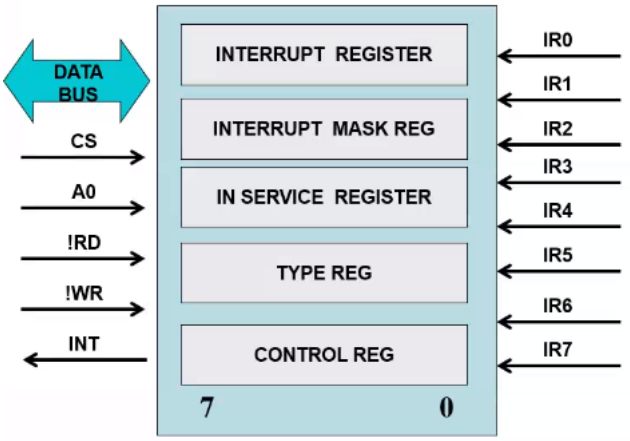
ESERCITAZIONE PIC

Con il PIC simuliamo un dispositivo molto semplificato rispetto a quello visto. **Il device ha 8 linee di richiesta, con una certa priorità, in cui il livello 0 è il più prioritario**. Nel modello di programmazione abbiamo i registri visti nel datasheet. Possiamo accedere all'***interrupt mask register (IMR), che ci consente di avere una maschera ai livelli di interruzioni che vogliamo mascherare.*** Poi abbiamo ***l'interrupt request register (IRR) che ha tanti bit quante le linee di interruzione***, si alzano quando arriva un'interruzione su quella linea. ***L'In Service Register (ISR) tiene traccia della linea di interruzione correntemente servita***, poi abbiamo ***un registro di controllo e type register, in cui memorizziamo la base degli indirizzi dei vettori delle eccezioni***, si parla di **interrupt vettorizzato** in cui possiamo definire dove stanno gli indirizzi delle ISR da associare alle interruzioni. Abbiamo le 8 linee, il databus e tutti i segnali di controllo. **Il TR è la prima cosa che inizializziamo, ci da una base dei vettori.** *Dato che nel 68k i vettori definiti dagli utenti vanno dal 64° in poi, scriviamo come base proprio 64*, da cui poi ricaviamo il vettore vero dell'interruzione, gli 8 numeri dopo 64. Il numero di un vettore non è l'indirizzo in cui prendere l'indirizzo dell'ISR ma è l'indice, il 25° autovettore ad esempio si trova all'indirizzo 100. Anche in questo caso dobbiamo fare la conversione per capire dove si trova effettivamente in memoria l'indirizzo dell'ISR specifica da eseguire. **Man mano che il PIC serve le interruzioni si crea in modo automatico il numero vero del vettore da eseguire, in base alla linea interrompente**, se interrompe il dispositivo in 0 il PIC seleziona l'interruzione del vettore 64, se l'interrompente è quello della linea 3 del PIC va a prendere il vettore 67. **Prende il TR e aggiunge la codifica del numero 3**, per cui trova il 67° vettore. Questo si fa in modo automatico, dobbiamo solo dare al PIC la base dei vettori. ***Appena facciamo un reset, se facciamo un access in scrittura a indirizzo dispari selezioniamo il TR.*** Il PIC ha un indirizzo pari e uno dispari e tanti registri, per cui *l'indirizzamento è complesso, riguarda anche alcuni bit interni, come visto nella PIA.* Per il registro di maschera, si fa sempre un accesso all'indirizzo dispari ma dopo il primo fatto per il TR. **Dal secondo accesso in poi all'indirizzo dispari si legge/scrive nell'IMR**. Per mascherare una linea mettiamo 1, con tutti 0 non mascheriamo niente. **Al registro di controllo accediamo quando accediamo in scrittura a indirizzo pari**, invece in lettura, per accedere a CNTRL, dobbiamo settare un apposito bit, dobbiamo fare un indirizzamento più complesso**. Il registro di controllo ha i primi bit che dicono quale bit cancellare nell'ISR, quindi vogliamo ripulire un bit relativo al servizio delle interruzioni**. Nel PIC possiamo avere due modalità, *in una il bit di richiesta/servizio delle interruzione viene resettato automaticamente dopo averla servita (**automatic end of interrupt**), nell'altra modalità dobbiamo farlo in **modo manuale**, dobbiamo azzerare il bit dell'interruzione eseguita, dobbiamo farlo nel registro di controllo*. Per quando riguarda **la prima modalità, la settiamo con il bit 4**, quando lo poniamo a 1 automaticamente viene cancellato il bit relativo alla linea appena servita, se è 0 va fatto manualmente. **Per cancellarlo allora dovremo codificare la linea da cancellare, in bit da 0 a 2, poi in 3 diciamo che vogliamo cancellarlo**. I primi 5 bit servono a queste due modalità. **Il bit 6 serve per fare la selezione, in base a questo faremo la lettura in ISR o in IRR, il bit 7 invece a 0 vuol dire che vogliamo accedere al registro di controllo, se è 1 agli altri due**. *Se scriviamo in indirizzo pari siamo sicuri di andare in controllo, per la lettura per prima cosa dobbiamo vedere il bit 7, se è 0 siamo nel controllo, se è 1 dobbiamo vedere il bit 6, se è 0 andiamo in IRR se è 1 in ISR*. Abbiamo due bit che servono a discriminare dove facciamo l'accesso. IRR è il registro i cui bit si alzano man mano che arrivano le interruzioni su quella linea. **Quando andiamo a servire un'interruzione, il bit IRR viene abbassato e si alza il bit omologo in ISR.** IRR è la prima barriera, ogni volta che una linea di interruzione si alza si alza il bit di IRR, *per cui potremmo anche avere più bit alti,* man mano che decidiamo di servirli si alza anche il bit in ISR. **L'ISR ha il bit i alto quando stiamo servendo l'interruzione sulla linea 1.**

