACTS/1,3,3,3,3,3,5,1,3,3,3,6,1,3,3,3,3,7,3,3,3,3,8,3,3
     &    2,1,3,3,4,3,3,4,3,3,4/
      DATA NEFAL/8,19,29,29,29,29,-101,13,29,24,29,32,27,29,29,29,2
     &    35,29,29,29,29,29,29,29,29,28,-104,30,31,-105,33,34,-102,36,3
     &    -103/
    1 ISSIX=0
      IEROR=0
      ISTT=1
      IF(NEXPCT.NE.0)WRITE(101,600)NEXPCT
      CALL READCH
      IF(IEROR.NE.0)RETURN
      IF(NEXPCT.NE.0.AND.NEXPCT.NE.NGIVEN)GO TO 200
    5 CALL GETCH
   10 IF(ICHH(ISTT).GE.ICH.AND.ICHL(ISTT).LE.ICH)GO TO 15
      IFAIL=1
      NSTATE=NEFAL(ISTT)
      IACT=0
      GO TO 20
   15 NSTATE=NESUC(ISTT)
      IACT=NACTS(ISTT)
      IFAIL=0
   20 IF(IACT.NE.0)GO TO 30
   25 IF(NSTATE.LT.0)GO TO 1000
      ISTT=NSTATE
      IF(IFAIL.EQ.1)GO TO 10
      GO TO 5
   30 GO TO (110,120,130,140,150,160,170,180,190), IACT
 1000 NSTATE=-NSTATE
 1100 IEROR=NSTATE-100
      RETURN
  110 NCLASS=20
      GO TO 130
  120 NCLASS=21
  130 CALL PACK
      GO TO 25
  140 IF(NEXPCT.NE.0)GO TO 200
      CALL FNCTTB(JF)
      IF(JF.EQ.1)GO TO 145
      CALL IDTB(1)
      CALL AREXAN
      NCLASS=1
      RETURN
  145 IEROR=6
      RETURN
  150 IF(NGIVEN.NE.0)GO TO 155
      IEROR=28
      ICHCNT=6
      RETURN
  155 IF(NEXPCT.NE.0.AND.NEXPCT.NE.NGIVEN)GO TO 200
      IF(ICTYP.EQ.1)GO TO 250
      CALL FRMTS
      IF(IEROR.EQ.0)GO TO 157
      IEROR=IEROR+29
      RETURN
  157 NCLASS=1
      IF(NEXPCT.NE.NGIVEN)NCLASS=2
      NEXPCT=0
      NEWIFR=NOWIFR
      RETURN
  160 IUS=92
      ISP=0
      NCLASS=73
```

```
  110 IF(NCLASS.NE.3)GO TO 140
      CALL TRNOPS
      IF(IEROR.NE.0)RETURN
      IF(NTOS.NE.1.AND.NTIS.NE.2)GO TO 120
      CALL STRLHS(LHSC,LHSV)
      RETURN
  120 IF(IBRC.EQ.0)GO TO 130
      IEROR=70
      RETURN
  130 IEROR=71
      RETURN
  140 IF(NCLASS.LT.6.OR.NCLASS.GT.10)GO TO 150
      CALL TRNOPS
      IF(IEROR.NE.0)RETURN
      CALL STKOP
      GO TO 100
  150 IF(NCLASS.LT.20)GO TO 160
      CALL STKID
      GO TO 100
  160 IF(NVALUE.NE.-1)GO TO 170
      IBRC=IBRC+1
      CALL STKOP
      GO TO 100
  170 IBRC=IBRC-1
      CALL TRNOPS
      IF(IEROR.NE.0)RETURN
      IF(IPSTAK(NTOS,2).EQ.-1)GO TO 190
      IEROR=72
      RETURN
  190 NTOS=NTOS-1
      IF(IPSTAK(NTOS,1).NE.10)GO TO 100
      CALL GENCOD
      GO TO 100
      END
C
C     ARITMETIK KOD URETIM SURUCUSU
C
      SUBROUTINE TRNOPS
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON IC7(89)
      COMMON IDSTAK(15,2),IDSM,NTIS,NTISN
      COMMON IPSTAK(15,3),IPSM,NTOS
    1 IF(NTOS.EQ.1)RETURN
      IF(IPSTAK(NTOS,2).EQ.-1)RETURN
      IF(IPSTAK(NTOS,1).LT.NCLASS)RETURN
      CALL GENCOD
      IF(IEROR.EQ.0)GO TO 1
      RETURN
      END
C
C     ARITMETIK KOD URETICISI
C
      SUBROUTINE GENCOD
      COMMON IC1T3(351)
      COMMON VALTB(40),MAXCN,MAXTP,MACC,NEWTP,NEWCN
      COMMON IC5(63),RC5
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON IC7(89)
      COMMON IDSTAK(15,2),IDSM,NTIS,NTISN
      COMMON IPSTAK(15,3),IPSM,NTOS
```

```
      IV1=IDSTAK(NTISN,2)
      ITP1=IDSTAK(NTISN,1)
      IV2=IDSTAK(NTIS,2)
      ITP2=IDSTAK(NTIS,1)
      IOPT=IPSTAK(NTOS,1)
      IOPC=IPSTAK(NTOS,2)
      IPOS=IPSTAK(NTOS,3)
      IRT=ITP2
      IF(IOPT.LT.9)GO TO 30
      IF(IV2.EQ.-1)GO TO 20
      IF (IV1.NE.-1)GO TO 15
      I=NEWTP
      NEWTP=NEWTP+1
      CALL GENC(2,I,ITP1)
      IDSTAK(NTISN,2)=I
   15 CALL GENC(1,IV2,ITP2)
   20 IF(IOPT.EQ.9)GO TO 25
      IRT=20
      IF(IOPC.LE.47)GO TO 25
      IF(IOPC.NE.48)IRT=21
      IF(IOPC.EQ.48.AND.ITP2.EQ.20)IEROR=73
      IF(IOPC.EQ.58.AND.ITP2.EQ.21)IEROR=74
   25 CALL GENC(IOPC,0,0)
      GO TO 890
   30 IF(ITP1.NE.ITP2)IRT=20
      IF(IV1.EQ.-1.OR.IV2.EQ.-1)GO TO 50
      CALL GENC(1,IV1,ITP1)
      CALL GENC(IOPC,IV2,IRT)
      GO TO 880
   50 IF(IV1.NE.-1)GO TO 70
   55 CALL GENC(IOPC,IV2,IRT)
      GO TO 880
   70 IV2=IV1
      IF(IOPC.EQ.5.OR.IOPC.EQ.7)GO TO 55
      CALL GENC(2,NEWTP,ITP2)
      CALL GENC(1,IV1,ITP1)
      CALL GENC(IOPC,NEWTP,IRT)
  880 NTIS=NTISN
      NTISN=NTIS-1
      IF(MDCHK.EQ.1)CALL MODCHK(IOPC,ITP1,ITP2,IPOS)
  890 NTOS=NTOS-1
      IDSTAK(NTIS,1)=IRT
      IDSTAK(NTIS,2)=-1
      RETURN
      END
C
C     URETILEN KODLARIN ICODEX DIZISINE YAZILMASI
C
      SUBROUTINE GENC(IPCODE,IPRND,IPTYP)
      COMMON IC1T4(356),RC1T4(40)
      COMMON ICODEX(60),NEWCD,MAXCD,XMXINT,MXPOW
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      IF(NEWCD.LT.MAXCD)GO TO 10
      IEROR=75
      RETURN
   10 IOFF=0
      IF(IPTYP.EQ.21)IOFF=10
      ICODEX(NEWCD)=IOFF+IPCODE
      NEWCD=NEWCD+1
      IF(IPRND.EQ.0)RETURN
      ICODEX(NEWCD)=IPRND
      NEWCD=NEWCD+1
```

```
C
C     DEYIM COZUMUNDE ISLENENLERIN TORBALANMASI
C
      SUBROUTINE STKID
      COMMON IC1T3(351)
      COMMON VALTB(40),MAXCN,MAXTP,MACC,NEWTP,NEWCN
      COMMON IC5(63),RC5
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON IC7(89)
      COMMON IDSTAK(15,2),IDSM,NTIS,NTISN
      IF(IDSTAK(NTISN,2).NE.-1)GO TO 10
      I=NEWTP
      NEWTP=NEWTP+1
      IRT=IDSTAK(NTISN,1)
      CALL GENC(2,I,IRT)
      IDSTAK(NTISN,2)=I
   10 NTISN=NTIS
      NTIS=NTIS+1
      IF(NTIS.GT.IDSM)GO TO 20
      IDSTAK(NTIS,1)=NCLASS
      IDSTAK(NTIS,2)=NVALUE
      RETURN
   20 IEROR=77
      RETURN
      END
C
C     DEYIM COZUMUNDE ISLEMLERIN TORBALANMASI
C
      SUBROUTINE STKOP
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      COMMON IDSTAK(15,2),IDSM,NTIS,NTISN
      COMMON IPSTAK(15,3),IPSM,NTOS
      NTOS=NTOS+1
      IF(NTOS.GT.IPSM)GO TO 20
      IPSTAK(NTOS,1)=NCLASS
      IPSTAK(NTOS,2)=NVALUE
      IPSTAK(NTOS,3)=ICHCNT
      RETURN
   20 IEROR=78
      RETURN
      END
C
C     DEYIM SONUNDA ATAMA ISLEMINI YAPACAK KODUN URETILMESI
C
      SUBROUTINE STRLHS(LHSC,LHSV)
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON IC7(89)
      COMMON IDSTAK(15,2),IDSM,NTIS,NTISN
      IV=IDSTAK(NTIS,2)
      IT=IDSTAK(NTIS,1)
      IF(IV.EQ.-1)GO TO 20
      CALL GENC(1,IV,IT)
   20 CALL GENC(2,LHSV,LHSC)
      IF(ICTYP.EQ.1)CALL GENC(71,LHSV,LHSC)
      IF(ICTYP.NE.1)CALL GENC(0,0,0)
      RETURN
      END
```

