

YILDIZ TEKNİK ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayarlar Arası İletişim

YÜKSEK LİSANS TEZİ

Ceydet Dengi

1990

YILDIZ TEKNİK ÜNİVERSİTESİ
KÜTÜPHANE DOKÜMANTASYON
DAİRE BAŞKANLIĞI

Kot	: R 368 55
Alındığı Yer	: Fen Bil. Ens.
Tarih	: 13.4.95
Fatura	:
Fiyatı	: 35.000,-
Ayniyat No	: 1-4
Kayıt No	: 51033
UDC	:
Ek	:

YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YILDIZ ÜNİVERSİTESİ

D.D. No 49893

BİLGİSAYARLAR ARASI İLETİŞİM



YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜH. E. CEVDET DENGİ

TEZ YÖNETİCİSİ
DOÇ.DR. KIRKOR HARUTUNYAN

İSTANBUL , 1990

İÇİNDEKİLER

<u>Başlık</u>	<u>Sayfa numarası</u>
Özet	
İçindekiler	
Abstract	
Giriş	1
OSI Temel Referans Modeli	3
IBM System/370 mimarisi	7
CTCA	13
VM/SP İşletim Sistemi	15
IUCV	23
Sistemin Çalışma Şekli	33
Kaynak Programlar	39
Makrolar	67
Kaynaklar	83
Özgeçmiş	84

ÖZET

Bilgisayar kullanımının alabildiğine yaygınlaştığı günümüzde bilgisayar ağları giderek daha önemli bir yer almaya başlamıştır. İlk zamanlarında sadece büyük işlem zamanı gerektiren hesapları yapmak için bir araç olarak kullanılan bilgisayarlar, sonraları bilgi depolama ve işleme alanında da yaygın şekilde kullanılmaya başlanmışlardır. Günümüzde bankacılık gibi pek çok sektörde bilgisayar kullanımı kaçınılmazdır.

Bilgisayarların bir ağ oluşturacak şekilde biribirlerine bağlanması yöntemi yeni bir yöntem sayılmaz. Bilgisayar ağları alanında günümüze kadar gerek donanım ve gerekse yazılım olarak pekçok ürün ortaya çıkmıştır. Ağlar büyüdükçe ve sayıları arttıkça bu değişik özellikteki ağların bir arada kullanılması büyük bir sorun olarak belirmiştir.

İşte bu sorun büyük bilgisayar üreticilerini ve uluslararası standart kuruluşlarını genel bir standart oluşturmaya itmiştir. Günümüzde belli başlı bazı standartların varlığından söz edilebilmekteyse de tam bir standartlaşma henüz sağlanamamıştır. Oluşturulan standartlardan en yaygınlarından biri ISO (International Organization of Standarts) tarafından geliştirilen ve 1983 yılında ilan edilen OSI (Open System Interconnection) dur.

Projenin amacı IBM System/370 mimarisindeki bilgisayarlar ve VM/SP işletim sistemi altında çalışacak, OSI referans modelinin katmanlı yapısına sahip bir bilgisayar iletişim ağı prototipinin ortaya çıkarılmasıdır.

ABSTRACT

Nowadays, use of computers and also computer networks has become very important. Computers used as a device to do big amount of calculations in their early years started to be used for data processing purposes frequently. In some sectors such as banking to avoid using computers is not possible anymore.

The method of connecting computers to built a network cannot be mentioned as new. Many software hardware products have been developed. As computer networks got larger and number of networks increased, the problem of using those different type of computer networks all together has appeared as a big problem.

This problem, forced big computer vendors and international standart organizations to built a general standart. Now, although some standarts can be mentioned still a complete standardization could not be established. One of the well known standarts have been developed is the OSI (Open System Interconnection) developed and published in 1983 by ISO (International Organization of Standarts).

The aim of this project is to develop a computer communication network prototype which has the layered structure of OSI reference model and runs on IBM System/370 architecture computers under VM/SP operating system.

GİRİŞ

İçinde bulunduğumuz çağa bir isim verilmesi düşünüldüğünde akla ilk gelecek isimlerden ikisi şüphesiz 'İletişim çağı' ve 'Bilgisayar çağı' dır. Yirminci yüzyılda bilgisayar ve iletişim sistemleri konusunda yapılan gelişme yaşam tarzımızı ve kültürümüzü derinden etkilemiştir. Bilgisayarların giderek iletişim teknolojisinin vazgeçilmez bir parçası haline gelmesi ve bilgisayarların kendilerinden son derece etkin bir iletişim aracı olarak yararlanılmaya başlamasından sonra bilgisayar ve iletişim sözcükleri biribirinden ayrılmaz olmuştur.

Bilgisayarlar iletişim sistemleri bilgisayarların ilk zamanlarından beri var olmakla beraber o zamandan beri gerek dijital bilgi transferi konusunda sağlanan ilerleme gerekse bilgisayar ağlarının bilinen telefon, telegraf, telex yada posta gibi yöntemlerden çok daha hızlı ve güvenilir olduğunun ortaya çıkması bilgisayar ağlarının gerek boyut gerekse sayı olarak çığ gibi artmasına sebep olmuştur. Günümüzde bilgi iletişiminin olduğu her alanda bilgisayar ağları kullanılabilmektedir. Bunlara örnek olarak Üniversiteler arası akademik ve araştırma amaçlı bilgisayar ağları (Avrupada EARN, Amarikada BITNET, ARPANET, Kanadada BITNORTH, İngilterede JNET v.b.), turizm şirketlerinin, havayolu şirketlerinin, bankaların kullandığı uluslararası bilgisayar ağları, birçok ülkede yaygın olarak kullanılan electronk posta servisleri, bilgi bankaları, askeri amaçlı bilgisayar ağları ve daha pekçok irili ufaklı örnekler verilebilir.

Bilgisayar ağlarına olan gereksinim önceleri bilgisayar üreticilerinin ayrı ayrı yaptıkları çalışmaların ürünleri olan yazılım ve donanımlar ile karşılanıyordu. Bilgisayar

ağları gerek amaç gerekse yapı olarak çeşitlendikçe bu ağların gerektiğinde biribirlerine bağlanmaları büyük problemler yaratmaya başladı. Özellikle çok değişik marka ve büyüklükteki bilgisayarları bir araya getirmesi düşünülen ağların ortaya çıkması bu problemi daha da içinden çıkılmaz bir hale getirdi. Bütün bu gelişmeler bilgisayar üreticisi şirketleri ve uluslararası kuruluşları bilgisayar ağları konusunda bazı standartlar belirlemeye zorladı. Her ne kadar tek bir standart üzerinde birleşmek bugün için imkansız görünüyorsa da çok yaygın olarak kullanılan bazı standartların varlıkları da yadsınamaz.

Bilgisayar ağları konusunda ortaya çıkan standartlardan en bilinenlerinden biri ISO (International Organization of Standards) tarafından hazırlanan ve 1983 yılında yayınlanan OSI (Open System Interconnection) dur. Bu sistem bilgisayar ağlarını yedi ayrı katman olarak tarif etmektedir. Katmanların kullanıldığı bu yaklaşım son derece karmaşık sistemler olan bilgisayar ağlarının tarifini bir ölçüde kolaylaştırmıştır.

Bu projenin konusu olan bilgisayar ağı sistemi ticari bir ürün olmaya aday değildir. Ticari bir bilgisayar ağı sisteminin gerçekleşmesi bir yüksek lisans tezinin boyutlarının dışına taşan büyük bir grup çalışması ve zaman gerektirmektedir. Bu projenin amacı daha çok, katmanlı yapıya sahip basit bir bilgisayar ağının tasarlanmasıdır. Bu ağ daha sonra bilgisayar ağları konusundaki diğer araştırma çalışmaları için bir deney ortamı olarak kullanılabilir.

Bir uygulama OSI ortamına eriştiği yoldur. Bu katmanın işlevi, uygulamaların OSI ortamı arasında bir pencere görevi yapmaktır. Temel fonksiyonları ve dağıtık uygulamalar (Distributed Applications) için kullanışlı mekanizmalar içerir. Bu katman örnek olarak file sistemi, elektronik

OSI Temel Referans Modeli :

OSI temel referans modeli (Open System Interconnection Basic Reference Model), ISO (The International Organization for Standardization) nun bilgisayarlar arası iletişimde bir standart getirme çabalarının ürünü olarak ortaya çıkmış ve 1983 yılında yayınlanmış bir modeldir. Bu model bir ürünün tarifi değildir. Zaten bir iletişim sistemi kurmaya yetecek kadar ayrıntılı bilgi de içermez. Bu modelin amacı daha sonraki standardizasyon çalışmaları için bir temel oluşturmaktır.

OSI Temel Referans Modelinin Yapısı :

OSI temel referans modelinin yapısal tekniği katmanlamadır. Bu tekniğe göre her açık sistem mantıksal olarak sıralanmış alt sistemlerden oluşmuştur. Komşu alt sistemler ortak sınırları yoluyla haberleşirler. Bu yapıda her katman kendisinden üst düzeydeki katmana servis verir ve iki açık sistemin aynı seviyeli katmanları birbirleriyle haberleşirler. OSI temel referans modelinde yedi adet katman bulunur. Bunlar İngilizce isimleriyle, Application Layer, Session Layer, Transport Layer, Network Layer, Data Link Layer ve Physical Layer dır.

Application Layer :

Bu katman, referans modelinin en üst katmanını oluşturur. Bir uygulamanın OSI ortamına eriştiği yoldur. Bu katmanın işlevi, uygulamalarla OSI ortamı arasında bir pencere görevi yapmaktır. Yönetim fonksiyonları ve dağıtık uygulamalar (Distributed Applications) için kullanışlı mekanizmalar içerir. Bu katmana örnek olarak file aktarımı, elektronik

mektup gibi servisler verilebilir.

Presentation Layer : Ortak entity lerin arasında kayıpsız ve
veri transferi sağlar. Boylaca Transport Layer'dir.

Bu katman, uygulamalar arasında aktarılan bilginin
sentaksından sorumludur. Uygulamalar arasında kullanılacak
sentaksı belirler ve gerekli sentaks dönüşümlerini yapar.
Uygulamalar herhangi bir sentaks kullanabilirler.
Presentation Layer, bu sentakslarla ortak sentaks arasındaki
dönüşümü üstlenir. Bu katmana örnek olarak şifreleme ve
virtual terminal protokolleri verilebilir.

Session Layer : Fiksel katmanın görevleri arasında sağlar.
Katkıların kabulünü, dışa aktarılmasını, diğ. katmanların aktif

Bu katmanın görevi, karşılıklı presentation entity lerin
birlikte çalışabilmeleri için gerekli organizasyonu ve
senkronizasyonu sağlamaktır. Bunu yapabilmek için session
layer, presentation entity ler arasındaki bağlantıyı ve
düzgün veri akışını sağlar. Sağlanan bağlantı isteğe göre
simplex, half duplex yada full duplex olabilir. Ayrıca bu
katman hataların düzeltilmesi ile de uğraşır.

Bu katman, kayıpsız bilginin fiziksel ortama taşınmasını
Transport Layer : Sorumludur. Bu katmana örnek olarak kayıpsız
protokoller olan RS-232 ve RS-432 verilebilir.

Bu katman, verinin hatalardan arınmış, doğru sırada,
kayıpsız ve tekrarsız ulaştırılmasını garanti eder. Aynı
zamanda bilginin ekonomik şekilde taşınması ile de
ilgilidir. Transport Layer sayesinde onun üzerindeki
katmanlar veri iletişiminin ne şekilde yapıldığından
bağımsız çalışabilirler.

Network Layer :

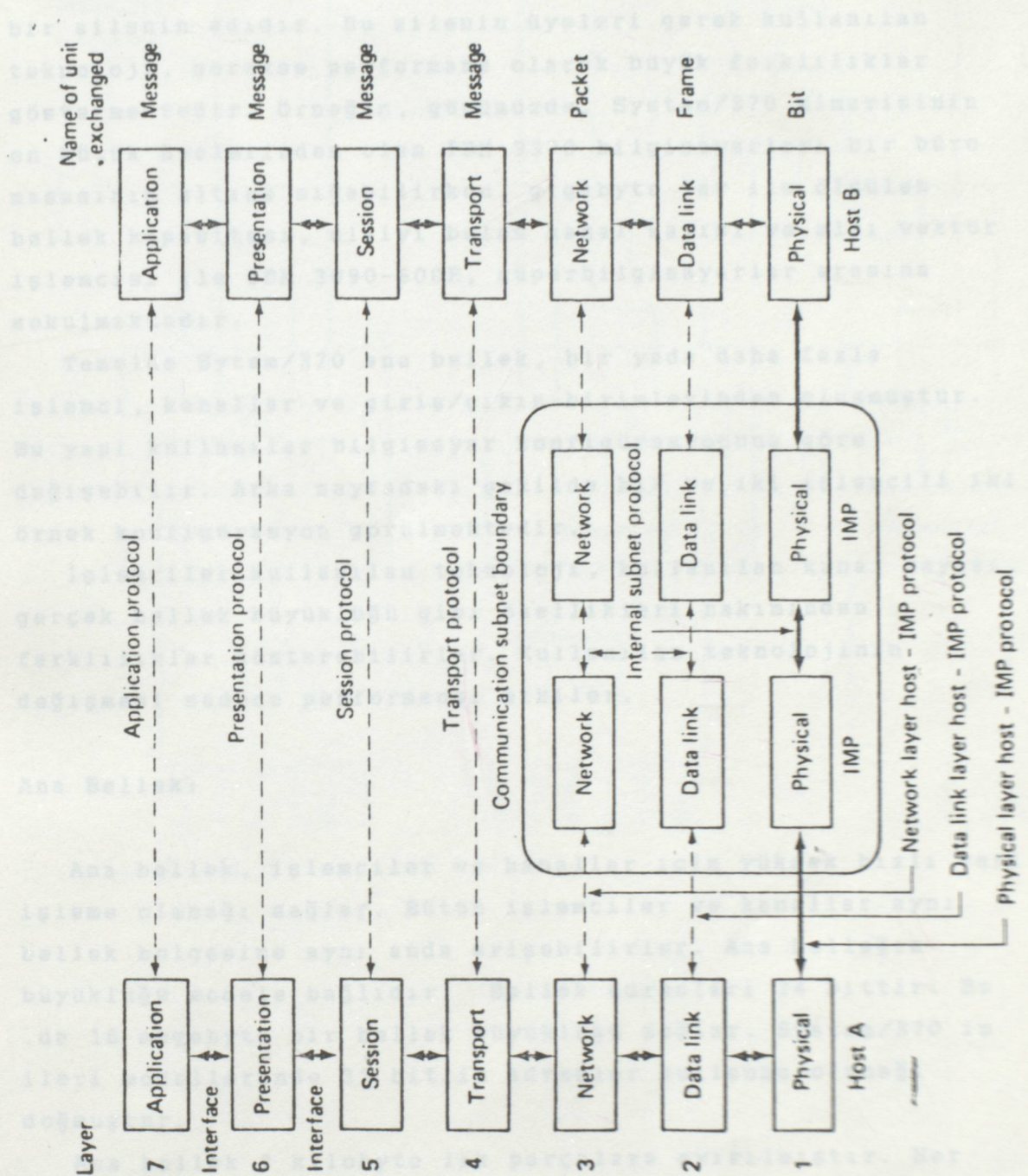
Bu katman, transport entity lerin arasında saydam bir veri transferi sağlar. Böylece transport layer ın sistemlerin bağlanmasında kullanılan veri transferi yöntemlerini bilmesi gerekmez. Bu katman, iletişim ağının topolojisi hakkında detaylı bilgiye sahiptir. Verilerin ağdaki düğümlerin üzerinden aktarılması bu katmanın sorumluluğundadır.

Data Link Layer :

Bu katman, fiziksel katmanın güvenilir olmasını sağlar. Hataların bulunmaları, düzeltilmeleri, linklerin aktive edilmeleri bu katmanın sunduğu hizmetler arasındadır. Data link layer üst katmanlara hatalardan arınmış bilgi ulaştırmakla yükümlüdür. Bu katmana örnek olarak HDLC ve BSC gibi protokoller verilebilir.

Physical Layer :

Bu katman, sayısal bilginin fiziksel ortam üzerinden aktarılması ile görevlidir. Bu katmana örnek olarak tanınmış protokoller olan RS-232 ve RS-422 verilebilir.



IBM System/370 mimarisi:

IBM System/370 IBM tarafından 1970 ten beri üretilen değişik büyüklüklerde genel amaçlı bilgisayarlardan oluşan bir ailenin adıdır. Bu ailenin üyeleri gerek kullanılan teknoloji, gerekse performans olarak büyük farklılıklar göstermektedir. Örneğin, günümüzde, System/370 mimarisinin en küçük üyelerinden olan IBM 9370 bilgisayarları bir büro masasının altına sığabilirken, gigabyte lar ile ölçülen bellek kapasitesi, elliye bulan kanal sayısı ve altı vektör işlemcisi ile IBM 3090-600E, süperbilgisayarlar arasına sokulmaktadır.

Temelde Sytem/370 ana bellek, bir yada daha fazla işlemci, kanallar ve giriş/çıkış birimlerinden oluşmuştur. Bu yapı kullanılar bilgisayar konfigürasyonuna göre değişebilir. Arka sayfadaki şekilde bir ve iki işlemcili iki örnek konfigürasyon görülmektedir.

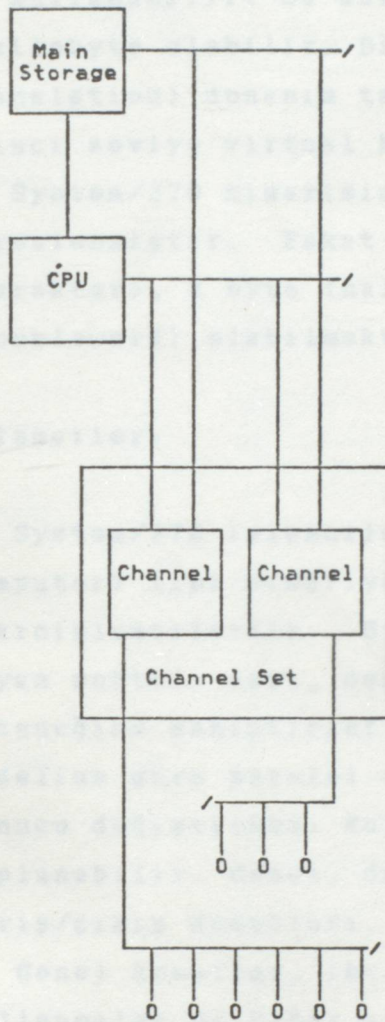
İşlemciler kullanılan teknoloji, kullanılan kanal sayısı, gerçek bellek büyüklüğü gibi özellikleri bakımından farklılıklar gösterebilirler. Kullanılan teknolojinin değişmesi sadece performansı etkiler.

