Kesme (*Interrupt*)

Kesme kaynağında (örneğin G/Ç arabirimi) belli bir koşul oluştuğunda MİB'e kesme isteği gönderilir.

Bir MİB'ne birden fazla kesme kaynağı bağlı olabilir.

Aynı anda birden fazla kaynak kesme isteğinde bulunabilir.

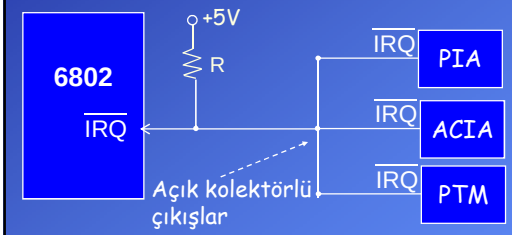
Kaynak belirleme ile ilgili yapılması gerekenler:

1. Aynı anda birden fazla kaynaktan istek geldiğinde öncelik belirlemek.

2. Kesme isteği kabul edilen kaynağa ilişkin kesme hizmet programının (*interrupt service routine ISR*) başlangıç adresini belirleme. Vektörlü/vektörsüz çalışma.

MİB'lerin Kesme düzenleri:

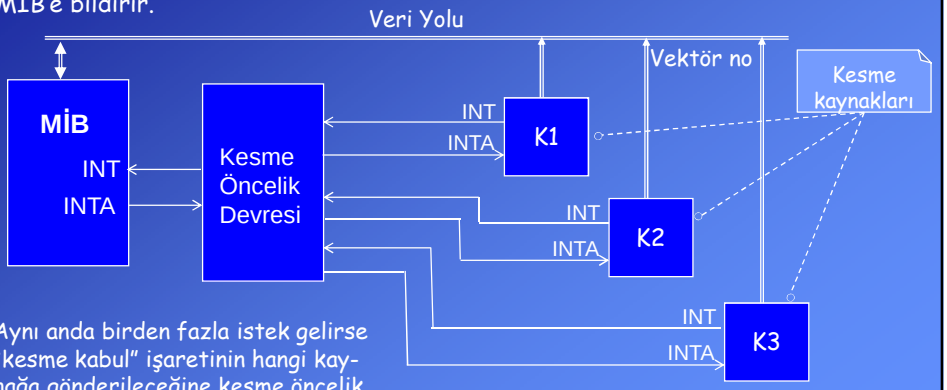
a) Bir kesme isteği girişi var, kesme kabul çıkışı yok. Örnek 6802.



- Tek bir kesme hizmet programı var.
- Kesme kaynağı, kesme hizmet programının içinde yoklama (*polling*) ile belirlenir.
- Kesme isteğinden vazgeçirme (kesme kabul) yazılım ile yapılır.

MİB'lerin Kesme düzenleri devamı:

b) Bir kesme isteği girişi (INT) var, bir kesme kabul çıkışı (INTA) var, vektörlü çalışıyor.
Kesme kaynakları MİB'e kesme öncelik devreleri üzerinden bağlanırlar.
İsteği kabul edilen kaynak kendi hizmet programının başlangıç adresini (ya da başlangıç adresinin vektör tablosunda nereden alınacağını) veri yolu üzerinden MİB'e bildirir.



Aynı anda birden fazla istek gelirse "kesme kabul" işaretinin hangi kaynağa gönderileceğine kesme öncelik devresi "karar" verir.

Vektör adresi:

MİB'ler bir kesme isteği kabul edildiğinde hangi hizmet programının çalıştırılacağına ilişkin bilgileri bir kesme **vektör tablosunda** tutarlar.
Bu bilgiler iki farklı şekilde tutulabilir:

1. Vektör tablosunda kesme hizmet programlarının başlangıç adresi bulunur. Kesme isteği kabul edilen kaynak MİB'e vektör numarasını yani hizmet programının bulunduğu satır numarasını gönderir.
İlgili satırdan hizmet programının başlangıç adresi alınır. (MC 68000)

2. Tabloda kesme hizmet programlarının kendisi (ilk satırı) bulunur.
Tüm programları tek tabloya sığdırmak mümkün olmayacağından bu satırlara asıl hizmet programına gitmeyi sağlayan **JMP hizmet_programi** komutları yerleştirilir.

Sabit vektörlü (otovektörlü) çalışmada vektör numarası kaynak birimden alınmaz. Her kesme kaynağı için önceden belirlenmiş bir vektör numarası vardır.

Örnek: 6802'de IRQ ve NMI kesmeleri için vektör adresleri sabittir.

Tasarımcı ilgili kesme hizmet programlarının başlangıç adreslerini tablolardaki ilgili gözlere önceden yazar.

