Excepciones en SPIM

Prof. Mariela Curiel,

Marzo 2013


Excepciones
Vectores de Interrupción
Niveles de Prioridad
Llamadas al sistema
Traps
Eventos Síncronos
Interrupciones
Manejadores de Interrupciones
Interrupciones por software
Sistema Operativo
FETCH-DECODE-EXECUTE
Eventos Asíncronos
Servicios del SOP
Eficiencia en el uso de los recursos

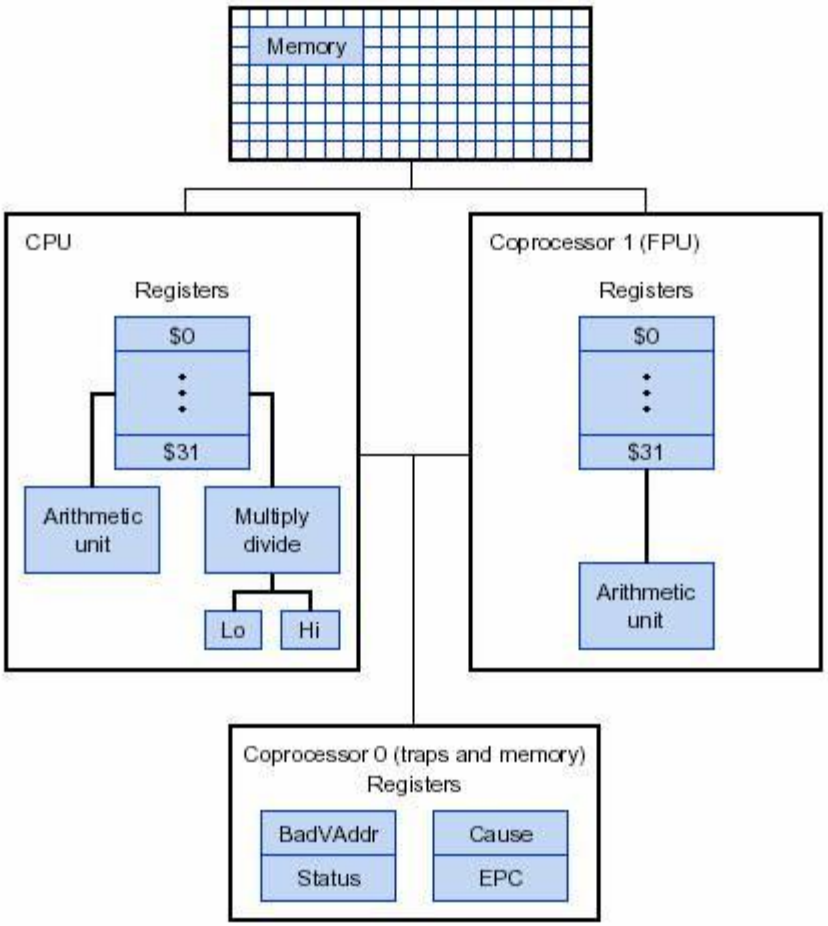


FIGURE A.10.1 MIPS R2000 CPU and FPU.

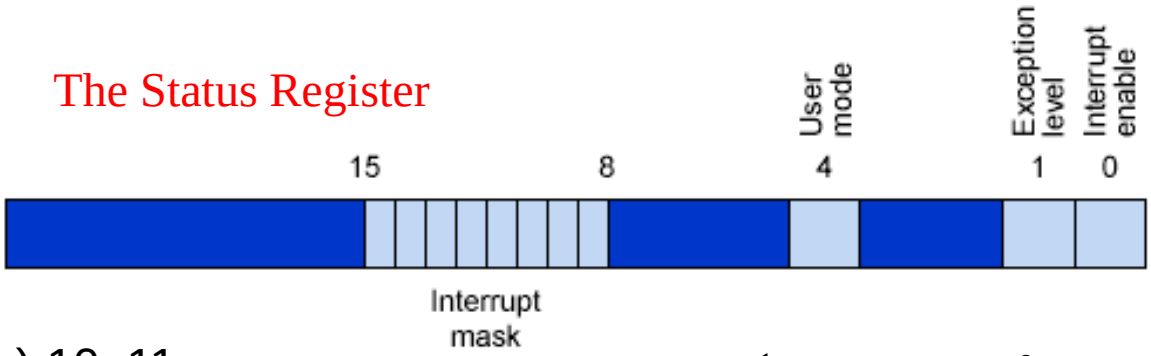
- Para acceder a los cuatro registros se emplean las instrucciones:
  - ✓ mfc0 rt,rd
    - Transfiere el contenido del registro *rd* del coprocesador 0 al registro *rt* de la CPU.
  - ✓ mtc0 rd,rt
    - Transfiere el contenido del registro *rt* de la CPU al registro *rd* del coprocesador 0.