```
C
C    ISLENEN TIPLERININ DENETIMI
C
      SUBROUTINE MODCHK(IP,ITP1,ITP2,IPOS)
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      IF(IP.EQ.4)GO TO 10
      IF(ITP1.NE.ITP2)IEROR=76
      GO TO 15
   10 IF(ITP1.GT.ITP2)IEROR=76
   15 IF(IEROR.NE.0)ICHCNT=IPOS
      RETURN
      END
C
C    ARITMETIK DEYIM OGELERINI SAPTAYAN GSM MODELI
C
      SUBROUTINE GSM2(IWAS)
      COMMON IDTAB(20,6),NOIDTB,MAXID,NEWIDT
      COMMON IC2T3(228)
      COMMON VALTB(40),MAXCN,MAXTP,MACC,NEWTP,NEWCN
      COMMON ICODEX(60),NEWCD,MAXCD,XMXINT,MXPOW
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      DIMENSION ICHH(23),ICHL(23),NSUC(23),NACTS(23),NFAL(23)
      DIMENSION NACTF(23)
      DATA ICHL/'I','A','A','0','0','0',',','0',',','0','E','+','-',
     &  '0','0','+','-','*','*','/','(',')','@'/
      DATA ICHH/'N','Z','Z','9','9','9','.','9','.','9','E','+','-',
     &  '9','9','+','-','*','*','/','(',')','@'/
      DATA NSUC/3,3,3,3,6,6,8,10,10,10,12,14,14,15,15,1,-8,19,-9,-11,
     &  -12,-13,-14/
      DATA NACTS/2,1,3,3,5,5,0,6,0,6,0,0,8,9,9,19,11,0,12,14,15,16,17
     &  /
      DATA NFAL/2,5,4,-4,7,9,16,-108,-6,11,-7,13,14,-109,-7,17,18,20,
     &  -10,21,22,23,-110/
      DATA NACTF/0,0,0,4,0,0,0,0,7,0,10,0,0,0,10,0,0,0,13,0,0,0,0/
      DATA IEQ/'='/
    1 NADP=0
      NEDP=0
      ISSIX=0
      ISTT=1
      VAL=0
      IEXPV=0
      IEXSGN=1
    5 CALL GETCH
   10 IF(ICHH(ISTT).GE.ICH.AND.ICHL(ISTT).LE.ICH)GO TO 15
      IFAIL=1
      NSTATE=NFAL(ISTT)
      IACT=NACTF(ISTT)
      GO TO 20
   15 NSTATE=NSUC(ISTT)
      IACT=NACTS(ISTT)
      IFAIL=0
   20 IF(IACT.NE.0)GO TO 30
   25 IF(NSTATE.LT.0)GO TO 1000
      ISTT=NSTATE
      IF(IFAIL.EQ.1)GO TO 10
      GO TO 5
   30 IF(IACT-9)32,190,33
   32 GO TO (110,120,130,140,150,160,170,180),IACT
   33 JACT=IACT-9
      GO TO (200,210,220,230,240,250,260,270,280),JACT
```

```
1000 NSTATE=-NSTATE
     IF(NSTATE.GT.100)GO TO 1100
     NCLASS=NSTATE
     IF(IFAIL.EQ.1)ICHCNT=ICHCNT-1
     RETURN
1100 IEROR=NSTATE-100
     IF(ICH.EQ.IEQ)IEROR=16
     RETURN
 110 NCLASS=20
     GO TO 130
 120 NCLASS=21
 130 CALL PACK
     IF(IEROR.NE.0)RETURN
     GO TO 25
 140 CALL FNCTTB(JF)
     IF(JF.NE.1)CALL IDTB(0)
     GO TO 300
 150 NBDP=NBDP+1
     GO TO 165
 160 NADP=NADP+1
 165 VAL=VAL*10.0+IDIG(ICH)
     GO TO 25
 170 NCLASS=21
     IF(VAL.GT.XMXINT)IEROR=18
     GO TO 205
 180 IEXSGN=-1
     GO TO 25
 190 IEXPV=IEXPV*10+IDIG(ICH)
     GO TO 25
 200 NCLASS=20
     IPOW=NBDP+IEXPV*IEXSGN
     IF(IPOW.GT.MXPOW)IEROR=19
     IPOW=IEXSGN*IEXPV-NADP
     IF(IEROR.NE.0)GO TO 300
     VAL=VAL*10.0**IPOW
 205 VALTB(NEWCN)=VAL
     NVALUE=NEWCN
     NEWCN=NEWCN+1
     IF(NEWCN.GT.MAXCN)IEROR=69
     GO TO 300
 210 NVALUE=6
     NCLASS=6
     GO TO 300
 220 NCLASS=8
     NVALUE=4
     GO TO 300
 230 NCLASS=7
     NVALUE=7
     GO TO 300
 240 NCLASS=7
     NVALUE=8
     GO TO 300
 250 NCLASS=5
     NVALUE=-1
     GO TO 300
 260 NCLASS=5
     NVALUE=-2
     GO TO 300
 270 NCLASS=3
     NVALUE=0
     GO TO 300
 280 NCLASS=4
```

```
1000 NSTATE=-NSTATE
     IF(NSTATE.GT.100)GO TO 1100
     NCLASS=NSTATE
     IF(IFAIL.EQ.1)ICHCNT=ICHCNT-1
     RETURN
1100 IEROR=NSTATE-100
     IF(ICH.EQ.IEQ)IEROR=16
     RETURN
 110 NCLASS=20
     GO TO 130
 120 NCLASS=21
 130 CALL PACK
     IF(IEROR.NE.0)RETURN
     GO TO 25
 140 CALL FNCTTB(JF)
     IF(JF.NE.1)CALL IDTE(0)
     GO TO 300
 150 NBDP=NBDP+1
     GO TO 165
 160 NADP=NADP+1
 165 VAL=VAL*10.0+IDIG(ICH)
     GO TO 25
 170 NCLASS=21
     IF(VAL.GT.XMXINT)IEROR=18
     GO TO 205
 180 IEXSGN=-1
     GO TO 25
 190 IEXPV=IEXPV*10+IDIG(ICH)
     GO TO 25
 200 NCLASS=20
     IPOW=NBDP+IEXPV*IEXSGN
     IF(IPOW.GT.MXPOW)IEROR=19
     IPOW=IEXSGN*IEXPV-NADP
     IF(IEROR.NE.0)GO TO 300
     VAL=VAL*10.0**IPOW
 205 VALTB(NEWCN)=VAL
     NVALUE=NEWCN
     NEWCN=NEWCN+1
     IF(NEWCN.GT.MAXCN)IEROR=69
     GO TO 300
 210 NVALUE=6
     NCLASS=6
     GO TO 300
 220 NCLASS=8
     NVALUE=4
     GO TO 300
 230 NCLASS=7
     NVALUE=7
     GO TO 300
 240 NCLASS=7
     NVALUE=8
     GO TO 300
 250 NCLASS=5
     NVALUE=-1
     GO TO 300
 260 NCLASS=5
     NVALUE=-2
     GO TO 300
 270 NCLASS=3
     NVALUE=0
     GO TO 300
```

```
  300 CALL MATRIX(IFLG,IWAS)
      IF(IFAIL.EQ.1)ICHCNT=ICHCNT-1
      IF(IEROR.NE.0)RETURN
      IF(IFLG.EQ.1)GO TO 1
      RETURN
      END
C
C     DEGISKEN KARAKTERLERININ IDTEMP'DE BIRIKTIRILMESI
C
      SUBROUTINE PACK
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      DATA IB/' '/
      ISSIX=ISSIX+1
      IF(ISSIX.NE.1)GO TO 10
      DO 5 M=1,6
      IDTEMP(M)=IB
    5 CONTINUE
   10 IF(ISSIX.GT.6)GO TO 20
      IDTEMP(ISSIX)=ICH
      RETURN
   20 IEROR=7
      RETURN
      END
C
C     FONKSIYONLARIN SAPTANMASI
C
      SUBROUTINE FNCTTB(JF)
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      DIMENSION IFUNCS(6,10),NVAL(10)
      DATA IFUNCS/'S','I','N',' ',' ',' ','C','O','S',' ',' ',' ',
&     'A','L','O','G',' ',' ','A','L','O','G','1','0',
&     'S','Q','R','T',' ',' ','E','X','P',' ',' ',' ','A','B','S',
&     ' ',' ',' ','F','L','O','A','T',' ','I','A','B','S',' ',
&     ' ','I','N','T',' ',' ',' '/
      DATA NVAL/41,42,43,44,45,46,47,48,57,58/
      JF=0
   25 DO 100 NN=1,10
      DO 50 MM=1,6
      IF(IDTEMP(MM).NE.IFUNCS(MM,NN))GO TO 100
   50 CONTINUE
      GO TO 150
  100 CONTINUE
      RETURN
  150 JF=1
      NCLASS=10
      NVALUE=NVAL(NN)
      RETURN
      END
C
C     DEYIMLERDE SOZDIZIM HATALARININ DENETIMI
C
      SUBROUTINE MATRIX(IFLG,IWAS)
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      DIMENSION IMAT(8,8),ICLVAL(10),ICLMN(10)
      DATA ICLVAL/3,4,5,6,7,8,9,10,20,21/
```

```
      DATA IMAT/0,0,0,0,0,0,0,0,-170,-175,0,-174,1,-174,0,0,-170,-175
     &    ,0,-174,1,-174,0,0,0,-175,-174,0,0,0,-172,-172,-170,-175,0,-173
     &    ,-174,-174,0,0,-170,-175,0,-173,-174,-174,0,0,-171,-175,0,-171,
     &    -171,-171,-171,171,0,-175,-176,0,0,0,-176,-176/
      DATA ICLB/')'/
      IFLG=0
      DO 10 JM=1,10
      IF(NCLASS.EQ.ICLVAL(JM))GO TO 20
   10 CONTINUE
      IEROR=10
      RETURN
   20 INOW=ICLMN(JM)
      IF(NCLASS.NE.5)GO TO 30
      IF(ICH.EQ.ICLB)INOW=INOW+1
   30 MAT=IMAT(INOW,IWAS)
      IF(MAT.EQ.0)GO TO 40
      IF(MAT.LT.0)GO TO 50
      IFLG=1
      IF(NVALUE.EQ.5)GO TO 40
      IFLG=2
      NCLASS=9
      NVALUE=3
   40 IWAS=INOW
      RETURN
   50 IEROR=-(159+MAT)
      RETURN
      END
C
C    DEGISKEN TABLOSU MANIPULASYONU
C
      SUBROUTINE IDTB(IN)
      COMMON IDTAB(20,6),NOIDTB,MAXID,NEWIDT
      COMMON IC2T5(296),RC2T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      IF(NOIDTB.EQ.0)GO TO 51
      DO 50 JS=1,NOIDTB
      DO 25 KS=1,6
      IF(IDTEMP(KS).NE.IDTAB(JS,KS))GO TO 50
   25 CONTINUE
      GO TO 100
   50 CONTINUE
   51 IF(IN.EQ.1)GO TO 60
      IEROR=25
      RETURN
   60 NOIDTB=NOIDTB+1
      DO 75 J=1,6
      IDTAB(NOIDTB,J)=IDTEMP(J)
   75 CONTINUE
      NVALUE=NOIDTB
      IF(NOIDTB.GE.MAXID)IEROR=20
      RETURN
  100 NVALUE=JS
      RETURN
      END
C
C    RAKAMIN KARAKTER KODUNDAN NUMERIK KODA DONUSTURULMESI
C
      FUNCTION IDIG(ICH)
      DATA IZ/'0'/,IO/'1'/
      IDIG=(ICH-IZ)/(IO-IZ)
      RETURN
```