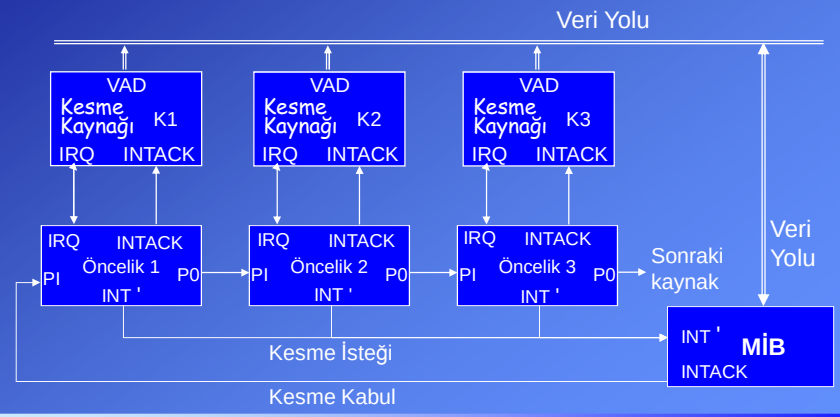
MC 68000 Vektör Tablosu:

Vectors Numbers		Address			
Hex	Decimal	Dec	Hex	Space[6]	Assignment
0	0	0	000	SP	Reset: Initial SSP[2]
1	1	4	004	SP	Reset: Initial PC[2]
2	2	8	008	SD	Bus Error
3	3	12	00C	SD	Address Error
4	4	16	010	SD	Illegal Instruction
5	5	20	014	SD	Zero Divide
6	6	24	018	SD	CHK Instruction
7	7	28	01C	SD	TRAPV Instruction
8	8	32	020	SD	Privilege Violation
9	9	36	024	SD	Trace
A	10	40	028	SD	Line 1010 Emulator
B	11	44	02C	SD	Line 1111 Emulator
C	12[1]	48	030	SD	(Unassigned, Reserved)
D	13[1]	52	034	SD	(Unassigned, Reserved)
E	14	56	038	SD	Format Error[5]
F	15	60	03C	SD	Uninitialized Interrupt Vector
10–17	16–23[1]	64	040	SD	(Unassigned, Reserved)
		92	05C		—
18	24	96	060	SD	Spurious Interrupt[3]
19	25	100	064	SD	Level 1 Interrupt Autovector
1A	26	104	068	SD	Level 2 Interrupt Autovector
1B	27	108	06C	SD	Level 3 Interrupt Autovector
1C	28	112	070	SD	Level 4 Interrupt Autovector
1D	29	116	074	SD	Level 5 Interrupt Autovector
1E	30	120	078	SD	Level 6 Interrupt Autovector
1F	31	124	07C	SD	Level 7 Interrupt Autovector
20–2F	32–47	128	080	SD	TRAP Instruction Vectors[4]
		188	0BC		—
30–3F	48–63[1]	192	0C0	SD	(Unassigned, Reserved)
		255	0FF		—
40–FF	64–255	256	100	SD	User Interrupt Vectors
		1020	3FC		—

Kesme Öncelik Devreleri

a) Seri Öncelik Devresi (Papatya Zinciri "*Daisy Chain*") 1

- Öncelik devresi (zincir) değişik önceliklere sahip "halka"lardan oluşur.
- Her kesme kaynağı bu halkalardan birine (*Interrupt Request* - IRQ) bağlıdır.
- Zincirin başındaki halka (öncelik 1) en yüksek önceliğe sahiptir.

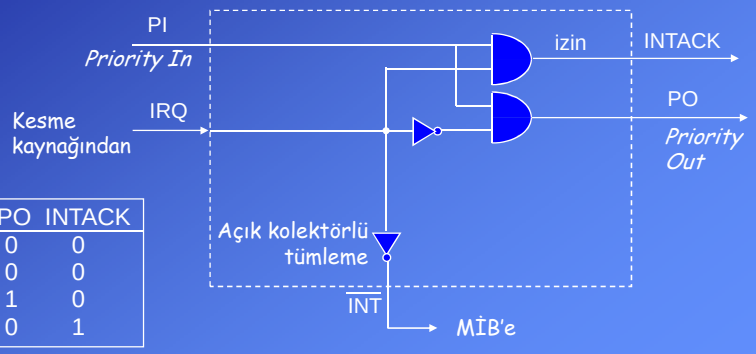


Öncelik zincirinin çalışması:

•MİB'den "kesme kabul" işareti önce baştaki halkaya (Priority in: PI) gelir. Eğer bu halka istekte bulunduysa kendi kendisine bağlı olan kesme kaynağına INTACK işaretini gönderir. Böylece bu kaynağın vektör adresi (VAD) veri yoluna çıkar.