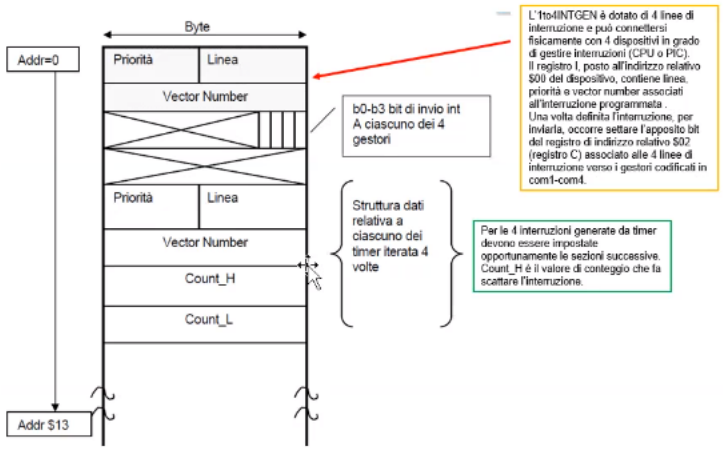


bit	significato
0,1,2	determinano il bit n da cancellare in **ISR**
3	Bit **EOI** (End Of Interrupt), se posto ad 1 fa cancellare il bit n-esimo del registro **ISR** puntato dai bit b0-b2
4	Bit **AEOI** (Automatic End Of Interrupt), se posto ad 1 fa cancellare automaticamente il bit in **ISR** dopo la trasmissione dell'interruzione
5	non utilizzato
6	Bit **RIS** (Register Interrupt Selector), se il bit **RR**=1 seleziona l'accesso in lettura del registro **ISR** se è posto a 1 e di **IRR** se a 0
7	Bit **RR**, permette se posto ad 1 la lettura dei registri **ISR** o **IRR**