Interrupciones en SPIM

Registros del Coprocesador 0 para el manejo de las excepciones e interrupciones

Register name	Register number	Usage
BadVAddr	8	memory address at which an offending memory reference occurred
Count	9	timer
Compare	11	value compared against timer that causes interrupt when they match
Status	12	interrupt mask and enable bits
Cause	13	exception type and pending interrupt bits
EPC	14	address of instruction that caused exception
Config	16	configuration of machine

The Status Register



0 deshabilita interrupciones
1 habilita interrupciones

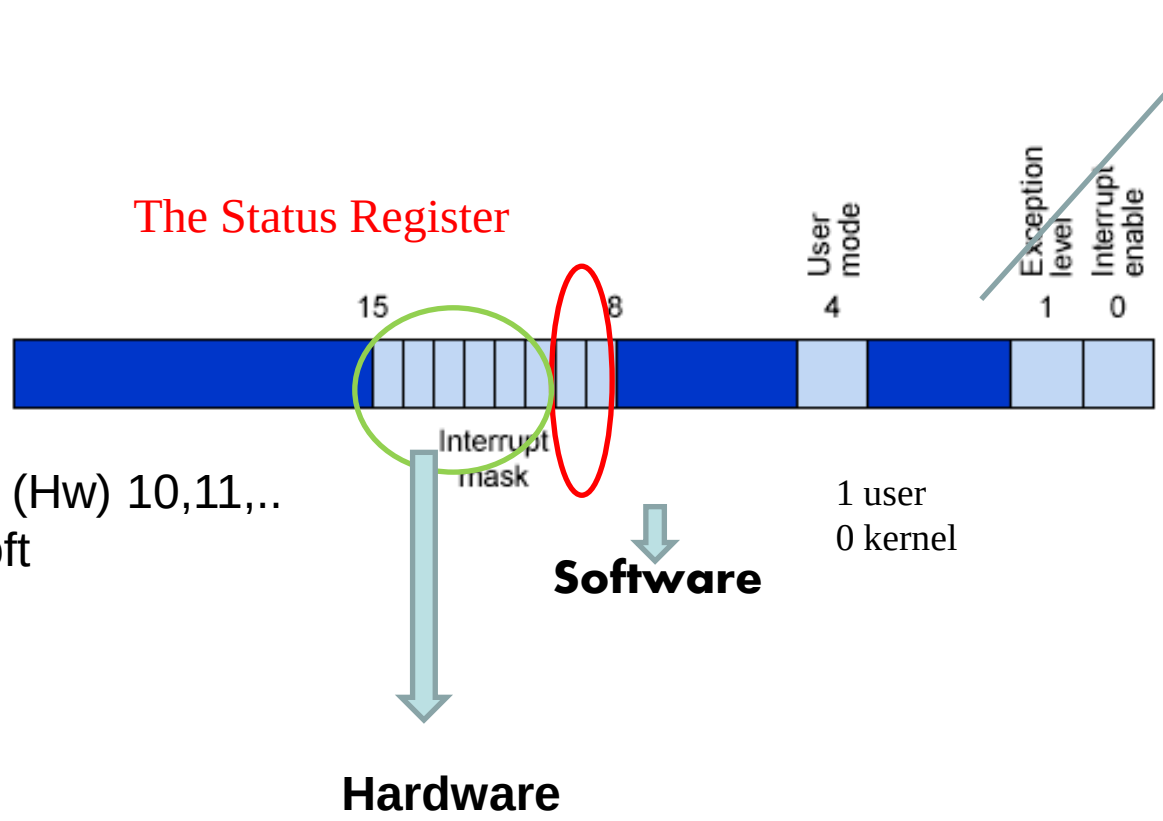
6 bits Int (Hw) 10, 11,..
2 bits Soft

1 habilita la interrupción
0 la deshabilita

1 user
0 kernel

0
1cuando ocurre una excepción (debería Inicializarse al finalizar el manejador de interrupciones)

Interrupciones en SPIM



6 bits Int (Hw) 10,11,..
2 bits Soft

Si un bit está encendido ocurre la interrupción a ese nivel.