• Eğer bu halka istekte bulunmadıysa (Priority out: PO) çıkışını etkin yaparak bir sonraki halkaya (kaynağa) olanak tanır.

Papatya Zincirinin bir halkasının iç yapısı:



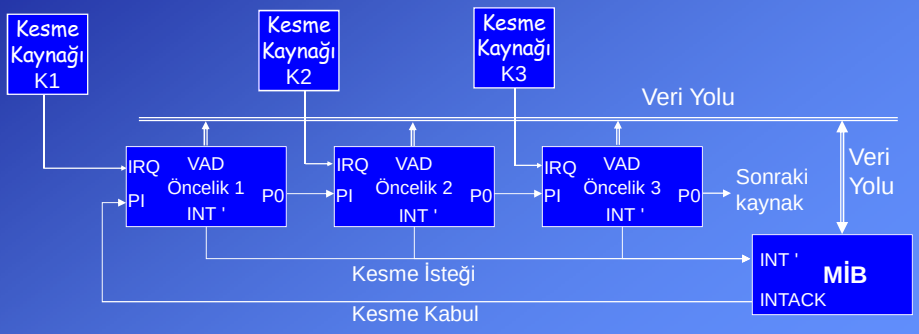
PI	IRQ	PO	INTACK
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

Seri Öncelik Devresi (Papatya Zinciri "*Daisy Chain*") 2

• Önceki devrede kesme kaynakları kendi vektör numaralarına sahiptirler ve INTACK işareti geldiğinde bu adresi veri yoluna çıkartabilmektedirler.

• Aşağıdaki devrede ise kesme kaynakları vektör numarasına sahip değildirler. Bu adresi tutma işi öncelik devresine verilmiştir.

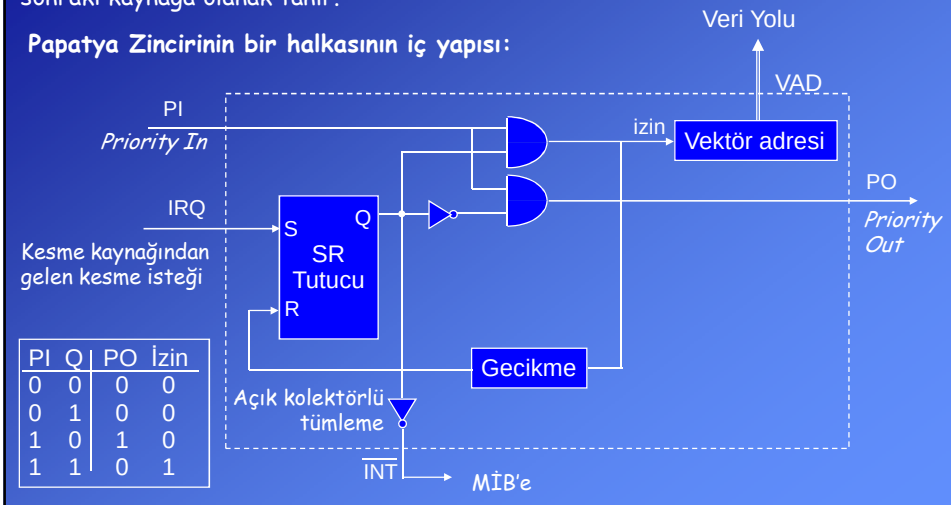
•Ayrıca kesme kaynakları kesme isteğini kabul işareti gelinceye kadar sürdürmemektedirler. İsteği tutma (sürdürme) görevi de öncelik devresindedir.



•MİB'den "kesme kabul" işareti önce baştaki halkaya (Priority in: PI) gelir. Eğer bu halka istekte bulunduysa kendi vektör adresini (VAD) veri yoluna koyar.

• Eğer bu halka istekte bulunmadıysa (Priority out: PO) çıkışını etkin yaparak bir sonraki kaynağa olanak tanır.

