

Kesme (Interrupt) nedir?

1- Geçitli kriterler ve önceliklerle program işleyişinin kesintiye uprotector es zamanda çalışmaının sağlanması interrupt mekanizması ile sağlanır.

Diyelimki bizim tek çekirdekli bir işlemcimiz var
8086 tek çekirdek setinde çalışıyor mesela
bu işlemciyle anda sadece tek bir işlem yapabiliyoruz

yani siz müzik dinliyorsanız aynı anda internete-
de pişmezsiniz veya youtube'dan bir video izli-
yorsanız aynı anda 'mouse' hareket ettiremez-
siniz

Diyelimki 8 çekirdekli olsun işlemciniz fiziksel olarak
8 farklı işlemci den oluşuyor olsun. böyle bir durumda
aslında aynı anda maksimum 8 farklı işlem yapabiliyoruz

hem müzik dinlerken hem orta taraftaki
worddteki dosyayı nasıl doldurcaz iste bunun
için turulormuş geçitli mekanizmalar var
ama şimdi görer yöneticisini açtığınızda en
az 100 tane işlem yapıldığını görürsünüz
e peki 8 çekirdekli bunu nasıl yapar
işte bunun için turulormuş geçitli mekanizmalar
var interrupt mekanizması buna göre birisi

Nor. interrupt mekanizmaları birebir aynı

2-yavaş birimler ile hızlı birimler arasındaki
bağlantıyı sıklıkla zaman kaybetmeyi
engellemek için kullanılır

3-Gerre birimleri işlemci den daha yavaş

Girişir

mouse - Klavye

yazıcı

monitor

4-difer birimler

Ram

interrupt mekanizması

temel amacı ne?

yavaş birimler ile hızlı birimlerin
sentrü bir şekilde veya hızdan
tasarruf edicek şekilde çalışması
ne demek istiyorum

ekran 60hz → her saniye benim ekranımı 60 kez yeniliyor

işlemcinin horisindeki diğer donanım birimleri işlemci den
yavaş girişir. işlemci en hızlı giriş donanım birimi. disk'e
veri yazmak dahil. yani hard-disk'e veri yazıcınızı zaman
sadece işlemcisi kullanırsınız dahi hard-disk'in kapasitesi
%100 çalıştırınca yani ona daha fazla yüklenmeyecek durum-
dayken işlemciniz hala %8-%4'lerde işleyecek

İşlemcinin zamanını farklı donorim birimleri arasında bölmemiz gerekiyor.

İşlemcinin hizinde collision en hızlı donorim birimi RAM

İşlemci ile yarası birimler arasındaki ilişkisi

1- Busy waiting

2- polling

3- interrupt

↳ neyi garantilemez o zaman hizmet
almak istediginde zile basıyor
digeri de zile basıyor böyle herası
bir kuyruğa girer ve siz o kuyrupu
işleriniz o kuyrupu tüketmeye
golisirdiniz bazen yetişemezsiniz
iste o yetişemediginiz zaman o
kum saatı dönmeye baslar
bazen yetişebilirsin bazen lütfen
bekleyiniz der

Biz genelde interruptları 2'ye
ayırırız

1- Zaman kesmesi

2- grafik kesmesi

→ her bir milisaniye sonra
bilgisayarın zamanını + arttır

→ Bazı interruptlar vardır bunun
gibi geçikemezsin önceliklidir

Kesme Geçitleri

Temel kesmeler

- Donanım kesmeleri
- Yazılım kesmeleri

→ Bunu kendinizde üretebilirsiniz
yazılım kesmesi ne işe yorar
mesela siz programınızın içinde
etrafa bir şey yazdırırsanız istiyorsunuz
böyle bir durumda naptıksınız bir
kesme üreticeksiniz. ekranı kullanma
kesmesi üreticeksiniz. veya klavyeden
bir şeyler okumak istiyorsunuz.

interruptların enable ya da disable
olmasını saglayon bir interrupt
flagımız var

İobi işletim sistemlerinin interrupt
handler'ı var

interrupt handler ne demek → interruptlara
iletilen

Bütün bu istekleri ve donanım
isteklerini, yazılım isteklerini
biriktiren gerekirse bir kuyruğa
olan hatta bazılarda öncelik
mekanizması kullandırtabilen
bir interrupt handlerı var
işletim sistemlerinde

Bu bir yazılım

Bu bir yazılım ama bu
yazılımın bağlı olduğu
donanım birimleri de
var.