Interrupciones en SPIM

Después de una excepción:

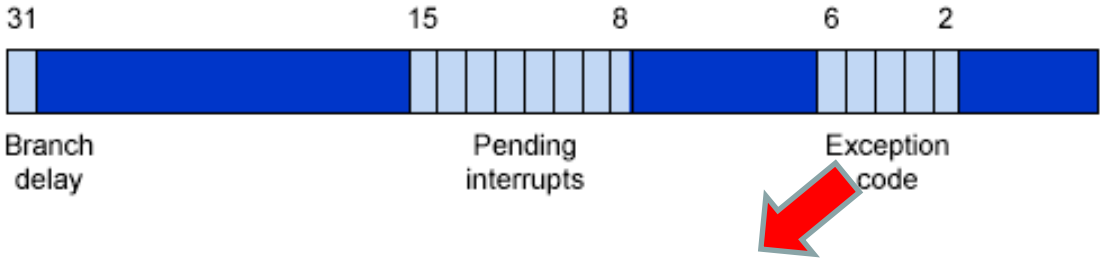
- EL EPC contiene la dirección de la instrucción que se estaba ejecutando al momento que ocurrió la interrupción.

- Si la excepción es una interrupción, la instrucción aún no se ha ejecutado.

- Todas las otras excepciones son causadas por la dirección de la instrucción que está en el EPC (excepto cuando la instrucción problema está en el slot delay de un *branch* o un *jump*).

Interrupciones SPIM

eret



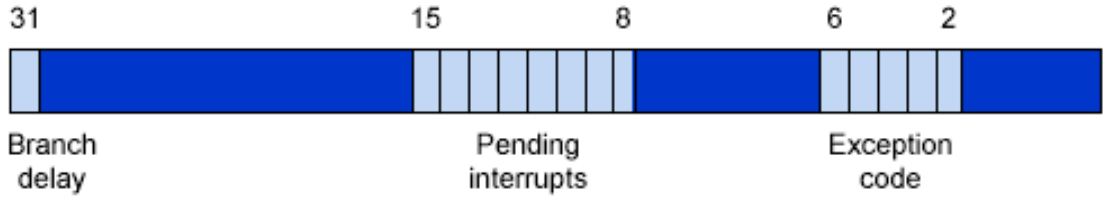
The Cause Register

Exception Code

Number	Name	Cause of exception
0	Int	interrupt (hardware)
4	AdEL	address error exception (load or instruction fetch)
5	AdES	address error exception (store)
6	IBE	bus error on instruction fetch
7	DBE	bus error on data load or store
8	Sys	syscall exception
9	Bp	breakpoint exception
10	RI	reserved instruction exception
11	CpU	coprocessor unimplemented
12	Ov	arithmetic overflow exception
13	Tr	trap
15	FPE	floating point

- ✓ Los cinco bits de *interrupciones pendientes* se corresponden con los cinco niveles de interrupción:
  - Cada uno de los bits se pone a 1 cuando se ha producido una interrupció en el nivel correspondiente y todavía no ha sido atendida.

Transmitter (0): Se prende el bit 10: 0x400
Receiver (1): Se prende el bit 11: 0x800
Timer (5): Se prende el bit 15: 0x8000



Interrupciones

- Cuando ocurre una excepción en SPIM, el procesador salta a un pedazo de código en la dirección 80000180hex (en el espacio de direcciones del kernel): el manejador de excepciones.
- El sistema operativo responde a las excepciones de diversas formas: termina el proceso (si hay un error), responde al servicio solicitado, hace un cambio de contexto, etc.

Manejador de Excepciones Simple

```
.ktext 0x80000180
# Select the appropriate one for the mode in which SPIM is compiled.
.set noat
move $k1 $at                # Save $at (también se usa en el manejador)
.set at

 sw $v0 s1                  # Not re-entrant and we can't trust $sp
 sw $a0 s2                  # But we need to use these registers

 mfc0 $k0 $13               # Cause register
 srl $a0 $k0 2              # Extract ExcCode Field

 andi $a0 $a0 0x1f   // máscara con los 5 bits del codigo de excepcion.

# Print information about exception.
#
li $v0 4                    # syscall 4 (print_str)
la $a0 __m1_
syscall
```

