ANEXO B

HOJA DE DATOS RISCKER

1. SET DE INSTRUCCIONES RISCKER

1.1. SINTAXIS DE LA INSTRUCCIÓN RISKCER

Una Instrucción RISCKER - ASM está compuesta de hasta cuatro campos:

[Etiqueta] **:** *[Operación] [Operandos]* **;** *[Comentario]*

[Etiqueta]: Nombre simbólico de la localidad de memoria en que se encuentra la instrucción.

[Operación]: Indica la operación que se va a ejecutar *(Ver Tabla 1)*

[Operandos]: El campo operandos tiene hasta cuatro sub-campos dependiendo del tipo de operación.

Instrucción Completa:

```
[Etiqueta] : [Operación] [Rd],  [Rs], [Rt], #[Imm/Dir]  ;[Comentario]
```

[Rd] Registro Destino: Indica el registro en el cual se guardará el resultado de la operación.
[Rs] Registro Fuente (Source): Indica el registro en el que se encuentra un primer operando.
[Rt]* Registro Fuente: Indica un segundo registro con un dato a operar, sólo si la operación lo requiere.
#[Imm/Dir]* Valor Inmediatio o Dirección: Indica un número o dirección de 16bits.

(*) Estos campos dependen del tipo de operación.

[Comentario]: Se escribe al final de la instrucción precedido por un ";".

Ejemplos:

```
beq     rg1, rg0, #16    ;Salto condicional, si rg1 es igual a rg0 a la dirección 16
lw      rg3, r0, #0      ;Cargar el registro rg3 con el dato Memoria[r0+0]
inc     op               ;Incrementar el puerto de salida (op) y almacenar el resulta en op
```

La Tabla 2 muestra el formato de las distintas instrucciones RISCKER diviendo los campos que harán la función de operandos, código de operación (*Opcode*) y en el caso de instrucciones de asignación, los bits que identifian la función a realizar por la ALU.

Tabla 1. Listado de instrucciones RISCKER

Operación	Operandos	Descripción	
R-Type			
add	Rd, Rs, Rt	Rd = Rs + Rt	Rd = Rd + Rt
and		Rd = Rs & Rt	Rd = Rd & Rt
or		Rd = Rs | Rt	Rd = Rd | Rs
xor	Rd, Rt	Rd = Rs ^ Rt	Rd = Rd ^ Rs
nand		Rd = ~(Rs & Rt)	Rd = ~(Rd & Rs)
nor		Rd = ~(Rs | Rt)	Rd = ~(Rd | Rs)
sub		Rd = Rs - Rt	Rd = Rd - Rs
neg	Rd,Rs Rd	Rd = -Rs	Rd = -Rd
inc		Rd = Rs + 1	Rd = Rd+1
mult	Rd,Rs,Rt	Rd[15:0] = Rs[7:0] * Rt[7:0] Rd[15:0] = Rd[7:0] * Rs[7:0]	
div	Rd,Rt	Rd = {Rs[15:0] / Rt[15:0],Rs[15:0] % Rt[15:0]} Rd = {Rd[15:0] / Rs[15:0],Rd[15:0] % Rs[15:0]}	
rot	Rd, Rt, shamt Rd,shamt	Rd = Rt rotates left by shamt	Rd = Rd rotates left by shamt
sll		Rd = Rt << Shift Amount	Rd = Rd << Shift Amount
srl		Rd = Rt >> Shift Amount	Rd = Rd >> Shift Amount
sra		Rd = Rt>>Arithmetic by shamt	Rd = Rd>>Arithmetic by shamt
I-Type			
addi	Rd,Rs,imm Rd,imm	Rd = Rs + imm	Rd = Rd + imm
andi		Rd = Rs & imm	Rd = Rd & imm
ori		Rd = Rs | imm	Rd = Rd | imm
xori		Rd = Rs ^ imm	Rd = Rd ^ imm
nandi		Rd = ~(Rs & imm)	Rd = ~(Rd & imm)
nori		Rd = ~(Rs | imm)	Rd = ~(Rd | imm)
multi		Rd = Rs[15:0] * imm	Rd = Rd[15:0] * imm
divi	Rd,Rs,imm Rd, imm	Rd = {Rs[15:0] / imm, Rs[15:0] % imm} Rd = {Rd[15:0] / imm, Rd[15:0] % imm}	
I-Type			
sw	Rd, Rs, imm Rd, imm	Memory[Rs+imm] = Rd Memory[Rd+imm] = Rd	
lw	Rd, Rs, imm Rd, imm	Rd = Memory[Rs+imm] Rd = Memory[Rd+imm]	
blt	Rd, Rs, target	((Rd-Rs)<0) ? PC = target	
beq	Rd, Rs, target Rd, target	((Rd-Rs)=0) ? PC = target ((Rd-Rd)=0) ? PC = target	