```
C
C    READ/WRITE KOMUTLARINI COZUMLEYEN GSM MODELI
C
      SUBROUTINE GSM3
      COMMON IC1(123)
      COMMON IFRTB(10,2),MAXIFR,NEWIFR,NGIVEN,NEXPCT,NOWIFR
      COMMON IC3T4(208),RC3T4(40)
      COMMON ICODEX(60),NEWCD,MAXCD,XMXINT,MXPOW
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      DIMENSION ICML(13),ICMH(13),NSUC(13),NFAL(13),IACTS(13)
      DATA ICML/'0',',','0','0',')','I','A','A','0',',','@','I','A'/
      DATA ICMH/'9',',','9','9',')','N','Z','Z','9',',','@','N','Z'/
      DATA NSUC/2,3,4,4,6,8,8,8,8,6,-1,8,8/
      DATA NFAL/-58,-59,-60,5,-61,7,11,9,10,11,-62,13,-66/
      DATA IACTS/1,2,3,3,4,5,6,7,7,8,8,5,6/
      DATA MS/'@'/
      IOFF=NCLASS
      ISTT=1
    1 IDP=0
      NCLASS=20
    5 CALL GETCH
   10 IF(ICML(ISTT).LE.ICH.AND.ICMH(ISTT).GE.ICH)GO TO 15
      ISTT=NFAL(ISTT)
      IF(ISTT.GT.0)GO TO 10
      IEROR=-ISTT
      RETURN
   15 ICT=IACTS(ISTT)
      GO TO(110,120,130,140,150,160,170,180),ICT
   20 ISTT=NSUC(ISTT)
      IF(ISTT.GT.0)GO TO 5
      RETURN
  110 IDEVCD=IDIG(ICH)
      JOFF=IOFF-73
      IDVCH=IABS(JOFF)+5
      IF(IDEVCD.EQ.IDVCH)GO TO 20
      IEROR=63
      RETURN
  120 VLAB=0.
      GO TO 20
  130 VLAB=VLAB*10.0+IDIG(ICH)
      IF(VLAB.LE.XMXINT)GO TO 20
      IEROR=64
      RETURN
  140 NLAB=VLAB
      IF(NOWIFR.EQ.0)GO TO 143
      DO 142 L=1,NOWIFR
      IF(IFRTB(L,1).EQ.NLAB)GO TO 144
  142 CONTINUE
  143 NEXPCT=NLAB
      L=NOWIFR+1
      GO TO 146
  144 IF(IFRTB(L,2).EQ.0)GO TO 148
  146 CALL GENC(L,0,0)
      GO TO 20
  148 IEROR=65
      RETURN
  150 NCLASS=NCLASS+1
  160 IDP=1
      ISSIX=0
  170 CALL PACK
      IF(IEROR.NE.0)RETURN
```

```
  180 IF(IDP.EQ.0)RETURN
      IN=IOFF-72
      CALL IDTB(IN)
      IF(IEROR.NE.0)RETURN
      CALL GENC(IOFF,NVALUE,NCLASS)
      IF(ICH.EQ.MS)RETURN
      ISTT=12
      GO TO 1
      END
C
C     FORMAT COZUMLEYEN GSM MODELI
C
      SUBROUTINE FRMTS
      COMMON IC1(123)
      COMMON IFRTB(10,2),MAXIFR,NEWIFR,NGIVEN,NEXPCT,NOWIFR
      COMMON IFRMT(200),MAXFRT,NEWFRT,NOWFRT
      COMMON IC4T5(68),RC4T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      DIMENSION ILOW(34),IHIG(34),NSC(34),NAS(34),NFL(34)
      DATA ILOW/'0','0','I','0','0','F','0','0','.','0','0',
&     'E','0','0',',','0','0','X','/','H','(',')',',','I',
&     'F','E','X','/','(',')',',',10016,'H','@'/
      DATA IHIG/'9','9','I','9','9','F','9','9','.','9','9',
&     'E','9','9','.','9','9','X','/','H','(',')',',','I',
&     'F','E','X','/','(',')',',',10016,'H','@'/
      DATA NSC/2,2,4,5,5,7,8,8,10,11,11,13,14,14,16,17,17,-4,
&     -109,-5,-7,-110,-110,4,7,13,-4,-6,-7,-8,-9,-5,-112,-116/
      DATA NAS/1,1,0,2,2,0,2,2,0,3,3,0,2,2,0,3,3,0,0,0,0,0,0,4,
&     4,4,0,0,0,0,0,0,0,0/
     ,DATA NFL/24,3,6,-101,-1,12,-101,9,-103,-104,-2,18,-101,15,
&     -106,-107,-3,19,20,21,22,23,-113,25,26,27,28,29,30,31,32,
&     33,34,-113/
      DATA MS/'@'/,IAP/10016/
      NOWFRT=NEWFRT
      CALL GENF(-5)
      IBRC=1
      IWAS=3
    1 NRPT=0
      NWID=0
      NDEC=0
      NST=1
    5 CALL GETCH
   10 IF(ICH.LT.ILOW(NST).OR.ICH.GT.IHIG(NST))GO TO 20
      NAC=NAS(NST)
      NST=NSC(NST)
      IFL=0
      GO TO 30
   20 NST=NFL(NST)
      NAC=0
      IFL=1
   30 IF(NST.LT.0)GO TO 300
      IF(NAC.NE.0)GO TO 50
      IF(IFL.EQ.0)GO TO 5
      GO TO 10
   50 ID=IDIG(ICH)
      GO TO(110,120,130,140),NAC
  110 NRPT=NRPT*10+ID
      IF(NRPT.GT.80)GO TO 337
      GO TO 140
  120 NWID=NWID*10+ID
```

```
130   NDEC=NDEC*10+ID
      IF(NDEC.GT.80)GO TO 337
      GO TO 5
140   IF(IWAS.EQ.1)IEROR=14
      IF(IWAS.EQ.4)IEROR=19
      IF(IEROR.EQ.0)GO TO 5
      RETURN
300   NST=-NST
      IF(NST.GT.100)GO TO 999
      CALL MATFOR(IWAS,NST)
      IF(IEROR.NE.0)RETURN
      NRFS=NRPT
      IF(NRPT.EQ.0)NRPT=1
      IFC=NRPT*10+NST
      NW=NWID-NDEC
      IF(NWID.EQ.0.OR.NW.GT.0)GO TO 305
      IEROR=5
      RETURN
305   GO TO(330,310,320,340,350,340,370,380,1),NST
310   IF(NW.GE.3)GO TO 330
      IEROR=23
      RETURN
320   IF(NW.GE.7)GO TO 330
      IEROR=24
      RETURN
330   IF(NWID.NE.0)GO TO 335
      IEROR=28
      RETURN
335   CALL GENF(IFC)
      CALL GENF(NWID)
      IF(NST.EQ.1)GO TO 336
      CALL GENF(NDEC)
      IF(NDEC.EQ.0)GO TO 998
336   ICHCNT=ICHCNT-1
      IF((NRPT*NWID).LE.80)GO TO 1
337   IEROR=2
      RETURN
340   CALL GENF(IFC)
      GO TO 1
350   IF(ICH.EQ.IAP)GO TO 400
      IF(NRPS.EQ.0)GO TO 997
      CALL GENF(IFC)
      DO 355 I=1,NRPT
      ICHCNT=ICHCNT+1
      IF(ICHCNT.GT.72)GO TO 990
      IFC=ISTRNG(ICHCNT)
      CALL GENF(IFC)
355   CONTINUE
      GO TO 1
370   IBRC=IBRC+1
      IFC=-NRPT*10-7
      GO TO 340
380   IBRC=IBRC-1
      IFC=-8
      IF(IBRC.NE.0)GO TO 340
      CALL GETCH
      IF(ICH.NE.MS)GO TO 993
      CALL GENF(-9)
      IFRTB(NOWIFR,2)=NEWFRT
      NEWFRT=NOWFRT
      RETURN
400   NRPT=0
```

```
      I=NOWFRT
      NOWFRT=NOWFRT+1
  410 ICHCNT=ICHCNT+1
      IF(ICHCNT.GE.72)GO TO 995
      ICH=ISTRNG(ICHCNT)
      IF(ICH.EQ.IAP)GO TO 420
      CALL GENF(ICH)
      NRPT=NRPT+1
      GO TO 410
  420 IF(NRPT.EQ.0)GO TO 997
      IFRMT(I)=NRPT*10+5
      GO TO 1
  990 IEROR=25
      RETURN
  993 IEROR=22
      RETURN
  995 IEROR=26
      RETURN
  997 IEROR=27
      RETURN
  998 IEROR=11
      RETURN
  999 IEROR=NST-100
      RETURN
      END
C
C     URETILEN DESEN OGE KODLARININ IFRMT DIZISINI YERLESTIRILMESI
C
      SUBROUTINE GENF(I)
      COMMON IC1T2(148)
      COMMON IFRMT(200),MAXFRT,NEWFRT,NOWFRT
      COMMON IC4T5(68),RC4T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      IF(NOWFRT.GE.MAXFRT)GO TO 10
      IFRMT(NOWFRT)=I
      NOWFRT=NOWFRT+1
      RETURN
   10 IEROR=8
      RETURN
      END
C
C     DESEN OGE DIZIM DENETIMI
C
      SUBROUTINE MATFOR(IWAS,IST)
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      DIMENSION MATF(5,5)
      DATA MATF/14,0,0,19,0,0,0,0,0,0,15,0,0,19,0,0,0,17,0,20,
     &     0,0,18,0,21/
      INOW=IST-4
      IF(INOW.LE.0)INOW=1
      MAT=MATF(IWAS,INOW)
      IF(MAT.NE.0)GO TO 10
      IWAS=INOW
      RETURN
   10 IEROR=MAT
      RETURN
      END
```