Vediamo in Asim, dobbiamo dare i due indirizzi, pari e dispari, poi **nella com1 diciamo qual è il dispositivo che gestisce le interruzioni del PIC**, che *potrebbe essere collegato o al processore o a un altro PIC, in cascata*. Nel nostro caso va al processore, per cui troviamo l'identificativo del processore. **In com2 troviamo le informazioni sul vettore delle interruzioni, priorità e linea di priorità a cui è collegato il PIC rispetto al processore**, nel nostro caso è 7 (alta priorità). *Le interruzioni che arrivano al PIC le gestisce lui e ne manda solo una al processore.* Ci vorrebbero più dispositivi per vedere bene il comportamento del PIC, così che possa scegliere quale interruzione mandare**. Nello sviluppo di Asim è stato introdotto uno strumento apposito *1to4INTGEN* per produrre interruzioni**, non perché senza questo il PIC non funzioni ma per testarlo. Questo dispositivo **è in grado di generare 5 interruzioni diverse**, in un modo particolare. **La prima interruzione la genera in modo programmatico**, ad un certo punto scriviamo 1 in un registro specifico e si genera l'interruzione, che attaccheremo opportunamente a una delle linee di interruzione in ingresso al PIC. **Le altre 4 interruzioni invece sono generate allo scadere del timer**, settiamo un valore di partenza di 4 timer diversi e quando ognuno scade genera un'interruzione. Abbiamo 4 fili che escono dal timer che possiamo collegare a 4 fili diversi in ingresso al PIC, con quello programmatico poi *possiamo connettere 5 fili diversi al PIC, come se fossero 5 dispositivi diversi che generano interruzioni su 5 linee diverse.* Il generatore delle interruzioni (figura in basso a sinistra) ha parecchi byte allocati, con delle sezioni ben precise che andremo a programmare, il primo pezzo sopra è legato alla configurazione generica del dispositivo che dice chi sono i gestori delle interruzioni. **Poi abbiamo i 4 pezzi dei timer, count_H e count_L sono dei registri in cui scriviamo il valore iniziale e finale del timer. Faremo questo per 4 volte di seguito**. Il dispositivo ha 4 linee di interruzione che possiamo connettere a 4 dispositivi che gestiscono le interruzioni, possono essere 4 diversi oppure usiamo il PIC, che gestisce internamente la priorità. **Il registro I, iniziale, contiene linea e priorità e vector number dell'interruzione programmatica, la inviamo e poi con un bit specifico la facciamo partire**. Il sistema simulato (figura a destra) ha il processore, la memoria e il bus, poi il PIC a cui sono collegate le 4 linee del dispositivo 1to4, poi c'è un terminale che serve per vedere qual è l'interruzione che stiamo servendo, è solo un fatto grafico.

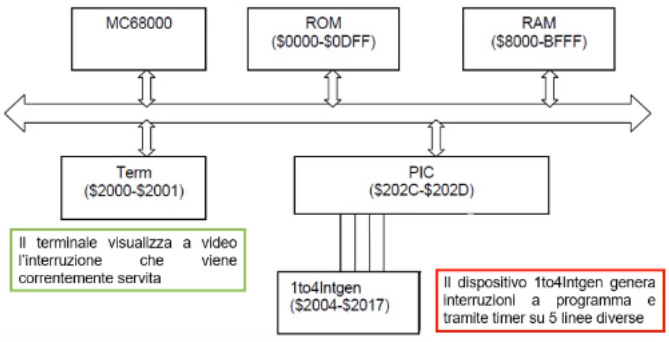
```
Name        I8259PIC
Type        Device
Identif     Intero ($01..$FF) che identifica univocamente l'oggetto in una
            configurazione
Address1    Indirizzo base per il dispositivo
Address2    Indirizzo base+1
BUS         Ident. del bus a cui il dispositivo è connesso
COM1        Ident. del dispositivo PIC o CPU gestore delle interruzioni
COM2        |vettore (b15-b8)|priorità (b7-b4)|linea Int (b3-b0)|
COM3        N.U.
COM4        N.U
```

```
Name        1to4INTGEN.
Type        Device
Identif     Intero ($01..$FF) che identifica univocamente l'oggetto in una
            configurazione;
Address1    Valore iniziale dello spazio indirizzi dell'oggetto;
Address2    Valore finale dello spazio indirizzi (Address1 + $13);
BUS         Identificatore di bus a cui il dispositivo è connesso;
COM1        Identificatore dei dispositivi in grado di gestire le
            interruzioni, se 0 la linea è n.c.;
COM2        Identificatore dei dispositivi in grado di gestire le
            interruzioni, se 0 la linea è n.c.;
COM3        Identificatore dei dispositivi in grado di gestire le
            interruzioni, se 0 la linea è n.c.;
COM4        Identificatore dei dispositivi in grado di gestire le
            interruzioni, se 0 la linea è n.c.;
```