Bizim işlemcinin (8086)

40 tane bacağı var

2 tane interruptla

ilgili bacağı var

Sadece 2 bacağı

interruptla ilgili bu işlemcinin

Dahili kesmeler

lokal interrupt

Sifira bölme \rightarrow sifira böldüğünde otomatik olarak
bir kesme üretilir. İşlemciler o'a bölmek

adım adım çalıştırma

vb.

icin dizayn edilmediği
için sifira bölme işlemi
olursa diye hemen bir
kesme üretirler.

\rightarrow Mantıksal interruptlar
bunların bir kısmı

yazılımla bir türmi
donanımla da olabili
olabilir.

→ Vega dizin programınız
interrupt olarak çalışıyo
olabilir

→ Adım Adım Çalıştırmada
bir Geçit interrupttır

→ 8259A bütünülesik entegresi
Horici kesmeler → üzerinden sürdürüyor
maskelenemez kesmeler (NMI)
maskelenebilir kesmeler (INTR)
IF durumu önemli (interrupt flag)

→ Burda sen muhakkak o kitobın
İgine o ograçını koyıcak şin
gidip o işlemi yapıcaksın

→ Zaman kesmesi maskelenemez

→ Bozen mouse'in klavyen donabiliyor
yani bu işlemler maskelenebiliyor
daha forte işlemlerin varsa onları
yap demek

Bir kesmenin maskelenebilir olduğunu

olmaması interrupt flag'ı olوكلو

işlemcinin içinde-
ki 2 bacak

KESİME ÖNCELİKLERİ

Öncelik

kesme

en öncelikli
kesme

çıkarma

1

divide by zero

hatana neden olan
program sonlandırılır
Sistem güvenliğini yitir.
tekrar başlatılması
gerekebilir

number
onlomina
geliyor

2

INT #

yazılım olarak numara-
sı verilen kesmenin
çalıştırılması

3

INTO

Aritmetik işlemin (mul / imul)
sonucunu değerlendirmek içindir.
J0 / JNO koşullu döllenme komutla-
rı da bu amaçla kullanılır

4 NMI

int 02H - Bellek üzerinde oluşan kritik hatalarda kullanılır

5 INTR

8259A entegresi üzerinde oluşan donanım kesmeleridir

6 Single step int 01H programın odur odur çalışmamasını sağlayarak kesmedir

Kesme oluşturupunda yapılan işlemler

1- Bayraklar yığında saklanır (PushF)

→ Bir mesela dizi sıralama işlemi yapıyoruz tam yarısında bir kesme geldi bu yüzden bayraklar yığında saklanır bunu biz yapmıyoruz. Sistem otomatik olarak kendisi yapıyor

2- TF=0

→ Bunlar otomatik olarak orta tarafda olan şeyler

3- IF=0 (CLI)

4- CS kayıtları sıfır olursa (Push CS)

1- CS yazmaya yipinə otılır (Push IP)

5- IP yazmaya yipinə otılır (Push IP)

6- INT çalıştırılır (# x 4 sonucunda vektör tablosundaki adrese erişilir)

- olunan ilk word deperi IP yazmaya
- ikinci word deperi CS yazmaya yerleştirilir

↪ int çağrıldığında gyri bir program çalışır o da bir yordamdır. Ve eger siz harici yordamınız interrupt dahi olsunız harici yordamınız o harici yordamın içeriğinde registerların değerleri korunur programınızı yazmaya başlamadan önce o interrupt kodunu yazmaya başlamadan önce hanpi registerleri kullanıcısın onları stack'e otarsın programını interrupttan bitirmeden önce o stack'e ottıklarını stackten peri getersin

Vektör tablosu → hanpi interruptin kodu bellekte hanpi adresinde

256 tane interruptımız var.
bunların alt fonksiyonları var
hanpi interruptımız mesela
17 no'lu interrupt bellekte
nerde işletim sistemi çalışmaya başladığında bunu pidip