Tabla 1. Listado de instrucciones RISCKER (continuación)

J-Type		
j	target	PC = target
jrlw	Rs, Rd, imm Rd, imm	PC = Rs ; Rd = imm PC = Rs ; Rt0 = 0
jal	Rd, target	Rd = PC+1; PC = target

Tabla 2. Formato de los distintos tipos de instrucciones de la arqutiectura RISCKER

R-Type					
Assembler	**Opcode 4bits**	**Rd 6bits**	**Rs 6bits**	**Rt 6bits**	**Function 10bits**
add	0000	Rx	Rx	Rx	0000000000
and	0000	Rx	Rx	Rx	0000000001
or	0000	Rx	Rx	Rx	0000000010
xor	0000	Rx	Rx	Rx	0000000011
nand	0000	Rx	Rx	Rx	0000000100
nor	0000	Rx	Rx	Rx	0000000101
sub	0000	Rx	Rx	Rx	0000000110
neg	0000	Rx	Rx		0000000110
inc	0000	Rx	Rx		0000000111
mult	0000	Rx	Rx	Rx	0010001000
div	0000	Rx	Rx	Rx	0011001001
rot	0000	Rx		Rx	000100sham
sll	0000	Rx		Rx	000101sham
srl	0000	Rx		Rx	000110sham
sra	0000	Rx		Rx	000111sham
I-Type Operations					
Assembler	**Opcode 4bits**	**Rd 6bits**	**Rs 6bits**	**Immediate (16 bits) Two's Complement**	
addi	1000	Rx	Rx	imm	
andi	1001	Rx	Rx	imm	
ori	1010	Rx	Rx	imm	
xori	1011	Rx	Rx	imm	
nandi	1100	Rx	Rx	imm	
nori	1101	Rx	Rx	imm	
multi	1110	Rx	Rx	imm	
divi	1111	Rx	Rx	imm	

Tabla 2. Formato de los distintos tipos de instrucciones de la arquitectura RISCKER (continuación)

I-Type				
sw	0010	Rx	Rx	imm
lw	0011	Rx	Rx	imm
blt	0100	Rx	Rx	target
beq	0101	Rx	Rx	target
J-Type				
Assembler	**Opcode 4bits**	**Rd 6bits**	**Rs 6bits**	**Immediate (16 bits) Two's Complement**
j	0001	target		
jrlw	0110	Rx		
jal	0111	Rx	target	

1.2. CARACTERÍSTICAS PRINCIPALES DEL HARDWARE

1.2.1. Registros

RISCKER cuenta con 64 registros organizados como se muestran en la tabla 3, divididos en tres tipos.

1.2.1.1. Registros Modificables

Estos registros permiten escritura y lectura en todas las instrucciones. Están divididos para cumplir distintas funciones de programa pero su implementación en hardware es la misma. El buen uso de estos registros permite la reutilización sencilla de código.

Valores de Retorno (rv0-rv7)
Registros destinados para retornar valores de subrutinas.

Argumentos de subrutina (rp0-rp7)
Registros destinados para llevar valores como argumentos de subrutinas.

Registros de uso Temporal (rt0-rt7)
Registros para uso temporal como intermediario en algoritmos complejos o de subrutinas.

Registros Globales (rg0-rg21)
Registros para definir constantes y variables de programa.

Tabla 3. Archivo de registros RISCKER

# of Registers	Name	Index	Descripción
1	r0	0	Valor 0
8	rv0-rv7	1:8	Valores de Retorno
8	rp0-rp7	9:16	Argumentos de Subrutina
8	rt0-rt7	17:24	Registros de uso Temporal
22	rg0-rg21	25-46	Registros Globales
4	ra0-ra3	47-50	Dirección de Retorno
1	sp	51	Puntero de la Pila
2	pr1-pr2	52-53	Periodo de Timers
2	ti1-ti2	54-55	Direcciones de Interrupciones por Timers
2	eir1-eir2	56-57	Dirección de Interrupciones Externas
1	op	58	Registro de Salida
1	rc	59	Registro de Control
2	tmr1-tmr2	60-61	Registros de Timers
1	ip	62	Registro de Entrada
1	ipc	63	PC en el momento de la interrupción
64			

Modificables
Proposito Especial
Sólo Lectura

Dirección de Retorno (ra0-ra3)
Registros para direcciones de retorno de procedimientos. Se permiten hasta cuatro procedimientos en cadena teniendo en cuenta la dirección de retorno correspondiente.