Facciamo uscire sul terminale il codice della linea che stiamo servendo.

```
Configuration name: pic.cfg

CHIP Name: MEMORY
Type: MMU/BUS.   Identif: 01.          BUS: 0000.
Addres 1: 00008000.          Address 2: 00000000.
Com1: 0000.  Com2: 0010.  Com3: 0008.  Com4: 0000.

CHIP Name: M68000
Type: CPU.       Identif: 02.          BUS: 0001.
Addres 1: 00009000.          Address 2: 00009200.
Com1: 0000.  Com2: 0000.  Com3: 0000.  Com4: 0000.

CHIP Name: TERMINAL
Type: Device.    Identif: 03.          BUS: 0001.
Addres 1: 00002000.          Address 2: 00002001.
Com1: 0006.  Com2: 0006.  Com3: 0007.  Com4: 0000.

CHIP Name: 1TO4INTGEN
Type: Device.    Identif: 04.          BUS: 0001.
Addres 1: 00002004.          Address 2: 00002017.
Com1: 0006.  Com2: 0006.  Com3: 0006.  Com4: 0006.

CHIP Name: I8259PIC
Type: Device.    Identif: 06.          BUS: 0001.
Addres 1: 0000202C.          Address 2: 0000202D.
Com1: 0002.  Com2: 0077.  Com3: 0000.  Com4: 0000.
```



Il PIC ha come suo gestore delle interruzioni il processore, con identificativo 2, ed è connesso alla linea di priorità 7.

Noi tipicamente sappiamo che il processore ha un encoder di priorità, 7 linee con dei livelli fissi, su ogni linea in realtà potremmo avere più dispositivi con priorità diversa, sulla linea 1 possiamo avere dispositivi con priorità da 0 a 7, *la priorità sulla linea si può gestire con meccanismi di daisy chain, posizione fisica rispetto al processore*. **Essendo tutto legato alla linea interrompente, quando riceviamo l'interruzione e la dobbiamo servire, servirebbero dei meccanismi aggiuntivi per capire chi ha interrotto**, si complicherebbe il tutto perché la linea con gli autovettori dice il vettore da usare, anche con il PIC dice direttamente l'ISR da invocare, unica per tutti i dispositivi connessi nella linea. **Nell'ISR dovremmo inventarci qualcosa per capire chi ha interrotto e chi è il più prioritario, interroghiamo i registri di stati dei dispositivi, nel daisy chain, e appena troviamo chi ha interrotto facciamo il pezzettino dell'ISR, relativa a tutti, che è legata proprio a quel dispositivo**.

Questo è per capire perché Asim usa sempre linea e priorità, noi abbiamo messo 77 ma anche se avessimo messo 72 non sarebbe cambiato perché è l'unico dispositivo, nel nostro caso, sulla linea 7. Facciamo partire il programma, carichiamo il listato, le memorie. Troveremo nella memoria del PIC tante interruzioni diverse, associate ai vari timer del sistema 1to4.