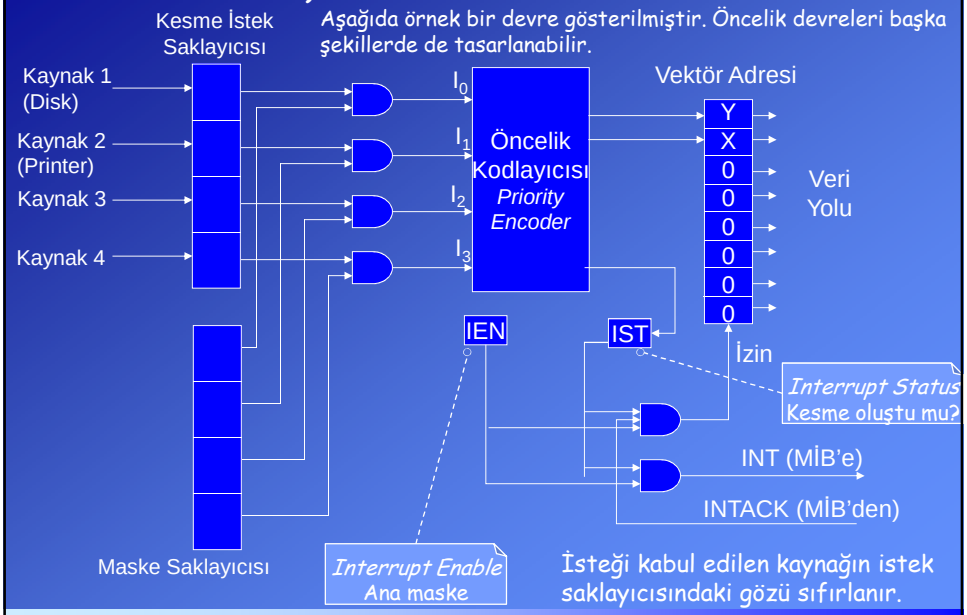
Papatya Zincirinin bir halkasının iç yapısı:



PI	Q	PO	İzin
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

b) Örnek Bir Paralel Öncelik Devresi

Aşağıda örnek bir devre gösterilmiştir. Öncelik devreleri başka şekillerde de tasarlanabilir.



İsteği kabul edilen kaynağın istek saklayıcısındaki gözü sıfırlanır.

Bu örnek sistemde kaynakların vektör adresleri:

Kaynak 1: 0000 0000
Kaynak 2: 0000 0001
Kaynak 3: 0000 0010
Kaynak 4: 0000 0011

Öncelik Kodlayıcısının doğruluk tablosu:

Girişler				Çıkışlar		
I_0	I_1	I_2	I_3	x	y	IST
1	x	x	x	0	0	1
0	1	x	x	0	1	1
0	0	1	x	1	0	1
0	0	0	1	1	1	1
0	0	0	0	Φ	Φ	0

Lojik ifadeler:

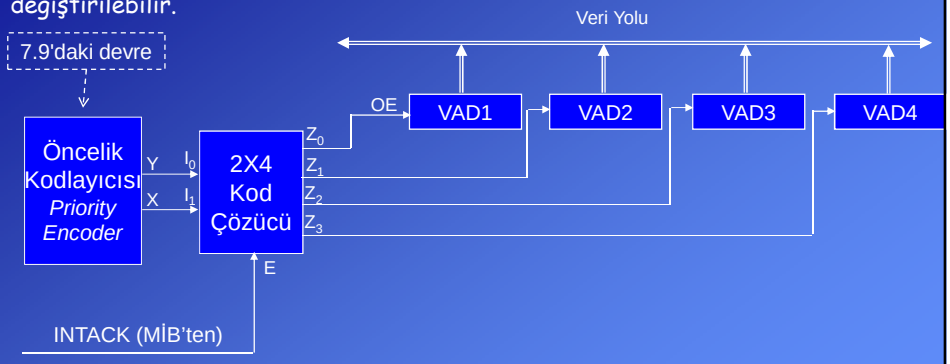
$x = I_0' I_1'$

$y = I_0' I_1 + I_0' I_2'$

$(IST) = I_0 + I_1 + I_2 + I_3$

Önceki örnek sistemde (7.9) kaynakların vektör numaraları önceden belirlenmiştir.

Sistemde aşağıdaki değişiklik yapılarak her kaynak için ayrı bir vektör adres saklayıcısı yerleştirilebilir. Böylece her kaynağın vektör numarası esnek olarak değiştirilebilir.



Kesme isteği kabul edilen kaynağa ilişkin vektör adresi saklayıcısına çıkış izni (*Output enable* OE) verilir.

Bu saklayıcılara gerektiğinde MİB tarafından yazma yapılabilecek şekilde bir tasarım oluşturulur. Böylece vektör adresleri esnek olarak değiştirilebilir.

Kesme Hizmet Programına gidilip dönülürken yapılan işler:

Komut yürütme çevriminden sonra kesme istekleri değerlendirilir.

Eğer bir kesme isteği kabul edilirse kesme hizmet programına dallanmadan önce MİB aşağıdaki işleri gerçekleştirir:

(Bu işler MİB'in denetim birimi tarafından iç işlem olarak yapılır. Kullanıcı programı ile yapılmazlar.)

SP ← SP-1 (Yığın işaretçisi gerektiği kadar –adres kaç sekizli ise- azaltılır)
M[SP] ← PC (Geri dönüş adresi yığına)
INTACK ← 1 (Kesme kabul)
PC ← VAD (Vektör adresi) ya da PC ← Tablo[VNo] (Vektör tablosundan)
SP ← SP-1
M[SP] ← SR (Durum saklayıcısı yığına)
IEN ← 0 (Diğer kesmeler maskelendi, bu bayrak SR'nin içindedir)

Bundan sonraki komut alma çevriminde Kesme Hizmet Programından devam edilir.