Puntero de la Pila (sp)
Registro que debe apuntar a la última localidad ocupada de la pila. Su utilización es completamente por software, decrementando su valor antes de una escritura e incrementándolo después de una lectura.

1.2.1.2. Registros de Sólo Lectura

Son registros especiales que contienen información relacionada al funcionamiento del microprocesador. Su entrada está conectada directamente a señales externas. Y por lo tanto sus valores no pueden ser modificados por software.

Registro 0 (r0)
Constante cero mapeada en el archivo de registros.

Timers (tmr1, tmr2)
Contiene el valor del contador de cada temporizador.

Puerto de Entrada (ip)
Contiene la información del puerto de entrada.

Interrupt PC (ipc)
Contiene el valor del Contador de programa antes de ejecutar una interrupción.

1.2.1.3. Registros de propósito especial

Son registros que configuran características del procesador. Su contenido puedo ser leído pero estas señales también están conectadas a componentes externos.

Periodo del Timer (pr1, pr2)
Contiene el valor en el cual ocurre el desbordamiento de cada temporizador con lo cual genera una señal de interrupción.

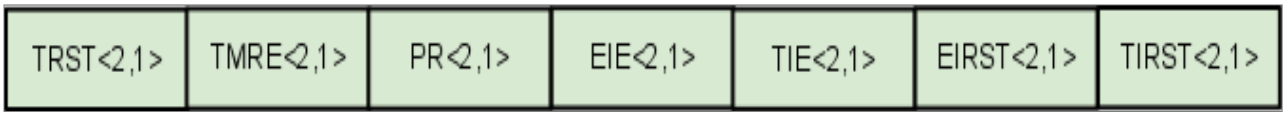
Direcciones de Interrupción por desbordamiento de Timer (ti1, ti2)
Contiene la dirección en la que se encuentra el servicio de interrupción por desbordamiento de cada timer. Esta señal está conectada al circuito controlador de interrupciones.

Direcciones de Interrupción Externa (eir1, eir2)
Contiene la dirección en la que se encuentra el servicio de cada interrupción externa. Esta señal está conectada al circuito controlador de interrupciones.

Puerto de Salida (op)
Registro conectado como salida del microprocesador.

Registro de control (rc)
Registro de banderas de configuración de interrupciones y temporizadores:

Figura 1. Organización de señales en el registro de control RISCKER



[0:1] TIRST<1,2> = Timer Interrupt Reset Flag
Borra la bandera de interrupción por termporizador 1 o 2

[2:3] EIRST<1,2> = External Interrupt Reset Flag
Borra la bandera de interrupción externa 1 o 2

[4:5] TIE<1,2> = Timer Interrupt Enable
Habilita la Interrupción por temporizador 1 o 2

[6:7] EIE<1,2> = External Interrupt Enable
Habilita la Interrupción externa 1 o 2

[8:9] PR1<1,2> = Timer1 Prescaler

[10:11] PR2<1,2> = Timer2 Prescaler

[12:13] TMRE<1,2> = Timer Enable
Habilita el temporizador 1 o 2

[14:15] TRST<1,2> = Timer Reset
Borra el contenido del temporizador 1 o 2

1.2.2. Ruta de Datos y Unidad de Control

La figura 2 es el diagrama de la ruta de datos RISCKER con las señales de control provenientes de la HCU (Hardware Controlled Unit), en esta imagen se pueden observar el archivo de registros, la ALU (Arithmetic Logic Unit) y la unidad de memoria. Las señales de control están descritas en la tabla 4 y su comportamiento está determinado por los estados descritos en la figura 3 y las tablas 5 y 6.

La tabla 5 muestra las señales que dependen de los estados de la unidad de control. La tabla 6 muestra las señales que dependen únicamente del opcode de la instrucción y por lo tanto no cambian durante los ciclos de ejecución de la misma.

La figura 3 representa la máquina de estados de la HCU RISCKER. Esta contempla todas las instrucciones en los estados *FETCH, DECODE* y *EXECUTE*, como también los estados de reconocimiento y retorno de interrupción.