Ana Bellek:

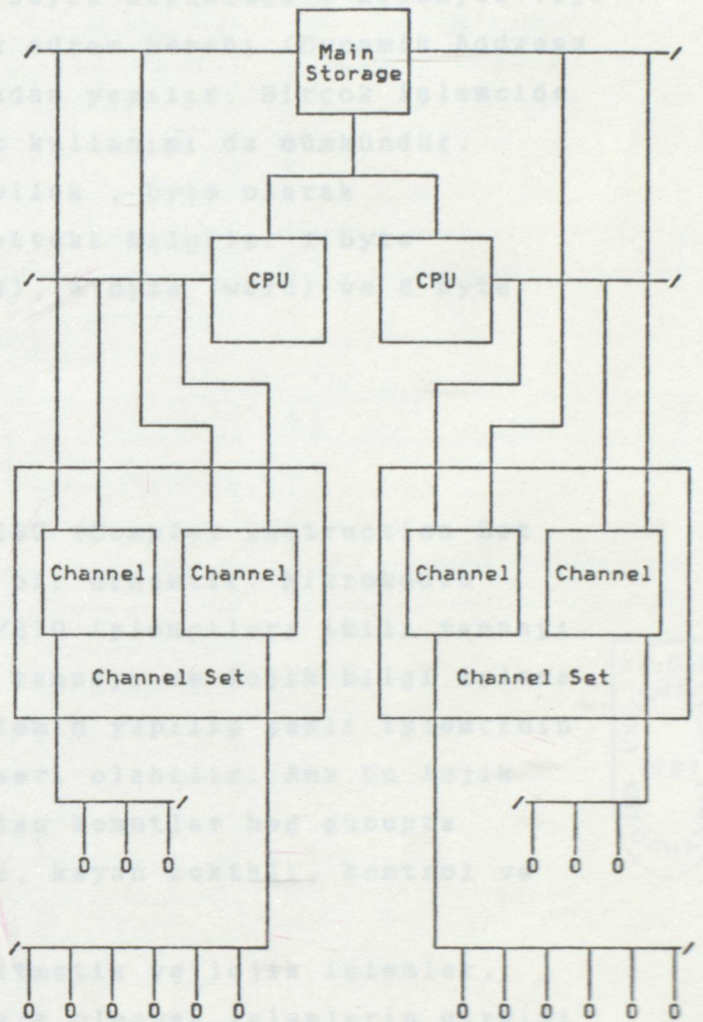
Ana bellek, işlemciler ve kanallar için yüksek hızlı veri işleme olanağı sağlar. Bütün işlemciler ve kanallar aynı bellek bölgesine aynı anda erişebilirler. Ana belleğin büyüklüğü modele bağlıdır. Bellek adresleri 24 bittir. Bu .da 16 megebyte bir bellek büyüklüğü sağlar. System/370 in ileri modellerinde 32 bitlik adresler kullanma olanağı doğmuştur.

Ana bellek 2 kilobyte lık parçalara ayrılmıştır. Her blok bir koruma anahtarına sahiptir. Bu anahtarlar

Single-CPU Configuration



2-CPU Multiprocessing Configuration



Logical Structure

kullanılarak programların yalnızca belli bellek bölgelerine erişebilmeleri sağlanır. Ayrıca System/370, virtual adresler de kullanılabilir. Bu durumda sayfa büyüklüğü 2 kilobyte veya 4 kilobyte olabilir. Dinamik adres hesabı (Dynamik Address Translation) donanım tarafından yapılır. Birçok işlemcide ikinci seviye virtual bellek kullanımı da mümkündür.

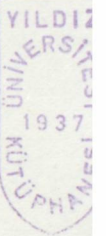
System/370 mimarisinde bellek , byte olarak adreslenmiştir. Fakat bellekteki bilgiler 1 byte (karakter), 2 byte (halfword), 4 byte (word) ve 8 byte (doubleword) olabilmektedir.

İşlemciler:

System/370 işlemcileri CISC (Complex Instruction Set Computer) tipi mimariye iyi bir örnektir. Mikrokodlu mikroişlemcilerdir. System/370 işlemcileri ikili tamsayı, kayan noktalı sayı, desimal tamsayı ve lojik bilgi işleme yeteneğine sahiptirler. İşlemin yapılış şekli işlemcinin modeline göre paralel veya seri olabilir. Ama bu lojik sonucu değiştirmez. Kullanılan komutlar beş grupta toplanabilir. Genel, desimal, kayan noktalı, kontrol ve giriş/çıkış komutları.

Genel komutlar, ikili aritmatik ve lojik işlemler, dallanmalar ve diğer aritmatik olmayan işlemlerin girdiği gruptur. Desimal komutlar, desimal formatlı veriler üzerinde kullanılır. Desimal format özellikle sayıların sık sık EBCDIC koduna çevirildiği durumlarda tercih edilirler. Kayan noktalı işlemler ise daha çok mühendislik uygulamalarında kullanılan kayan noktalı sayılar için kullanılırlar. Yetki isteyen kontrol komutları ve giriş/çıkış komutları sadece işlemci, yönetici durumunda iken icra edilebilirler.

System/370 mimarisinde işlemciler, değişik amaçlara



R Field	Register Number	Control Registers	General Registers	Floating-Point Registers
		←32 bits→	←32 bits→	←64 bits→
0000	0			
0001	1			
0010	2			
0011	3			
0100	4			
0101	5			
0110	6			
0111	7			
1000	8			
1001	9			
1010	10			
1011	11			
1100	12			
1101	13			
1110	14			
1111	15			

General, Floating-Point, and Control Registers

hizmet eden saklayıcı guruplarına sahiptir. Bunlar:

Programın durumunu saklayan PSW (program Status Word), genel saklayıcılar, kontrol saklayıcıları, saat karşılaştırıcısı ve işlemci saati (TOD) için kullanılan saklayıcılardır.

Genel saklayıcılar, 32 bit uzunluktadır ve 16 adettir. Bu saklayıcılar veri ve adres taşımak için kullanılabilirler. System/370 de adresleme görelidir. yani bir bellek bölgesini adresliyeabilmek için bir saklayıcının baz olarak kullanılması gereklidir. Bazı komutlarda genel saklayıcıların sadece düşük anlamlı 16 biti kullanılır. Ayrıca arka arkaya iki saylayıcıyı birlikte 64 bitlik bir saklayıcı olarak kullanan komutlar da mevcuttur.

Kayan noktalı saklayıcılar, herbiri 64 bit uzunluğunda olmak üzere dört tanedir. Bu saklayıcılar, 32 bitlik sayılar için de kullanılabilirler. Bu durumda saklayıcıların düşük anlamlı 32 biti kullanılmaz. Arka arkaya iki kayan noktalı saklayıcıyı bir arada 128 bitlik bir saklayıcı olarak kullanan komutlar da vardır.

System/370 herbiri 32 bitlik 16 kontrol saklayıcısına sahiptir. Bu saklayıcılar bilgisayar sisteminin değişik kısımlarının (Örneğin 6. kanaldan sonraki kanalların kesmelerini maskelenmesi) kontrolünde kullanılırlar.

Giriş/Çıkış:

System/370 te giriş/çıkış işlemleri kanallar tarafından yürütülür. Kanallar, giriş/çıkış birimleri ile ana bellek arasında veri taşımak üzere tasarlanmış özel işlemcilerdir. Kanallar, kanal komutları ile programlanırlar. Bir kanal işlevsel olarak asıl işlemciye bağlıdır. İşlemci kanallar üzerinde bir giriş/çıkış programı başlatabilir, yada durdurabilir. Kanallar giriş/çıkış işlemi ile ilgili bilgileri giriş/çıkış kesmeleri oluşturarak işlemciye haber

verirler. İşlemci bu kesmeleri maskeleyebilir.

Kanallar giriş/çıkış birimleri ile programlar arasında standart bir iletişim sağlar. Yani, bir yazıcıya bir satır yazdırmak ile bir disk üzerinden bir sektör okumak arasında program açısından bir yöntem farkı yoktur. Sadece kullanılan kanal komutları değişiktir. Standart kanallarla yapı ve veri iletişim hızları birbirlerinden çok farklı aygıtları birbirlerine uydurmak kontrol birimlerinin görevidir. Kontrol birimleri, bağlı oldukları aygıttan bağımsız olarak kanallara standart bir veri akışı sağlarlar. Kontrol birimleri bazen aygıtlarla birlikte bazen de ayrı bir gövde içinde bulunurlar.

Kanallar veri iletim yöntemlerine göre üçe ayrılırlar. Bunlar genellikle terminal veya printerler gibi yavaş aygıtlarda kullanılan byte multiplexer kanallar, teyp gibi aygıtlarda kullanılan selector kanallar ve son olarak disklerde kullanılan block multiplexer kanallardır.

System/370 mimarisinde her aygıtın bir adresi vardır. Bu adresler iki byte uzunluğundadır. Yüksek anlamlı byte, bu aygıtın bağlı olduğu kanalı, düşük anlamlı byte ise kontrol ünitesi ve aygıtın kendisini belirtir.

System/370 te bir giriş/çıkış işleminin gidişi şöyledir:

1. Kanal programı oluşturulur.
2. Kanal programının adresi CAW (Channel Address Word) yazılır.
3. Giriş/Çıkış işlemi SIO yada SIOF komutu ile başlatılır.
4. Kanal, giriş/çıkış programını işlemeye başlar.
5. Kanal, giriş/çıkış işlemini bitirince bir kesme oluşturur ve gerekli bilgiyi CSW (Channel Status Word) e yazar.

(Disable Count)	1	1	X	X	0	1	1
(Enable Count)	0	0	X	X	X	0	1
Source Command Byte	X	X	X	1	0	1	0
Read Forward	X	0	0	0	1	1	0

CTCA :

CTCA (Channel to Channel Adapter) iki kanal arasında hızlı ve doğrudan veri iletişimini sağlamak üzere tasarlanmış bir aygıttır. İki ayrı bilgisayarın haberleşmesinde kullanılabileceği gibi aynı bilgisayarın içinde iki bellek bölgesi arasında veri transferi için de kullanılabilir.

CTCA ya bağlı iki kanal birbirleri ile işlem çiftleri kullanarak haberleşirler. Örneğin kanallardan birinin okuma komutuna diğer kanalın bir yazma komutu ile karşılık vermesi gerekir.

Taraflardan birinin bir giriş/çıkış işlemini başlatması eğer karşı taraf daha önce bir komut icra etmemişse o tarafta bir kesmeye sebep olur. Kesmeyi alan karşı taraf, Sense Command Byte komutu ile hangi komutun aktif olduğunu öğrenebilir ve veri transferi için gereken komutu icra edebilir.

CTCA nın komutları ve değişik durumlarda verilen komutlara tepkisi aşağıdaki tablolarda gösterilmiştir.

Komut	Komut Byte'ı Bit Pozisyonları							
	0	1	2	3	4	5	6	7
Write	X	X	X	X	0	0	0	0
Write End-of-File	0	M	M	M	M	M	0	1
Read	1	X	X	X	X	X	0	1
No Operation	0	X	X	X	X	0	1	1
Control	X	X	X	0	0	1	0	0
Sense Adapter State	X	X	X	0	0	1	0	0
Modified No-Op								
(Disable Compt)	1	1	X	X	X	0	1	1
(Enable Compt)	0	1	X	X	X	0	1	1
Sense Command Byte	X	X	X	1	0	1	0	0
Read Backward	M	M	M	M	1	1	0	0

Channel X issues:		Resulting status when channel Y has previously issued:			
	X Ctrl and Y sns Cmd	Y read	Y write	Y control	Y Wr EOF
X read	Busy, DE clear Adpt	Busy and Attn	Y-go-X reset Attn	Busy and Attn	Unit Exc
X write	Busy, DE clear Adpt	X-go-Y reset Attn	Busy and Attn	Busy and Attn	Reset X EOF Attn to Y
X control	Busy, DE clear Adpt	Busy and Attn	Busy and Attn	Busy and Attn	Reset X EOF Attn to Y CE to X
X Wr EOF	Busy, DE clear Adpt	CE, DE to both sides unit Exc to Y	Busy and Attn	Busy and Attn	Reset X EOF set Y EOF CE, DE to X
X test I/O	DE, clear Adpt	Attn	Attn	Attn	Reset X EOF Adpt idle zero status to X
X No-op	Busy, DE clear Adpt	Busy and Attn	Busy and Attn	Busy and Attn	Reset X EOF CE, DE to X
X Sns Cmd	Busy, DE	Force 0 status Y Cmd byte to X	Force 0 status Y Cmd byte to X	Y Cmd byte to X, Rst Attn DE Intr to Y.	Reset X EOF zero Cmd byte to X
X sns state	Busy, DE clear Adpt	Busy and Attn	Busy and Attn	Busy and Attn	Reset X EOF Sns bits to X

VM/SP İşletim Sistemi:

VM/SP (Virtual Machine / System Product) IBM tarafından IBM Ssystem/370 mimarisindeki bilgisayarlar için geliştirilmiş bir işletim sistemidir. Bu işletim sistemi ilk olarak IBM içinde işletim sistemlerini geliştiren çalışma guruplarının gereksinimlerini karşılamak üzere düşünülmüştü. Bu guruplar, üzerinde çalıştıkları işletim sistemlerini deneyebilmek için çok sayıda bilgisayara gereksinim duyuyorlardı. Bu bilgisayarların sağlanması gerek fiziksel, gerekse ekonomik olarak büyük bir problem olduğu için, kullanıcılarına System/370 mimarisindeki bilgisayarların eşdeğeri bir çalışma ortamı sağlayabilecek çok kullanıcılı bir işletim sisteminin geliştirilmesi gerekti. İşte VM/SP işletim sisteminin atası, yalnızca IBM içinde kullanılmak üzere geliştirilen bu işletim sistemidir.

VM/SP nin temelinde o günlerin popüler işletim sistemi OS/VS1 vardır. Bu sistemin pek çok kısmı VM/SP de kullanılmıştır. Ek olarak VM/SP gerçek anlamda bir zaman paylaşımı, komut simülasyonu, giriş/çıkış simülasyonu gibi yeteneklere sahiptir.

Zamanla birkaç amerikan üniversitesinde denenen bu ürün, tahminlerin çok üzerinde bir ilgi görmüş, bu durumda IBM, bu işletim sistemini ticari bir ürün olarak pazarlama kararı almıştır.

VM/SP işletim sistemi günümüzde başta üniversiteler olmak üzere etkileşimli bilgisayar kullanımı gereksinimi olan kuruluşlarda, gerçek zamanlı uygulamalarda ve aynı bilgisayarda birden fazla işletim sistemini kullanılmasının gerektiği durumlarda yaygın olarak kullanılmaktadır.

VM Nedir ?

VM/SP işletim sistemi her kullanıcısına IBM System/370 mimarisindeki bilgisayarların eşdeğeri bir çalışma ortamı sağlar. Bu ortama Virtual Machine (VM) denir. Her VM in tıpkı gerçek bir System/370 bilgisayar gibi işlemcisi, belleği, giriş/çıkış birimleri vardır. Kullanıcılara verilen bu kaynakların miktarı (örneğin VM in bellek büyüklüğü) VM/SP sistem yöneticisi tarafından belirlenir.

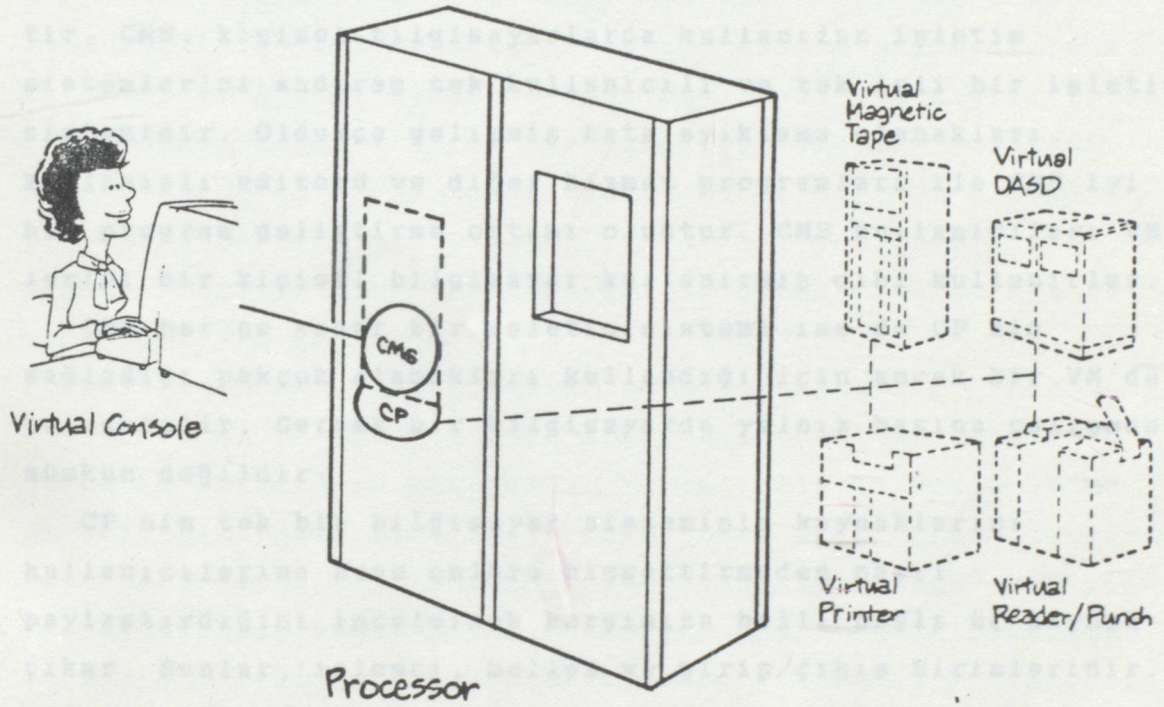
Kullanıcıya işlevsel olarak System/370 konsolunun eşdeğeri bir konsol sağlanmıştır. Gerçek bilgisayarlarda bir takım anahtarlar yada konsol menüleri yardımı ile yürütülen, gerçek belleğin içeriğini görmek, saklayıcıların içeriğini değiştirmek gibi işlevler VM lerde CP (Control Program) ye gönderilen konsol komutları ile gerçekleştirilir. Örneğin gerçek bilgisayara bir işletim sistemi yüklenmek istendiğinde, işletim sisteminin okunacağı aygıtın adresinin birtakım anahtarlarla bilgisayara bildirilmesi ve IPL (Initial Program Load) butonu ile işlemin başlatılması gerekir. VM de ise bu işlevi konsoldan girilen IPL komutu karşılar.