```
C
C    ISLEM KODLARINI YORUMLAYAN MODUL
C
      SUBROUTINE EXECCD
      COMMON IDTAB(20,6),NOIDTB,MAXID,NEWIDT
      COMMON IFRTB(10,2),MAXIFR,NEWIFR,NGIVEN,NEXPCT,NOWIFR
      COMMON IFRMT(200),MAXFRT,NEWFRT,NOWFRT
      COMMON VALTB(40),MAXCN,MAXTP,MACC,NEWTP,NEWCN
      COMMON ICODEX(60),NEWCD,MAXCD,XMXINT,MXPOW
      COMMON NCLASS,NVALUE,IEROR,MDCHR,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      COMMON IFSTAK(10,2),ITFS,IREW,IFDP,IRDWR,IWIDTH,IDCP
      DATA ALXMRL/170.28593/,ALALXM/5.1374790/,XMRL10/.9E73/,IB/' '/
      X=0.0
      I=0
   10 I=I+1
   15 VALTB(MACC)=X
   20 IP=ICODEX(I)
      IF(IP.EQ.0)RETURN
      I=I+1
      IT=IP/10
      IP=IP-IT*10
      IT=IT+1
      IR=ICODEX(I)
      IF(IR.GT.0.AND.IR.LT.MACC)Y=VALTB(IR)
      GO TO(25,25,400,400,120,115,400,260,260,350),IT
C
C    TEMEL ISLEM KODLARI
C
   25 GO TO(28,30,60,70,80,90,100,100),IP
   28 X=Y
      GO TO 10
   30 IF(IT.NE.2)GO TO 50
      Y=ABS(X)
      IF(Y.LE.XMXINT)GO TO 40
      IEROR=1
      RETURN
   40 IX=X
      X=IX
   50 VALTB(IR)=X
      GO TO 10
   60 X=-X
      GO TO 15
   70 IF(X.EQ.0.0.OR.X.EQ.1.0.OR.X.EQ.-1.0.OR.Y.EQ.0.0)GO TO 72
      Z=ABS(X)
      Z=ALOG(Z)
      Z=ABS(Z)
      Z=ALOG(Z)
      W=ABS(Y)
      W=ALOG(W)+Z
      IF(ABS(W).LE.ALALXM)GO TO 72
      IEROR=2
      RETURN
   72 IF(X.GE.0)GO TO 75
      IF(ABS(Y).GT.XMXINT)GO TO 73
      IY=Y
      IF ((Y-IY).NE.0.0)GO TO 73
      X=X**IY
      GO TO 10
   73 IEROR=13
      RETURN
```

```
   80 Z=X/10.0+Y/10.0
      IF(ABS(Z).LE.XMRL10)GO TO 81
      IEROR=3
      RETURN
   81 X=X+Y
      GO TO 10
   90 Z=X/10.0-Y/10.0
      IF(ABS(Z).LE.XMRL10)GO TO 91
      IEROR=4
      RETURN
   91 X=X-Y
      GO TO 10
  100 W=ABS(X)
      IF(W.NE.0.0)W=ALOG(W)
      Z=ABS(Y)
      IF(Z.NE.0.0)Z=ALOG(Z)
      IF(IP.NE.7)GO TO 110
      IF(X.EQ.0.0.OR.Y.EQ.0.0)GO TO 101
      Z=W+Z
      IF(ABS(Z).LE.ALXMRL)GO TO 101
      IEROR=5
      RETURN
  101 X=X*Y
      GO TO 10
  110 IF(Y.EQ.0.0)GO TO 111
      IF(X.EQ.0.0)GO TO 112
      Z=W-Z
      IF(ABS(Z).LE.ALXMRL)GO TO 112
  111 IEROR=6
      RETURN
  112 X=X/Y
      IF(IT.NE.2)GO TO 10
  113 IX=X
      X=IX
      GO TO 10
C
C     FONKSIYON KODLARI
C
  115 IP=IP+2
  120 GO TO(121,130,140,160,180,200,220,10,230,250),IP
  121 IF(X.GE.0.0)GO TO 125
      X=ABS(X)
      X=-SIN(X)
      GO TO 15
  125 X=SIN(X)
      GO TO 15
  130 IF(X.LT.0.0)X=-X
      X=COS(X)
      GO TO 15
  140 IF(X.GT.0.0)GO TO 150
      IEROR=7
      RETURN
  150 X=ALOG(X)
      GO TO 15
  160 IF(X.GT.0.0)GO TO 170
      IEROR=8
      RETURN
```

```
  170 X=ALOG(X)/ALOG(10.0)
      GO TO 15
  180 IF(X.GE.0.0)GO TO 190
      IEROR=9
      RETURN
  190 X=SQRT(X)
      GO TO 15
  200 Y=ABS(X)
      IF(Y.LE.ALXMRL)GO TO 210
      IEROR=10
      RETURN
  210 X=EXP(X)
      GO TO 15
  220 X=ABS(X)
      GO TO 15
  230 X=ABS(X)
      IF(X.LE.XMXINT)GO TO 255
      IEROR=11
      RETURN
  250 Y=ABS(X)
      IF(Y.LE.XMXINT)GO TO 255
      IEROR=12
      RETURN
  255 IX=X
      X=IX
      GO TO 15
C
C    GIRIS/CIKIS KODLARI
C
  260 DO 265 J=1,6
      ICH=IDTAB(IR,J)
      IF(ICH.EQ.IB)ICH=0
      IDTEMP(J)=ICH
  265 CONTINUE
C
C    "SERG","SERT"
C
      GO TO(270,290,310),IP
  270 ICHCNT=1
      DO 272 I=1,12
      ISTRNG(I)=0
  272 CONTINUE
      IF(IT.EQ.8)GO TO 275
      IX=X
      CALL IFORW(7,IX,0)
      GO TO 285
  275 Y=ABS(X)
      IF(Y.EQ.0.0)GO TO 277
      IF(Y.LT.0.001.OR.Y.GT.1.0E4)GO TO 280
  277 IWIDTH=12
      IDCP=2
      CALL FFORW(X)
      GO TO 285
  280 IWIDTH=12
      IDCP=5
      CALL EFORW(X)
  285 WRITE(101,999)(IDTEMP(J),J=1,6),(ISTRNG(J),J=1,12)
      RETURN
```

```
  380 IF(IREW.EQ.0)CALL IOLINE(1)
  400 RETURN
  996 FORMAT(1X,80A1)
  999 FORMAT(1X,6A1,' DEGERI: ',12A1)
      END
C
C    AKTIF OGEYI SAPTAYAN ALTRUTIN
C
      SUBROUTINE EXACFD
      COMMON IC1T2(148)
      COMMON IFRMT(200),MAXFRT,NEWFRT,NOWFRT
      COMMON IC4TS(68),RC4TS(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      COMMON IFSTAK(10,2),ITFS,IREW,IFDP,IRDWR,IWIDTH,IDCP
      ICLS=0
   10 IRPT=IFSTAK(ITFS,1)
      JFLD=IFSTAK(ITFS,2)
      IFLD=IFRMT(JFLD)
      IR=IFLD/10
      IRIPIT=IABS(IR)
      IFLD=IFLD-IR*10
      IF(IRPT.GE.IRIPIT)GO TO 20
      IRPT=IRPT+1
      IFSTAK(ITFS,1)=IRPT
      IF(IFLD.LT.0)GO TO 30
      IFDP=IFLD
      RETURN
   20 ITFS=ITFS-1
      IF(IFLD.LT.0.AND.ICLS.NE.0)JFLD=ICLS
   30 CALL ADVANC(IFLD,JFLD)
      IFLD=IABS(IFLD)
      GO TO (100,100,100,40,40,40,70,80,90),IFLD
   40 CALL WRPAS(IFLD,JFLD)
      IF(IEROR.NE.0)RETURN
      GO TO 30
   70 ITFS=ITFS+1
      IFSTAK(ITFS,1)=1
      IFSTAK(ITFS,2)=JFLD
      GO TO 30
   80 ICLS=JFLD
      GO TO 10
   90 IREW=1
      CALL REWND(IFLD,JFLD)
      IRPT=0
      IFDP=JFLD
      ITFS=0
      GO TO 200
  100 IREW=0
      IFDP=IFLD
      IRPT=1
  200 ITFS=ITFS+1
      IFSTAK(ITFS,1)=IRPT
      IFSTAK(ITFS,2)=JFLD
      M=JFLD+1
      L=M+1
      IF(IFLD.NE.1)IDCP=IFRMT(L)
      IWIDTH=IFRMT(M)
      RETURN
      END
```

```
C
C    "YAZG","YAZT"
C
  290 IF(IREW.NE.1)GO TO 293
      CALL EXACFD
      IF(IREW.NE.1)GO TO 293
  291 IEROR=17
      GO TO 372
  293 GO TO(295,297,299),IFDP
  295 IF(IT.EQ.8)GO TO 340
      IY=Y
      CALL IFORW(IWIDTH,IY,0)
      GO TO 300
  297 IF(IT.EQ.9)GO TO 340
      CALL FFORW(Y)
      GO TO 300
  299 IF(IT.EQ.9)GO TO 340
      CALL EFORW(Y)
  300 IF(IEROR.NE.0)GO TO 372
  305 CALL EXACFD
      IF(IREW.NE.1)GO TO 370
      CALL IOLINE(1)
      GO TO 10
C
C    "OKUG","OKUT"
C
  310 IF(IREW.NE.1)GO TO 315
      CALL EXACFD
      IF(IREW.EQ.1)GO TO 291
      CALL IOLINE(1)
  315 GO TO(320,322,324),IFDP
  320 IF(IT.EQ.8)GO TO 340
      CALL IFORR(X)
      Y=ABS(X)
      IF(Y.GT.XMXINT)IEROR=25
      GO TO 327
  322 IF(IT.EQ.9)GO TO 340
      CALL FFORR(X)
      GO TO 327
  324 IF(IT.EQ.9)GO TO 340
      CALL EFORR(X)
  327 VALTB(IR)=X
      IF(IEROR.NE.0)GO TO 372
      CALL EXACFD
      GO TO 370
  340 IEROR=15
      GO TO 372
C
C    "YDESEN","ODESEN","DURY"
C
  350 IF(IP.EQ.3)GO TO 380
      IREW=0
      ITFS=1
      IFSTAK(1,1)=1
      IFSTAK(1,2)=IFRTB(IR,2)
      IRDWR=IP
      CALL IOLINE(0)
      IF(IP.EQ.1)GO TO 305
      CALL EXACFD
  370 IF(IEROR.EQ.0)GO TO 10
  372 IF(ICHCNT.GT.78)ICHCNT=78
      IF(IRDWR.EQ.1)WRITE(101,996)(ISTRNG(J),J=2,ICHCNT)
```

```
C
C    DESEN GOSTERGESINI BULUNAN OGEYE GORE ILERLETEN ALTRUTIN
C
      SUBROUTINE ADVANC(IFLD,JFLD)
      COMMON IC1T2(148)
      COMMON IFRMT(200),MAXFRT,NEWFRT,NOWFRT
      L=1
      IF(IFLD.EQ.2.OR.IFLD.EQ.3)L=3
      IF(IFLD.EQ.1)L=2
      IF(IFLD.EQ.5)L=IFRMT(JFLD)/10+1
      JFLD=JFLD+L
      IFLD=IFRMT(JFLD)
      IFLD=IFLD-(IFLD/10)*10
      RETURN
      END
C
C    DESENI GERI SARAN ALTRUTIN
C
      SUBROUTINE REWND(IFLD,JFLD)
      COMMON IC1T2(148)
      COMMON IFRMT(200),MAXFRT,NEWFRT,NOWFRT
      ICLB=0
    1 JFLD=JFLD-1
      IFLD=IFRMT(JFLD)
      IF(IFLD.GT.0)GO TO 1
      IFLD=IFLD-(IFLD/10)*10
      IF(IFLD.EQ.-5)RETURN
      IF(IFLD.EQ.-8)GO TO 20
      ICLB=ICLB-1
      IF(ICLB.NE.0)GO TO 1
      RETURN
   20 ICLB=ICLB+1
      GO TO 1
      END
C
C    PASIF DESEN OGELERIN YORUMLANMASI
C
      SUBROUTINE WRPAS(IFLD,JFLD)
      COMMON IC1T2(148)
      COMMON IFRMT(200),MAXFRT,NEWFRT,NOWFRT
      COMMON IC4T5(68),RC4T6(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      COMMON IFSTAK(10,2),ITFS,IREW,IFDP,IRDWR,IWIDTH,IDCP
      IRPT=IFRMT(JFLD)/10
      IF(IFLD.EQ.6)GO TO 1
      IF(IFLD.EQ.5)GO TO 3
      L=IRPT+ICHCNT
      IF(L.GT.80)GO TO 44
      ICHCNT=L
      RETURN
    1 CALL IOLINE(1)
      RETURN
    3 DO 33 I=1,IRPT
      L=JFLD+I
      ISTRNG(ICHCNT)=IFRMT(L)
      ICHCNT=ICHCNT+1
      IF(ICHCNT.GT.80)GO TO 44
   33 CONTINUE
      RETURN
   44 IEROR=16
      RETURN
```