Figura 2. Ruta de Datos RISCKER

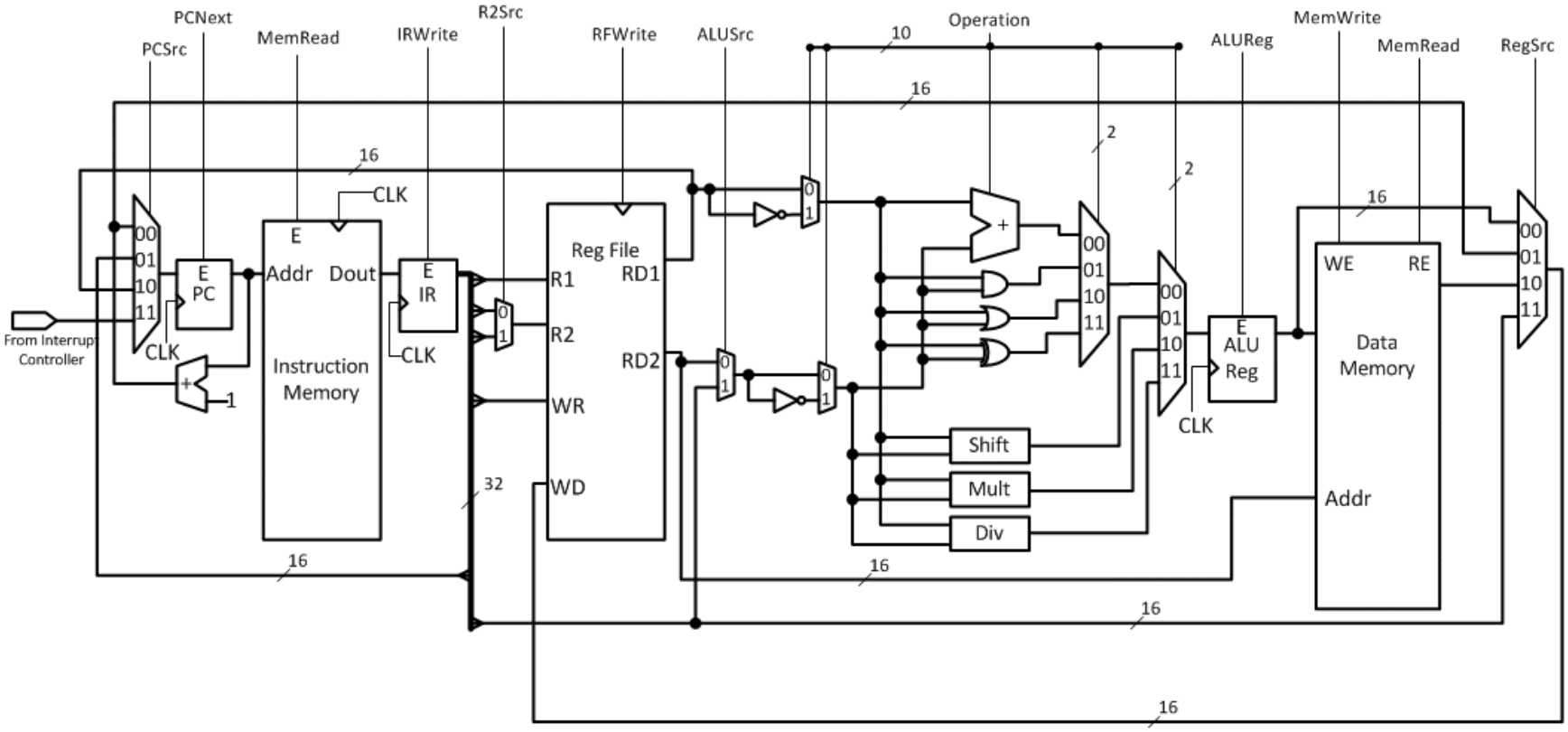


Tabla 4. Señales de Control

PCsrc	Se encarga de seleccionar la fuente de carga del registro PC
PCNext	Flanco de reloj del registro PC
IRwrite	Flanco de reloj del registro de instrucción
R2src	Fuente de la dirección del registro 2 del archivo de registros
RFwrite	Flanco de reloj de escritura del archivo de registros
ALUsrc	Fuente del operando B de la unidad de operaciones
Operation	Bus de control que selecciona la operación indicada
ALUreg	Flanco de reloj que registra el resultado de la ALU
MemWrite	Habilitador de escritura de la memoria de datos
MemRead	Habilitador de lectura de la memoria de datos
RegSrc	Fuente de datos a escribir en el archivo de registros
IPCWrite	Flanco de reloj del registro PC' de interrupción
INToggle	Señal que indica el inicio o final de servicio de interrupción.