Bir VM de çalışan programlar için çalıştıkları ortamın gerçek bir bilgisayar mı yoksa bir VM mi olduğunu anlamak mümkün değildir. Başka bir deyişle bir program gerçek bilgisayarda yapabildiği herşeyi VM de de aynen yapabilir. Buna ek olarak VM/SP, VM lerde çalışan programların kullanabilecekleri çok kullanışlı bir takım olanaklar da sağlar. Örneğin iki VM arasındaki iletişim. gerçek bir bilgisayarda olduğu gibi giriş/çıkış aygıtları (örneğin bir seri iletişim aygıtı) sağlanabileceği gibi CP nin sağladığı çok daha kullanışlı IUCV (Inter User Communication Vehicle) ile de sağlanabilir. IUCV nin gerçek bilgisayarlarda bir karşılığı yoktur. Ayrıca VM lerin bazı bellek bölgelerinin

örneğin, kullanıcılar bu ortak bellek bölgesi üzerinden birbirleriyle konuşabilir.

CP ve CMS

VM/SP işletim sistemi başlıca iki parçaya ayrılır. Bunlardan ilki, bütün sistem kaynaklarını (işlemci, bellek, giriş/çıkış birimleri) denetleyen ve CP (Control Program) ve CP'ye ait yazılımları VM ortamında bir işletim sistemi olarak çalıştıran CMS (Conversational Monitor System) dir. CMS, kullanıcıların VM ortamında çalıştırılan



İşlemci Yönetimi:

VM/SP işletim sisteminde işlemci, bellek ve giriş/çıkış birimleri gibi donanımsal kaynaklar, işletim sistemi tarafından denetlenir ve kullanıcılar bu kaynakları VM ortamında çalıştırılan programlar aracılığıyla kullanabilir. Bu sayede, kullanıcılar gerçek donanıma erişim olmadan, VM ortamında çalıştırılan programlar aracılığıyla işlemci, bellek ve giriş/çıkış birimlerini kullanabilir. İşletim sistemi, bir uygulama işletim sisteminin büyük bir kısmını oluşturur ve kullanıcılar bu işletim sistemi aracılığıyla VM ortamında çalıştırılan programları çalıştırabilir.

ortak olarak kullanmaları ve bu ortak bellek bölgesi üzerinden haberleşmeleri mümkündür.

CP ve CMS

VM/SP işletim sistemi başlıca iki parçadan oluşur. Bunlardan ilki bütün sistem kaynaklarının (işlemci, bellek, giriş/çıkış birimleri) denetimini elinde tutan CP (Control Program) ve CP nin yarattığı VM ortamında bir işletim sistemi olarak çalışan CMS (Conversational Monitor System) tir. CMS, kişisel bilgisayarlarda kullanılan işletim sistemlerini andıran tek kullanıcı ve tek işli bir işletim sistemidir. Oldukça gelişmiş hata ayıklama olanakları, kullanışlı editörü ve diğer hizmet programları ile CMS iyi bir program geliştirme ortamı oluşturur. CMS kullanıcıları VM lerini bir kişisel bilgisayar kullanırmış gibi kullanırlar.

CMS her ne kadar bir işletim sistemi ise de CP nin sağladığı pekçok olanakları kullandığı için ancak bir VM de çalışabilir. Gerçek bir bilgisayarda yalnız başına çalışması mümkün değildir.

CP nin tek bir bilgisayar sisteminin kaynaklarını kullanıcılarına bunu onlara hissettirmeden nasıl paylaştığını incelersek karşımıza belli başlı üç kaynak çıkar. Bunlar, işlemci, bellek ve giriş/çıkış birimleridir.

İşlemci Yönetimi:

VM/SP işletim sisteminde işlemci, birçok işlemciyi simüle eder. İşlemcinin kontrolü, belli zaman aralıklarında VM lere verilir. Bu zaman aralıklarının uzunluğu ve sıklığı herşeyden önce VM de çalışan programın tabiatına bağlıdır. İletişimli bir uygulama işlemcinin kontrolünü büyük işlemci zamanı gerektiren bir uygulamaya göre daha sık alır. Bir VM

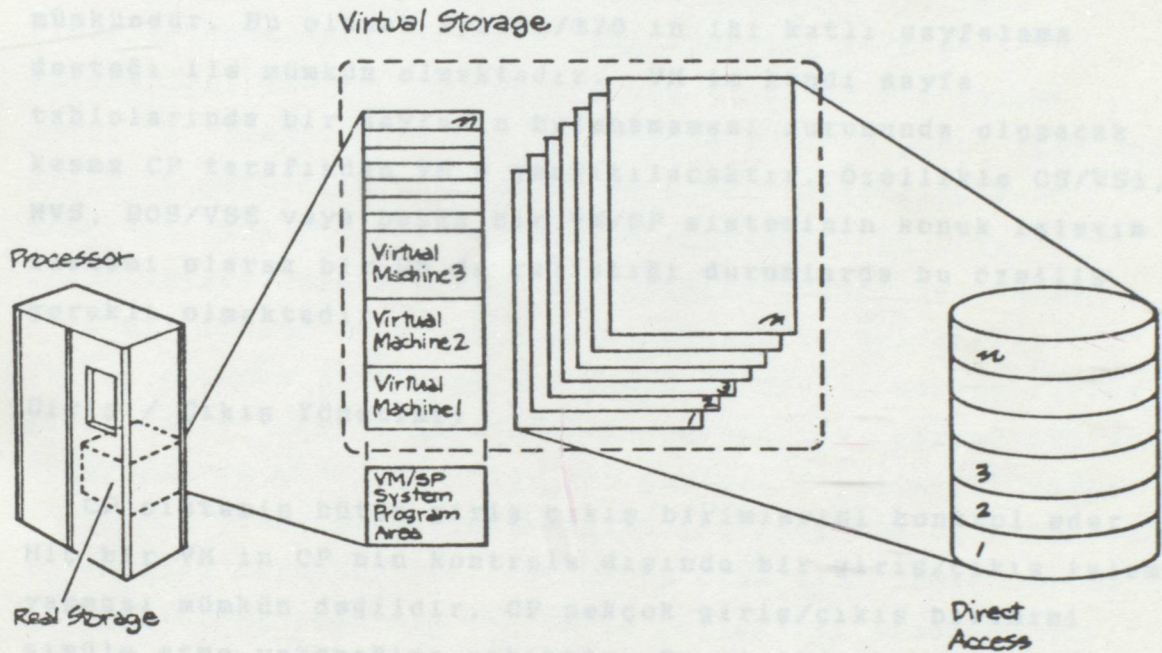
in işlemcinin kontrolünü alabilmesi için, bir kaynak yada olay için beklemiyor olması gerekir. Eğer VM bir bellek sayfası için veya bir kesme için bekliyorsa CP bu durum değişmeden işlemcinin kontrolünü bu VM e vermez. Bir VM in toplam işlemci zamanının ne kadarını kullanabileceği ise o anda işlemci isteği ile bekleyen diğer VM lerin sayısına ve VM in göreceli önceliğine bağlıdır. Öncelikler sistem operatörü tarafından dinamik olarak değiştirilebilir.

VM/SP işletim sisteminde bütün VM ler işlemciyi problem durumunda kullanırlar. İşlemciyi yönetici durumunda sadece işletim sisteminin kendisi kullanır. Bir VM yönetici yetkisi gerektiren, sistem kaynaklarının kullanımı ile ilgili bir komut işlediğinde kontrol bir kesme ile işletim sistemine geçer. CP VM in durumunu kontrol eder ve eğer VM o anda problem durumunda çalışıyorsa, VM de bir yetki aşıldı kesmesi yaratır. Eğer VM yönetici durumunda ise bu kez CP komutu simüle eder.

Yetki gerektiren komutların simülasyonuna bir örnek olarak VM de çalışmakta olan programın kendi durumunu öğrenmek istediğini düşünelim. Durum saklayıcısının (PSW) belleğe yazılması ile ilgili komut icra edildiğinde bu bir kesme yaratacaktır. CP programın kullandığı gerçek PSW yi alacak ve üzerindeki durum göstergelerini değiştirecektir. Bu durumda programın gerçekte problem durumunda çalıştığını anlaması mümkün değildir.

Bellek Yönetimi:

VM/SP işletim sistemi, sayfalı bellek sistemi kullanır. Bellek 4 kilobyte'lık sayfalara ayrılmıştır. İşletim sisteminin bir kısmı gerçek bellekte sürekli dururken, bir kısmı sayfalanabilir bellek kısmına koyulmuştur. VM lerin herbirinin kendilerine ait bir sayfa tablosu vardır.



Sayfalama işlemleri VM in kontrolu ve bilgisi dışında CP tarafından gerçekleştirilir. Temelde en yakın zamanda kullanılan sayfaların bellekte tutulduğu bir bellek yönetimi sistemi kullanılmıştır. Eğer VM o anda ana bellekte olmayan bir sayfaya erişmek isterse bu bir kesmeye sebep olacaktır. Kontrolu alan CP bu sayfayı yan bellekten alıp gerçek belleğe taşıyacak ve daha sonra VM deki programın kaldığı yerden çalışmasına izin verecektir. Bu durumda VM de çalışan program gerçek bellek üzerinde çalışıyormuş izlenimini edinecektir.

VM lerin kendi sayfalama işlemlerini yapması da mümkündür. Bu olanak System/370 in iki katlı sayfalama desteği ile mümkün olmaktadır. VM in kendi sayfa tablolarında bir sayfanın bulunamaması durumunda oluşacak kesme CP tarafından VM e yansıtılacaktır. Özellikle OS/VS1, MVS, DOS/VSE veya başka bir VM/SP sisteminin konuk işletim sistemi olarak bir VM de çalıştığı durumlarda bu özellik gerekli olmaktadır.

Giriş / Çıkış Yönetimi:

CP sistemin bütün giriş çıkış birimlerini kontrol eder. Hiç bir VM in CP nin kontrolu dışında bir giriş/çıkış işlemi yapması mümkün değildir. CP pek çok giriş/çıkış birimini simüle etme yeteneğine sahiptir. Bu giriş/çıkış birimlerini üç grupta toplayabiliriz.

Bunlardan ilki disklerdir. <genellikle bir diskin tümü bir kullanıcıya verilmez. VM lere sadece belli disklerin belli bölgelerine erişme yetkisi verilir. VM ler bu bölgeleri gerçek bir disk gibi görürler. VM de çalışan program bu diskler üzerinde bir giriş/çıkış işlemi başlatmak istediğinde CP kontrolü alır ve programın hazırladığı kanal programını inceler. Kanal programı

üzerindeki VM in gerçek diskte kapladığı alanın başına göre relatif olan adresleri gerçek disk adresleri ile, VM in kullandığı disk adresini de gerçek diskin adresi ile değiştirir. CP daha sonra bu giriş/çıkış işlemini aynı disk için yapılan diğer işlemlerin bulunduğu kuyruğa koyar. İşlem tamamlandığında, CP gerekli bilgiyi VM in belleğine yerleştirip diskten gelen kesmeyi simüle eder. Bu yöntem ile her VM in sadece disk üzerindeki kendi bölgesine erişmesi garantilenmiş olur.

İkinci gurubu reader, punch, printer gibi aygıtlar oluşturur. Bunlar yavaş ve sıralı erişimli aygıtlar olduklarından, simülasyonlarında spooling tekniği kullanılır. Bu tip giriş/çıkış aygıtlarının bir gerçek karşılığı olması gerekmez. Örneğin bir VM in punch aygıtına bir file yazdığını düşünelim. Bu kütük kullanılan kanal komutları ile birlikte disklerde saklanır. Bu file gerektiğinde gerçek bir punch aygıtına gönderilebileceği gibi bir reader'in giriş bilgisi olarak ta kullanılabilir. İkinci durumda ne bir gerçek punch ne de bir gerçek reader gereklidir. Eğer simüle edilen aygıtın tipi gerçek aygıta uymuyorsa CP kanal komutlarında gerekli değişikliği yapar.

İkinci gruba girebilecek başka bir aygıt ta CTCA (Channel to Channel Adapter) dır. İki VM in CTCA ları lojik olarak birbirlerine bağlanıp iki VM in bellekleri arasında hızlı veri transferi sağlanır. Gerçek CTCA larda iki kanal arasında olan veri transferi simülasyon sırasında CP tarafından bellekten belleğe yapılır.

Üçüncü gurup giriş/çıkış simülasyonu da bir aygıtın bütün kullanım yetkilerinin tek bir VM e verildiği, yani paylaşmanın olmadığı durumda yapılır. Bu durumda CP sadece gerekli adres değişikliklerini yapar. Aygıt bir disk, bir teyp, bir printer yada herhangi bir aygıt olabilir. Bu yöntem genellikle bir konuk işletim sistemi çalıştırıldığında kullanılır.

IUCV :

IUCV iki VM in biribirleriyle, bir VM in kendisiyle veya bir sistem servisi ile haberleşmesini sağlayan bir olanaktır. IUCV haberleşmesi bir kaynak ve bir hedef arasında önceden tanımlanan bir yol (path) üzerinden sağlanır. Her VM birçok yola sahip olabilir ve aynı yol üzerinden aynı anda hem mesaj alıp hem de mesaj yollayabilirler.

IUCV, kullanıcıya dört gurup fonksiyon sağlar. Bunlar:

- o Yolların yaratılması veya yok edilmesi
- o Mesajların gönderilmesi veya cevaplanması
- o Bekleyen bir mesajın yada cevabın durumunun öğrenilmesi
- o Mesajların seçilerek kabul edilmesi yada geri çevrilmesi

IUCV Mesajları :

IUCV haberleşmesi mesaj olarak anılır. Kaynak SEND fonksiyonunu çağırdığında iletişim başlatılır ve mesaj yaratılır. Hedef, mesajı RECEIVE fonksiyonunu çağırarak kabul eder. Hedef ayrıca isterse DESCRIBE fonksiyonunu kullanarak mesajlar hakkında bilgi alabilir. Kurulan yolun özelliğine bağlı olarak, gelen mesajı REP fonksiyonunu çağırarak cevaplar. Kaynak, TEST COMPLETION fonksiyonunu kullanarak mesajının cevabının gelip gelmediğini kontrol edebilir. Mesajın gelişi kaynağa bildirildikten sonra mesaj yokedilir. Bir IUCV mesajı, CP tarafından tutulan bir kontrol bloku (MSGBLOK) tarafından temsil edilir.

Mesaj Kuyrukları :

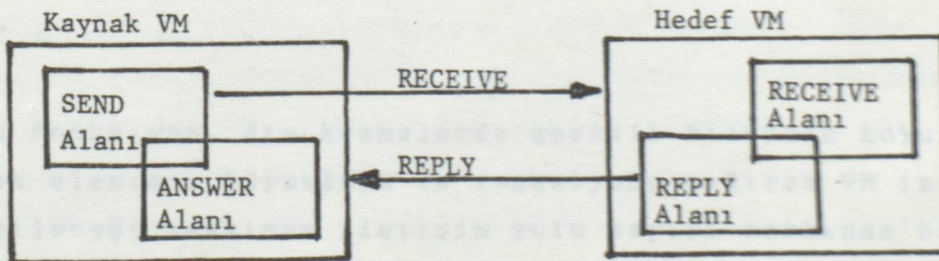
Bir IUCV mesajı yaratılmasından yokedilmesine kadar geçen sürede kuyruk arasında dolaşır. Bunlar:

- o SEND kuyruğu : Bu kuyruk hedeflere gönderilen fakat henüz alınmamış mesajlar içindir.
- o RECEIVE kuyruğu : Bu kuyruk ise hedef tarafından alınmış fakat henüz cevabı verilmemiş mesajlar içindir.
- o REPLY kuyruğu : Bu kuyruk ise cevapları verilmiş ve kaynak tarafından yokedilmeyi bekleyen mesajlar içindir.

Kaynak SEND fonksiyonunu çağırdığında IUCV bir mesaj bloğu yaratır ve bunu hedefin SEND kuyruğuna koyar. Hedef RECEIVE fonksiyonunu çağırdığında IUCV, MSGBLOK u bu kez hedefin RECEIVE kuyruğuna koyar. Hedef Reply fonksiyonunu çağırdığında MSGBLOK un gideceği yer kaynağın REPLY kuyruğudur. Kaynak, TEST COMPLETION fonksiyonunu çağırdığında mesaj REPLY kuyruğundan alınır ve yokedilir. Böylece kaynak ve hedef arasındaki haberleşme bitmiş olur.

Mesaj Veri Transferi :

MSGBLOK IUCV kuyrukları arasında dolaşırken, IUCV asıl bilgiyi hedef RECEIVE ve REPLY fonksiyonlarını çağırdığında taşır. Hedef RECEIVE fonksiyonunu çağırdığında IUCV bilgiyi kaynağın SEND alanından hedefin RECEIVE alanına taşır. Benzer şekilde hedef REPLY fonksiyonunu çağırdığında cevap hedefin REPLY alanından kaynağın ANSWER alanına taşınır.



IUCV Veri Transferi

IUCV Dış Kesmeleri :

Kullanıcı, bekleyen IUCV mesajlarını haber almak için iki yoldan birini seçer. Bunlardan biri özel bir dış kesme, ikincisi ise DESCRIBE fonksiyonudur. IUCV dış kesmelerini kullanabilmek için kullanıcının,

- o DECLARE BUFFER fonksiyonu ile IUCV nin dış kesme ile ilgili bilgileri koyacağı bellek alanının belirlenmesi,
- o PSW nin 7. bitinin 1 yaparak dış kesmelere izin vermesi,
- o 0. kontrol saklayıcısının 30. bitini 1 yaparak IUCV kesmelerine izin vermesi gerekir.

Buna ilave olarak kullanıcı, SETMASK ve SETCMASK fonksiyonları ile bazı IUCV kesmelerine izin verip diğerlerini engelleyebilir.

IUCV X'4000' kodlu bir dış kesme oluşturur ve kesmeye sebep olan durum ile ilgili bilgiyi, daha önce DECLARE BUFFER fonksiyonu ile belirlenen bellek bölgesine koyar.

IUCV Kesmelerinin Kullanılması :

Bir assembler programdan IUCV fonksiyonları IUCV makro komutu yardımı ile çağırılır. Aşağıdaki IUCV fonksiyonları tipik bir iletişimde kullanıldıkları sıra ile kısaca açıklanmışlardır.

QUERY :

Bu fonksiyon, dış kesmelerde gerekli bilginin koyulacağı bellek alanının büyüklüğü ve fonksiyonu çağırarak VM in kurabileceği maksimum iletişim yolu sayısı hakkında bilgi

almak için kullanılır.

DECLARE BUFFER :

Bu fonksiyon, dış kesmelerde IUCV nin kesme ile ilgili bilgileri koyacağı bellek bölgesinin yerini belirtmek için kullanılır. Bu, QUERY dışındaki IUCV fonksiyonlarının kullanılabilmesi için gereklidir.