' bellege yükleyor hemen'

Benim aradığım interrupt bellekte
hərpi odreste

1-hərpi code segmentte

2- o code segmentin hərpi offsetinde

code segment deperimi 4 tane hexa decimal
deperden olusuyordu

4 tane hexa decimal deperden olusuyorsa 16
byte tər. 2 Byte boyutunda yer kəpəliyər

2 byte ta offset deperi etti mi size 4 byte
yəni hərpi code segmentin hərpi offsetinde
oldığını söyləmək için 4 byte lik vektor
tablosunda her bir interrupt için yer ayri-
liyər, yəni vektor tablosunun boyutu 16
byte tər.

1024 byte lik bir olan vektor tablosu
256 tane interruptımız var her bir interrupt
için o interruptın bellekte hərpi odreste
oldığını tutmak için 4 byte lik yer kullanı-
yoruz, yəni vektor tablosunun boyutu 256×4
ten 1024 byte tər.

Simdi biz o interruptı göstərmədən önce diyelim
ki 17 numaralı interruptı göstərmək
istiyoruz. Bunu 17'gi 4 ile çarpıyarız
68 oldu. işte vektor tablosundakı decimal
olarak 68. byte'a gidiyorsun orda bulduğun
ilk word deperini instruction pointer
kaçına salırsanız işe yaramaz

degerine set ediyordun 2. word deperini
4 byte lik yer kaplıyor dediğimde
ilk word deperi 2 byte lik yer
kaplıyor. Onu aldın instruction
pointer olsaydı 2. word deperi işe
code segmentte yazmışsun böyle olunca
senin code segment re instruction pointer yazmak
larının içerisinde sun gidişimin kesmenin
kodu var, ondan sonra onu iteratif bir şekilde
gelistirmeye başlıyorsun.

Kesme servis programını yazın

- yazmocolorla ilgili durumları değerlendirilmeli
- RET komutu ile kesmeden dönmemelidir

ret ile retf arasındaki fark

near dölleriyorsak ret'le beraber dönüyorduk
near döllerdiğimiz için extradan code segmentin
degerini stack'in içeriğine atmayı orduktan, yani böyle
olduğu için içinde offset deperi stack'in
icerasına giriyyordu neorda ret'le beraber
dönerken 2 byte lik bir veri getiyorduk içinde
stackten retf'lo beraber dönerken harici
bir segmentten dönüpümüz için o segmente
gitmeden önce code segmentin degerini de

Stackin içerişine attığımızdan ötürü retf'la dönerken 4 byte'luk bir değer stack'ten getiyoruz. IRET ile dönerken IP'yi atmışız (Push IP) neyi atmışız code segmenti atmışız (Push CS) neyi atmışız bayrakları atmışız (PushF). IRET ile dönerken 6 byte'luk bir değer getiliyor stackin içerişinden.

interrupt oldığında PushF, Push CS, Push IP komutları otomatik olarak çağrılır ve bu değerler bizim stack'ımızın içeriğine yazılır. Biz eger kullandığımız sayıda stack aktıysak bizim kodumuz interrupt oldığında bu komutlar çağrılırsa sistemimiz overflow hatası olur 3 tane word burdan kesin geliyor

Sinarda söyle gitir bir kod verilir bu kod bussuru push pop yapar deriz ki bu stackin minimum değeri kaç olur

→ hesapladınız 8 word atılıyor 3 word'te burdan 11 diceksiniz

→ O 8 word'u geçmemeliyiz ama interrupt olur diye o 3

word'u 605 bıratmın perek

→ IRet (interrupt return) stackten
kaç byte veri geter

→ 3 word getiyor o da 6 byte
eder

↳ biz kendi kesmelerimizi yazabilirim

↳ hali hazırda olan bir kesmeyi Jen
depistirebilirsin

Vektör tablosunun görünüm konumu

1- Kesme servis programları perek BIOS gereksiz
işletim sistemi tarafından sağlanan depistik
uzunluktaki kod parçalarıdır

→ mesela 17 numaralı interrupt kodu 30 byte'tır
18 350 byte'tır, yani forte uzunluğa olabilir

→ omo nesi aynı her birisine erişmek için
kullanabileceğimiz adres aynı kaç byte 4 byte
1'si kod segmenti için 2'si o kod segmenti
offset için kullanıyoruz

2- Bilgisayar çalışmasında bellekte yerleştirilirler.