Tabla 5. Señales de control de la máquina de estados

Estado	ipc_write	intoggle	pc_next	ir_write	rf_write	alu_reg	mem_write
idle	0	0	0	0	0	0	0
fetch	0	0	0	1	0	0	0
execute_op	0	0	1	0	0	1	0
execute_jrlw	0	0	1	0	1	0	0
write_op	0	0	0	0	1	0	0
write_sw	0	0	0	0	0	0	1
jump	0	0	1	0	0	0	0
interrupt	1	1	0	0	0	0	0
intjump	1	0	1	0	0	0	0
retfi	0	1	0	0	0	0	0

Figura 3. Diagrama de estados RISCKER

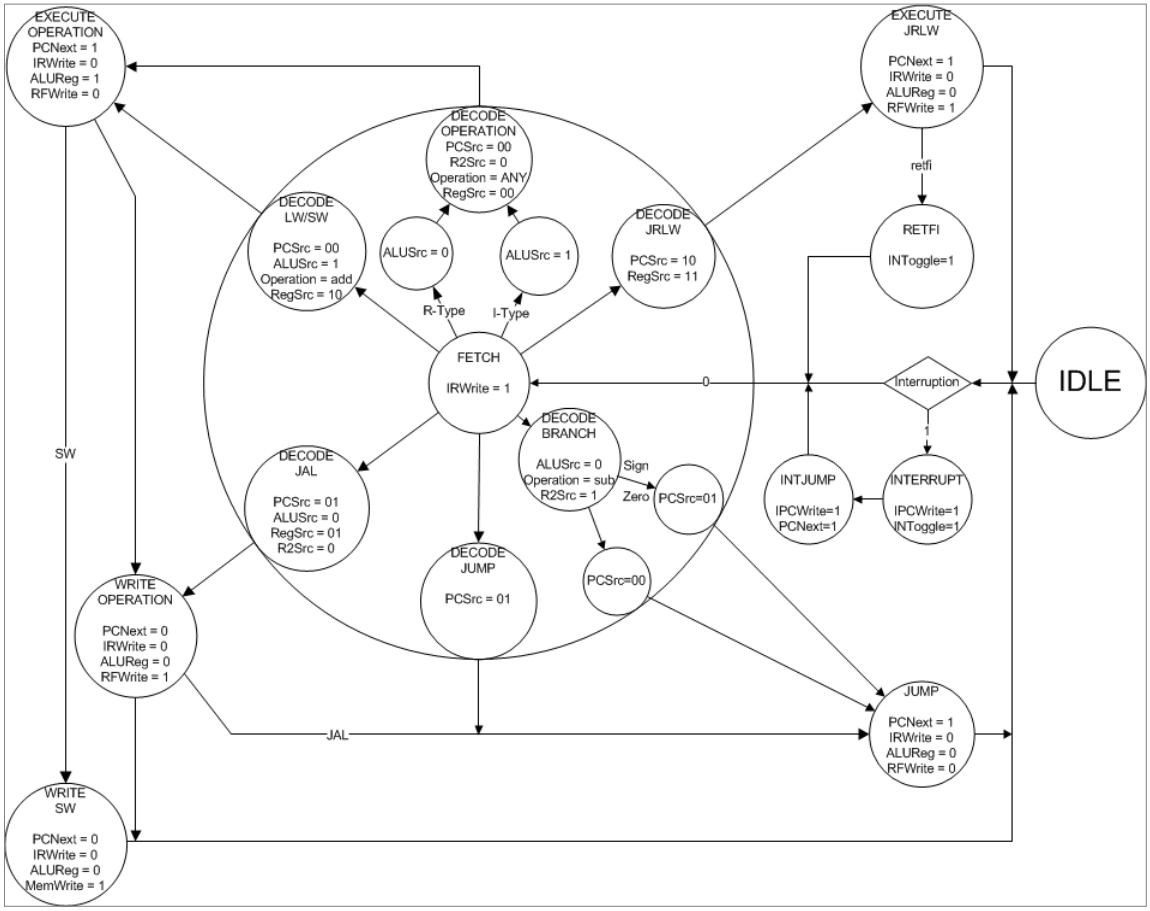


Tabla 6. Señales de control independientes de la máquina de estados

Instruction	PCSrc	R2Src	ALUSrc	Operation (11 bits)				RegSrc
				aluop	shamt	shift_mode	select	
add	00	0	0	000	XXXX	XX	00	00
and	00	0	0	001	XXXX	XX	00	00
or	00	0	0	010	XXXX	XX	00	00
xor	00	0	0	011	XXXX	XX	00	00
nand	00	0	0	100	XXXX	XX	00	00
nor	00	0	0	101	XXXX	XX	00	00
sub	00	0	0	110	XXXX	XX	00	00
neg	00	0	0	110	XXXX	XX	00	00
inc	00	0	0	111	XXXX	XX	00	00
mult	00	0	0	XXX	XXXX	XX	10	00
div	00	0	0	XXX	XXXX	XX	11	00
rot	00	0	0	XXX	SHAM	00	01	00
sll	00	0	0	XXX	SHAM	01	01	00
srl	00	0	0	XXX	SHAM	10	01	00
sra	00	0	0	XXX	SHAM	11	01	00
Instruction	**PCSrc**	**R2Src**	**ALUSrc**	**aluop**	**shamt**	**shift_mode**	**select**	**RegSrc**
addi	00	X	1	000	XXXX	XX	00	00
andi	00	X	1	001	XXXX	XX	00	00
ori	00	X	1	010	XXXX	XX	00	00
xori	00	X	1	011	XXXX	XX	00	00
nandi	00	X	1	100	XXXX	XX	00	00
nori	00	X	1	101	XXXX	XX	00	00
multi	00	X	1	XXX	XXXX	XX	10	00
divi	00	X	1	XXX	XXXX	XX	11	00
Instruction	**PCSrc**	**R2Src**	**ALUSrc**	**aluop**	**shamt**	**shift_mode**	**select**	**RegSrc**
lw	00	X	1	000	XXXX	XX	00	10
sw	00	X	1	000	XXXX	XX	00	XX
beq	01	1	0	110	XXXX	XX	00	XX
blt	01	1	0	110	XXXX	XX	00	XX
jrlw	10	0	X	XXX	XXXX	XX	XX	11