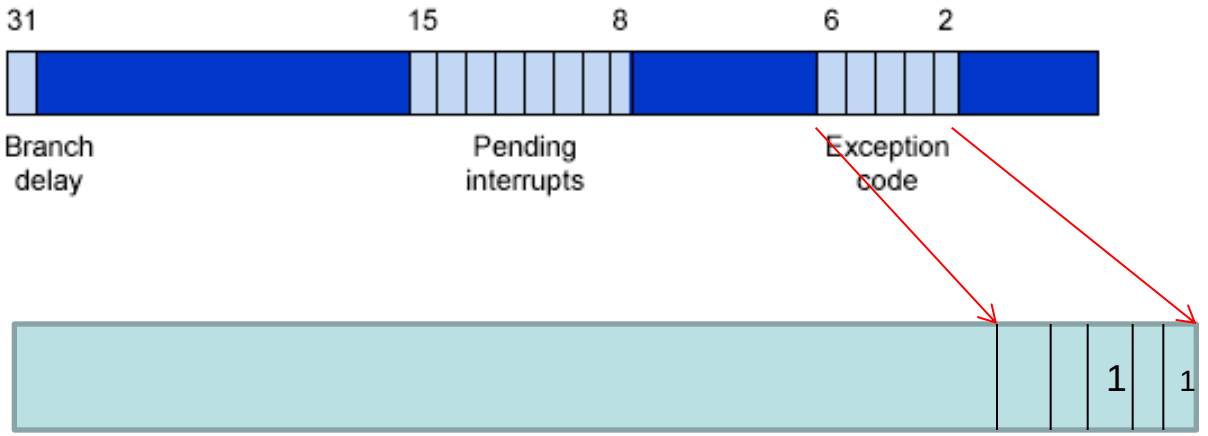
Register name	Register number	Usage
BadVAddr	8	memory address at which an offending memory reference occurred
Count	9	timer
Compare	11	value compared against timer that causes interrupt when they match
Status	12	interrupt mask and enable bits
Cause	13	exception type and pending interrupt bits
EPC	14	address of instruction that caused exception
Config	16	configuration of machine

$at

```
[00400060] 3c011001  lui $1, 4097 [mesg1]     ; 23: la $a0, mesg1
[00400064] 34240014  ori $4, $1, 20 [mesg1]
[00400068] 34020004  ori $2, $0, 4            ; 24: li $v0, 4
```

.

```
mfc0 $k0 $13                # Cause register
srl $a0 $k0 2               # Extract ExcCode Field

andi $a0 $a0 0x1f   // máscara con los 5 bits del codigo de excepcion.

# Print information about exception.
#
li $v0 4                    # syscall 4 (print_str)
la $a0 __m1_
Syscall

li $v0 1
srl $a0 $k0 2
andi $a0 $a0 0x1f
syscall
```



1 1 1 1 1

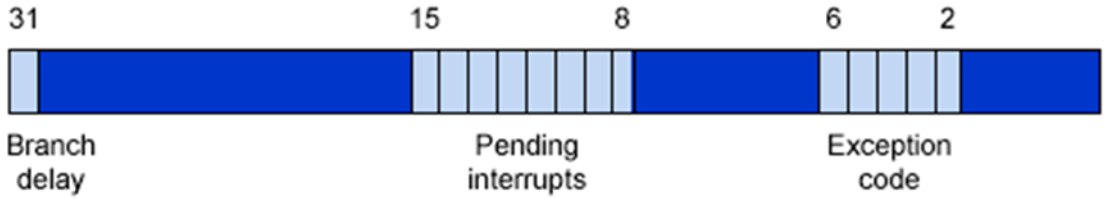
0 0 1 0 1

```
li $v0 4                # syscall 4 (print_str)
andi $a0 $k0 0x3c  //          111100 se haçe la máscara sin el shift
                        //          sólo se carga el código
lw $a0 __excp($a0)  // Carga la direccion del mensaje a imprimir
nop
syscall

bne $k0 0x18 ok_pc          # Bad PC exception requires special checks
nop

mfc0 $a0 $14                # EPC
andi $a0 $a0 0x3 # Is EPC word-aligned?
beq $a0 0 ok_pc
nop

li $v0 10               # Exit on really bad PC
syscall
```