```
C
C    OKUMA VEYA YAZMA TAMPONUNUN YENILENMESI
C
      SUBROUTINE IOLINE(IN)
      COMMON IC1T6(424),RC1T6(41)
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      COMMON IFSTAK(10,2),ITFS,IREW,IFDP,IRDWR,IWIDTH,IDCP
      DATA IB/' '/
      IF(IRDWR.NE.1)GO TO 10
      IF(IN.EQ.0)GO TO 5
      WRITE(101,20)(ISTRNG(I),I=2,ICHCNT)
    5 DO 7 I=1,80
      ISTRNG(I)=IB
    7 CONTINUE
      ICHCNT=1
      RETURN
   10 I=IRDWR-1
      WRITE(101,30)I
      IRDWR=IRDWR+1
      READ(100,40)(ISTRNG(I),I=1,80)
      ICHCNT=1
      RETURN
   20 FORMAT(1X,80A1)
   30 FORMAT(' VERI SATIRI',I3,/,1X)
   40 FORMAT(80A1)
      END
C
C    I-DESENIYLE TAMSAYI OKUMA
C
      SUBROUTINE IFORR(RNUM)
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      COMMON IFSTAK(10,2),ITFS,IREW,IFDP,IRDWR,IWIDTH,IDCP
      DIMENSION NUMCOD(5)
      DATA NUMCOD/'E','+','-','.',' '/,NL/'0'/,NH/'9'/
      IF((ICHCNT+IWIDTH).GT.80)GO TO 15
      RNUM=0
      MSGN=1
      DO 9 I=1,IWIDTH
      IBF=ICHCNT+I-1
      ITOKEN=ISTRNG(IBF)
      DO 2 K=1,5
      IF(ITOKEN.EQ.NUMCOD(K))GO TO 4
    2 CONTINUE
      IF(ITOKEN.GE.NL.AND.ITOKEN.LE.NH)GO TO 8
    3 IEROR=18
      RETURN
    4 GO TO (3,7,6,3,5),K
    5 ITOKEN=NL
      GO TO 8
    6 MSGN=-1
    7 IF(RNUM.NE.0.OR.ITOSGN.EQ.2)GO TO 10
      ITOSGN=2
      GO TO 9
    8 RNUM=IDIG(ITOKEN)+RNUM*10.0
    9 CONTINUE
      RNUM=MSGN*RNUM
      ICHCNT=ICHCNT+IWIDTH
      RETURN
   10 IEROR=19
      ICHCNT=IBF
```

```
   15 IEROR=16
      RETURN
      END
C
C    F-DESENIYLE GERCEKSAYI OKUMA
C
      SUBROUTINE FFORR(FNUM)
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      COMMON IFSTAK(10,2),ITFS,IREW,IFDF,IRDWR,IWIDTH,IDCP
      DIMENSION NUMCOD(5)
      DATA NUMCOD/'E','+','-','.',' '/,NL/'0'/,NH/'9'/
      IF((ICHCNT+IWIDTH).GT.80)GO TO 25
      FNUM=0.0
      MSGN=1
      ITOSGN=1
      IDP=0
      IPDCP=IDCP
      DO 15 I=1,IWIDTH
      IBF=ICHCNT+I-1
      ITOKEN=ISTRNG(IBF)
      DO 1 K=1,5
      IF(ITOKEN.EQ.NUMCOD(K))GO TO 3
    1 CONTINUE
      IF(ITOKEN.GE.NL.AND.ITOKEN.LE.NH)GO TO 14
    2 IEROR=20
      RETURN
    3 GO TO (2,12,11,13,10),K
   10 ITOKEN=NL
      GO TO 14
   11 MSGN=-1
   12 IF(FNUM.NE.0.0.OR.ITOSGN.EQ.2)GO TO 19
      ITOSGN=2
      GO TO 15
   13 IF(IDP.EQ.1)GO TO 20
      IDP=1
      IPDCP=IWIDTH-I
      GO TO 15
   14 FNUM=IDIG(ITOKEN)+FNUM*10.0
   15 CONTINUE
      FNUM=MSGN*FNUM
      FNUM=FNUM/(10.0**IPDCP)
      ICHCNT=ICHCNT+IWIDTH
      RETURN
   19 IEROR=19
      GO TO 21
   20 IEROR=21
   21 ICHCNT=IBF
      RETURN
   25 IEROR=16
      RETURN
      END
C
C    E-DESENIYLE TAMSAYI OKUMA
C
      SUBROUTINE EFORR(ENUM)
      COMMON IC1T4(419),RC1T4(40)
      COMMON ICODEX(30),NEWCD,MAXCD,XMXINT,MXPOW
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      COMMON IFSTAK(10,2),ITFS,IREW,IFDF,IRDWR,IWIDTH,IDCP
      DIMENSION NUMCOD(5)
```

```
      DATA NUMCOD/'E','+','-','.',' '/,NL/'0'/,NH/'9'/
      IF((ICHCNT+IWIDTH).GT.80)GO TO 66
      IDP=0
      IE=1
      ENUM=0.0
      MSGN=1
      ITOSGN=1
      IEXP=0
      DO 55 I=1,IWIDTH
      IBF=ICHCNT+I-1
      ITOKEN=ISTRNG(IBF)
      DO 100 K=1,5
      IF(ITOKEN.EQ.NUMCOD(K))GO TO 200
  100 CONTINUE
      IF(ITOKEN.GE.NL.AND.ITOKEN.LE.NH)GO TO 6
      IEROR=22
      RETURN
  200 GO TO (1,3,2,4,5),K
    1 IF(IE.EQ.2)GO TO 77
      IE=2
      NDCP=NDCP-IWIDTH+I-1
      TENUM=MSGN*ENUM
      ENUM=0.0
      ITOSGN=1
      MSGN=1
      GO TO 55
    2 MSGN=-1
    3 IF(ENUM.NE.0.0.OR.ITOSGN.EQ.2.OR.IEXP.NE.0)GO TO 88
      ITOSGN=2
      GO TO 55
    4 IF(IDP.EQ.1.OR.IE.EQ.2)GO TO 111
      IDP=1
      NDCP=IWIDTH-I
      GO TO 55
    5 ITOKEN=NL
    6 IF(IE.EQ.2)GO TO 7
      ENUM=IDIG(ITOKEN)+ENUM*10.0
      GO TO 55
    7 IEXP=IDIG(ITOKEN)+IEXP*10
   55 CONTINUE
      IF(IE.EQ.1.OR.IEXP.GE.MXPOW)GO TO 99
      IF(IDP.EQ.0)NDCP=IDCP
      IEXP=MSGN*IEXP-NDCP
      ENUM=TENUM*10.0**IEXP
      ICHCNT=ICHCNT+IWIDTH
      RETURN
   66 IEROR=16
      RETURN
   77 IEROR=23
      GO TO 112
   88 IEROR=19
      GO TO 112
   99 IEROR=24
      GO TO 112
  111 IEROR=21
  112 ICHCNT=IBF
      RETURN
      END
C
C    I-DESENIYLE TAMSAYI YAZMA
C
      SUBROUTINE IFORW(IFWID,INUMW,NULLSP)
      COMMON IC1T5(419),RC1T5(41)
```

```
      DIMENSION NUMCOD(10)
      DATA NUMCOD/'0','1','2','3','4','5','6','7','8','9'/
      DATA MINSGN/'-'/
      I=0
      IF((ICHCNT+IFWID).GT.80)GO TO 10
      IDVD=IABS(INUMW)
    1 IQUOT=IDVD/10
      IDIG=IDVD-IQUOT*10
      I=I+1
      IT=ICHCNT+IFWID-I
      ISTRNG(IT)=NUMCOD(IDIG+1)
      IF(I.EQ.IFWID)GO TO 3
      IF(IQUOT.EQ.0.AND.NULLSP.EQ.0)GO TO 2
      IDVD=IQUOT
      GO TO 1
    2 IF(INUMW.GE.0)GO TO 4
      IT=IT-1
      ISTRNG(IT)=MINSGN
      GO TO 4
    3 IF(INUMW.GE.0.AND.IQUOT.EQ.0)GO TO 4
      IEROR=14
      RETURN
    4 ICHCNT=ICHCNT+IFWID
      RETURN
   10 IEROR=16
      RETURN
      END
C
C     F-DESENIYLE GERCEKSAYI YAZMA
C
      SUBROUTINE FFORW(RNUMW)
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      COMMON IF$TAK(10,2),ITFS,IREW,IFDP,IRDWR,IWIDTH,IDCP
      DATA IDP/'.'/,MSGN/'-'/
      IF((ICHCNT+IWIDTH).GT.80)GO TO 25
      IEXP=0
      EMULT=RNUMW
      CALL EXPON(EMULT,IEXP)
      IF(IEXP.LT.5)GO TO 10
      IFREE=IWIDTH-IEXP-IDCP-2
      IF(IFREE.LT.0)GO TO 20
      KMSG=ICHCNT+IFREE
      ICHCNT=KMSG+1
      IF(RNUMW.LT.0.0)ISTRNG(KMSG)=MSGN
      RNUM=ABS(RNUMW)*(10.0**(-IEXP))
      CALL LARGE(IEXP,RNUM,0)
      RNUM=RNUM*(10.0**IEXP)
      INUMW=RNUM
      POSTPT=RNUM-INUMW
      IF(IEXP.EQ.0)GO TO 11
      CALL IFORW(IEXP,INUMW,1)
      GO TO 11
   10 INUMW=RNUMW
      POSTPT=RNUMW-INUMW
      IFWID=IWIDTH-IDCP-1
      CALL IFORW(IFWID,INUMW,0)
      IF(INUMW.NE.0.OR.RNUMW.GE.0.0)GO TO 11
      IMSGN=ICHCNT-2
      ISTRNG(IMSGN)=MSGN
   11 ISTRNG(ICHCNT)=IDP
      IPOSTP=ICHCNT+1
```