Tabla 6. Señales de control independientes de la máquina de estados (continuación)

Instruction	PCSrc	R2Src	ALUSrc	aluop	shamt	shift_mode	select	RegSrc
jal	01	X	X	XXX	XXXX	XX	XX	01
j	01	1	0	XXX	XXXX	XX	XX	01

1.2.3. Controlador de interrupciones

El manejo de las interrupciones en el microprocesador RISCKER se logra con el circuito expuesto en la figura 4. Dispone de cuatro fuentes de interrupción implementadas por defecto como dos interrupciones por temporizadores y dos interrupciones externas. El funcionamiento de este circuito está determinado por las siguientes señales.

Control Register: Registro de control del archivo de registros. En él se encuentran las señales “Enable” y “Reset” de cada interrupción.

Interruption Toggle: Señal proveniente de la unidad de control como bandera de entrada y salida de la rutina de servicio de interrupción.

Interruption: Bandera de entrada a la máquina de estados para reconocimiento de interrupciones.

IPC Write: Señal de escritura del contador de programa en el registro de almacenamiento para retorno de interrupciones. Esta señal borra automáticamente la señal “Interruption”.

Return from Interruption (Retfi): Señal que indica el retorno de la interrupción al programa mediante el “Reset” de cualquiera de las banderas.

Address Interruption X: Registros del archivo de registros que contienen la dirección de la rutina de servicio de interrupción correspondiente a la interrupción reconocida. Estas direcciones tienen un orden de prioridad configurable en hardware.

Interruption Address: Dirección de la rutina de servicio de interrupción correspondiente a la interrupción reconocida cuando se activa la señal *Interruption*.