CONNECT :

Bu fonksiyon, bir başka VM ile bağlantı kurmak istenildiğinde kullanılır. Kaynak CONNECT fonksiyonunu çağırdığında, hedef durumdan haberdar edilir. Yol hedef ACCEPT fonksiyonunu çağırıncaya kadar tam olarak kurulmuş sayılmaz.

ACCEPT :

Bu fonksiyon, bekleyen bir CONNECT e olumlu cevap vermek için kullanılır. ACCEPT çağırıldıktan sonra IUCV yolu mesaj trafiğine açar. Hedef, ACCEPT yerine SEVER fonksiyonunu kullanarak bu yolu reddedebilir.

SEND :

Bu fonksiyon, bir başka VM le daha önce kurulmuş bir yol üzerinden bir haberleşme başlatmak için kullanılır. Kaynak, SEND fonksiyonunu çağırdığında, IUCV bir mesaj bloğu yaratarak bunu hedefin SEND kuyruğuna koyar. Hedef RECEIVE fonksiyonunu çağırmadıkça asıl mesaj transfer edilmez. SEND fonksiyonunu çağıran kaynak bu mesajın karşılığında bir yanıt bekleyip beklemediğini, başka bir değişle mesajın bir

yönlü mü yosa iki yönlü mü olduğunu da belirtir.

DESCRIBE :

Bu fonksiyon, SEND kuyruğunda bekleyen mesajları sorgulamak için kullanılır. DESCRIBE fonksiyonu tarafından sağlanan bilgi ışığında hedef, bu mesajı kabul edip etmeyeceğine karar verir. Verilen karara göre RECEIVE yada REJECT fonksiyonlarından biri çağırılır.

RECEIVE :

Bu fonksiyon, SEND kuyruğunda bekleyen bir mesaj bloğunun temsil ettiği veriyi kaynağın SEND alanından hedefin RECEIVE alanına aktarmak için kullanılır. Eğer hedefin bellek alanı mesajın tamamını almaya yetmiyorsa, mesajın kalan kısmı ikinci bir RECEIVE ile alınabilir.

REPLY :

Bu fonksiyon, kaynak tarafından gönderilen ve alınan bir mesaja cevap yollamak için kullanılır. REPLY fonksiyonu cevabı hedefin REPLY alanından, kaynağın ANSWER alanına taşır. Mesaj bloğu da hedefin RECEIVE kuyruğundan hedefin REPLY kuyruğuna geçer.

TEST COMPLETION :

Bu fonksiyon, herhangi bir mesajın tamamlanıp tamamlanmadığını anlamak için kullanılır. TEST COMPLETION çağırıldıktan sonra IUCV mesajı REPLY kuyruğundan alarak yok eder.

TEST MESSAGE :

Bu fonksiyon, herhangi bir mesaj veya bir cevabın işlenmeyi bekleyip beklemediğini kontrol etmek için kullanılır. TEST MESSAGE fonksiyonu çağırıldığında eğer bekleyen bir mesaj yada cevap yoksa bekleme konumuna girer. Bir mesaj geldiğinde VM, TEST MESSAGE komutunu yeniden işleterek çalışmaya başlar. Bu fonksiyonun en önemli faydası, VM i dış kesmeleri değerlendirme derdinden kurtarmasıdır.

REJECT :

Bu fonksiyon, bir mesajı geri çevirmeye yarar. DESCRIBE veya RECEIVE fonksiyonlarını çağırdıktan sonra hedef, mesajı işlememeye karar verebilir. Bu durumda kullanılan REJECT fonksiyonu, mesajı hedefin SEND veya RECEIVE kuyruğundan alıp, kaynağın REPLY kuyruğuna koyar ve kaynağı bu mesajın reddedildiği konusunda uyarır.

PURGE :

Bu fonksiyon, herhangi bir kuyruktaki mesajın yok edilmesi için kullanılır. IUCV eğer gerekiyorsa hedefi mesajın yok edildiği konusunda uyarır.

SET CONTROL MASK :

Bu fonksiyon, IUCV kontrol fonksiyonlarının meydana getirdiği kesmelerin hangilerine izin verileceğini belirlemede kullanılır.

SET MASK :

Bu fonksiyon, deęişik türde IUCV kesmelerinin hangilerine izin verileceęini belirlemede kullanılır.

QUIESCE :

Bu fonksiyon, IUCV yollarından gelen mesajları geçici olarak durdurur. Mesaj trafięi RESUME fonksiyonu ile yeniden başlatılabilir.

RESUME :

Bu fonksiyon, mesaj trafięi QUIESCE fonksiyonu ile geçici olarak durdurulmuş yolları tekrar eski haline getirir.

SEVER :

Bu fonksiyon, bir IUCV yolunun ortadan kaldırılması için kullanılır. Bir yolun ortadan kaldırılması için her iki tarafın da aynı yol için SEVER fonksiyonunu çağırması gerekir.

RETRIEVE BUFFER :

Bu fonksiyon, bütün aktif mesajları, yolları yok eder ve IUCV haberleşmesini durdurur.

VM ler Arası İletişim :

Aşağıdaki şekil, iki VM in IUCV yolu ile haberleşmeleri sırasında IUCV fonksiyonlarını hangi sırayla kullandıklarını göstermektedir.

(Virtual Machine X)

1 DECLARE BUFFER

2 Y ye CONNECT

5 Dış kesme...

6 Y ye SEND

8 TEST COMPLETION

11 Dış kesme...

veya

TEST COMPLETION

12 SEVER

15 RETRIEVE BUFFER

(Virtual Machine Y)

1 DECLARE BUFFER

3 Dış kesme...

4 ACCEPT

7 Dış kesme...

veya

DESCRIBE

9 RECEIVE

10 REPLY

13 Dış kesme...

14 SEVER

15 RETRIEVE BUFFER

1.. VM X, VM Y ile haberleşmek ister. Her iki VM de bağımsız olarak DECLARE BUFFER fonksiyonunu çağırmalıdır.

2.. VM X, CONNECT fonksiyonunu çağırır ve hedef olarak Y VM ini gösterir. IUCV Y için X in kendisi ile haberleşmek istediğini belirten bir dış kesme yaratır. VM X te kontrol IUCV komutundan sonraki komuta geçer.

3.. VM Y nin dış kesme altprogramları kontrolü alırlar.

4.. VM Y kesmeyi değerlendirir ve ACCEPT fonksiyonunu

çağırır.

IUCV bağlantıyı tamamlar ve VM X için bir bağlantı tamamlandı kesmesi oluşturur. IUCV VM Y de kontrolü IUCV komutundan sonraki komuta verir.

5.. VM X in dış kesme altprogramları kontrolü alırlar ve kesmeyi değerlendirerek haberleşmenin sağlandığını anlarlar.

6.. VM X bir SEND icra eder. Bu da VM Y de bir dış kesme oluşturur. VM X IUCV komutundan sonraki komut ile çalışmasına devam eder.

7.. Eğer VM Y nin dış kesmeleri maskeli değilse VM Y kesme programını işlemeye başlar. Eğer kesmeler maskeli ise VM Y DESCRIBE fonksiyonu yardımıyla gelen mesajdan haberdar olabilir.

8.. Eğer VM X bu sırada TEST COMPLETION fonksiyonu ile mesajın durumunu kontrol ederse mesajın henüz alınmadığını öğrenir.

9.. VM Y RECEIVE fonksiyonunu çağırır. Kullandığı parametre listesi mesajın nereye koyulacağını göstermektedir.

10.. Mesajın işlenmesi bittiğinde VM Y, VM X e REPLY fonksiyonu ile yanıt gönderir. Bu ise VM X te bir dış kesme yaratır. VM Y, çalışmasına IUCV komutundan sonra gelen komutla devam eder.

11.. Eğer VM X dış kesmeleri maskelemedi ise bir cevabın beklediğini dış kesme ile öğrenir. Aksi taktirde TEST COMPLETION fonksiyonu kullanarak bekliyen bu cevap görülür.

12.. VM X işini bitirdiği için ilişkiyi kesmek üzere SEVER fonksiyonunu çağırır. IUCV VM Y için bir dış kesme oluşturur. VM X, çalışmasına IUCV komutundan sonraki komut ile devam eder.

13.. Dış kesmeyi alan VM Y, bu haberleşme için ayırdığı bellekleri temizler.

14.. VM Y, işini bitirdikten sonra SEVER fonksiyonunu çağırır ve haberleşmeyi bitirir. IUCV, kontrol bloklarını temizler.

15.. Haberleşme sona erdikten sonra her iki taraf ta RETRIEVE BUFFER fonksiyonunu kullanarak IUCV ile ilgilerini keserler.

ROUTE Tablosu

Bu tablo, ağı oluşturan düğümlerin yollarını belirler. Network Layer, kendisine gelen bir mesajla hangi link üzerinden gönderileceğine bu tabloya bakılarak karar verir. Tablo doğru ve link isimlerinden oluşan sıralar içerir. Bu tabloda her düğüm için bir kez geçer. Bir tek düğüm link isimleri aynı olabilir. Bu tablonun kayıtlarının oluşturulmasında ROUTE makrosu kullanılır. Makrosun formatı şöyledir:

ROUTE Nodeid,Linkid

Sistemin Çalışma Şekli :

Bu sistem, katmanlar arasındaki düzenleyen bir ana program, muhtelif katmanları oluşturan modüller, bu modüller arasındaki mesaj trafiğini sağlayan bir kuyruk sistemi ve sistemin ve ağın yapısı hakkında önceden belirlenen bazı bilgileri taşıyan bir veri tabanından oluşmuştur.

Veri Tabanı :

Sistemin veri tabanı TABLES isimli bir assembler program derlenerek oluşturulur. Tables programının içinde sistemin veri tabanını meydana getiren route tablosu, link tablosu, lokal veriler, ve adres tablosu vardır. Oluşturulan amaç kod sistemin tüm modülleri tarafından kullanılır. Veri tabanının oluşturulmasını kolaylaştırmak amacı ile makrolar kullanılmıştır. Tabloların route tablosu, link tablosu, lokal veriler ve adres tablosu şeklinde sıralı oluşturulması zorunludur.

ROUTE Tablosu :

Bu tablo, ağı oluşturan düğümlerin yönlerini belirler. Network Layer, kendisine gelen bir mesajın hangi link üzerinden gönderileceğine bu tabloyu inceleyerek karar verir. Tablo düğüm ve link isimlerinden oluşan eşler içerir. Bu tabloda her düğümün adı bir kez geçer. Bir çok düğümün link isimleri aynı olabilir. Bu tablonun kayıtlarının oluşturulmasında ROUTE makrosu kullanılır. Makronun formatı şöyledir:

ROUTE Nodeid,Linkid

Burada 'Nodeid', en fazla sekiz karakter uzunluğunda olan bir isimdir. Ağ üzerindeki her düğüm böyle bir isimle anılır. 'Linkid' ise yine en fazla sekiz karakter uzunluğunda olabilen link ismidir. Sistemin komşu düğümlere bağlantısını oluşturan linkleri temsil eder. Bu tablodaki düğüm sayısı sınırlı değildir.

LINK Tablosu :

Bu tablo, sistemin linklerini tanımlamakta kullanılır. Her link e sekiz karaktere kadar uzun olabilen bir isim verilmiştir. Bu sistemde bütün linkler full duplex tir. Link tablosunun kayıtları LINK makrosu yardımı ile oluşturulur. Bu makronun formatı şöyledir.

LINK Linkid,Bufflen,Inctca,Outctca

Burada 'Linkid' linkin ismidir. 'Bufflen', bu link için kullanılacak giriş/çıkış tamponlarının boyudur. 'Inctca' ve 'Outctca' ise giriş ve çıkış için kullanılacak CTCA ların adresleridir. Bu tablodaki linklerin sayısı konusunda da bir sınırlama yoktur.

Lokal Bilgiler :

Lokal bilgiler: Sistemin kendi düğüm ismi ve Session Layer ın kullanacağı tamponların boyutudur. Bu bilgiler LOCAL makrosu ile verilir. Makronun formatı şöyledir.

LOCAL Nodeid,Bufflen

Burada 'Nodeid', lokal düğüm ismi, Bufflen ise tampon uzunluğudur. Bu tablo ayrıca sistem kuyrukları ile ilgili

bilgiler de içerir. Herhangi bir mesajın alınması ve iletmesi için mesaj bloğunun yok edilmesi için kullanılır.

Adres Tablosu :

Sistem Modülleri :

Bu tablo, route tablosu, link tablosu ve lokal bilgilerin adreslerinin yanında sistemi oluşturan modüllerin adresleri ve başka bilgiler de taşır. Tablonun oluşturulmasında ADDRLST makrosu kullanılır. Bu makronun formatı şöyledir:

ADDRLST Routes, Links, Local
Bu sistemde Physical Layer ad CTCL, EXTEND ciftidir. Bu sistemde Physical Layer ad CTCL, EXTEND ciftidir. Bu sistemde Physical Layer ad CTCL, EXTEND ciftidir.

Burada 'Routes', route tablosu, 'Links', link tablosu, 'Local' ise lokal bilgiler için kullanılan assembler etiketleridir.

Kuyruk Sistemi :

Bu sistemde başlıca üç adet kuyruk vardır. Bunlar, linkler üzerinden gitmek isteyen mesajların kuyrukları, Network Layer a gelen ve ya bir link e yada daha üst katmanlara gönderilmeyi bekleyen mesajların kuyruğu ve lokal kullanıcılar da dağıtılmak üzere Session Layer a yollanan mesajların kuyruğudur.

Kuyrukların hepsi FIFO (First In First Out) prensibi ile çalışır. Yani bir kuyruğa gelen mesaj kendinden önce gelenleri bekler. Öncelikli mesajlar söz konusu değildir.

Kuyrukta bekleyen her mesaj için, içinde mesajın kendisi ile diğer mesajlara doğru ileri ve geri pointer lar olan bir mesaj bloğu oluşturulur. Mesajın bir kuyruktan diğerine aktarılması sırasında sadece pointer ları değiştirilir. Mesaj yerine ulaştığında mesaj bloğu yok edilir.

Kuyrukların denetimi için üç altprogram kullanılmıştır. Bunlardan QUEMSG, bir mesaj bloğunun kuyruğa sokulması için,

POPMSG, kuyruğun en üzerindeki mesajın alınması ve KILLMSG bir mesaj bloğunun yok edilmesi için kullanılır.

Sistemin Modülleri :

Sistemde dört adet ana modül ve bunların alt programları vardır. Bu modüller sırasıyla, Diğer modülleri kontrol eden ana program, Data Link Layer görevi yapan LNKSEND, IOIHND çifti, Network Layer görevi yapan NETSEND, ile Session Layer ve Presentation Layer görevlerini birlikte yapan SESSEND, EXTIHND çiftidir. Bu sistemde Physical Layer ın CTCA, Application Layer ın ise IUCV olduğunu söyleyebiliriz. Transport Layer ın fonksiyonları ise bu sistemde kullanılmamıştır.

Ana Program :

Ana program, kontrolü işletim sisteminden aldıktan sonra INIT altprogramını çalıştırarak sistemi çalışmaya hazır hale getirir. Daha sonra, bir döngü içinde LNKSEND, SESSEND ve NETSEND modüllerini çağırarak sistemdeki mesaj akışını sağlar. Sistemde yapılacak bir iş olmadığı durumlarda ana program işlemciyi bekleme durumuna getirerek bir kesme bekler. Sistemde meydana gelebilecek anormal bir durumda, bütün operasyonu RESET altprogramı ile durdurarak kontrolü işletim sistemine verir.

IOIHND :

Bu program, sistemin giriş/çıkış kesmelerini değerlendirir. Komşu bir düğüm linklerden biri üzerinden bir mesaj yolladığında bu mesaj alınarak Network Layer ın kuyruğuna koyulur.

LNKSEND :

Bu program, linklerin kuyruklarındaki mesajları komşu düğümlere yollar. Ana program tarafından çağırıldığında bütün linkleri tarayarak kuyruklarında mesaj olan ve bilgi göndermeye hazır linklerde gönderme işlemini başlatır. Bütün linkler kontrol edildiğinde kontrolü ana programa geri verir.

NETSEND :

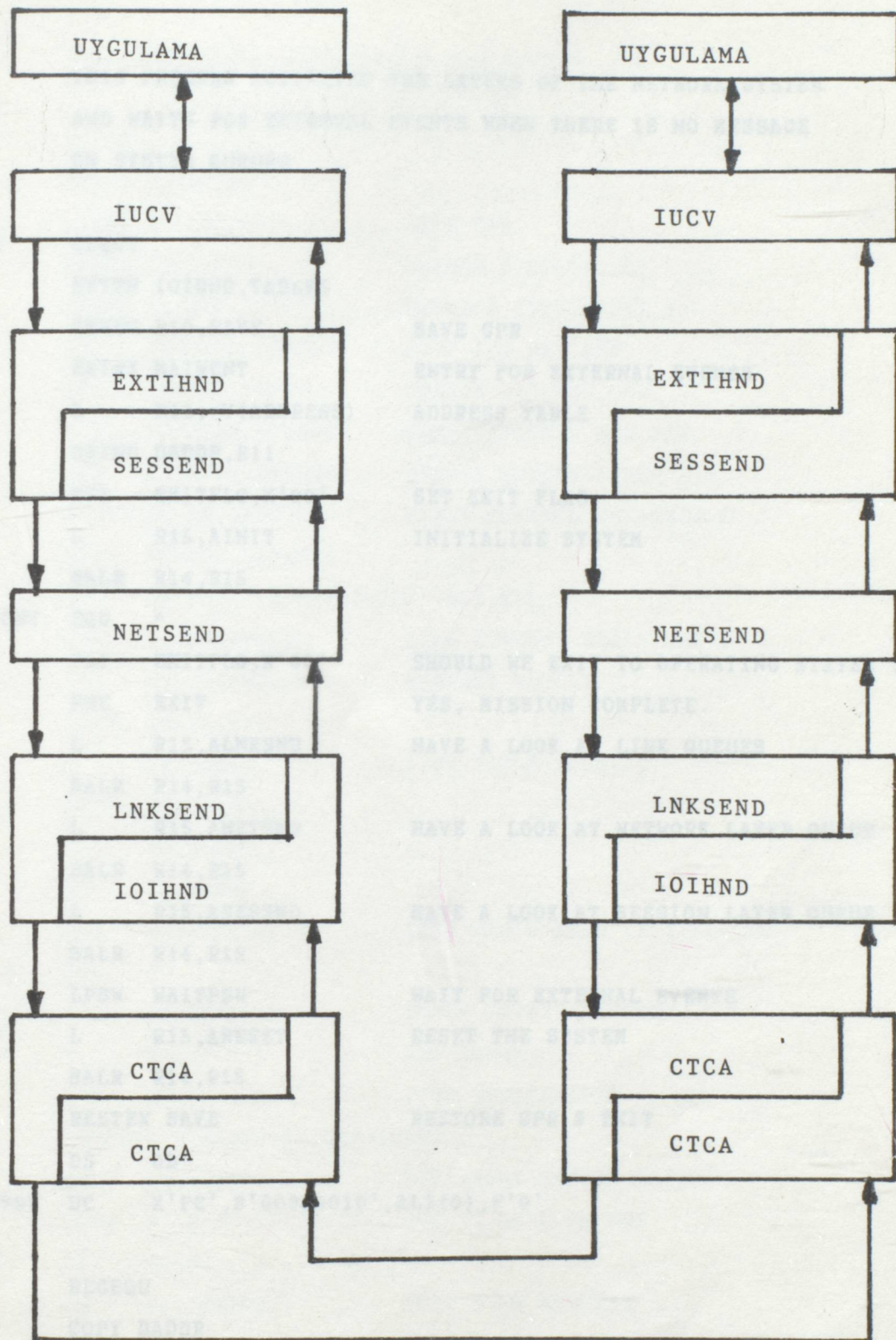
Bu program, Network Layer ın kuyruğundaki mesajlara bakarak bunları ait oldukları yerlere yollar. Eğer mesaj lokal düğüm içinse mesajın koyulacağı kuyruk Session Layer ın kuyruğudur. Mesaj bir uzak düğüme gidiyorsa düğüme giden link bulunur ve mesaj o linkin kuyruğuna koyulur.