Bazı MİB'ler iç saklayıcılarını da yığında saklarlar. Bazıları bu işlemi programcıya bırakır.

Kesme hizmet programından dönülürken:

SR ← M[SP] (Durum saklayıcısı geri alındı)
SP ← SP+1
PC ← M[SP] (Geri dönüş adresi alındı)
SP ← SP+1 (Gerektiği kadar –adres kaç sekizli ise- arttırılır)

MC68000'de Sıra Dışı Durumlar (*Exceptions*)

Dış Kaynaklı:
- Reset
- Yol Hatası (*Bus Error* - BERR)
- Kesme İstekleri (*Interrupts*) Vektörlü, otovektörlü

İç Kaynaklı:
- İzleme (*trace*) Adım adım komut yürütme
- Adres hatası : Tek adreslere word/long erişimi
- Yazılım kesmesi (Trap 0 -15), TRAPV (Trap on overflow), CHK
- Geçersiz komut: Makine dili karşılığı olmayan işlem kodu (op code)
- Komut emulasyonu (1010 ve 1111 ile başlayan komutlar)
- Yetkili komut çalıştırma
- Sıfıra bölme

Sıra dışı durum oluştuğunda yapılanlar:
- SR → Temp (SR'nin kopyası çıkartılır)
- S←1, T←0 (İşlemci yönetici (*supervisor*) konumuna geçiyor.
- PC (geri dönüş adresi) yığına yazılır.
- SR nin Temp'teki kopyası yığına (S ve T'nin değişmemiş önceki değerleri) yazılır.
- Hizmet programının adresi vektör tablosundan alınır.
- Saklayıcılar yığında işlemci tarafından saklanmaz. Programcı kullandığı saklayıcıları yığında kendisi korumalıdır.

Geri dönüşte:
- Hizmet programları RTE (*Return from Exception*) komutu ile sonlandırılır.
- Durum saklayıcısı SR yığından alınır.
- Geri dönüş adresi yığından alınır.

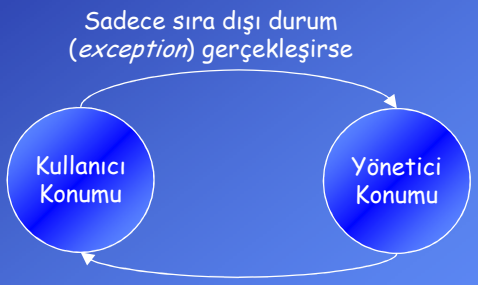
RESET durumunda bu işlemlerin hepsi yapılmaz.

Bazı sıra dışı durumlarda ise (BERR, kesme gibi) ek işler de yapılır.

Sıra dışı durum oluştuğunda S←1 işleminden dolayı işlemci yönetici (*supervisor*) konumuna geçer.
Bu nedenle hizmet programları yönetici (*supervisor*) konumunda yürütülür.

Hizmet programından geri dönüldüğünde, eğer sıra dışı durum kullanıcı konumunda oluştuysa tekrar kullanıcı konumuna geçilir, çünkü SR'nin orijinal değeri yığından alınır.

Ayrıca SR'ye yazan komutlar ile de yönetici konumundan kullanıcı konumuna geçilebilir.



RTE ya da durum saklayıcısına yazan komutlar ile MOVE, ANDI, EORI (S=0 yapılır)

Yol Hatası (BERR) ve Adres Hatası:

68000'nin BERR' girişi etkin yapıldığında yol hatası sıra dışı durumu oluşur.

Bir program tek numaralı bir adrese word veya long erişimi yapmak isterse adres hatası sıra dışı durumu oluşur.

Bu durumlarda kesmelerde olduğu gibi komutun tamamlanması beklenmez, yol çevrimi kesilerek sıra dışı durum işlemlerine geçilir.

Diğer sıra dışı durumlardan farklı olarak yığında daha fazla bilgi saklanır.

BERR sıra dışı işlemleri yapılırken tekrar BERR oluşursa işlemci HALT konumuna geçer ve kendini sistemden yalıtır.

Bu konumdan RESET ile çıkılır.

MC68000'de Kesmeler (*Interrupts*):

68000'nin kesme istekleri için üç bitlik bir girişi vardır (IPL2, IPL1, IPL0).

Bu girişlerden gelen üç bitlik değere (kesme isteği düzeyine) göre kesmenin kabul edilip edilmeyeceğine karar verilir.