Algoritmo para el servicio de interrupciones:

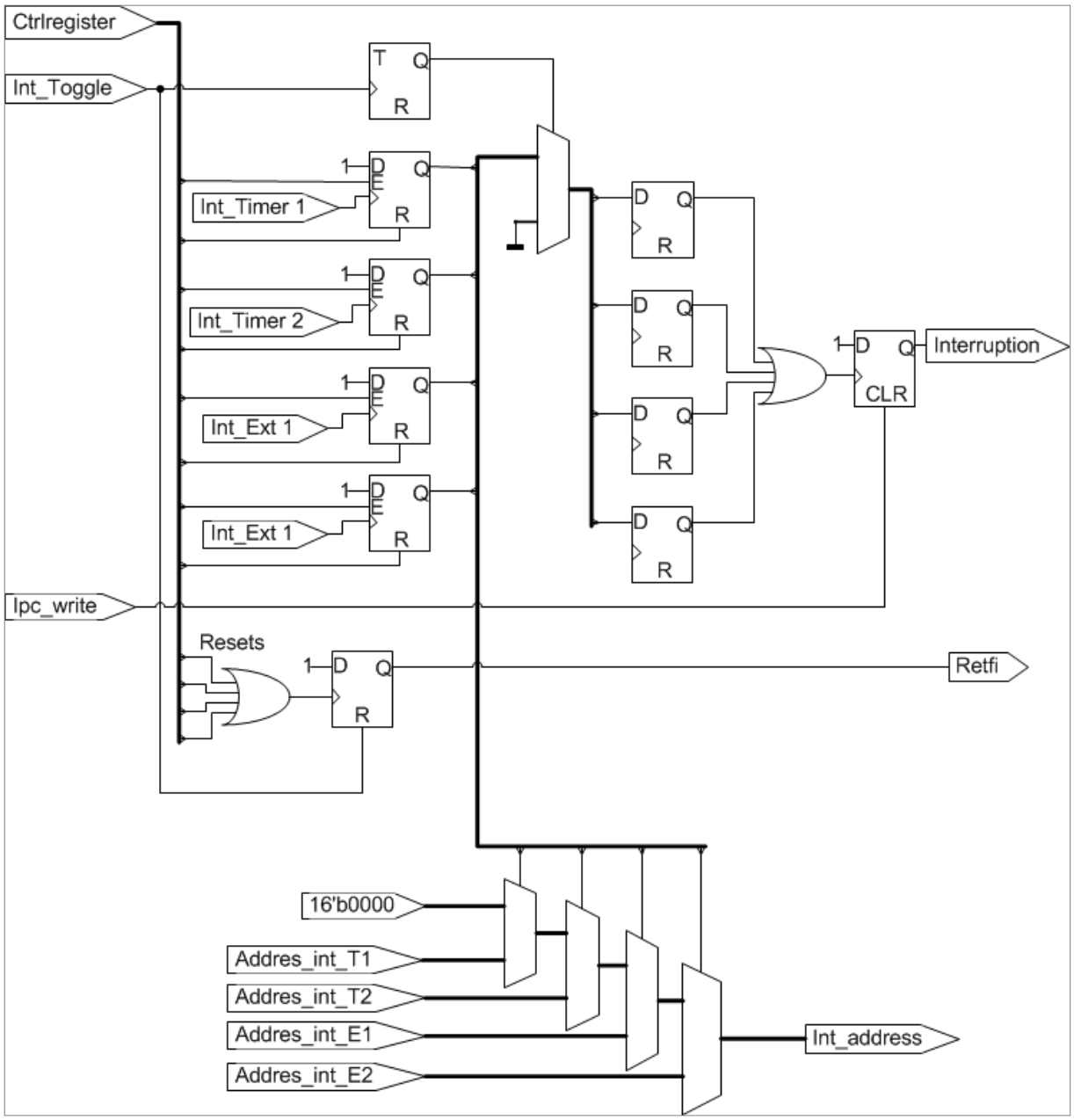
Deshabilitar temporalmente las interrupciones.

Guardar en memoria los registros que se modifican dentro de la interrupción.
Ejecutar la subrutina de interrupción.
Recuperar los valores de los registros que se guardaron en memoria.
Habilitar las interrupciones.
Hacer reset a la bandera de la interrupción que se generó.
Cargar el contador de programa con el valor guardado en el registro ipc.

Manejo de la pila:

```
addi sp, sp, -3       ;Dejar el espacio en la pila para almacenar registros
sw rg0, sp, 2         ;Guardar rg0 de último en la pila
sw rg1, sp, 1         ;Guardar rg1 de segundo en la pila
sw rg2, sp, 0         ;Guardar rg2 de primero en la pila
...
;Subrutina de servicio de interrupción
;que utiliza los registros rg0,rg1 y rg2
...
lw rg2, sp, 0         ;Restaurar rg2 desde la pila
lw rg1, sp, 1         ;Restaurar rg1 desde la pila
lw rg0, sp, 2         ;Restaurar rg0 desde la pila
addi sp, sp, 3        ;Retornar el puntero de la pila
jrlw ra               ;Retornar a la rutina que llamo este procedimiento
```