EXTIHND :

Bu program, sistemi kullanmak isteyen uygulamaların IUCV kanalıyla yarattıkları kesmeleri değerlendirir. IUCV den gelen mesajları network layer a iletir. İki uygulama arasındaki session ı sağlar.

SESSEND :

Bu program, Network Layer dan gelen mesajları ait oldukları yere IUCV ile yollar. Session ların denetimini EXTIHND ile paylaşır.




```

*      THIS PROGRAM ACTIVATES THE LAYERS OF THE NETWORK SYSTEM
*      AND WAITS FOR EXTERNAL EVENTS WHEN THERE IS NO MESSAGE
*      ON SYSTEM QUEUES
*
MAIN    CSECT
        EXTRN IOIHND, TABLES
        SAVEG R10, SAVE          SAVE GPR
        ENTRY MAINCNT           ENTRY FOR EXTERNAL EVENTS
        L      R11, =V(ADDRESS)  ADDRESS TABLE
        USING DADDR, R11
        MVI    EXITFLG, X'00'    SET EXIT FLAG
        L      R15, AINIT        INITIALIZE SYSTEM
        BALR   R14, R15
MAINCNT EQU    *
        CLI    EXITFLG, X'00'    SHOULD WE EXIT TO OPERATING SYSTEM ?
        BNE    EXIT             YES, MISSION COMPLETE.
        L      R15, ALNKSND      HAVE A LOOK AT LINK QUEUES
        BALR   R14, R15
        L      R15, ANETSND     HAVE A LOOK AT NETWORK LAYER QUEUE
        BALR   R14, R15
        L      R15, ASENSND     HAVE A LOOK AT SESSION LAYER QUEUE
        BALR   R14, R15
        LPSW   WAITPSW          WAIT FOR EXTERNAL EVENTS
EXIT     L      R15, ARESET      RESET THE SYSTEM
        BALR   R14, R15
        RESTEX SAVE             RESTORE GPR & EXIT
        DS     OD
WAITPSW DC     X'FE', B'00000010', AL2(0), F'0'
*
        REGEQU
        COPY DADDR
        END

```



```

*      THIS SUBROUTINE INITIALIZES THE SYSTEM
*
INIT   CSECT
        USING NUCON,0
        SAVEG R10,SAVEINI      SAVE GPR
        DMSKEY NUCLEUS         CHANGE STORAGE KEY
        STNSM SAVESM,X'00'     DISABLE INTERRUPTS
        MVC   SAVEION,IONPSW   SAVE I/O NEW PSW
        MVC   IONPSW,IOPSW     CHANGE I/O NEW PSW
        MVC   SAVEEXN,EXTNPSW  SAVE EXTERNAL NEW PSW
        MVC   EXTNPSW,EXTPSW   CHANGE EXTERNAL NEW PSW
        STCTL C0,C0,SAVECTL    STORE CONTROL REGISTER 0
        MVC   NEWCTL,SAVECTL
        OI    NEWCTL+3,B'00000010' SET BIT 30
        LCTL C0,C0,SAVECTL     LOAD CONTROL REGISTERS 0
        L      R11,=V(ADDRESS) ADDRESS TABLE
        USING DADDR,R11
        L      R4,LINKS        LINK TABLE ADDRESS
        USING DLINK,R4
        L      R2,LNLEN
        LA     R3,LNEND
        MVC   EXBADDR,=F'0'    IUCV HAS'T BEEN INITIALIZED
LINKLP EQU   *
        L      R0,INBUFLN      INPUT BUFFER LENGTH
        A      R0,=F'7'        ADD SEVEN
        SRA    R0,3            FIND DOUBLEWORDS
        DMSFREE DWORDS=(0),ERR=ERREX
        ST     R1,INBUF        PUT INPUT BUFFER ADDRESS
        STCM   R1,B'0111',INCCW+1 PUT IT ALSO IN CCW
        SR     R1,R1
        LH     R1,INCTCA       LOAD CTCA ADDRESS
        HIO    0(R1)           HALT I/O
        TIO    0(R1)           CLEAR PENDING INTERRUPTS

```



```

BC      1,ERREX          QUIT IF DEVICE NOT EXIST
L       R0,OUTBUFLN      INPUT BUFFER LENGTH
A       R0,=F'7'         ADD SEVEN
SRA     R0,3             FIND DOUBLEWORDS
DMSFREE DWORDS=(0),ERR=ERREX
ST      R1,OUTBUF        PUT OUTPUT BUFFER ADDRESS
STCM    R1,B'0111',OUTCCW+1 PUT IT ALSO OUT CCW
SR      R1,R1
LH      R1,OUTCTCA       LOAD CTCA ADDRESS
HIO     0(R1)            HALT I/O
TIO     0(R1)            CLEAR PENDOUTG OUTTERRUPTS
BC      1,ERREX          QUIT IF DEVICE NOT EXIST
MVC     LQUETOP,NULL     QUEUE IS
MVC     LQUEBOT,NULL     EMPTY
BXLE    R4,R2,LINKLP     LOOP
L       R4,LOCAL         LOAD CONTROL BLOK ADDRESS
USING   DLOC,R4
MVC     NQUETOP,NULL     QUEUE IS
MVC     NQUEBOT,NULL     EMPTY
MVC     SQUETOP,NULL     QUEUE IS
MVC     SQUEBOT,NULL     EMPTY
MVC     SCHNTOP,NULL     CHAIN IS
MVC     SCHNBOT,NULL     EMPTY
EXTINI  EQU      *
IUCV    QUERY           ASK NECESSARY BUFFER SIZE ?
ST      R1,MAXCONN      MAXIMUM CONNECTIONS ALLOWED
ST      R0,EXBSIZE      EXTERNAL INTERRUPT BUFFER SIZE
A       R0,=F'7'         ADD SEVEN
SRA     R0,3             FIND DOUBLEWORDS
DMSFREE DWORDS=(0),ERR=ERREX
ST      R1,EXBADDR
USING   IPARML,R1
IUCV    DCLBFR,BUFFER=(1),PRMLIST=IUCVPARM

```



```

BNZ    ERREX
IUCV   SETMASK,MASK=MYMASK,PRMLIST=IUCVPARM
IUCV   SETCMASK,MASK=MYCMASK,PRMLIST=IUCVPARM
DROP   R1
B       INITEX
ERREX   EQU    *
MVI     EXITFLG,X'FF'      EXIT TO OPERATING SYSTEM
INITEX  RESTEX SAVEINI
*
*      THIS SUBROUTINE RESETS THE SYSTEM
*
ENTRY RESET          ENTRY POINT FOR RESET
RESET  SAVEG R10,SAVERES  SAVE GPR
MVC     IONPSW,SAVEION   RESTORE I/O NEW PSW
MVC     EXTNP SW,SAVEEXN  RESTORE EXTERNAL NEW PSW
LCTL    CO,CO,SAVECTL    SAVE CONTROL REGISTER 0
CLC     EXBADDR,=F'0'
BE       SKIPIUCV
IUCV    RTRVBFR          RESET IUCV OPERATION
SKIPIUCV SSM  SAVESM      RESTORE SYSTEM MASK
DMSKEY  RESET           RESTORE STORAGE KEY
RESTEX  SAVERES
DS       OD
SAVEION  DS      D
IOPSW    DC      AL4(0),V(IOIHND)
SAVEEXN  DS      D
EXTPSW   DC      AL4(0),V(EXTIHND)
NULL     DC      F'0'
SAVECTL  DS      F
NEWCTL   DS      F
SAVESM   DS      F
MYMASK   DC      AL1(IPSNDN+IPRPYN+IPCTRL)
MYCMASK  DC      AL1(IPCLPC+IPCLCC+IPCLPS)

```


DS OD LOOKS THE DATA LINK LAYER DOWN
 IUCVPARM DS CL40 THE MESSAGES THROUGH LINKS

*

PRINT NOGEN
 REGEQU SAVE SAVE SPK
 COPY DADDR NUCLEUS CONSTANT 1001
 COPY DLINK ADDRESS BLOCK
 COPY DLOC LINKS LINK TABLE ADDRESS
 COPY IPARML LINK, R4
 NUCON R1, LNLEN LOAD LINK ENTRY LENGTH
 END R3, LNEND LOAD LINK TABLE END
 LINECH EQU LINK CHECK LOOP
 STR R1, R3, PARM5V SAVE R1R3
 LA R1, LOUENDP TOP OF QUEUE
 LA R1, LOUEBOT BOTTOM OF QUEUE
 L R15, APOENDG WE WILL CALL POPMSG
 BALR R14, R15
 USING BRSC, R1 R1 POINTS MESSAGE BLOCK
 C R1, NULL IS QUEUE EMPTY ?
 BE NEXT YES, LOOK FOR NEXT LINK
 CLX OUTSTAT, JOLB IN DEVICE INDEX ?
 BNE NEXT NO, LOOK FOR NEXT LINK
 LA R6, MSGNET BEGINNING OF THE MESSAGE
 L R5, MSGLEN TEXT LENGTH
 LA R9, 16(R9) EXACT MESSAGE LENGTH
 L R6, OUTSUP OUTPUT BUFFER ADDRESS
 LA R7, OUTROFLM LENGTH OF OUTPUT BUFFER
 MVSL R6, R6 MOVE THE MESSAGE
 STR R9, OUTCCW+4 PUT DATA LENGTH TO THE COW
 L R13, ANILN50 WE WILL CALL BILLING
 BALR R14, R15
 LA R1, OUTCCW PUT THE COW ADDRESS

* THIS SUBROUTINE LOOKS THE DATA LINK LAYER QUEUES
 * AND SENDS THE MESSAGES THROUGH LINKS
 *

LNKSEND	CSECT	
	SAVEG R10,SAVE	SAVE GPR
	USING NUCON,0	NUCLEUS CONSTANT AREA
	L R11,=V(ADDRESS)	ADDRESS BLOK
	USING DADDR,R11	
	L R4,LINKS	LINK TABLE ADDRESS
	USING DLINK,R4	
	L R2,LENL	LOAD LINK ENTRY LENGTH
	LA R3,LENL	LOAD LINK TABLE END
LNKCHK	EQU *	LINK CHECK LOOP
	STM R2,R3,PARMSV	SAVE R2&R3
	LA R2,LQUETOP	TOP OF QUEUE
	LA R3,LQUEBOT	BOTTOM OF QUEUE
	L R15,APOPMSG	WE WILL CALL POPMSG
	BALR R14,R15	
	USING DMSG,R1	R1 POINTS MESSAGE BLOK
	C R1,NULL	IS QUEUE EMPTY ?
	BE NEXT	YES, LOOK FOR NEXT LINK
	CLI OUTSTAT,IDLE	IS DEVICE IDLE ?
	BNE NEXT	NO, LOOK FOR NEXT LINK
	LA R8,MSGNET	BEGINNING OF THE MESSAGE
	L R9,MSGLEN	TEXT LENGTH
	LA R9,16(R9)	EXACT MESSAGE LENGTH
	L R6,OUTBUF	OUTPUT BUFFER ADDRESS
	LA R7,OUTBUFLN	LENGTH OF OUTPUT BUFFER
	MVCL R6,R8	MOVE THE MESSAGE
	STH R9,OUTCCW+6	PUT DATA LENGTH TO THE CCW
	L R15,AKILMSG	WE WILL CALL KILLMSG
	BALR R14,R15	
	LA R5,OUTCCW	PUT THE CCW ADDRESS +

	ST	R5,CAW	INTO CAW
	SR	R1,R1	ZERO R1
	LH	R1,OUTCTCA	TAKE CTCA ADDRESS
	SIO	0(R1)	START I/O
	BNZ	CHKCND	IF CC<>0 CHECK CONDITIONS
	MVI	OUTSTAT,IOWAIT	CHANGE STATUS AS BUSY
	B	NEXT	CONTINUE WITH THE NEXT LINK
CHKCND	BC	3,LNSNERR	SKIP IF FATAL ERROR OCCURED
	TM	CSW+4,B'00000100'	CHECK FOR DEVICE END
	BC	14,LNSNERR	NO, THERE MUST BE AN ERROR
	MVI	OUTSTAT,IDLE	I/O COMPLETE
	B	NEXT	
LNSNERR	HIO	0(R1)	HALT I/O
	TIO	0(R1)	CLEAR INTERRUPTS
NEXT	LM	R2,R3,PARMSV	RESTORE R2&R3
	BXLE	R4,R2,LNKCHK	LOOP
		RESTEX SAVE	
PARMSV	DS	2F	
NULL	DC	F'0'	
		*	
		PRINT NOGEN	
		REGEQU	
		COPY DADDR	
		COPY DLINK	
		COPY DMSG	
		NUCON	
		END	


```

IOIHND  CSECT
        USING NUCON,0
        ST      R10,NUCRSV2          SAVE R10
        BALR    R10,0                WHERE AM I ?
        USING *,R10
        STM     R0,R15,IOSAVE        SAVE GPR
        MVC     SAVE+40(4),NUCRSV2   MOVE R10
        LA      R13,SAVE              LOAD SAVE AREA ADDRESS
        SR      R6,R6                ZERO R6
        LH      R6,IOOPSW+2          INTERRUPTING DEVICE
        L       R11,=V(ADDRESS)      ADDRESS BLOK
        USING DADDR,R11
        L       R4,LINKS              LINKS TABLE
        USING DLINK,R4
        L       R2,LNLEN              LENGTH OF LINKS TABLE
        L       R3,LENEND             END OF LINKS TABLE

FNDLP   EQU     *
        CH      R6,INCTCA            IS IT THIS DEVICE ?
        BE      FNDIN                YES, IT IS AN INPUT DEVICE.
        CH      R6,OUTCTCA           IS IT THIS DEVICE ?
        BE      FNDOUT               YES, IT IS AN OUTPUT DEVICE.
        BXLE    R4,R2,FNDLP          TRY THE NEXT LINK
        B       IOEXIT               NO LINKS FOUND. IGNORE

FNDOUT  EQU     *
        CLI     OUTSTAT,IOWAIT       AM I WAITING FOR AN I/O
        BNE     IOERR                NO,ERROR.
        TM      CSW+4,B'0000100'     DEVICE END ?
        BC      14,IOERR              NO,ERROR.
        MVI     OUTSTAT,IDLE          I/O COMPLETE
        B       IOEXIT               EXIT

FNDIN   EQU     *
        CLI     INSTAT,IOWAIT        AM I WAITING FOR AN I/O
        BNE     CHKATT                NO,CHECK FOR ATTENTION

```


TM	CSW+4,B'0000100'	DEVICE END ?
BC	14,IOERR	NO,ERROR.
L	R6,INBUF	INPUT BUFFER ADDRESS
USING	DMSG+MSGQHLN,R6	
L	R7,MSGLEN	LENGTH OF THE TEXT
LA	R7,16(R7)	LENGTH OF THE MESSAGE BLOCK
LR	R9,R7	DESTINATION DATA LENGTH
LR	R0,R7	LOAD DATA LENGTH
A	R0,=F'7'	ADD SEVEN
SRA	R0,3	FIND DOUBLEWORDS
	DMSFREE DWORDS=(0)	RESERVE STORAGE FOR MESSAGE BLOK
USING	DMSG,R1	
LA	R8,MSGDNOD	DESTINATION LOCATION
MVCL	R8,R6	MOVE DATA
MVI	INSTAT,IDLE	I/O COMPLETE
L	R4,LOCAL	LOCAL CONTROL BLOK
USING	DLOC,R4	
LA	R2,NQUETOP	TOP OF NETWORK LAYER QUEUE
LA	R3,NQUEBOT	BOTTOM OF NETWORK LAYER QUEUE
L	R15,AQUEMSG	QUEUE THIS MESSAGE
BALR	R14,R15	
B	IOEXIT	QUIT
CHKATT	EQU *	
*		
IOERR	EQU *	
	HIO 0(R6)	HALT I/O
	TIO 0(R6)	CLEAR PENDING INTERRUPTS
IOEXIT	EQU *	
	LM R0,R15,SAVE	
	LPSW RETPSW	
	DS 0D	
RETPSW	DC AL4(0),V(MAINCNT)	
IOSAVE	DS 24F	


```

SAVE    DS    24F    TIME LOOPS THE NETWORK LAYER QUEUE
*
*    PRINT NOGEN LINK LAYER
*    REGEQU
*    COPY DADDR
*    COPY DLINK
*    COPY DLOC
*    COPY DMSG
*    NUCON
*    END
LOOP    EQU    *
        LA    R2, REQUEST
        LA    R1, REQUEST
        L      R15, APOPNED
        BNE   R14, R15
        C      R1, NULL
        BE    EXIT
        USING DMSG, R1
        CLC   R5, LOCALID
        BE    LCLASS
        L      R1, ROUTES
        USING ROUTES, R1
        L      R2, RLEN
        LA    R1, RTEND
        FWDOUT EQU    *
        CLC   R5, R5, LONGID
        BE    ROUTED
        BNE   R5, R2, FWDOUT
        L      R15, R5, LONGID
        BNE   R14, R15
        B     LOOP
ROUTED  EQU    *
        LD    R5, R5, LONGID