```
      ENRGW=ABS(POSTPT)*(10.0**IDCP)+0.5
      CALL IFORW(IDCP,INUMW,1)
      RETURN
   20 IEROR=14
      RETURN
   25 IEROR=16
      RETURN
      END
C
C    E-DESENIYLE GERCEK SAYI YAZMA
C
      SUBROUTINE EFORW(ENUMW)
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      COMMON ISTRNG(80),IDTEMP(6),ICH,ICHCNT,ISSIX
      COMMON IFSTAK(10,2),ITFS,IREW,IFDP,IRDWR,IWIDTH,IDCP
      DATA IDCPNT/'.'/,IE/'E'/,MINSGN/'-'/,IZERO/'0'/
      IF((ICHCNT+IWIDTH).GT.80)GO TO 15
      IEXP=0
      EMULT=ABS(ENUMW)
      IF(EMULT.EQ.0.0)GO TO 11
      IF(EMULT.LT.1.0)GO TO 3
      CALL EXPON(EMULT,IEXP)
   11 ICHCNT=ICHCNT+IWIDTH-IDCP-7
      IF(ENUMW.GE.0.0)GO TO 22
      ISTRNG(ICHCNT)=MINSGN
   22 ICHCNT=ICHCNT+1
      ISTRNG(ICHCNT)=IZERO
      ICHCNT=ICHCNT+1
      ISTRNG(ICHCNT)=IDCPNT
      ICHCNT=ICHCNT+1
      IF(IDCP.LE.4)GO TO 222
      POSTPT=ABS(ENUMW)*10.0**(-IEXP)
      CALL LARGE(IDCP,POSTPT,1)
      INUMW=ABS(POSTPT)*(10.0**IDCP)+0.5
      IF(IDCP.EQ.0)GO TO 20
      GO TO 2
  222 INUMW=ABS(ENUMW)*10.0**(IDCP-IEXP)+0.5
    2 CALL IFORW(IDCP,INUMW,1)
   20 ISTRNG(ICHCNT)=IE
      ICHCNT=ICHCNT+1
      IF(IEXP.LT.0)ISTRNG(ICHCNT)=MINSGN
      IEXWD=2
      ICHCNT=ICHCNT+1
      EXP=IEXP
      IEXP=ABS(EXP)
      CALL IFORW(IEXWD,IEXP,1)
      RETURN
    3 EPROD=EMULT*10.0
      IF(EPROD.GE.1.0)GO TO 11
      IEXP=IEXP-1
      EMULT=EPROD
      GO TO 3
   15 IEROR=16
      RETURN
      END
C
C    GERCEKSAYI YAZARKEN DORT RAKAMLIK DIZGININ YAZILMASI
C
      SUBROUTINE LARGE(IFD,POSTPT,NULLSP)
   10 PSTNEW=ABS(POSTPT)*10000.0
      KNUMW=PSTNEW
      CALL IFORW(4,KNUMW,NULLSP)
```

```
      IF(NULLSP.LT.1)RETURN
      NULLSP=1
      GO TO 10
      END
C
C     E-DESENIYLE YAZARKEN ONDALIK USSU HESAPLAYAN ALTRUTIN
C
      SUBROUTINE EXPON(EMULT,IEXP)
    1 PROD=EMULT*0.1
      IEXP=IEXP+1
      IF(PROD.LT.1.0)RETURN
      EMULT=PROD
      GO TO 1
      END
C
C     DERLEYICI HATA MESAJLARININ IEROR'A GORE YAZILMASI
C
      SUBROUTINE ERRCOM
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      IT=(IEROR-1)/10
      IE=IEROR-IT*10
      IT=IT+1
      GO TO(100,210,320,430,540,650,760,870),IT
  100 GO TO(101,102,103,104,105,106,107,108,109,110),IE
  101 WRITE(101,1001)
      RETURN
  102 WRITE(101,1002)
      RETURN
  103 WRITE(101,1003)
      RETURN
  104 WRITE(101,1004)
      RETURN
  105 WRITE(101,1005)
      RETURN
  106 WRITE(101,1006)
      RETURN
  107 WRITE(101,1007)
      RETURN
  108 WRITE(101,1008)
      RETURN
  109 WRITE(101,1009)
      RETURN
  110 WRITE(101,1010)
      RETURN
  210 GO TO(211,212,213,214,215,216,217,218,219,220),IE
  211 WRITE(101,1011)
      RETURN
  212 WRITE(101,1012)
      RETURN
  213 WRITE(101,1013)
      RETURN
  214 WRITE(101,1014)
      RETURN
  215 WRITE(101,1015)
      RETURN
  216 WRITE(101,1016)
      RETURN
  217 WRITE(101,1017)
      RETURN
  218 WRITE(101,1018)
      RETURN
  219 WRITE(101,1019)
```

```
320 GO TO(321,322,323,324,325,326,327,328,329,330),IE
321 WRITE(101,1021)
    RETURN
322 WRITE(101,1022)
    RETURN
323 WRITE(101,1023)
    RETURN
324 WRITE(101,1024)
    RETURN
325 WRITE(101,1025)
    RETURN
326 WRITE(101,1026)
    RETURN
327 WRITE(101,1027)
    RETURN
328 WRITE(101,1028)
    RETURN
329 WRITE(101,1029)
    RETURN
330 WRITE(101,1030)
    RETURN
430 GO TO(431,432,433,434,435,436,437,438,439,440),IE
431 WRITE(101,1031)
    RETURN
432 WRITE(101,1032)
    RETURN
433 WRITE(101,1033)
    RETURN
434 WRITE(101,1034)
    RETURN
435 WRITE(101,1035)
    RETURN
436 WRITE(101,1036)
    RETURN
437 WRITE(101,1037)
    RETURN
438 WRITE(101,1038)
    RETURN
439 WRITE(101,1039)
    RETURN
440 WRITE(101,1040)
    RETURN
540 GO TO(541,542,543,544,545,546,547,548,549,550),IE
541 WRITE(101,1041)
    RETURN
542 WRITE(101,1042)
    RETURN
543 WRITE(101,1043)
    RETURN
544 WRITE(101,1044)
    RETURN
545 WRITE(101,1045)
    RETURN
546 WRITE(101,1046)
    RETURN
547 WRITE(101,1047)
    RETURN
548 WRITE(101,1048)
    RETURN
549 WRITE(101,1049)
    RETURN
550 WRITE(101,1050)
    RETURN
```

```
      RETURN
  653 WRITE(101,1053)
      RETURN
  654 WRITE(101,1054)
      RETURN
  655 WRITE(101,1055)
      RETURN
  656 WRITE(101,1056)
      RETURN
  657 WRITE(101,1057)
      RETURN
  658 WRITE(101,1058)
      RETURN
  659 WRITE(101,1059)
      RETURN
  660 WRITE(101,1060)
      RETURN
  760 GO TO(761,762,763,764,765,766,767,768,769,770),IE
  761 WRITE(101,1061)
      RETURN
  762 WRITE(101,1062)
      RETURN
  763 WRITE(101,1063)
      RETURN
  764 WRITE(101,1064)
      RETURN
  765 WRITE(101,1065)
      RETURN
  766 WRITE(101,1066)
      RETURN
  767 WRITE(101,1067)
      RETURN
  768 WRITE(101,1068)
      RETURN
  769 WRITE(101,1069)
      RETURN
  770 WRITE(101,1070)
      RETURN
  870 GO TO(871,872,873,874,875,876,877,878),IE
  871 WRITE(101,1071)
      RETURN
  872 WRITE(101,1072)
      RETURN
  873 WRITE(101,1073)
      RETURN
  874 WRITE(101,1074)
      RETURN
  875 WRITE(101,1075)
      RETURN
  876 WRITE(101,1076)
      RETURN
  877 WRITE(101,1077)
      RETURN
  878 WRITE(101,1078)
      RETURN
 1001 FORMAT(1X,
&     '"FORMAT"TAN SONRA "(" GELMELI')
 1002 FORMAT(1X,
&     '"READ"DEN SONRA "(" GELMELI')
 1003 FORMAT(1X,
&     '"WRITE"DAN SONRA "(" GELMELI')
 1004 FORMAT(1X,
&     'HER KOMUTUN ILK KARAKTERI BIR HARF OLMALI')
```

```
     'FONKSIYON ISMI SOL TARAFTA OLAMAZ')
1007 FORMAT(1X,
     'DEGISKEN TANITICI ALTI KARAKTERI ASAMAZ')
1008 FORMAT(1X,
     'GECERSIZ SAYI')
1009 FORMAT(1X,
     'GECERSIZ US')
1010 FORMAT(1X,
     'GECERSIZ KARAKTER')
1011 FORMAT(1X,
     'KOMUT SONUNDA ARTIK ISLEM ISARETI')
1012 FORMAT(1X,
     'FONKSIYON ISMININ ARDINDAN PARANTEZ ICINDE ARGUMAN VERILMELI')
1013 FORMAT(1X,
     '")"DEN SONRA ISLEM ISARETI VERILMELI')
1014 FORMAT(1X,
     '")"DEN ONCE ARTIK ISLEM ISARETI')
1015 FORMAT(1X,
     'GECERSIZ ISLEM ISARET DIZGESI')
1016 FORMAT(1X,
     'ATAMA KOMUTUNDA YALNIZ BIR "=" OLMALI')
1017 FORMAT(1X,
     'ISLENENDEN SONRA ISLEM ISARETI VERILMELI')
1018 FORMAT(1X,
     'TAMSAYI DEGISMEZ COK BUYUK')
1019 FORMAT(1X,
     'GERCEK SAYI DEGISMEZ COK BUYUK')
1020 FORMAT(1X,
     '* DEGISKEN SAKLANAN ALANDA TASMA VAR. "RESET" KULLAN')
1021 FORMAT(1X,
     'KOMUT ISARETI TAMSAYI OLMALI')
1022 FORMAT(1X,
     'DEVAM POZISYONU BOS OLMALI')
1023 FORMAT(1X,
     '* ISARET TABLOSUNDA TASMA VAR. "RESET" KULLAN')
 024 FORMAT(1X,
     'ISARET BOS KOMUTU TANIMLIYOR')
 025 FORMAT(1X,
     'DEGISKENDE HENUZ BIR DEGER YOK')
 026 FORMAT(1X,
     'KOMUT ISARETI DAHA ONCE TANIMLANMISTI')
 027 FORMAT(1X,
     'KOMUT ISARETI EN BUYUK TAMSAYIDAN DAHA BUYUK OLAMAZ')
 028 FORMAT(1X,
     'FORMAT KOMUTU ISARETLENMELI')
 029 FORMAT(1X)
 030 FORMAT(1X,
     'ALAN GENISLIGI TANIMLANMALI')
 031 FORMAT(1X,
     'ALAN GENISLIGI 80 KARAKTERI ASMAMALI')
 032 FORMAT(1X,
     'ONDALIK KESIR ALANIN GENISLIGI BELIRTILMELI')
 033 FORMAT(1X,
     '"."DAN SONRA ONDALIK KESIR GENISLIGI BELIRTILMELI')
 034 FORMAT(1X,
     'ALAN GENISLIGI ONDALIK KESIR GENISLIGINDEN DAHA KUCUK OLAMAZ')
 035 FORMAT(1X,
     'MANTIS GENISLIGI BELIRTILMELI')
 036 FORMAT(1X,
     '"."DAN SONRA MANTIS GENISLIGI BELIRTILMELI')
 037 FORMAT(1X,
     '* FORMAT TABLOSUNDA TASMA VAR. "RESET" KULLAN')
 038 FORMAT(1X,
```