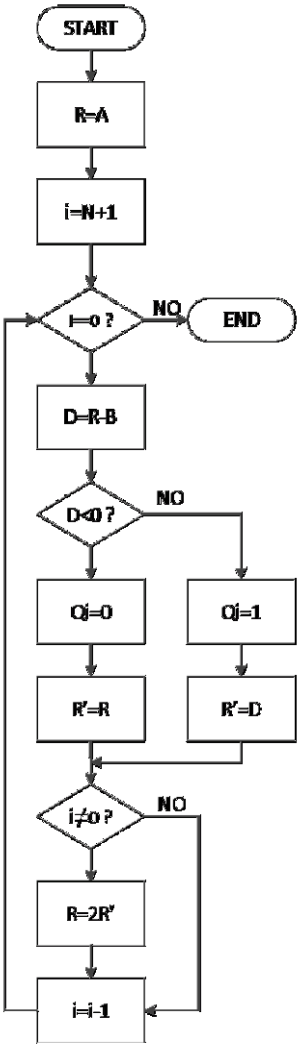
Figura 4. Controlador de Interrupciones RISCKER



1.2.4. Divisor

El procesador RISCKER cuenta con un módulo de división por hardware que consiste en la implementación del algoritmo para dividir números enteros sin signo que se muestra en el siguiente diagrama de flujo. Dónde N es número de bits, A es dividendo, B el divisor, Q el cociente y R el residuo.

Figura 5. Algoritmo de división

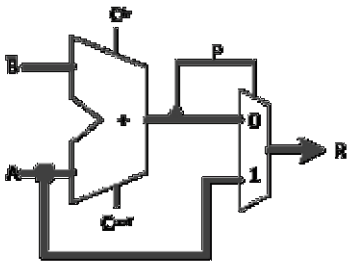


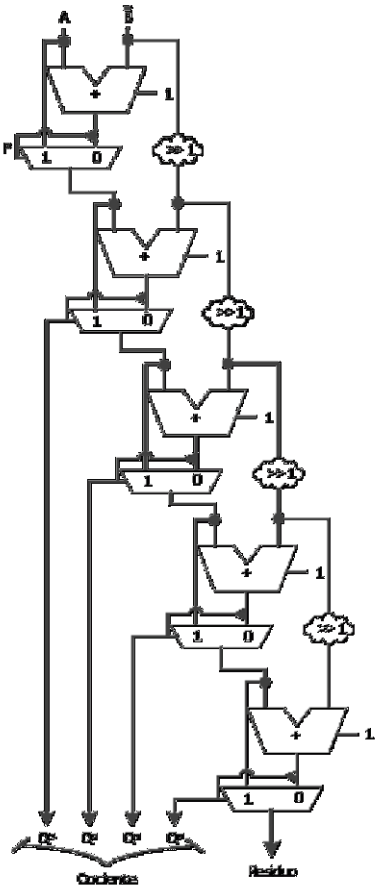
Para llevar a cabo la división se inicializa el residuo con el valor del dividendo. Este residuo parcial se le resta al divisor repetidamente para determinar cuántas veces cabe. Si la diferencia entre residuo y divisor es negativa, se asigna cero al cociente y se descarta el resultado de la resta. Si por el contrario la diferencia es positiva, se asigna uno al cociente y el residuo recibe la diferencia. En cualquier caso el residuo se multiplica por dos y se repite el proceso N+1 veces.

Para implementar en Hardware el algoritmo descrito, se entiende cada iteración como una etapa del circuito, cada etapa lleva a cabo todas las operaciones correspondientes a una repetición del algoritmo, por lo tanto el circuito tendrá N+1 etapas. La Figura 5.a muestra la implementación en hardware de una sola etapa, dónde A y B son números enteros, P el bit de signo del resultado de la resta, Cin el carry de entrada y Cout el de salida y R el residuo parcial de cada etapa.

Figura 6. Diagrama esquemático de un divisor paralelo de 4 bits.

(a) Celda del Divisor paralelo (b) Sistema completo





1.2.5. Multiplicador

Igualmente se implementa un circuito de multiplicación en hardware de 16 bits. Un ejemplo con operandos de 4 bits se muestra en la figura 6. El algoritmo realiza, bit a bit, la multiplicación con la función lógica AND y suma los resultados con un corrimiento.

Figura 7. Diagrama esquemático de un multiplicador paralelo de 4 bits