Eğer kesme isteğinin düzeyi SR'deki maskelerin düzeyinden büyükse kesme isteği kabul edilir.

Kesme Maskeleri (I_0 I_1 I_2)

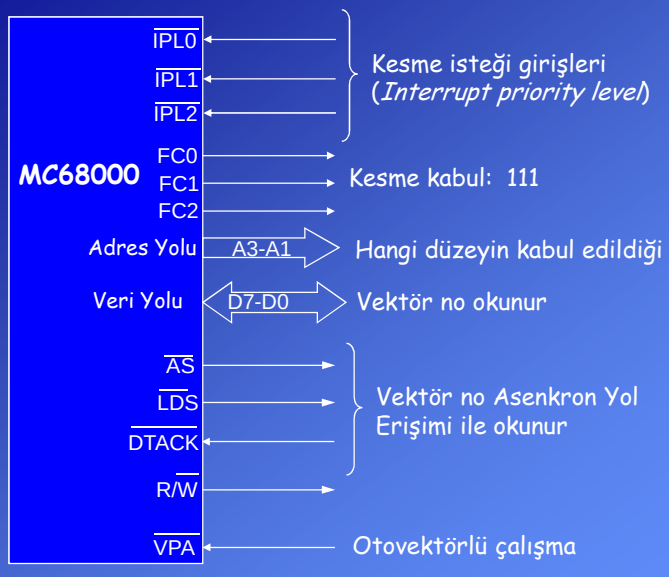
IPL2, IPL1, IPL0 > I2, I1, I0 ise kesme kabul edilir.

Eşitlik durumunda da kesme kabul edilmez.

Bunun istisnası 7. düzeyden gelen kesmedir.

7. düzeyden gelen kesme **maskelenemez**.

MC68000'nin Kesmelerde kullanılan hatları:



68000'de kesme kabul edildiğinde yapılan işlemler:

- SR → Temp (SR'nin kopyası çıkartılır)
- S←1, T←0
- PC yığına yazılır
- SR'nin Temp'teki kopyası yığına (S ve T'nin değişmemiş önceki değerleri) yazılır.
- I2, I1, I0 ← IPL2, IPL1, IPL0 Kabul edilen düzey maskeye yazılır. Böylece eşit ve daha düşük öncelikli kesmeler maskelenmiş olur.
- FC2, FC1, FC0 ← 111 (Kesme Kabul)
- A3, A2, A1 ← Kabul edilen düzey

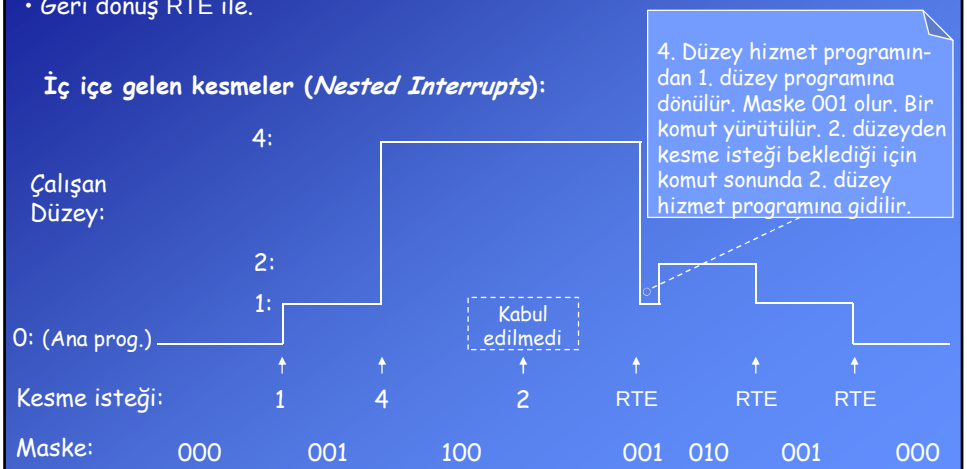
a) Vektörlü çalışma:
  - Dışarıdan asenkron yol erişimi ile 8 bitlik vektör no okunur.
  - Vektör tablosunun vektör no satırından (vno x 4) hizmet programının başlangıç adresi alınır.