0011000

```
            li $v0 4              # syscall 4 (print_str)
            andi $a0 $k0 0x3c  //        111100 se haçe la máscara sin el shift
                                //       sólo se carga el código
            lw $a0 __excp($a0)  // Carga la direccion del mensaje a imprimir
            nop
            syscall

__e1_:   .asciiz    "  [TLB]"
__e2_:   .asciiz    "  [TLB]"
__e3_:   .asciiz    "  [TLB]"
__e4_:   .asciiz    "  [Address error in inst/data fetch] "
__e5_:   .asciiz    "  [Address error in store] "
__e6_:   .asciiz    "  [Bad instruction address] "
__e7_:   .asciiz    "  [Bad data address] "
__e8_:   .asciiz    "  [Error in syscall] "
__e9_:   .asciiz    "  [Breakpoint]

……..

__excp:  .word __e0_, __e1_, __e2_, __e3_, __e4_, __e5_, __e6_, __e7_, __e8_, __e9_
         .word __e10_, __e11_, __e12_, __e13_, __e14_, __e15_, __e16_, __e17_,
__e18_,
         .word __e19_, __e20_, __e21_, __e22_, __e23_, __e24_, __e25_, __e26_,
__e27_,
         .word __e28_, __e29_, __e30_, __e31_
```

```
li $v0 4                # syscall 4 (print_str)
andi $a0 $k0 0x3c  //          111100 se haçe la máscara sin el shift
                   //          sólo se carga el código
lw $a0 __excp($a0)  // Carga la direccion del mensaje a imprimir
nop
syscall

bne $k0 0x18 ok_pc          # Bad PC exception requires special checks
nop

mfc0 $a0 $14                # EPC
andi $a0 $a0 0x3 # Is EPC word-aligned?
beq $a0 0 ok_pc
nop

li $v0 10               # Exit on really bad PC
syscall
```



000110 00

6 = bus error on instruction fetch

```
ok_pc:
        li $v0 4            # syscall 4 (print_str)
        la $a0 __m2_      // Ocurrio la excepcion y se ignora
        syscall

        srl $a0 $k0 2                # Extract ExcCode Field
        andi $a0 $a0 0x1f
        bne $a0 0 ret                # 0 means exception was an interrupt
        nop

# Interrupt-specific code goes here!
# Don't skip instruction at EPC since it has not executed.
```

// Codigo del manejador de interrupciones

```
ret:
# Return from (non-interrupt) exception. Skip offending instruction
# at EPC to avoid infinite loop.
#
        mfc0 $k0 $14                # Bump EPC register
        addiu $k0 $k0 4             # Skip faulting instruction
                                    # (Need to handle delayed branch case here)
        mtc0 $k0 $14                # nuevo EPC


# Restore registers and reset procesor state
#
        lw $v0 s1           # Restore other registers
        lw $a0 s2

        .set noat
        move $at $k1                # Restore $at
        .set at

        mtc0 $0 $13                 # Clear Cause register

        mfc0 $k0 $12                # Set Status register
        ori  $k0 0x1                # Interrupts enabled
        mtc0 $k0 $12

# Return from exception on MIPS
        eret
```

Retorna a la instruccion apuntada por EPC.
No se re-ejecuta la instruccion que fallo.
Coloca el bit Exection Level en 0 (del registro status)

E/S (memory mapped I/O)