```



```

*      THIS SUBROUTINE LOOKS THE NETWORK LAYER QUEUE
*      AND DIRECT MESSAGES TO THE SESSION LAYER
*      OR THE DATA LINK LAYER
*
NETSEND  CSECT
        SAVEG R10,SAVE          SAVE GPR
        L      R11,=V(ADDRESS)  ADDRESS BLOK
        USING DADDR,R11
        L      R4,LOCAL          LOAD CONTROL BLOK ADDRESS
        USING DLOC,R4

LOOP     EQU      *              LINK CHECK LOOP
        LA     R2,NQUETOP        TOP OF QUEUE
        LA     R3,NQUEBOT        BOTTOM OF QUEUE
        L      R15,APOPMSG       POP A MESSAGE FROM QUEUE
        BALR   R14,R15
        C      R1,NULL           IS QUEUE EMPTY ?
        BE     EXIT              YES, EXIT
        USING DMSG,R1            R1 POINTS MESSAGE BLOK
        CLC    MSGDNOD,LOCALID   IS IT LOCAL
        BE     LCLMSG            YES, SEND IT TO SESSION LAYER
        L      R5,ROUTES         LOAD ROUTE TABLE ADDRESS
        USING DROUTE,R5
        L      R2,RTLEN          LENGTH OF ROUTE ENTRY
        LA     R3,RTEND          END OF ROUTES TABLE

FNDROUT  EQU      *
        CLC    NODENM,MSGDNOD    FOUND ?
        BE     ROUTOK            YES, GO ON.
        BXLE   R5,R2,FNDROUT     TRY THE NEXT ONE
        L      R15,AKILMSG       LINK COULDNT BE FOUND
        BALR   R14,R15           DESTROY THIS MESSAGE
        B      LOOP

ROUTOK   EQU      *
        LD     R6,LINKS          LOAD LINK TABLE ADDRESS

```



```

      USING DLINK,R6
      L   R2, LNLEN      LENGTH OF LINK ENTRY
      LA  R3, LNEND      END OF LINKS TABLE

FNDLINK EQU *
      CLC LINKNM, LINKID  FOUND ?
      BE LINKOK           YES, GO ON.
      BXLE R6, R2, FNDLINK TRY NEXT ONE
      L   R15, AKILMSG    LINK COULDNT BE FOUND
      BALR R14, R15       DESTROY THIS MESSAGE
      B   LOOP

LINKOK EQU *
      LA  R2, LQUETOP     TOP OF LINKS QUEUE
      LA  R3, LQUEBOT     BOTTOM OF LINKS QUEUE
      L   R5, AQUEMSG     QUEUE THIS MESSAGE
      BALR R14, R15
      B   LOOP

LCLMSG EQU *
      LA  R2, SQUETOP     TOP OF SESSIONS QUEUE
      LA  R3, SQUEBOT     BOTTOM OF SESSIONS QUEUE
      L   R5, AQUEMSG     QUEUE THIS MESSAGE
      BALR R14, R15

EXIT   RESTEX SAVE
NULL   DC   F'0'
*
      REGEQU
      COPY DADDR
      COPY DLINK
      COPY DROUTE
      COPY DLOC
      COPY DMSG
      END

```



```

*      THIS SUBROUTINE LOOKS THE SESSION LAYER QUEUE
*      AND DIRECT MESSAGES TO THE PRESENTATION LAYER
*      VIA IUCV
*
SESSEND  CSECT
          SAVEG R10,SAVE          SAVE GPR
          L     R11,=V(ADDRESS)    ADDRESS BLOK
          USING DADDR,R11
          L     R4,LOCAL            LOAD CONTROL BLOK ADDRESS
          USING DLOC,R4
          USING IPARML,R1
LP1      EQU    *                  LINK CHECK LOOP
          LA    R2,SQUETOP          TOP OF QUEUE
          LA    R3,SQUEBOT          BOTTOM OF QUEUE
          L     R15,APOPMMSG        POP A MESSAGE FROM QUEUE
          BALR  R14,R15
          C     R1,NULL             IS QUEUE EMPTY ?
          BE    EXIT               YES, EXIT
          LR    R7,R1
          USING DMSG,R7            R7 POINTS MESSAGE BLOK
          LH    R1,MSGDSES          SESSION ID OF THE MESSAGE
          LA    R2,SCHNTOP          SESSION CHAIN TOP
          LA    R2,SCHNBOT          SESSION CHAIN BOTTOM
          L     R15,AFNDSSES        FIND THIS SESSION
          BALR  R14,R15
          C     R1,NULL             FOUND ?
          BE    CHKCON              NO, IT MAY BE CONNECT REQUEST
          LR    R6,R1
          USING DSESS,R6
CHKACC   CLI    MSGTYPE,ACCEPT     IS IT ACCEPT ?
          BNE   CHKSND              NO, CHECK FOR SEND
          MVC    SESRID,MSGSSES     STORE REMOTE SESSION ID
          IUCV   ACCEPT,PATHID=SESLID,PRMLIST=IUCVPARM

```


	B	DESTMSG	
CHKSND	CLI	MSGTYPE,SEND	IS IT SEND ?
	BNE	CHKSVR	NO, CHECK FOR SEVER
	IUCV	SEND,BUFFER=MSGTEXT,BUFLEN=MSGLEN,PATHID=SESLID,	X
		PRMLIST=IUCVPARM,TYPE=1WAY	
	B	DESTMSG	
CHKSVR	CLI	MSGTYPE,SEVER	IS IT SEVER ?
	BNE	DESTMSG	NO, DON'T CARE
	IUCV	SEVER,PATHID=SESLID,PRMLIST=IUCVPARM	
	LA	R2,SCHNTOP	TOP OF SESSION CHAIN
	LA	R3,SCHNTOP	BOTTOM OF SESSION CHAIN
	L	R15,AKILSES	DESTROY THIS SESSION
	BALR	R14,R15	
	B	DESTMSG	
CHKCON	CLI	MSGTYPE,CONNECT	IS IT CONNECT ?
	BNE	DESTMSG	NO, DON'T CARE
	LA	R0,SESBLN+7	LOAD SESSION BLOCK LENGTH PLUS SEVEN
	SRA	R0,3	FIND DOUBLEWORDS
		DMSFREE DWORDS=(0)	
	LR	R6,R1	SESSION BLOCK ADDRESS
		USING DSESS,R6	
	MVC	SESLUSR,MSGDUSR	LOCAL USERID
	MVC	SESRNOD,MSGSNOD	REMOTE NODE
	MVC	SESRUSR,MSGSUSR	REMOTE USERID
	MVC	SESRID,MSGSSSES	REMOTE SESSION ID (UNKNOWN)
	L	R0,SBUFLN	SESSION LAYER BUFFER LENGTH
	ST	R0,SESRBLN	LENGTH OF RECEIVE BUFFER
	A	R0,=F'7'	ADD SEVEN
	SRA	R0,3	FIND DOUBLEWORDS
		DMSFREE DWORDS=(0)	
	ST	R1,SESRECB	SESSION RECEIVE BUFFER
	LR	R1,R6	LOAD ADDRESS OF SESSION BLOK
	LA	R2,SCHNTOP	TOP OF SESSION CHAIN

	LA	R3, SCHNBOT	BOTTOM OF SESSION CHAIN
	L	R15, APUTSES	PUT IT IN THE CHAIN
	BALR	R14, R15	
	IUCV	CONNECT, USERID=SESLUSR, PRMLIST=IUCVPARM	
DESTMSG	EQU	*	
	LR	R1, R7	MESSAGE BLOCK ADDRESS
	L	R15, AKILMSG	DESTROY THIS MESSAGE
	BALR	R14, R15	
	B	LP1	TAKE THE OTHER MESSAGE
EXIT	RESTEX		SAVE
NULL	DC	F'0'	
	DS	OD	
IUCVPARM	DS	CL40	
	*		
	REGEQU		
	COPY	DADDR	
	COPY	DSESS	
	COPY	DLOC	
	COPY	DMSG	
	COPY	IPARML	
	END		
	LA	R4, R1	SESSION BLOCK ADDRESS
	USING	DSESS, R4	
	MVC	SESLUSR, IPVNI0	LOCAL USERID
	MVC	SESLID, IPPATNID	LOCAL SESSION ID
	MVC	SESRMOD, IPUSER	REMOTE MODE
	MVC	SESRUSR, IPUSER+4	REMOTE USERID
	MVC	SESRID, +R'0'	REMOTE SESSION ID (UNKNOWN)
	L	R0, DBUFLN	SESSION LAYER BUFFER LENGTH
	BT	R0, DBUFLN	LENGTH OF RECEIVED BUFFER
	A	R0, +7	ADD SEVEN

EXTIHND	CSECT	
	USING NUCON,0	
	ST R10,NUCRSV2	SAVE R10
	BALR R10,0	WHERE AM I ?
	USING *,R10	
	STM R0,R15,EXTSAVE	SAVE GPR
	MVC SAVE+40(4),NUCRSV2	MOVE R10
	LA R13,SAVE	LOAD SAVE AREA ADDRESS
	LH R6,EXTOPSW+2	INTERRUPT CODE
	CH R6,=X'2000'	IS IT AN IUCV INTERRUPT ?
	BNE EXIT	NO. DON'T CARE
	L R11,=V(ADDRESS)	ADDRESS BLOC
	USING DADDR,R11	
	L R4,EXBADDR	EXTERNAL INTERRUPT BUFFER
	USING IPARML,R4	
	L R5,LOCAL	LOCAL CONTROL BLOC
	USING DLOC,R5	
	CLI IPTYPE,X'01'	IS IT FOR CONNECTION REQUEST ?
	BNE FINDIT	NO,FIND THE SESSION INFORMATION.
CONREQ	EQU *	
	LA R0,SESBLN+7	LOAD SESSION BLOCK LENGTH PLUS SEVEN
	SRA R0,3	FIND DOUBLEWORDS
	DMSFREE DWORDS=(0)	
	LR R6,R1	SESSION BLOCK ADDRESS
	USING DSESS,R6	
	MVC SESLUSR,IPVMID	LOCAL USERID
	MVC SESLID,IPPATHID	LOCAL SESSION ID
	MVC SESRNOD,IPUSER	REMOTE NODE
	MVC SESRUSR,IPUSER+8	REMOTE USERID
	MVC SESRID,=H'0'	REMOTE SESSION ID (UNKNOWN)
	L R0,SBUFLN	SESSION LAYER BUFFER LENGTH
	ST R0,SESRBLN	LENGTH OF RECEIVE BUFFER
	A R0,=F'7'	ADD SEVEN


```

SRA    R0,3          FIND DOUBLEWORDS
DMSFREE DWORDS=(0)

ST     R1,SESRECB    SESSION RECEIVE BUFFER
LR     R1,R6          LOAD ADDRESS OF SESSION BLOK
LA     R2,SCHNTOP     TOP OF SESSION CHAIN
LA     R3,SCHNBOT     BOTTOM OF SESSION CHAIN
L      R15,APUTSES    PUT IT IN THE CHAIN
BALR   R14,R15
LA     R0,MSGHDRL+8+7 MESSAGE BLOCK LENGTH+7
SRA    R0,3          FIND DOUBLEWORDS
DMSFREE DWORDS=(0)

LR     R8,R1
USING DMSG,R8

MVC    MSGDNOD,SESRNOD DESTINATION NODE
MVC    MSGDSES,SESRID  DESTINATION SESSION ID
MVC    MSGSNOD,LOCALID SOURCE NODE
MVC    MSGSSES,SESLID  SOURCE SESSION ID
MVI    MSGTYPE,CONNECT THIS IS A CONNECT REQUEST
MVC    MSGLEN,=AL4(CONRLN) LENGTH OF PARAMETERS
MVC    MSGDUSR,SESRUSR  DESTINATION USERID
MVC    MSGSUSR,SESLUSR  SOURCE USERID
LA     R2,NQUETOP      NETWORK LAYER QUEUE TOP
LA     R3,NQUEBOT      NETWORK LAYER QUEUE BOTTOM
L      R15,AQUEMSG      PUT THIS MESSAGE IN +
BALR   R14,R15          THE NETWORK LAYER QUEUE
B      EXIT
FINDIT EQU    *
LA     R2,SCHNTOP      TOP OF SESSION CHAIN
L      R15,AFNDSES      FIND THIS SESSION
BALR   R14,R15
C      R1,NULL          ANY SESSION EXIST ?
BNE    CHKSVR          NO, LOOK WHAT THIS IS.
*      CHKACC          SEVER THIS UNKNOWN PATH !

```



```

MVC    SAVEPATH,IPPATHID    LOAD MESSAGE LENGTH
PUSH   USING    CAN IT FIT OUR RECEIVE BUFFERS ?
USING  IPARML,R1    NO, REJECT THIS REQUEST.
IUCV   SEVER,PATHID=SAVEPATH,PRMLIST=IUCVPARM
POP    USING
B      EXIT
CHKSVR EQU    *
CLI    IPTYPE,X'03'    IS IT SEVER REQUEST ?
BNE    CHKSND    NO, CHECK FOR SEND REQUEST ?
LA     R0,MSGHDRL+7    MESSAGE HEADER LENGTH PLUS SEVEN
SRA    R0,3    FIND DOUBLEWORDS
DMSFREE DWORDS=(0)    ADD TEXT LENGTH
LR     R8,R1    FIND DOUBLEWORDS
USING  DMSG,R8
MVC    MSGDNOD,SESRNOD    DESTINATION NODE
MVC    MSGDSSES,SESRID    DESTINATION SESSION ID
MVC    MSGSNOD,LOCALID    SOURCE NODE
MVC    MSGSSSES,SESLID    SOURCE SESSION ID
MVI    MSGTYPE,SEVER    THIS IS A SEVER REQUEST
MVC    MSGLEN,=F'0'    NO TEXT
LA     R2,NQUETOP    NETWORK LAYER QUEUE TOP
LA     R3,NQUEBOT    NETWORK LAYER QUEUE BOTTOM
L      R15,AQUEMSG    PUT THIS MESSAGE IN +
BALR   R14,R15    THE NETWORK LAYER QUEUE,
LR     R1,R6    LOAD ADDRESS OF SESSION BLOK
LA     R2,SCHNTOP    TOP OF SESSION CHAIN
LA     R3,SCHNBOT    BOTTOM OF SESSION CHAIN
L      R15,AKILMSG    DESTROY THIS SESSION
BALR   R14,R15
B      EXIT
CHKSND EQU    *
CLI    IPTYPE,X'09'    IS IT SEND REQUEST ?
BNE    CHKACC    NO, CHECK FOR ACCEPT

```



```

L      R7,IPBFLN1F      LOAD MESSAGE LENGTH
C      R7,SESRBLN      CAN IT FIT OUR RECEIVE BUFFER ?
BH     REJMSG          NO, REJECT THIS REQUEST.
*      *              RECEIVE THE MESSAGE.
MVC    SAVEMSGI,IPMSGID  MOVE LOCAL SESSION ID
PUSH   USING           MESSAGE HEADER LENGTH PLUS SEVEN
USING  IPARML,R1        FIND DOUBLEWORDS
IUCV   RECEIVE,BUFFER=SESRECB,BUFLEN=SESRBLN,
      MSGID=SAVEMSGI,PRMLIST=IUCVPARM
POP    USING
LA     R0,MSGHDRL+7      MESSAGE HEADER LENGTH PLUS SEVEN
AR     R0,R7            ADD TEXT LENGTH
SRA    R0,3            FIND DOUBLEWORDS
DMSFREE DWORDS=(0)
LR     R8,R1            THIS IS A ACCEPT REQUEST
USING  DMSG,R8          NO TEXT
MVC    MSGDNOD,SESRNOD  DESTINATION NODE
MVC    MSGDSES,SESID    DESTINATION SESSION ID
MVC    MSGSNOD,LOCALID  SOURCE NODE
MVC    MSGSSES,SESLID   SOURCE SESSION ID
MVI    MSGTYPE,SEND     THIS IS A SEND REQUEST
ST     R7,MSGLEN        MESSAGE TEXT LENGTH
LA     R2,NQUETOP       NETWORK LAYER QUEUE TOP
LA     R3,NQUEBOT       NETWORK LAYER QUEUE BOTTOM
IUCV   L      R15,AQUEMSG PUT THIS MESSAGE IN +
      BALR   R14,R15    THE NETWORK LAYER QUEUE
      B      EXIT
REJMSG EQU    *
MVC    SAVEMSGI,IPMSGID
PUSH   USING
USING  IPARML,R1
IUCV   REJECT,MSGID=SAVEMSGI,PRMLIST=IUCVPARM
POP    USING

```



```

      B EQU EXIT
CHKACC EQU *
      CLI IPTYPE,X'02'      IS IT ACCEPT ?
      BNE EXIT              NO, EXIT
      MVC SESLID,IPPATHID   MOVE LOCAL SESSION ID
      LA  R0,MSGHDRL+7      MESSAGE HEADER LENGTH PLUS SEVEN
      SRA R0,3              FIND DOUBLEWORDS
      DMSFREE DWORDS=(0)
      LR  R8,R1
      USING DMSG,R8
      MVC MSGDNOD,SESRNOD   DESTINATION NODE
      MVC MSGDSES,SESRID    DESTINATION SESSION ID
      MVC MSGSNOD,LOCALID   SOURCE NODE
      MVC MSGSSES,SESLID    SOURCE SESSION ID
      MVI MSGTYPE,ACCEPT    THIS IS A ACCEPT REQUEST
      MVC MSGLEN,=F'0'      NO TEXT
      LA  R2,NQUETOP        NETWORK LAYER QUEUE TOP
      LA  R3,NQUEBOT        NETWORK LAYER QUEUE BOTTOM
      L   R15,AQUEMSG        PUT THIS MESSAGE IN +
      BALR R14,R15          THE NETWORK LAYER QUEUE
EXIT EQU *
      LM  R0,R15,SAVE
      LPSW RETPSW
      DS  0D
IUCVPARM DS CL40
RETPSW DC AL4(0),V(MAINCNT)
EXTSAVE DS 24F
SAVE DS 24F
NULL DC F'0'
SAVEMSGI DS F
SAVEPATH DS H

```

*

PRINT NOGEN


```

REGEQU ROUTINE PUTS A SESSION BLOCK
COPY DADDR OF THE CHAIN
COPY DSESS
COPY DLOC NEXT. R1: SESSION BLOCK ADDRESS
COPY DMSG R2: TOP OF CHAIN POINTER ADDRESS
COPY IPARML R3: BOTTOM OF CHAIN POINTER ADDRESS
PUTTES NUCON
END
    EC R10,SAVE      SAVE SPN
    USING DSESS,R1    SESSION BLOCK POINTER=RT
    MVC NEXT, NULL    THERE IS NO SESSION AFTER THAT
    L    R4,0(R3)      LOAD BOTTOM POINTER
    C    R4, NULL      IS IT NULL ?
    BNE PUTBOT        NO, PUT IT ON THE BOTTOM
    MVC NEXT, NULL     THERE IS NO SESSION BEFORE THAT
    ST   R1,0(R2)      THIS IS THE ONLY SESSION -
    ST   R1,0(R3)      ON THE CHAIN
    B    QUIT          QUIT
PUTBOT ST   R4,0(R3)    THE SESSION BEFORE
    ST   R1,0(R3)      CHANGE THE BOTTOM POINTER
    USING DSESS,R4
    ST   R1,NEXT       ESTABLISH POINTED CHAIN
QUIT   NEXT,SAVE      RESTORE SPN AND QUIT
NULL   DC   '0'