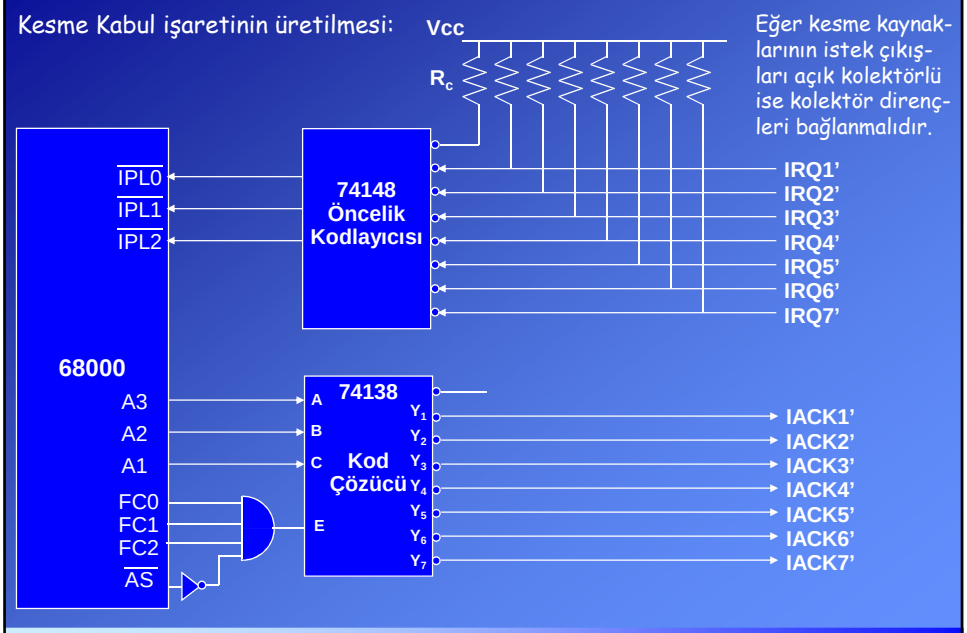
b) Otovektörlü çalışma:
  - Kesme kabulü sırasında dış birim 68000'in VPA' girişini etkin (sıfır) yaparsa dış birim vektör numarası göndermeyecek demektir.
  - Bu durumda kabul edilen düzeye göre vektör tablosunda belirli bir yerden kesme hizmet programının başlangıç adresi alınır.

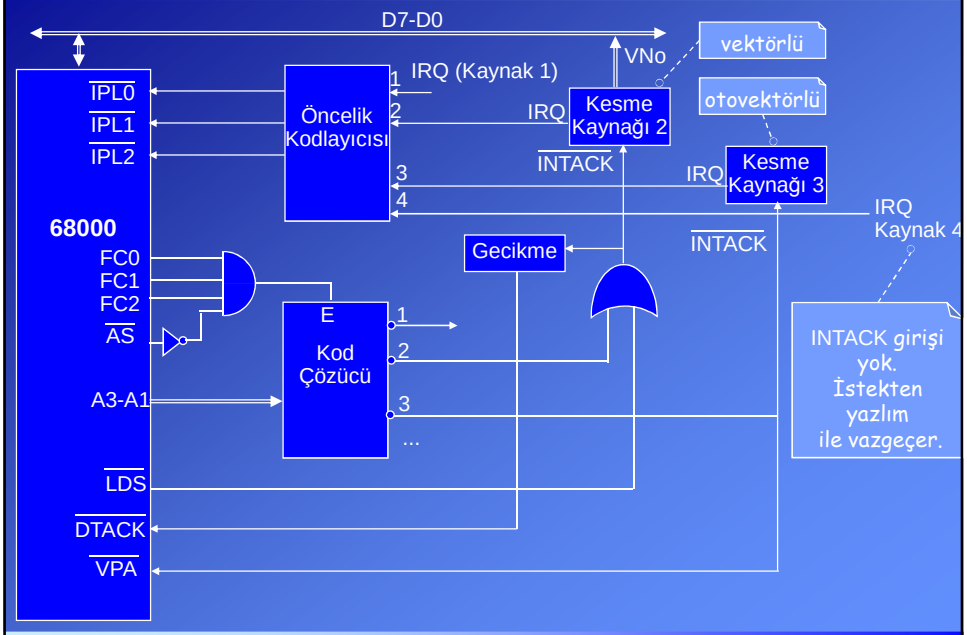
Programcının yapacağı işlemler:

- Kesme hizmet programında kullanılan saklayıcılar yığında saklanmalı (MOVEM).
- Eğer gerekirse kesme maskeleri değiştirilerek daha düşük öncelikli kesmelere izin verilebilir ya da daha yüksek öncelikli kesmeler (7 hariç) engellenebilir.
- Geri dönüş RTE ile.

İç içe gelen kesmeler (*Nested Interrupts*):







Yazılım Kesmeleri

MC68000'de 16 farklı yazılım kesmesi oluşturmak mümkündür:

TRAP #0 – TRAP #15

Sistemi tasarlayanlar, kullanıcının gerek duyacağı işlemleri, örneğin PIA'yı kullanmak, yazılım kesmesi hizmet programları şeklinde önceden hazırlarlar.