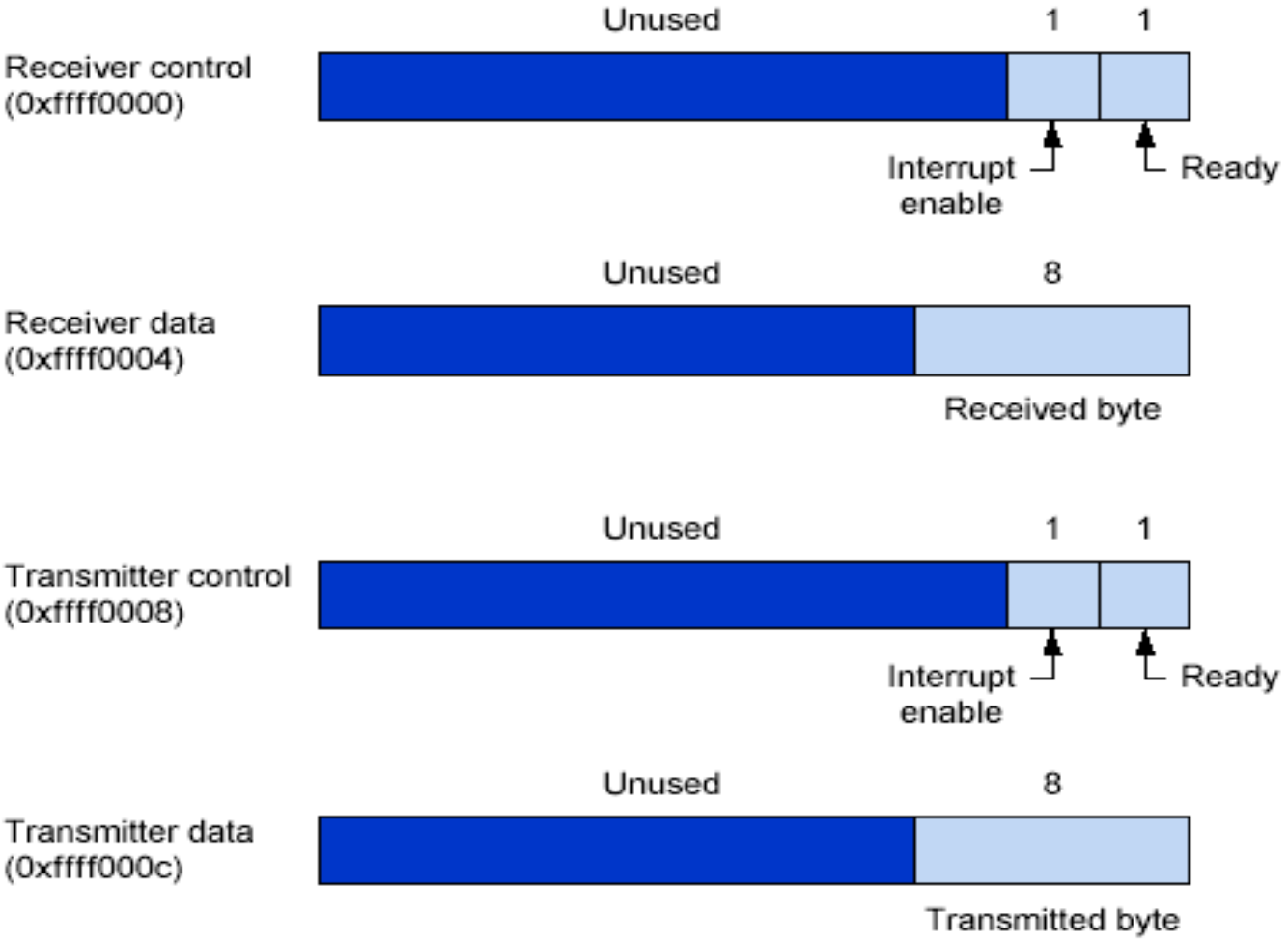


FIGURE A.8.1 The terminal is controlled by four device registers, each of which appears as a memory location at the given address. Only a few bits of these registers are actually used. The others always read as 0s and are ignored on writes.

Memory-mapped I/O

- Memory-mapped significa que cada registro aparece en una direccion de memoria especial
- Receiver: lee caracteres del teclado a medida que son tipeados.
- Transmitter: transmite caracteres al display. Las dos unidades son completamente independientes.

Memory-mapped I/O

- Receiver Control:
  - Bit Ready (Read-only): si es 1 implica que un carácter fue tipeado y se encuentra en el data register, aun sin procesar.Cambia de 0 a 1 si se tipea un carácter en el teclado y de 1 a 0 cuando se toma el carácter del registro de datos.
  - Interrupt enable (rw): Si es colocado en 1, se genera una interrupción (nivel 0) cuando el bit de ready está en 1

Memory-mapped I/O

- Receiver Data Register:
  - Los 8 bits menos significativos contienen el carácter tipeado y los otros bits contienen 0. Es R-only, una vez que se lee este valor, el bit de ready en el Receiver Control se coloca en 0.

Memory-mapped I/O

- Transmitter Control Register:
  - Ready (RO): si está en 1 significa que el transmitter está listo para aceptar un nuevo caracter. Si es 0 significa que el carácter anterior todavía no se ha impreso.
  - Interrupt enable (rw): si es colocado en 1 se solicita una interrupción a nivel 1 , cuando el bit de ready esté en 1. Cuando hay un carácter que escribir (en el Transmitter Data Register) el bit de ready se coloca en 0. El bit permanecerá en 0 hasta que haya transcurrido suficiente tiempo para enviar el carácter a la cónsola,

Memory-mapped I/O

- Transmitter Data Register:

  Se debe escribir sólo si el bit de ready está en 1, de lo contrario las escrituras se ignoran.