3- farklı bellek alanlarında bulunan bu kodlara erişmek için vektör tablosu kullanılır

4- vektör tablosunun tetiği üzerinde kesmelerin başlangıç adreslerini tutmaktadır

5- 00000H - 00FFFH fiziksel adresleri arasında bulunan 1024 byte'lik alanda bulunur

→ 8086'da fiziksel adresimiz 20 byte'ton oluşur

→ maksimum 1 mb adresleyebiliyorduk

→ 00000H - 003FFH arasında bizim interruptlarımız var

→ Bu alanları kullanamazsınız

→ Bu alanlar interruptların vektör tablosu için ayrılmış bir alan

→ Niye 1024 byte içinde 256 tane interruptınız var. $256 \times 4 \rightarrow 1024$ byte etmiş olur.

her interruptın adresi 2 word ile ifade ediliyor. offset ve kesim adresi olarak yani 2 word demet

4 byte demek

6- her servisin adresi 2 word ile ifade edilir (offset ve kesim adresi)

7- yani $1024/4 = 256$ tane kesme vardır

8- Kullanıcılar kesme yazıcotlarsa

int 60H - INT 67H arası bu iş için ayrılmıştır

Mercut olan servisin yerine yeni bir kesme yazılabilir

Buraya siz kendi kodunuzu yazabilirsiniz
Kendi kesmenizi yazabilirsiniz

Sizin kesme yordamınız bellette 100 byte
yer kaplıyodur yerine koymak istediginiz
80 byte yer kaplıyodur. Nasıl yapıldıracak-
siniz

Vektör tablosunda depositlik nosil yapılır

Programı tekrar kesmesini yapabilir

Programın son kesmesini yazdırır

Vor olan bir kesmeden hemen önce derreye girebilecek bir kesme yazabilir

her iki durumda da yazılın kesme vektor tablosuna yerlestirilmeli dir

Bunun için yapılması gereken işlemler:

1- Kesmenin TSR ile bellege yerleştirilmiş olması gereklidir. (TSR → Terminate and Stay resident)

2- yazılın kodun adresinin (offset + kesim) vektor tablosunda ilgili alanlara yazılması gereklidir.
(Kesmeler disable edilmelidir. edilmezse?)

3- depistirilecek kesmenin adresi saklanmalıdır

4- INT 21H - fonksiyon 15H (set interrupt vector) kullanılarak yazılın kesme kodunun adresi vektor tablosuna yazılmalıdır

5- Kesmelere izin verilmelidir

1- Kendi kesmemizi yarımışsa RET ile dönmeliyiz

2- Orijinal kesmeden önce galişmiş bir JMP ile orijinal kesme adresine gitmeli ve okuma ordan devam etmeliyiz

6- Kullanım tamamlandıktan sonra orijinal kesmenin adresi vektör tablosuna yazılmalıdır

Interrupt servis programı yazarken nelere dikkat edilmelidir?

Kesme kodu içerisinde girildiğinde kesmeye izin verilmelidir

Kesme içerisinde yazıcıların bir önceki değerlerinin korunması gerekmektedir

Kesme programı 1'den fazla kesim kullanacaktır ise bu değerler kesmeye gitmeden önce ilgili yazıcıları yüklemeli dir

Donanım kesmesi yerine BIOS) geçerek kesme yazılıcaksa işletim sisteme ait (DOS) hiçbir fonksiyon kullanılmamalıdır

Kesme programı TSR ile bellege yerleştirilmesi gerekmektedir

yapınca veri göndermişsek onları kesme içerişinde temizleyecektir. Setilde hareket serpilemeyeceğiz.

Kesmeleri kullanırken

00H - 1FH arasında BIOS kesmeleri

20H - 2FH arasında DOS kesmeleri bulunur

Kesmeleri kullanırken

fonksiyon numarası AH'a

eğer varsa alt fonksiyon numarası AL'ye aktarılır

INT komutu kullanılır

Örnek

mov AH, 1H

INT 21H