Codice: prendiamo l'indirizzo del PIC, intgen e terminale, inizializziamo il terminale, poi facciamo l'init del PIC. Per il PIC per prima cosa scriviamo **in TR, scriviamo 64, ovvero 40 in esadecimale, in modo che a seconda della linea interrompente nei 3 bit meno significativi il PIC scrive la codifica della linea che ha interrotto in quel momento.** Il successivo accesso sarà al registro di controllo, quando andiamo nel registro pari. Di tutte le configurazioni che ci possono servire ci interessa che nel bit di controllo ***abbiamo automatic end of interrumpt a 0, per cui ogni volta nelle interruzioni prima di tornare indietro dobbiamo resettare il bit dell'ISR***. Poi, settiamo la maschera, mettiamo tutti 0 per cui non mascheriamo niente, facciamo l'azzeramento del registro di stato e configuriamo il gestore delle interruzioni.

```
1) SCRITTURA NEL REGISTRO TR    ************
   < primo accesso in scrittura all'indiriz

    viene settato il valore $40 = 01000|000
   il vettore di base (primo vettore utile

   MOVE.B  #$40,1(A0)

2) SCRITTURA NEL REGISTRO CTRL    **********
   < accesso in scrittura all'indirizzo par
   < i prossimi accessi in lettura saranno
   *CTRL: se bit7=0, (è questo il caso)
   *IRR: se bit7=1 e bit6=0
   *ISR: se bit7=1 e bit6=1  >

    si pone AEOI=0, quindi appena prima di r

   MOVE.B  #$00,(A0)
```

```
*Inizializzazione registri timer 1

    MOVE.B  #$10,4(A1)      inizializza pr=1; linea=0
    MOVE.B  #$01,5(A1)      inizializza vect#=1
    MOVE.B  #$20,6(A1)      inizializza CountH
    MOVE.B  #$00,7(A1)      inizializza CountL

*Inizializzazione registri timer 2

    MOVE.B  #$11,8(A1)      inizializza pr=1; linea=1
    MOVE.B  #$02,9(A1)      inizializza vect#=2
    MOVE.B  #$1b,$a(A1)     inizializza CountH
    MOVE.B  #$00,$b(A1)     inizializza CountL

*Inizializzazione registri timer 3

    MOVE.B  #$12,$c(A1)     inizializza pr=1; linea=2
    MOVE.B  #$03,$d(A1)     inizializza vect#=3
    MOVE.B  #$1b,$e(A1)     inizializza CountH
    MOVE.B  #$00,$f(A1)     inizializza CountL

*Inizializzazione registri timer 4

    MOVE.B  #$13,$10(A1)        inizializza pr=1; linea=3
    MOVE.B  #$04,$11(A1)        inizializza vect#=4
    MOVE.B  #$1b,$12(A1)        inizializza CountH
    MOVE.B  #$00,$13(A1)        inizializza CountL
```