```

REGEQU
 COPY DSESS
 END

* THIS SUBROUTINE PUTS A SESSION BLOCK
 * ON THE BOTTOM OF THE CHAIN
 *

* INPUT PARAMETERS: R1: SESSION BLOCK ADDRESS
 * R2: TOP OF CHAIN POINTER ADDRESS
 * R3: BOTTOM OF CHAIN POINTER ADDRESS

PUTSES	CSECT	
	SAVEG R10,SAVE	SAVE GPR
	USING DSESS,R1	SESSION BLOCK POINTER=R1
	MVC SESNEXT,NULL	THERE IS NO SESSION AFTER THAT
	L R4,0(R3)	LOAD BOTTOM POINTER
	C R4,NULL	IS IT NULL ?
	BNE PUTBOT	NO, PUT IT ON THE BOTTOM.
	MVC SESPREV,NULL	THERE IS NO SESSION BEFORE THAT
	ST R1,0(R2)	THIS IS THE ONLY SESSION +
	ST R1,0(R3)	ON THE CHAIN
	B QUEXIT	QUIT
PUTBOT	ST R4,SESPREV	THE SESSION BEFORE
	ST R1,0(R3)	CHANGE THE BOTTOM POINTER
	USING DSESS,R4	
	ST R1,SESNEXT	ESTABLISH FORWARD CHAIN
QUEXIT	RESTEX SAVE	RESTORE GPR AND QUIT
NULL	DC F'0'	
*		
	REGEQU	
	COPY DSESS	
	END	


```

*      THIS SUBROUTINE FINDS A SESSION BLOCK
*      ON THE BOTTOM OF THE CHAIN
*
*      INPUT PARAMETERS:  R1: SESSION ID
*                        R2: TOP OF CHAIN POINTER ADDRESS
*                        R3: BOTTOM OF CHAIN POINTER ADDRESS
*      OUTPUT PARAMETERS: R1: SESSION BLOCK ADDRESS
*
FNDSES  CSECT  DSESS,R1      SESSION BLOCK POINTER=R1
        SAVEG R10,SAVE      SAVE GPR
        USING DSESS,R1      SESSION BLOCK POINTER=R1
        LR     R4,R1        LOAD SESSION ID
        L      R1,0(R2)     TOP OF THE CHAIN
LOOP    C      R1,NULL      END OF CHAIN ?
        BE     EXIT        YES, COULDNT FIND
        CH     R4,SESLID    IS IT THIS BLOK ?
        BE     EXIT        YES, WE GOT IT
        L      R1,SESNEXT   TRY NEXT ONE
        B LOOP
EXIT    L      R4,SAVE+4    LOAD OLD SAVE AREA
        ST     R1,24(R4)    REPLACE R1
        RESTEX SAVE        RESTORE GPR AND QUIT
NULL   DC     F'0'
*      LA     R3,SESLLEN-7  SESSION BLOCK LENGTH -7
        REGEQU
        COPY DSESS
        END
        DC     F'0'
        REGEQU
        COPY DSESS
        END

```



```

*      THIS SUBROUTINE REMOVES A SESSION BLOCK
*      FROM THE CHAIN
*
*      INPUT PARAMETERS:  R1: SESSION BLOCK ADDRESS
*                        R2: TOP OF CHAIN POINTER ADDRESS
*                        R3: BOTTOM OF CHAIN POINTER ADDRESS
KILLSES  CSECT
          SAVEG R10,SAVE          SAVE GPR
          USING DSESS,R1          SESSION BLOCK POINTER=R1
          L      R4,SESPREV(R1)    BACKWARD CHAIN
          L      R5,SESNEXT(R1)    FORWARD CHAIN
          C      R4,NULL           ANY PREVIOUS SESSION ?
          BE     NOPREV            NO, THERE IS NOT
          USING DSESS,R4          SESSION BLOCK POINTER=R4
          ST     R5,SESNEXT        ESTABLISH FORWARD CHAIN
          B      NEXT             LOOK FOR BACKWARD CHAIN
NOPREV    ST     R5,0(R2)          THIS MUST BE AT THE TOP
NEXT      C      R5,NULL          ANY NEXT SESSION ?
          BE     NONEXT           NO, THERE IS NOT
          USING DSESS,R5          SESSION BLOCK POINTER=R5
          ST     R4,SESPREV        ESTABLISH BACKWARD CHAIN
          B      KILL             DESTROY THIS SESSION BLOC
NONEXT    ST     R4,0(R3)          THIS MUST BE AT THE TOP
KILL      LA     R0,SESLEN+7       SESSION BLOCK LENGTH +7
          SRA    R0,3             FIND DOUBLEWORDS
          DMSFRET DWORDS=(0),LOC=(1)
          RESTEX SAVE
NULL      DC     F'0'
*
          REGEQU
          COPY DSESS
          END

```


* THIS SUBROUTINE PUTS A MESSAGE BLOCK
 * ON THE BOTTOM OF THE QUEUE
 *

* INPUT PARAMETERS: R1: MESSAGE BLOCK ADDRESS
 * R2: TOP OF QUEUE POINTER ADDRESS
 * OUTPUT PARAMETERS: R3: BOTTOM OF QUEUE POINTER ADDRESS

QUEMSG CSECT
 SAVEG R10,SAVE SAVE GPR
 USING DMSG,R1 MESSAGE BLOCK POINTER=R1
 MVC MSGNEXT,NULL THERE IS NO MESSAGE AFTER THAT
 L R4,0(R3) LOAD BOTTOM POINTER
 C R4,NULL IS IT NULL ?
 BNE PUTBOT NO, PUT IT ON THE BOTTOM.
 MVC MSGPREV,NULL THERE IS NO MESSAGE BEFORE THAT
 ST R1,0(R2) THIS IS THE ONLY MESSAGE +
 ST R1,0(R3) ON THE QUEUE
 B QUEXIT QUIT
 PUTBOT ST R4,MSGPREV THE MESSAGE BEFORE
 ST R1,0(R3) CHANGE THE BOTTOM POINTER
 USING DMSG,R4
 ST R1,MSGNEXT ESTABLISH FORWARD CHAIN
 QUEXIT RESTEX SAVE RESTORE GPR AND QUIT
 NULL DC F'0'
 *
 REGEQU
 COPY DMSG
 END


```

*      THIS SUBROUTINE POPS A MESSAGE BLOCK
*      FROM THE TOP OF THE QUEUE
*      OUTPUT PARAMETERS:  R1: MESSAGE BLOCK ADDRESS
*      INPUT PARAMETERS:  R2: TOP OF QUEUE POINTER ADDRESS
*                        R3: BOTTOM OF QUEUE POINTER ADDRESS
*      OUTPUT PARAMETERS: R1: MESSAGE BLOCK ADDRESS
*                        (NULL MEANS THERE IS NO MESSAGE)
POPMSG  CSECT
        SAVEG R10,SAVE          SAVE GPR
        USING DMSG,R1           MESSAGE BLOCK POINTER=R1
        L      R1,0(R2)         LOAD TOP POINTER
        C      R1,NULL          IS IT NULL ?
        BE     EXIT             YES, THERE IS NO MESSAGE ON THE QUEUE
        L      R4,MSGNEXT       NEXT MESSAGE
        ST     R4,0(R2)         IT WILL BE THE TOP
        C      R4,NULL          IS QUEUE EMPTY ?
        BE     EMPTY           YES, CHANGE BOTTOM POINTER.
        MVC    MSGPREV(R4),NULL PUT NULL TO BACKWARD CHAIN
        B      EXIT
EMPTY    MVC    0(R3),NULL       PUT NULL TO BOTTOM POINTER
EXIT     L      R4,SAVE+4        FIND THE LOCATION OF SAVE AREA
        ST     R1,24(R4)        CHANGE THE R1
        RESTEX SAVE             RESTORE GPR AND QUIT
NULL     DC     F'0'
*
        REGEQU
        COPY DMSG
        END

```



```

*   THIS SUBROUTINE DESTROYS A MESSAGE BLOCK
*
*   OUTPUT PARAMETERS:  R1: MESSAGE BLOCK ADDRESS
*
KILLMSG  CSECT
          SAVEG R10,SAVE          SAVE GPR
          USING DMSG,R1           MESSAGE BLOCK POINTER=R1
          LA    R0,MSGHDRL        LOAD HEADER LENGTH
          A     R0,MSGLEN         ADD TEXT LENGTH
          A     R0,=F'7'         ADD SEVEN
          SRA   R0,3             FIND DOUBLEWORDS
          DMSFRET DWORDS=(0),LOC=(1) FREE THIS STORAGE AREA
          RESTEX SAVE            RESTORE GPR AND QUIT
*
          REGEQU
          COPY DMSG
          END

```



```

TABLES      CSECT
            ENTRY ADDRESS
ROUTES      EQU      *
            ROUTE NODE1, LINK1
            ROUTE NODE2, LINK1
            ROUTE NODE3, LINK1
            ROUTE NODE4, LINK2
            ROUTE NODE5, LINK2
*
LINKS       EQU      *
            LINK  LINK1, 2048, 410, 420
            LINK  LINK2, 2048, 430, 440
*
LOCAL      LOCAL NODE0, 2048
*
ADDRESS     ADDRST ROUTES, LINKS, LOCAL
            END

```


DROUTE	DSECT		
	DS	0D	
NODENM	DS	CL8	NODE NAME
LINKNM	DS	CL8	LINK NAME
RTLEN	EQU	*-DROUTE	
INCCW	DS	B	
INCTCA	DS	ALL	
INSTAT	DS	ALL	
	MACRO		
&LABEL	ROUTE	&NODEID,&LINKID	
&LABEL	DS	0D	
	DC	CL8 '&NODEID'	NODE NAME
	DC	CL8 '&LINKID'	LINK NAME
	MEND		
LOUTTOP	DS	ALL	
LOUTBOT	DS	ALL	
LNLEN	EQU	*-RLINK	
LOUTATT	EQU	X'01'	
IBLE	EQU	X'00'	

DLINK	DSECT		
	DS	0D	
LINKID	DS	CL8	LINK NAME
INBUF	DS	AL4	INPUT BUFFER ADDRESS
INBUFLN	DS	AL4	INPUT BUFFER LENGTH
INCCW	DS	D	INPUT CCW
INCTCA	DS	AL2	INPUT CTCA ADDRESS
INSTAT	DS	XL1	INPUT STATUS
	DS	OF	
OUTBUF	DS	AL4	OUTPUT BUFFER ADDRESS
OUTBUFLN	DS	AL4	OUTPUT BUFFER LENGTH
OUTCCW	DS	D	OUTPUT CCW
OUTCTCA	DS	AL2	OUTPUT CTCA ADDRESS
OUTSTAT	DS	XL1	OUTPUT STATUS
	DS	OF	
LQUETOP	DS	AL4	TOP OF QUEUE
LQUEBOT	DS	AL4	BOTTOM OF QUEUE
LNLEN	EQU	*-DLINK	LENGTH OF LINK ENTRY
*			
IOWAIT	EQU	X'01'	
IDLE	EQU	X'00'	


```

MACRO
&LABEL LINK &LINKID,&BUFLen,&INDEV,&OUTDEV
&LABEL DS OD
&BUFLen DC CL8'&LINKID' LINK NAME
&INQUTOP DS AL4 INPUT BUFFER ADDRESS
&INQUTBOT DC AL4(&BUFLen) INPUT BUFFER LENGTH
&OUTQUTOP CCW X'02',0,X'20',0 INPUT CCW
&OUTQUTBOT DC XL2'&INDEV' INPUT CTCA ADDRESS
&INSTATOP DC XL1'00' INPUT STATUS
&INSTATBOT DS OF
&OUTSTATOP DS AL4 OUTPUT BUFFER ADDRESS
&OUTSTATBOT DC AL4(&BUFLen) OUTPUT BUFFER LENGTH
&OUTCCW CCW X'01',0,X'20',0 OUTPUT CCW
&OUTCTCATOP DC XL2'&OUTDEV' OUTPUT CTCA ADDRESS
&OUTCTCABOT DC XL1'00' OUTPUT STATUS
&OUTSTATBOT DS OF
&OUTSTATBOT DC AL4(0) TOP OF QUEUE
&OUTSTATBOT DC AL4(0) BOTTOM OF QUEUE
MEND

```


DLOC	DSECT	
	DS	0D
LOCALID	DS	CL8
SBUFLEN	DS	AL4
NQUETOP	DS	AL4
NQUEBOT	DS	AL4
SQUETOP	DS	AL4
SQUEBOT	DS	AL4
SCHNTOP	DS	AL4
SCHNBOT	DS	AL4
AINIT	DS	AL4
ARESET	DS	AL4
ALNENED	DS	AL4
ANENED	MACRO	AL4
&LABEL	LOCAL	&LOCALID,&SESLEN
&LABEL	DS	0D
EPOMSU	DC	CL8'&LOCALID'
ANILNSQ	DC	AL4(&SESLEN)
APUTSES	DC	AL4(0)
AFMUSE	DC	AL4(0)
AVILSES	DC	AL4(0)
EXITFLQ	DC	AL4(0)
	DC	AL4(0)
	DC	AL4(0)
	MEND	

DADDR	DSECT	
	DS	OD
ROUTES	DS	AL4
RTEND	DS	AL4
LINKS	DS	AL4
LNEND	DS	AL4
LOCAL	DS	AL4
MAXCONN	DS	AL4
EXBADDR	DS	AL4
EXBSIZE	DS	AL4
AINIT	DS	AL4
ARESET	DS	AL4
ALNKSND	DS	AL4
ANETSND	DS	AL4
ASESSND	DS	AL4
AQUEMSG	DS	AL4
APOPMSG	DS	AL4
AKILMSG	DS	AL4
APUTSES	DS	AL4
AFNDSES	DS	AL4
AKILSES	DS	AL4
EXITFLG	DS	XL1

MACRO

&LABEL ADDRST &RTLST,&LNKLST,&LOCAL

&LABEL DS OD

DC AL4(&RTLST)

DC AL4(&LNKLST-1)

DC AL4(&LNKLST)

DC AL4(&LOCAL-1)

DC AL4(&LOCAL)

DC AL4(0)

DC AL4(0)

DC AL4(0)

DC V(INIT)

DC V(RESET)

DC V(LNKSEND)

DC V(NETSEND)

DC V(SESSEND)

DC V(QUEMSG)

DC V(POPMSG)

DC V(KILLMSG)

DC V(PUTSES)

DC V(FNDSES)

DC V(KILLSES)

DC XL1'00'

MEND

DMSG	DSECT	
	DS	0D
MSGQUE	EQU	*
MSGNEXT	DS	AL4
MSGPREV	DS	AL4
MSGNET	EQU	*
MSGDNOD	DS	CL8
MSGDSES	DS	H
MSGSNOD	DS	CL8
MSGSSES	DS	H
MSGTYPE	DS	XL1
MSGLEN	DS	F
MSGTEXT	EQU	*
* THOSE ARE THE EXTENDED PARAMETERS FOR CONNECT REQUEST		
MSGDUSR	DS	CL8
MSGSUSR	DS	CL8
CONRLN	EQU	*-MSGTEXT
MSGHDRL	EQU	MSGTEXT-MSGQUE
MSGQHNL	EQU	MSGNET-MSGQUE
*		
* VALID MSGTYPES		
*		
CONNECT	EQU	X'01'
ACCEPT	EQU	X'02'
SEND	EQU	X'04'
SEVER	EQU	X'10'

NEXT MESSAGE
 PREVIOUS MESSAGE
 MESSAGE TRAVELLING THROUGH NETWORK
 DESTINATION NODE
 DESTINATION SESSION
 SOURCE NODE
 SOURCE SESSION
 MESSAGE TYPE
 SESSION STATUS
 SESSION BLOCK LENGTH
 DESTINATION USER
 SOURCE USER
 TOTAL LENGTH OF EXTENDED PARAMETERS

DSESS	DSECT		
SESSTRT	EQU	*	
SESNEXT	DS	AL4	NEXT SESSION ON THE CHAIN
SESPREV	DS	AL4	PREVIOUS SESSION ON THE CHAIN
SESLUSR	DS	CL8	LOCAL USER NAME
SESLID	DS	H	LOCAL SESSION ID
SESRNOD	DS	CL8	REMOTE NODE ID
SESRUSR	DS	CL8	REMOTE USER NAME
SESRID	DS	H	REMOTE SESSION ID
SESRECB	DS	AL4	RECEIVE BUFFER ADDRESS
SESRBLN	DS	AL4	RECEIVE BUFFER LENGTH
SESSTAT	DS	XL1	SESSION STATUS
SESBLN	EQU	*-SESSTRT	SESSION BLOCK LENGTH

*** IPARML - IUCV PARAMETER LIST AND EXTERNAL INTERRUPT MAPPING MACRO

IPATHID	DS	H		PATHID
IPFLAGS1	DS	X	1*1	FLAG BYTE
IPRCODE	DS	X	1*1	RETURN CODE
IPMSGID	DS	F		MESSAGE IDENTIFICATION
IPTRGCLS	DS	F		TARGET CLASS
IPRMSG1	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG2	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG3	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG4	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG5	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG6	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG7	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG8	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG9	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG10	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG11	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG12	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG13	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG14	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG15	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG16	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG17	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG18	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG19	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG20	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG21	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG22	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG23	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG24	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG25	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG26	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG27	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG28	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG29	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG30	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG31	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG32	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG33	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG34	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG35	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG36	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG37	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG38	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG39	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG40	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG41	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG42	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG43	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG44	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG45	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG46	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG47	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG48	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG49	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG50	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG51	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG52	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG53	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG54	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG55	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG56	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG57	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG58	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG59	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG60	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG61	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG62	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG63	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG64	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG65	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG66	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG67	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG68	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG69	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG70	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG71	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG72	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG73	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG74	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG75	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG76	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG77	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG78	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG79	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG80	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG81	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG82	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG83	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG84	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG85	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG86	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG87	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG88	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG89	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG90	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG91	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG92	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG93	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG94	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG95	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG96	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG97	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG98	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG99	DS	F		MESSAGE DATA IN PARAMETER LIST
IPRMSG100	DS	F		MESSAGE DATA IN PARAMETER LIST


```

IPBADDR1  PUSH  PRINT
IPBFLN1F  AIF  ('&SYSPARM' NE 'SUP').IPA01
          PRINT OFF,NOGEN
IPARML  DSECT  8
*-----+-----+-----+-----+-----+
*** IPARML - IUCV PARAMETER LIST AND EXTERNAL INTERRUPT MAPPING MACRO
*-----+-----+-----+-----+-----+
* 0 ! IPPATHID ! I*1 ! I*2 ! IPMSGID !
*-----+-----+-----+-----+-----+
* 8 ! IPTRGCLS ! IPBFADR1 !
*-----+-----+-----+-----+-----+
* 10 ! IPBFLN1F ! IPSRCCLS !
*-----+-----+-----+-----+-----+
* 18 ! IPMSGTAG ! IPBFADR2 !
*-----+-----+-----+-----+-----+
* 20 ! IPBFLN2F ! I*3 ! //////////////////// !
*-----+-----+-----+-----+-----+
* 28 ! //////////////////// ! IPNEXT !
*-----+-----+-----+-----+-----+
*** IPARML - IUCV PARAMETER LIST AND EXTERNAL INTERRUPT MAPPING MACRO
*-----+-----+-----+-----+-----+
IPPATHID DS  H  PATHID
IPFLAGS1 DS  X  I*1  FLAGS BYTE
IPRCODE DS  X  I*2  RETURN CODE
IPMSGID DS  F  MESSAGE IDENTIFICATION
IPTRGCLS DS  F  TARGET CLASS
IPRMSG DS  XL8  MESSAGE DATA IN PARAMETER LIST  $V6GN6G1
IPRMSG1 DS  F  FIRST FULLWORD OF PRMLIST DATA  $V6GN6G1
IPRMSG2 DS  F  SECOND FULLWORD OF PRMLIST DATA  $V6GN6G1
IPRMSG3 DS  F  MESSAGE DATA IN PARAMETER LIST  $V6GN6G1