```
 1040 FORMAT(1X,
&      'ONDALIK KESIR ALANININ GENISLIGI SIFIR OLAMAZ')
 1041 FORMAT(1X,
&      '"H"DEN ONCE DIZGI UZUNLUGU BELIRTILMELI')
 1042 FORMAT(1X,
&      'GECERSIZ FORMAT OGESI')
 1043 FORMAT(1X,
&      'OGELER ARASINDA "," VEYA "/" OLMALI')
 1044 FORMAT(1X,
&      'OGE ILE "(" ARASINDA AYIRICI OLMALI')
 1045 FORMAT(1X,
&      'FORMAT KOMUTU ")" ILE BITIRILMELI')
 1046 FORMAT(1X,
&      '"()" ICINDE OGE BELIRTILMEDI')
 1047 FORMAT(1X,
&      'HATALI "," KULLANIMI')
 1048 FORMAT(1X,
&      '")" VE OGE ARASINDA AYIRICI OLMALI')
 1049 FORMAT(1X,
&      '")"DAN SONRA HATALI OLARAK "," VAR')
 1050 FORMAT(1X,
&      'BIRDEN FAZLA "," ARKA ARKAYA GELMEMELI')
 1051 FORMAT(1X,
&      'BITIS BELIRTEN ")"DEN SONRA ARTIK KARAKTERLER VAR')
 1052 FORMAT(1X,
&      '"F" OGESINDE BELIRTILEN ALAN HATALI SONUCLAR VEREBILIR')
 1053 FORMAT(1X,
&      '"E" OGESINDE BELIRTILEN ALAN HATALI SONUCLAR VEREBILIR')
 1054 FORMAT(1X,
&      'HOLLERITH DIZGE SATIR SONUNU ASIYOR')
 1055 FORMAT(1X,
&      'DIZGI SONUNU BELIRLEYEN TIRNAK VERILMEDI')
 1056 FORMAT(1X,
&      'BOS KARAKTER DIZGESI KABUL EDILMIYOR')
 1057 FORMAT(1X,
&      'ALAN GENISLIGI SIFIRDAN BUYUK OLMALI')
 1058 FORMAT(1X,
&      'KANAL BELIRTEN RAKAM HATALI')
 1059 FORMAT(1X,
&      '"," EKSIK')
 1060 FORMAT(1X,
&      'GIRIS/CIKISTA KULLANILACAK FORMAT ISARETI HATALI')
 1061 FORMAT(1X,
&      '")" EKSIK')
 1062 FORMAT(1X,
&      'GECERSIZ KARAKTER')
 1063 FORMAT(1X,
&      'GECERSIZ KANAL KODU ( KANAL 5:READ, KANAL 6:WRITE )')
 1064 FORMAT(1X,
&      'KOMUT ISARETI EN BUYUK TAMSAYIDAN KUCUK OLMALI')
 1065 FORMAT(1X,
&      'KOMUT ISARETI DAHA ONCE ISLENEBILIR BIR KOMUTTA KULLANILDI'
 1066 FORMAT(1X,
&      '","DEN SONRA DEGISKEN GELMELI')
 1067 FORMAT(1X)
 1068 FORMAT(1X)
 1069 FORMAT(1X,
&      '** DEGISMEZ ALANINDA TASMA VAR. BU KO',
&                                    'MUTTAKI DEGISMEZ SAYISINI AZALT'
 1070 FORMAT(1X,
&      '")" EKSIK')
 1071 FORMAT(1X,
&      'BEKLENMEDIK ")"')
```

```
&       '"FLOAT" FONKSIYONUNUN ARGUMANI TAMSAYI OLMALI')
 1074 FORMAT(1X,
&       '"INT" FONKSIYONUNUN ARGUMANI GERCEK SAYI OLMALI')
 1075 FORMAT(1X,
&       '** KOMUT COK KARMASIK (YER KALMADI). LUTFEN SADELESTIR')
 1076 FORMAT(1X,
&       'DEYIMDE KARMA ARITMETIK KULLANILAMAZ')
 1077 FORMAT(1X,
&       '** ISLENEN ALANINDA TASMA VAR. LUTFEN DEYIMI SADELESTIR')
 1078 FORMAT(1X,
&       '** ISLEM ALANINDA TASMA VAR. LUTFEN DEYIMI SADELESTIR')
      END
C
C    ISLEM HATA MESAJLARININ IEROR DEGERINE GORE YAZILMASI
C
      SUBROUTINE ERREXC
      COMMON IC1T5(419),RC1T5(41)
      COMMON NCLASS,NVALUE,IEROR,MDCHK,ICTYP
      IT=(IEROR-1)/10
      IE=IEROR-IT*10
      IT=IT+1
      GO TO(100,210,320),IT
  100 GO TO(101,102,103,104,105,106,107,108,109,110),IE
  101 WRITE(101,1001)
      RETURN
  102 WRITE(101,1002)
      RETURN
  103 WRITE(101,1003)
      RETURN
  104 WRITE(101,1004)
      RETURN
  105 WRITE(101,1005)
      RETURN
  106 WRITE(101,1006)
      RETURN
  107 WRITE(101,1007)
      RETURN
  108 WRITE(101,1008)
      RETURN
  109 WRITE(101,1009)
      RETURN
  110 WRITE(101,1010)
      RETURN
  210 GO TO(211,212,213,214,215,216,217,218,219,220),IE
  211 WRITE(101,1011)
      RETURN
  212 WRITE(101,1012)
      RETURN
  213 WRITE(101,1013)
      RETURN
  214 WRITE(101,1014)
      RETURN
  215 WRITE(101,1015)
      RETURN
  216 WRITE(101,1016)
      RETURN
  217 WRITE(101,1017)
      RETURN
  218 WRITE(101,1018)
      RETURN
  219 WRITE(101,1019)
      RETURN
  220 WRITE(101,1020)
```

```
  320 GO TO(321,322,323,324,325),IE
  321 WRITE(101,1021)
      RETURN
  322 WRITE(101,1022)
      RETURN
  323 WRITE(101,1023)
      RETURN
  324 WRITE(101,1024)
      RETURN
  325 WRITE(101,1025)
      RETURN
 1001 FORMAT(1X,
&     'TAMSAYI UST TASMA')
 1002 FORMAT(1X,
&     'US ALIRKEN UST TASMA')
 1003 FORMAT(1X,
&     'TOPLARKEN UST TASMA')
 1004 FORMAT(1X,
&     'CIKARIRKEN UST TASMA')
 1005 FORMAT(1X,
&     'CARPARKEN UST TASMA')
 1006 FORMAT(1X,
&     'BOLERKEN UST TASMA')
 1007 FORMAT(1X,
&     'EKSI SAYILARIN "ALOG"U ALINAMAZ')
 1008 FORMAT(1X,
&     'EKSI SAYILARIN "ALOG10"U ALINAMAZ')
 1009 FORMAT(1X,
&     'EKSI SAYILARIN KARE KOKU ALINAMAZ')
 1010 FORMAT(1X,
&     '"EXP" ALIRKEN UST TASMA')
 1011 FORMAT(1X,
&     '"IABS" ALIRKEN TAM SAYI UST TASMA')
 1012 FORMAT(1X,
&     '"INT" ALIRKEN TAM SAYI UST TASMA')
 1013 FORMAT(1X,
&     'KARMASIK CEVAPLI US ALMA ISLEMI')
 1014 FORMAT(1X,
&     '"WRITE" SIRASINDA YETERSIZ ALAN GENISLIGI')
 1015 FORMAT(1X,
&     '"READ/WRITE" ESNASINDA UYUSMAYAN DEGISKEN/OGE TIPLERI')
 1016 FORMAT(1X,
&     '"READ/WRITE" ESNASINDA SATIRDAKI AZAMI KARAKTER SAYISI ',
&     '(80) ASILDI')
 1017 FORMAT(1X,
&     'SONSUZ FORMAT DONGUSU')
 1018 FORMAT(1X,
&     'I-DESENI ILE OKUNACAK ALANDA GECERSIZ KARAKTER BULUNDU')
 1019 FORMAT(1X,
&     'VERI CIZGISINDEKI SAYI ISARETI HATALI')
 1020 FORMAT(1X,
&     'F-DESENI ILE OKUNACAK ALANDA GECERSIZ KARAKTER BULUNDU')
 1021 FORMAT(1X,
&     'VERI CIZGISINDE HATALI "." KULLANIMI')
 1022 FORMAT(1X,
&     'E-DESENI ILE OKUNACAK ALANDA GECERSIZ KARAKTER BULUNDU')
 1023 FORMAT(1X,
&     'VERI CIZGISINDE HATALI "E" KULLANIMI')
 1024 FORMAT(1X,
&     'VERI CIZGISINDE HATALI "E" KULLANIMI (SAYI COK BUYUK)')
 1025 FORMAT(1X,
&     'TAMSAYI VERI COK BUYUK')
```

EK.B

FCN-F HATA MESAJLARI

EYICI HATA MESAJLARI

	MESAJ
	"FORMAT"TAN SONRA "(" GELMELI
	"READ"DEN SONRA "(" GELMELI
	"WRITE"DAN SONRA "(" GELMELI
	HER KOMUTUN ILK KARAKTERI BIR HARF OLMALI
	DEGISKEN TANITICIDAN SONRA "=" OLMALI
	FONKSIYON ISMI SOL TARAFTA OLAMAZ
	DEGISKEN TANITICI ALTI KARAKTERI ASAMAZ
	GECERSIZ SAYI
	GECERSIZ US
	GECERSIZ KARAKTER
	KOMUT SONUNDA ARTIK ISLEM ISARETI
	FONKSIYON ISMININ ARDINDAN PARANTEZ ICINDE ARGUMAN VERILMELI
	")"DEN SONRA ISLEM ISARETI VERILMELI
	")"DEN ONCE ARTIK ISLEM ISARETI
	GECERSIZ ISLEM ISARET DIZGESI
	ATAMA KOMUTUNDA YALNIZ BIR "=" OLMALI
	ISLENENDEN SONRA ISLEM ISARETI VERILMELI
	TAMSAYI DEGISMEZ COK BUYUK
	GERCEK SAYI DEGISMEZ COK BUYUK
*	DEGISKEN SAKLANAN ALANDA TASMA VAR. "RESET" KULLAN
	KOMUT ISARETI TAMSAYI OLMALI
	DEVAM POZISYONU BOS OLMALI
*	ISARET TABLOSUNDA TASMA VAR. "RESET" KULLAN
	ISARET BOS KOMUTU TANIMLIYOR
	DEGISKENDE HENUZ BIR DEGER YOK
	KOMUT ISARETI DAHA ONCE TANIMLANMISTI
	KOMUT ISARETI EN BUYUK TAMSAYIDAN DAHA BUYUK OLAMAZ
	FORMAT KOMUTU ISARETLENMELI
	ALAN GENISLIGI TANIMLANMALI
	ALAN GENISLIGI 80 KARAKTERI ASMAMALI
	ONDALIK KESIR ALANIN GENISLIGI BELIRTILMELI
	"."DAN SONRA ONDALIK KESIR GENISLIGI BELIRTILMELI
	ALAN GENISLIGI ONDALIK KESIR GENISLIGI'DEN DAHA KUCUK OLAMAZ
	MANTIS GENISLIGI BELIRTILMELI
	"."DAN SONRA MANTIS GENISLIGI BELIRTILMELI
*	FORMAT TABLOSUNDA TASMA VAR. "RESET" KULLAN
	"/" OGESINDEN ONCE TEKRAR SAYISI BELIRTILEMEZ