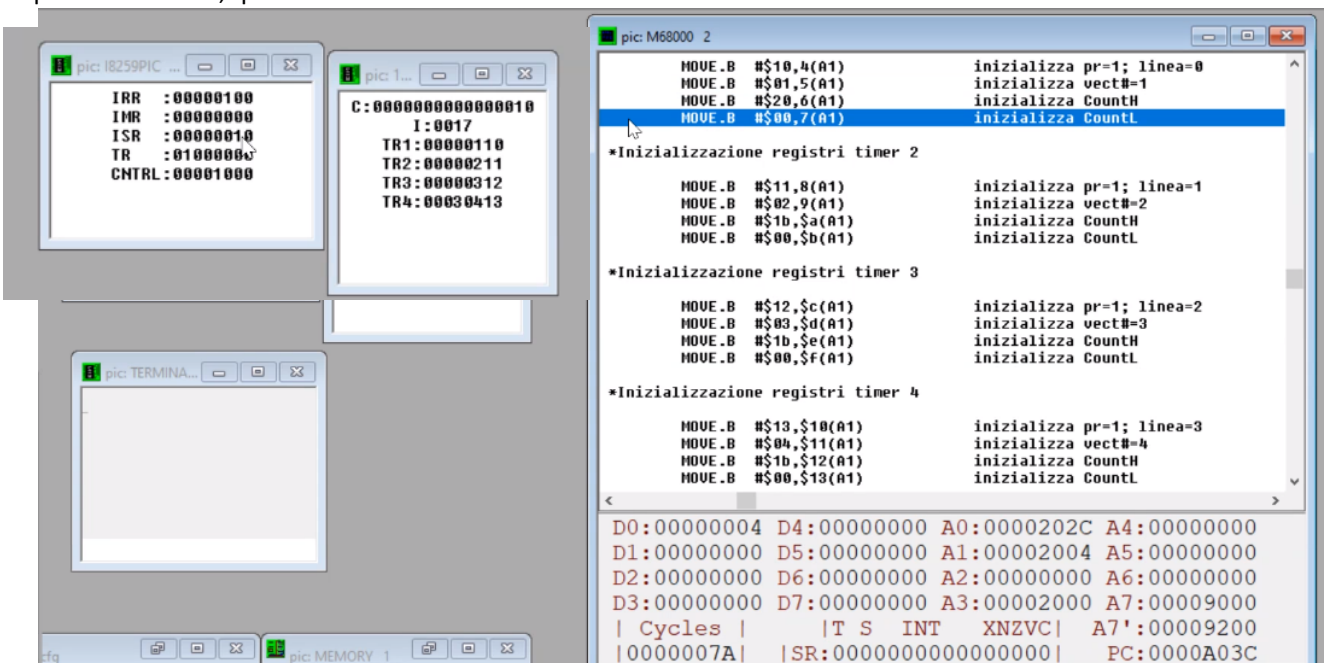
Ci sono 4 sezioni diverse per i 4 timer, *per ogni sezione, tramite gli indirizzi relativi del modello di programmazione, diciamo che il primo timer è associato alla linea 0, il secondo alla linea 1, il 3° alla linea 2 e il 4° alla linea 3. Come priorità abbiamo messo a tutti 1*, in tal modo vediamo solo la linea e non la priorità. **Per ogni timer abbiamo messo sia il valore L sia il valore H, come contatori timer, hanno tutto tempi diversi e scattano in momenti diversi**.

Poi c'è la sezione programmatica, per cui se scriviamo un bit in un registro si avvia, **facciamo allora delle move in 02, l'interruzione programmatica va sulla linea 7 del PIC e quando mettiamo 02 generiamo veramente l'interruzione**, di livello 7. Questo era il main, in cui abbiamo solo inizializzato le periferiche. Dobbiamo avere un'interruzione per ogni livello, non abbiamo usato la linea 4,5 e 6 ma troviamo il codice perché così possiamo sfruttarlo. ***Le ISR non fanno altro che stampare un numero al terminale***, dato dalla linea che servono, poi resettano manualmente il bit nel registro ISR. Quindi, dobbiamo settare eo1 a 1 e poi resettare i primi 3 bit, perché i primi 4 bit del registro di controllo servono per resettare manualmente il bit ISR che stiamo servendo (figura pagina successiva).

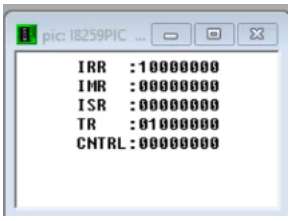
```
*Inizializzazione device 1to4intgen
    MOVE.B  #$17,(A1)       Intgen inizializza registro I con pr=1 e linea=7
    MOVE.B  #$02,2(A1)      Intgen registro C lancia int sulla linea fisica indicata

loop    JMP loop        <<<loop di attesa di int
```