Hizmet programları yönetici (*supervisor*) konumunda çalıştığından, kullanıcı doğrudan erişemeyeceği sistem kaynaklarını gerek duyduğunda yazılım kesmeleri ile kontrollü olarak kullanabilir.

Komut Emülasyonu (Benzetimi)

MC68000'in makine dilinde $A (1010) ve $F (1111) ile başlayan komut yoktur.

Sistemin belleğine bu değerler ile başlayan bir "komut" yerleştirilirse 68000 bu komutu alıp çözmeye çalıştığında sıra dışı bir durum (*exception*) oluşur ve bir hizmet programı çalıştırılır.

Tasarımcılar kendi makine dili komutlarını oluşturabilirler. Komut çözülürken oluşan sıra dışı durumda gidilen hizmet programının içinde o komutun yapması gerekenler program ile gerçekleştirilir.

Sıra dışı durum oluştuğunda hizmet programına gidilirken yığına yazılan PC değeri sıra dışı duruma neden olan komutun başına işaret eder.

Örnek:

MC68000 tabanlı bir bilgisayar sisteminde komut emülasyonu olanağından yararlanılarak aşağıda açıklanan komutlar gerçekleştirilecektir.

- ADD.B adres1,adres2,adres3 (adres3)←(adres1)+(adres2)

Bu komut adres1'deki ve adres2'deki **8** bitlik sayıları toplayarak sonucu adres3'e yazmaktadır. Adresler 32 bit uzunluğundadır.

- ADD.W adres1,adres2,adres3 (adres3)←(adres1)+(adres2)

Bu komut ise aynı işlemleri **16** bitlik sayılar üzerinde yapmaktadır. Burada da adresler 32 bit uzunluğundadır.

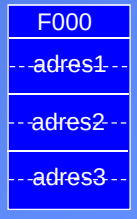
Çözüm:

Önce komutun makine dilindeki yapısını oluşturmak gerekir.

Örnek bir yapı:

ADD.B adres1,adres2,adres3 $F000 adres1,adres2,adres3

ADD.W adres1,adres2,adres3 $F001 adres1,adres2,adres3



Komut sözcüğünün (*Op word*) son biti boyut (*size*) için kullanılmıştır. 0:B, 1:W

Komutlarda istenen işleri yapacak hizmet programını yazmak gerekir.

Hizmet programından önce onu test etmek için kullanılacak bir ana program aşağıdaki gibi oluşturulur:

```
main  lea     stack,a7                          // Yığın işaretçisi başlangıç değeri
      adda.l  #40,a7                            // Yığın azalan adreslere doğru ilerler
      move.l  #service,($2C)                    // Vektöre hizmet programının adresi
      dc.w    $f000,0,$1000,0,$1100,0,$1200     //ADD.B $1000,$1100,$1200
      dc.w    $f001,0,$2000,0,$2100,0,$2200     //ADD.W $2000,$2100,$2200
      ....
      org     $500
stack ds.b    40                                // Yığın için bellekte yer ayrılıyor
```

Hizmet programının başında, gerekli saklayıcıları yığına yazmak gerekir.

Bunların hangi saklayıcılar olduğu program yazıldıktan sonra belli olur.

```
service  movem.l d0/a0-a3,-(a7)      D0, A0, A1, A2, A3 yığına yazılıyor
```


Hizmet programına gidilirken SR ve PC mikroişlemci tarafından yığına yazılmıştı.

Hizmet programının başında ise beş adet saklayıcı programla yığına yazıldı. Buna göre yığındaki bilgiler yandaki gibidir:

```
service movem.l  d0/a0-a3,-(a7)
        movea.l  22(a7),a0      Komuta işaret eden PC → a0
        move.w   (a0)+,d0       Komutun ilk 16 biti OpCode ->d0
        movea.l  (a0)+,a1       Adres1 → a1
        movea.l  (a0)+,a2       Adres2 → a2          Komut Alma
        movea.l  (a0)+,a3       Adres3 → a3
        tst.b    d0             B/W?
        bne      word           Komut çözme (Decoding)
        move.b   (a1),d0        Byte işlemleri
        add.b    (a2),d0
        move.b   d0,(a3)
        bra      ret                                 Komut Yürütme
word    move.w   (a1),d0        Word işlemleri
        add.w    (a2),d0
        move.w   d0,(a3)
ret     move.l   a0,22(a7)      Yığındaki PC güncelleniyor (update)
        movem.l  (a7)+,d0/a0-a3
        rte                     Yığındaki PC bir sonraki komuta işaret ettirilmeli.
```

SP →	A3_H
+2	A3_L
+4	A2_H
+6	A2_L
+8	A1_H
+10	A1_L
+12	A0_H
+14	A0_L
+16	D0_H
+18	D0_L
+20	SR
+22	PC_H
	PC_L