```
   ONDALIK KESIR ALANININ GENISLIGI SIFIR OLAMAZ
   "H"DEN ONCE DIZGI UZUNLUGU BELIRTILMELI
   GECERSIZ FORMAT OGESI
   OGELER ARASINDA "," VEYA "/" OLMALI
   OGE ILE "(" ARASINDA AYIRICI OLMALI
   FORMAT KOMUTU ")" ILE BITIRILMELI
   "()" ICINDE OGE BELIRTILMEDI
   HATALI "," KULLANIMI
   ")" VE OGE ARASINDA AYIRICI OLMALI
   ")"DAN SONRA HATALI OLARAK "," VAR
   BIRDEN FAZLA "," ARKA ARKAYA GELMEMELI
   BITIS BELIRTEN ")"DEN SONRA ARTIK KARAKTERLER VAR
   "F" OGESINDE BELIRTILEN ALAN HATALI SONUCLAR VEREBILIR
   "E" OGESINDE BELIRTILEN ALAN HATALI SONUCLAR VEREBILIR
   HOLLERITH DIZGE SATIR SONUNU ASIYOR
   DIZGI SONUNU BELIRLEYEN TIRNAK VERILMEDI
   BOS KARAKTER DIZGESI KABUL EDILMIYOR
   ALAN GENISLIGI SIFIRDAN BUYUK OLMALI
   KANAL BELIRTEN RAKAM HATALI
   "," EKSIK
   GIRIS/CIKISTA KULLANILACAK FORMAT ISARETI HATALI
   ")" EKSIK
   GECERSIZ KARAKTER
   GECERSIZ KANAL KODU ( KANAL 5:READ, KANAL 6:WRITE )
   KOMUT ISARETI EN BUYUK TAMSAYIDAN KUCUK OLMALI
   KOMUT ISARETI DAHA ONCE ISLENEBILIR BIR KOMUTTA KULLANILDI
   ","DEN SONRA DEGISKEN GELMELI


** DEGISMEZ ALANINDA TASMA VAR.BU KOMUTTAKI DEGISMEZ SAYISINI AZAL
   ")" EKSIK
   BEKLENMEDIK ")"
   FAZLA ")"
   "FLOAT" FONKSIYONUNUN ARGUMANI TAMSAYI OLMALI
   "INT" FONKSIYONUNUN ARGUMANI GERCEK SAYI OLMALI
** KOMUT COK KARMASIK (YER KALMADI). LUTFEN SADELESTIR
   DEYIMDE KARMA ARITMETIK KULLANILAMAZ
** ISLENEN ALANINDA TASMA VAR. LUTFEN DEYIMI SADELESTIR
** ISLEM ALANINDA TASMA VAR. LUTFEN DEYIMI SADELESTIR
```

M HATA MESAJLARI

MESAJ
TAMSAYI UST TASMA
US ALIRKEN UST TASMA
TOPLARKEN UST TASMA
CIKARIRKEN UST TASMA
CARPARKEN UST TASMA
BOLERKEN UST TASMA
EKSI SAYILARIN "ALOG"U ALINAMAZ
EKSI SAYILARIN "ALOG10"U ALINAMAZ
EKSI SAYILARIN KARE KOKU ALINAMAZ
"EXP" ALIRKEN UST TASMA
"IABS" ALIRKEN TAM SAYI UST TASMA
"INT" ALIRKEN TAM SAYI UST TASMA
KARMASIK CEVAPLI US ALMA ISLEMI
"WRITE" SIRASINDA YETERSIZ ALAN GENISLIGI
"READ/WRITE" ESNASINDA UYUSMAYAN DEGISKEN/OGE TIPLERI
"READ/WRITE" ESNASINDA SATIRDAKI AZAMI KARAKTER SAYISI (80) ASILD
SONSUZ FORMAT DONGUSU
I-DESENI ILE OKUNACAK ALANDA GECERSIZ KARAKTER BULUNDU
VERI CIZGISINDEKI SAYI ISARETI HATALI
F-DESENI ILE OKUNACAK ALANDA GECERSIZ KARAKTER BULUNDU
VERI CIZGISINDE HATALI "." KULLANIMI
E-DESENI ILE OKUNACAK ALANDA GECERSIZ KARAKTER BULUNDU
VERI CIZGISINDE HATALI "E" KULLANIMI
VERI CIZGISINDE HATALI "E" KULLANIMI (SAYI COK BUYUK)
TAMSAYI VERI COK BUYUK

KAYNAKLAR

AGAJANIAN 74: A bibliography of plasma display panels, Proc. Soc. Inf. Display, Vol. 15, 1974.

BALMAN, SMITH 77: Organisation of a Computer Assisted Teaching Laboratory 13. DECUS European Symposium, 1977.

BALMAN 78a: Teach Yourself FORTRAN, Computer Assisted Teaching Unit, Queen Mary College, London 1978.

BALMAN 78b: Logging and Accounting for RSX-11M, Computer Assisted Teaching Unit, Queen Mary College, London 1978.

BALMAN 78c: Modifications to RSX-11M Operating System, Computer Assisted Teaching Unit, Queen Mary College, LOndon 1978.

BALMAN 78d: Input/Output library routines for teaching programs, Computer Assisted Teaching Unit, Queen Mary College, LOndon 1978.

BALMAN 78e: MMP-Link to ICL 1905, Computer Assisted Teaching Unit, Queen Mary College, London 1978.

BALMAN 78f: FCN Programming notes, Computer Assisted Teaching Unit, Queen Mary College, London 1978.

BALMAN 79: Implementation Techniques for interactive CAL programs, Int. J. of Computers and Education, Vol.5, No. 1, 1980.

BALMAN 80a: Computer Assisted Teaching of FORTRAN, Int. J. of Computers and Education, Vol. 6, No.2, 1981.

BALMAN 80b: Eğitimde Bilgisayar, Boğaziçi Üniversitesi Dergisi, 1980.

BALMAN 80c: Compiler Design Techniques, Boğaziçi Üniversitesi Yayınları, 1980.

BCS 80: Strengths and weaknesses of BASIC, BCS Schools Committee, Programming Languages Working Party, 1980.

BERTHAUD, GRIFFITHS 73: Incremental compilation and conversational interpretation, Ann. Review in Automatic Programming, Vol. 7, 1973.

BITZER 77: The wide world of Computer Based Education, Advances in Computers Vol. 15, 1977.

BROADHURST 74: Contribution of Computers and CCTV to teaching and learning, Conference on Frontiers of Education, University College, LOndon 1974.

CAI 70: Computers and Educatıon. An international bibliography on Computers in Education, Int. Fed. Inf. Process., Amsterdam 1970.

CAI 73: Computer Aided Instruction in Japan, ACM Sigcue Bulletin, Vol.7,1973.

CET 75: Two Years on: The National Development Programme in Computer Assisted Learning, Ministry of Education, LOndon 1975.

CET 76: Computer Assisted Learning in the United Kingdom, Ministry of Education, London 1976.

COLLINS 77: Process in acquiring knowledge, R.C.Anderson (ed.), Schooling and Acquisition of Knowledge, 1977.

DUGDALE, KIBBEY 75: The Fractions Curriculum-PLATO elementary School Mathematics Project, Computer Based Education Research Laboratory, University of Illinois, 1975.

EDMONDS, SMITH, STAFFORD 75: Transferability-Why, What and Where, Symposium on CAL in Science and Engineering, Oxford 1975.

ESPCAL 77: Engineering Science Project in Computer Assisted Learning: Final Review, Computer Assisted Teaching Unit, Queen Mary College, London 1977.

FIELDEN 74: The Financial Evaluation of Computer Assisted Learning Projects, Int. J. of Math. Educ. Sci. and Technol., Vol. 5, 1974.

GLENNIE 60: On the syntax machine and the construction of a universal compiler Carnegie-Mellon University, Pittsburgh 1960.

GROS 78: Interpreters vs. Compilers, Interface Age 1978.

HOLT 77: A system for teaching computer programming, CACM, Vol. 20, 1977.

JOHNSON, BITZER, SLOTTOW 71: The device characteristic of the Plasma Display Element, IEEE Trans. Electron Devices, Vol. 18, 1971.

MC. INTOSH 74: Evaluation of multi-media education systems-some problems, Brit.J. of Educ. Technol., Vol.5, 1974.

MONTANELLI 77: Using CAI to teach introductory computer programming, ACM Bull. 11, 1977.

MORRISON 75: Design and use of the BAC Experimental manoevre and attack simulator, Conf. on Computer Simulation, London 1975.

MORRISON 77: The BAC air combat simulator, 13. DECUS Europe Symposium, 1977.

NCET 69a: Computer Based Learning Systems, A Feasibility study, National Council for Educational Technology, Lendon 1969.

NCET 69b: Computer Based Learning: Aprogramme for Action, National Council for Educational Technology, LOndon 1969.

PAPERT 71a: A computer laboratory for elementary schools, LOGO Memo No.1, M.I.T. 1971.

PAPERT 71b: Teaching children thinking, LOGO Memo No. 2, M.I.T. 1971.

PAPERT 71c: Twenty things to do with a computer, LOGO Memo No.3, M.I.T. 1971.

PAPERT 72: Teaching children to be mathematicians versus teaching about mathematics, Int. J.Math. Educ. Sci. Vol. 3, 1972.

SMITH 75: CAL in Engineering, Hooper (ed), Computer Assisted Learning in the U.K, Council for Educational Tecnology, London 1975.

SMITH 76a: A CAL Package in Reactor Kinetics, J.Inst. Nuclear Engineering, Vol. 17, 1976.