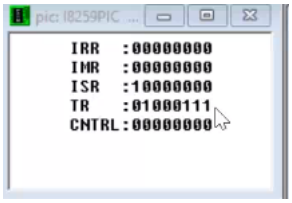
```
* ISR associata alla linea 0
    ORG $8000
int0    MOVE.B  #48,(A3)            stampa 0 a terminale
    MOVE.B  #$08,(A0)       setta EOI=1 per resettare il bit 0 di ISR (1000)
    RTE

* ISR associata alla linea 1
    ORG $8100
int1    MOVE.B  #49,(A3)            stampa 1 a terminale
    MOVE.B  #$09,(A0)       setta EOI=1 per resettare il bit 1 di ISR (1001)
    RTE

* ISR associata alla linea 2
    ORG $8200
int2    MOVE.B  #50,(A3)            stampa 2 a terminale
    MOVE.B  #$0A,(A0)       setta EOI=1 per resettare il bit 2 di ISR (1010)
    RTE

 * ISR associata alla linea 3
    ORG $8300
int3    MOVE.B  #51,(A3)            stampa 3 a terminal
    MOVE.B  #$0B,(A0)       setta EOI=1 per resettare il bit 3 di ISR (1011)
    RTE

 * ISR associata alla linea 4
    ORG $8400
int4    MOVE.B  #52,(A3)            stampa 4 a terminale
    MOVE.B  #$0C,(A0)       setta EOI=1 per resettare il bit 4 di ISR (1100)
    RTE
```

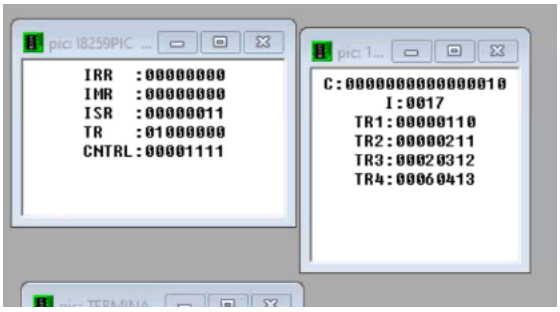
Vediamo l'esecuzione, inizializziamo il terminale, tutti 0 nel registro di controllo e nel registro di maschera e di stato, poi configuriamo i timer, ora in tr1,tr2,tr3,tr4 avremo dei valori, per il primo abbiamo messo un valore in counth ed esce 1f, che decrescerà dopo ogni tic, perché abbiamo fatto partire i timer, poi configuriamo il secondo dove mettiamo 1b, abbiamo già 1° perché decresce, quando le due cifre vanno a 0 si ha l'interruzione.



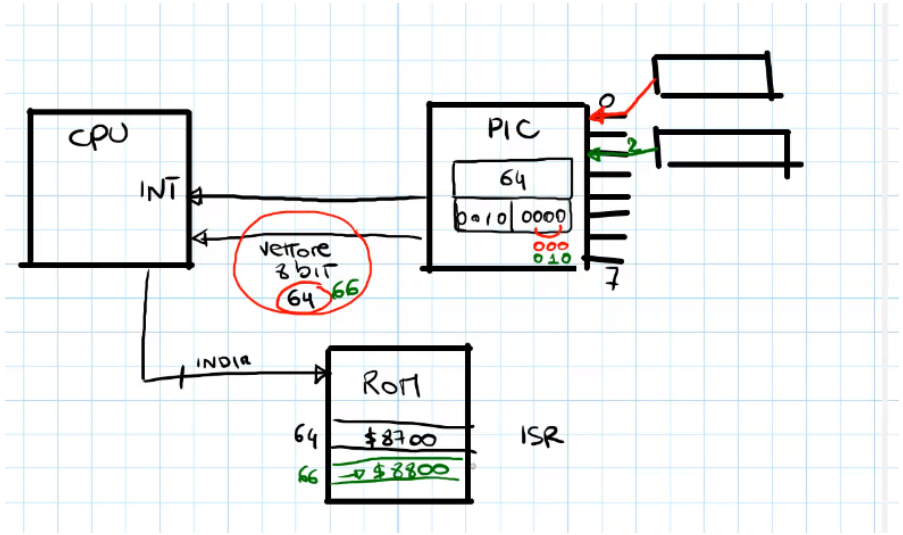
Ora generiamo l'interruzione sulla linea 7, **per l'interruzione programmatica, nell'IRR si è alzato il bit 7** (figura a destra) **quindi si è alzata un'interruzione in ingresso al PIC, IRR si è settato e dobbiamo eseguirlo, per farlo dobbiamo mettere in TR 111 che codifica il livello 7** (figura a sinistra), *così il PIC scatena sul processore l'identificativo del vettore delle interruzioni, ovvero 64+7, in tal modo il processore fa il suo ciclo di interrupt vettorizzato*, mette nel pc l'indirizzo dell'ISR di livello 7, ora sul terminale vediamo il numero 7. Ora dobbiamo azzerare manualmente il bit nel registro ISR. *Vediamo i timer, andiamo un poco avanti finchè non arriva il primo a 0, vediamo tr2, il più vicino, il secondo timer è collegato alla linea 1 per cui ora si alza in IRR il secondo bit (linea 1), allo stesso tempo va a 0 anche il tr1, il processore fa ad eseguire il codice della linea 1, in TR si mette 1*, nel codice serviamo la linea 1 ma nel frattempo diventa 0 anche il valore del primo timer che aveva priorità 0, che è più prioritaria di 1, allora ci aspettiamo che venga eseguito il codice non più della linea 1 ma della 0, perché stavamo eseguendo un codice meno prioritario, arriva