```


IPBFADR1	DS	F	ADDRESS OF BUFFER	
IPBFLN1F	DS	F	FULLWORD LENGTH OF IPBFADR1	\$V6DJ1G1
	ORG	IPBFLN1F+2		\$V6DJ1G1
IPBFLN1	DS	H	HALFWORD LENGTH OF IPBFADR1	\$V6DJ1G1
IPSRCCLS	DS	F	SOURCE CLASS	
IPMSGTAG	DS	F	MESSAGE TAG	
IPBFADR2	DS	F	ADDRESS OF SECOND BUFFER	
IPBFLN2F	DS	F	FULLWORD LENGTH OF IPBFADR2	\$V6DJ1G1
	ORG	IPBFLN2F+2		\$V6DJ1G1
IPBFLN2	DS	H	HALFWORD LENGTH OF IPBFADR2	\$V6DJ1G1
IPNQFLG	DS	X	I*3 ENQUEUE/DEQUEUE FLAGS FOR CP PATH	\$V6MVFM0
	DS	XL3	RESERVED	\$V6MVFM0
*				
IPSIZE	EQU	(*-IPARML+7)/8 PARAMETER LIST SIZE IN DOUBLEWORDS		
				\$V6MVFM0
	DS	F	RESERVED	\$V6MVFM0
IPNEXT	DS	F	ADDRESS OF NEXT PENDING INTERRUPT	
*				
	ORG	IPARML+8		
IPVMID	DS	CL8	TARGET VIRTUAL MACHINE ID	
IPUSER	DS	CL16	USER DATA	
	ORG	IPARML		
IPMASK	DS	X	ENABLE MASK	
	ORG	IPARML		\$V6GN6G1
IPCMASK	DS	X	ENABLE CONTROL MASK	\$V6GN6G1
	ORG	IPARML+8		
IPAUDIT	DS	XL3	AUDIT TRAIL	\$V6KR3L5
	ORG	IPAUDIT	REDEFINE AUDIT FIELDS	
IPAUDIT1	DS	X	AUDIT TRAIL BYTE 1	
IPAUDIT2	DS	X	AUDIT TRAIL BYTE 2	
IPAUDIT3	DS	X	AUDIT TRAIL BYTE 3	\$V6KR3L5
	ORG	IPARML+4		
IPMSG LIM	DS	H	MESSAGE LIMIT	

IPFCNCD	DS	X	FUNCTION CODE	
	ORG	IPARML+3		
IPTYPE	DS	X	EXTERNAL INTERRUPT CODE	
	ORG	IPMSGID		\$V6MVFM0
IPCODE	DS	XL2	ERROR/SEVER CODE	\$V6MVFM0
IPFLAGS2	DS	XL1	APPC FLAGS	\$V6MVFM0
IPSENDOP	DS	XL1	SEND OPTION CODE	\$V6MVFM0
	ORG	IPFLAGS2		\$V6MVFM0
IPWHATRC	DS	XL1	WHAT RECEIVED CODE	\$V6MVFM0
	ORG	IPUSER		\$V6MVFM0
IPRESID	DS	XL8	TARGET RESOURCE ID	\$V6MVFM0
	ORG			
* BITS DEFINED IN IPCHASK				
IPBSIZE	EQU	(*-IPARML+7)/8	BUFFER SIZE IN DOUBLEWORDS	\$V6MVFM0
* POLCS				
* BITS DEFINED IN IPFLAGS1				
IPALL	EQU	X'80'	QUIESCE, RESUME, SEVER ALL	
IPRMDATA	EQU	X'80'	MESSAGE IS IN PARAMETER LIST	\$V6GN6G1
IPQUSCE	EQU	X'40'	CONNECT IN QUIESCE MODE	
IPBUFLST	EQU	X'40'	BUFFER LIST OPTION	\$V6KR3L5
IPPRTY	EQU	X'20'	PRIORITY MESSAGE OR REPLY	
IPNORPY	EQU	X'10'	ONE WAY PROTOCOL	
IPANSLST	EQU	X'08'	ANSWER LIST OPTION	\$V6KR3L5
IPAPPC	EQU	X'08'	APPC FUNCTION (EXCEPT FOR SEND)	\$V6MVFM0
IPFGMID	EQU	X'04'	MESSAGE ID SPECIFIED	
IPCNTL	EQU	X'04'	CONTROL OPTION SPECIFIED	\$V6MU3ND
IPFGPID	EQU	X'02'	PATH ID SPECIFIED	
IPAPPCSN	EQU	X'02'	APPC SEND FUNCTION	\$V6MVFM0
IPFGMCL	EQU	X'01'	MESSAGE CLASS SPECIFIED	
* POLCS				
* BITS DEFINED IN IPFLAGS2				
IPWAIT	EQU	X'80'	WAIT=YES SPECIFIED	\$V6MVFM0
IPLVLCF	EQU	X'40'	SEND CNF/SEND CNFD OK ON PATH	\$V6MVFM0

IPCOMSRV EQU X'20' CONNECT ON BEHALF OF ANOTHER VM \$V6MVFM0

*

* BITS DEFINED IN IPMASK

IPSNDN	EQU	X'80'	ENABLE FOR NON-PRIORITY MESSAGES
IPSNDP	EQU	X'40'	ENABLE FOR PRIORITY MESSAGES
IPRPYN	EQU	X'20'	ENABLE FOR NON-PRIORITY REPLIES
IPRPYP	EQU	X'10'	ENABLE FOR PRIORITY REPLIES
IPCTRL	EQU	X'08'	ENABLE FOR IUCV CONTROL INTERRUPTS
*	EQU	X'04'	RESERVED
*	EQU	X'02'	RESERVED
*	EQU	X'01'	RESERVED

*

* BITS DEFINED IN IPCMASK

IPCLPC	EQU	X'80'	ENABLE FOR PENDING CONNECTION	\$V6GN6G1
IPCLCC	EQU	X'40'	ENABLE FOR COMPLETE CONNECTION	\$V6GN6G1
IPCLPS	EQU	X'20'	ENABLE FOR SEVER INTERRUPT	\$V6GN6G1
IPCLPQ	EQU	X'10'	ENABLE FOR QUIESCE INTERRUPT	\$V6GN6G1
IPCLPR	EQU	X'08'	ENABLE FOR RESUME INTERRUPT	\$V6GN6G1
*	EQU	X'04'	RESERVED	
*	EQU	X'02'	RESERVED	
*	EQU	X'01'	RESERVED	

*

* BITS DEFINED IN IPAUDIT1

IPADRPLE	EQU	X'80'	REPLY TOO LONG FOR BUFFER	
IPADSNPX	EQU	X'40'	PROTECTION EXCEPTION ON SEND BUFFER	
IPADSNAX	EQU	X'20'	ADDRESSING EXCEPTION ON SEND BUFFER	
IPADANPX	EQU	X'10'	PROTECTION EXCEPTION ON ANSWER BUFFER	
IPADANAX	EQU	X'08'	ADDRESSING EXCEPTION ON ANSWER BUFFER	
IPADRJCT	EQU	X'04'	MESSAGE WAS REJECTED	
IPADPRMD	EQU	X'02'	REPLY SENT IN PARAMETER LIST	\$V6GN6G1
*	EQU	X'01'	RESERVED	

IPADMSK1 EQU IPADSNPX+IPADSNAX+IPADANPX+IPADANAX APPC MASK \$V6MVFM0

*

MASK OF VALID APPC AUDIT FLAGS

* IPTYPE EQU 0000 EXT INT TYPE - (INCOMING PRIORITY REPLY

* ITPRND BITS DEFINED IN IPAUDIT2 INT TYPE - INCOMING REPLY

IPADRCPX EQU	X'80'	PROTECTION EXCEPTION ON RECEIVE BUFFER	
IPADRCAX EQU	X'40'	ADDRESSING EXCEPTION ON RECEIVE BUFFER	
IPADRPPX EQU	X'20'	PROTECTION EXCEPTION ON REPLY BUFFER	
IPADRPAX EQU	X'10'	ADDRESSING EXCEPTION ON REPLY BUFFER	
IPADSVRD EQU	X'08'	PATH WAS SEVERED	
IPADRLST EQU	X'04'	INVALID RECEIVE/REPLY LIST	\$V6KR3L5
* ITPPCA EQU	X'02'	RESERVED	
* ITPPSA EQU	X'01'	RESERVED	
IPADMSK2 EQU	IPADRCPX+IPADRCAX+IPADRPPX+IPADRPAX+IPADRLST APPC MASK		
*		MASK OF VALID APPC AUDIT FLAGS	\$V6MVFM0

* VALUES DEFINED IN IPRND

* BITS DEFINED IN IPAUDIT3

IPADBLEN EQU	X'80'	BAD LENGTH IN SEND BUFFER LIST	\$V6KR3L5
IPADALEN EQU	X'40'	BAD LENGTH IN SEND ANSWER LIST	\$V6KR3L5
IPADBTOT EQU	X'20'	INVALID TOTAL SEND BUFFER LENGTH	\$V6KR3L5
IPADATOT EQU	X'10'	INVALID TOTAL SEND ANSWER LENGTH	\$V6KR3L5
IPADTINV EQU	X'08'	INVALID LRECL IN PARTNER'S DATA	\$V6MVFM0
IPADIINV EQU	X'04'	INVALID LRECL IN YOUR DATA	\$V6MVFM0
IPADTTRN EQU	X'02'	LOGICAL RECORD TRUNCATION BY PARTNER	
*			\$V6MVFM0
IPADITRN EQU	X'01'	LOGICAL RECORD TRUNCATION BY YOU	\$V6MVFM0
IPADMSK3 EQU	IPADBLEN+IPADALEN+IPADBTOT+IPADATOT+IPADTINV+IPADIINV+IPX		
	ADTTRN+IPADITRN MASK OF VALID APPC AUDIT FLAGS		
			\$V6MVFM0

*

* VALUES DEFINED IN IPTYPE

IPDATA SPACE

IPTYPE EQU	X'01'	EXT INT TYPE - PENDING CONNECTION	
IPTYPE EQU	X'02'	EXT INT TYPE - CONNECTION COMPLETE	
IPTYPE EQU	X'03'	EXT INT TYPE - SEVERED CONNECTION	
IPTYPE EQU	X'04'	EXT INT TYPE - QUIESCED CONNECTION	
IPTYPE EQU	X'05'	EXT INT TYPE - RESUMED CONNECTION	

IPTYPRP	EQU	X'06'	EXT INT TYPE - INCOMING PRIORITY REPLY	
IPTYPRNP	EQU	X'07'	EXT INT TYPE - INCOMING REPLY	
IPTYMP	EQU	X'08'	EXT INT TYPE - INCOMING PRIORITY MSG	
IPTYPMNP	EQU	X'09'	EXT INT TYPE - INCOMING MESSAGE	
*				
IPTYPPCA	EQU	X'81'	APPC INT - PENDING CONNECTION	\$V6MVFM0
IPTYPPCA	EQU	X'82'	APPC INT - CONNECTION COMPLETE	\$V6MVFM0
IPTYPSVA	EQU	X'83'	APPC INT - SEVER INTERRUPT	\$V6MVFM0
IPTYPFCA	EQU	X'87'	APPC INT - FUNCTION COMPLETE	\$V6MVFM0
IPTYPSRA	EQU	X'88'	APPC INT - SENDREQ INTERRUPT	\$V6MVFM0
IPTYMPA	EQU	X'89'	APPC INT - INCOMING MESSAGE	\$V6MVFM0
*				
* VALUES DEFINED IN IPCODE				
SPACE				
IPCDDFLT	EQU	X'0610'	DEFAULT SEVER CODE	\$V6MVFM0
			\$VA21273	
*				
* BITS DEFINED IN IPNQFLG				
IPNQNO	EQU	X'80'	NO ENQUEUE FOR THIS TRANSACTION	\$VA21273
IPDQEND	EQU	X'40'	PERFORM DEQUEUE	\$VA21273
IPDQNO	EQU	X'20'	NO DEQUEUE FOR THIS TRANSACTION	\$VA21273
*				
* VALUES DEFINED IN IPSENDOP				
* NOTE:				
* SOME OF THE EQUATES USED IN IPSENDOP ARE ALSO USED IN				
* IPWHATRC WITH SLIGHTLY DIFFERENT MEANINGS				
*				
	EQU	X'00'	RESERVED	\$V6MVFM0
IPDATA	EQU	X'01'	SENDATA RECEIVE=NO REQUESTED	\$V6MVFM0
IPSNDRCV	EQU	X'02'	SENDATA RECEIVE=YES REQUESTED	\$V6MVFM0
IPERROR	EQU	X'03'	SENDERR REQUESTED	\$V6MVFM0
IPCNFRM	EQU	X'04'	SENDCNF TYPE=NORMAL REQUESTED	\$V6MVFM0
IPCNFSEV	EQU	X'05'	SENDCNF TYPE=SEVER REQUESTED	\$V6MVFM0
IPCNFRMD	EQU	X'06'	SENDCNFD REQUESTED	\$V6MVFM0

IPREQSND EQU	X'07'	SENDREQ REQUESTED	\$V6MVFM0
IPSNORM EQU	X'08'	SEVER TYPE=NORMAL REQUESTED	\$V6MVFM0
IPSABEND EQU	X'09'	SEVER TYPE=ABEND REQUESTED	\$V6MVFM0
IPRECV EQU	X'0A'	RECEIVE FUNCTION BEING COMPLETED	\$V6MVFM0
IPALLOCD EQU	X'0B'	SEND OF THE ALLOCATE DATA	\$V6MVFM0
SPACE			
* VALUES DEFINED IN IPWHATRC			
IPCOMP EQU	X'00'	FUNCTION COMPLETED NORMALLY	\$V6MVFM0
*IPDATA EQU	X'01'	ONLY DATA WAS RECEIVED	\$V6MVFM0
IPSEND EQU	X'02'	INDICATES NOW IN SEND STATE	\$V6MVFM0
*IPERROR EQU	X'03'	PARTNER ISSUED SENDERR	\$V6MVFM0
*IPCMFRM EQU	X'04'	PARTNER ISSUED SENDCNF TYPE=NORM	\$V6MVFM0
*IPCNFSEV EQU	X'05'	PARTNER ISSUED SENDCNF TYPE=SEVER	\$V6MVFM0
*IPCNFRMD EQU	X'06'	PARTNER ISSUED SENDCNFD	\$V6MVFM0
* EQU	X'07'	RESERVED	\$V6MVFM0
*IPSNORM EQU	X'08'	PARTNER ISSUED SEVER TYPE=NORMAL	\$V6MVFM0
*IPSABEND EQU	X'09'	PARTNER ISSUED SEVER TYPE=ABEND	\$V6MVFM0
* EQU	X'0A'	RESERVED	\$V6MVFM0
*IPALLOCD EQU	X'0B'	ALLOCATE DATA BEING RECEIVED	\$V6MVFM0
EJECT			
POP PRINT			
SPACE			


```

MACRO
&LABEL  SAVEG &BASE,&PLACE
&LABEL  STM   R14,R11,12(R13)      STORE GENERAL REGISTERS
*
      THE CALLERS SAVE AREA
      BALR   &BASE,0                FIND THE CURRENT ADDRESS
      USING *,&BASE                 ESTABLISH ADDRESSIBILITY
      ST     R13,&PLACE+4           STORE CALLERS SAVE AREA ADDRESS
      LA     R13,&PLACE             LOAD NEW SAVE AREA ADDRESS
MEND

```

```

MACRO
&LABEL  RESTEX &PLACE
&LABEL  L     R13,&PLACE+4          FIND CALLERS SAVE AREA ADDRESS
      LM     R14,R11,12(R13)       RESTORE GENERAL REGISTERS
      BR     R14                   RETURN TO CALLER
&PLACE  DS    24F                 NEW SAVE AREA
MEND

```


KAYNAKLAR

- William Stallings, Data and Computer Communications
Macmillan Publishing Company, 1988
ISBN 0-02-415451-2
- Wushow Chou, Computer Communications
Prentice Hall Inc., 1983
ISBN 0-13-165043-2
- Andrew S. Tanenbaum, Computer Networks
Prentice Hall Inc., 1981
ISBN 0-13-164699-0
- IBM Corp. (SC19-6200) VM/SP Introduction
- IBM Corp. (SC19-6201) VM/SP Planning Guide and Reference
- IBM Corp. (SC19-6211) VM/SP CP Command Reference
- IBM Corp. (SC24-5236) VM/SP CMS Primer
- IBM Corp. (GA22-7000) S/370 Principles of Operation

ÖZGEÇMİŞ

1965 yılında, Diyarbakırda doğdu. Orta öğrenimini 1983 yılında Maçka Teknik Lisesinde tamamladı. Aynı yıl Yıldız Üniversitesi Bilgisayar Bilimleri Mühendisliği bölümünde lisans öğrenimine başladı. Lisans programını 1987 yılında bölüm ikincisi olarak bitirdi ve yine Yıldız Üniversitesinin Fen Bilimleri Enstitüsünde Bilgisayar Bilimleri dalında lisans eğitime başladı. 1987 yılından beri Yıldız Üniversitesi Bilgisayar Bilimleri Mühendisliği bölümünde araştırma görevlisidir.