un'interruzione più prioritaria e passiamo a eseguirla**. Nell'ISR abbiamo entrambi i bit alti perché li stiamo eseguendo entrambi, ora andiamo con 0 in TR (vettore 64), azzeriamo il bit nell'ISR** (figura a sinistra)**e poi azzeriamo anche il secondo quando finiamo anche quella**. Nel frattempo si azzera anche tr4 e quindi continuiamo così. Possiamo cambiare il valore del conteggio h così da far arrivare le interruzioni in momenti differenti. Man mano che arriva un'interruzione si alza IRR, il PIC deve dare l'identificativo del vettore da eseguire. **Nell'interrupt vettorizzato il dispositivo scarica il proprio identificativo di 8 bit con cui il processore va in tabella, invece qui lo fa il PIC, è il PIC a dare al processore il vettore da attivare a quel momento, lo fa perché ogni volta appende alla parte iniziale del TR il codice su 3 bit della linea che sta servendo** e così crea un identificativo a 8 bit, dato al processore tramite il bus dati e usato per inizializzare la tabella delle eccezioni. Il processore esegue l'interruzione e poi ritorna il controllo. Se arriva un'interruzione più prioritaria il processore deve servire quell'interruzione*, il PIC di volta in volta gestisce le interruzioni, non più l'encoder interno al processore.*



Abbiamo da un lato il PIC e dall'altro il processore, nel PIC abbiamo 8 linee con priorità decrescente da 0 a 7, quello che entra dal PIC al processore è una linea di interruzione, il processore si aspetta di ricevere, nel vettorizzato, anche 8 bit che sono l'identificativo, il vettore, con cui il processore accede alla memoria rom e identifica, ad esempio se è 64, in 64 trova l'indirizzo della ISR da eseguire. Utilizziamo questo identificatore di 8 bit per accedere alla tabella delle eccezioni in memoria dove trova l'ISR da eseguire. **Il processore esegue l'interruzione, il PIC è solo un intermediario**. Noi sappiamo che i vettori dell'utente vanno da 64 in poi, il PIC contiene un registro che di base è 64, si scrive in binario 00100000***. Come fa il PIC a dare il vettore giusto, dato che è sempre la CPU ad interagire con la rom?*** *Se interrompe il dispositivo sulla linea 0 e il PIC decide che dev'essere servito perché sulla maschera non sono mascherate, scrive nei bit più significativi 000 e manda 64. Se invece interrompe un dispositivo nella linea 2, e il PIC decide di servirlo, codifica 010 (verde) non c'è 6 ma il numero 64+2=66, con questo numero il processore accedere all'altro vettore della rom, che ha l'indirizzo 8800, va a servire l'interruzione 8800, associata alla periferica verde mentre quella che sta in 64 è della rossa.* **Ora la priorità la gestisce il PIC, non la CPU, perché il PIC stesso è collegato a un livello di priorità, l'interruzione che manda però non è definita dalla priorità ma dal vettore**. Nel PIC c'è il registro ISR, il PIC con la maschera decide cosa passa e cosa no, poi mette in esecuzione un'interruzione alla volta, che manda al processore.