ANEXO C

HOJA DE DATOS CISCKER

1. SET DE INSTRUCCIONES CISCKER

1.1. SINTAXIS DE LA INSTRUCCIÓN CISKCER

Una Instrucción CISCKER - ASM está compuesta de hasta cinco campos:

[Etiqueta]: [Operación] [Operando], [MD] ;[Comentarios]

[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)*

[Operando]: Indica el operando con el que se va a ejecutar la operación
Se precede de "**#**" si el operando es un número.
Se precede de "$" si el operando es una dirección.

[MD] Modo de Direccionamiento: Se precede de ",". Indica el modo de direccionamiento de la instrucción de forma explícita: IX o IX+. Los demás modos de direccionamientos se infieren de la operación y los operandos.

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

Ejemplos:

```
ADDA    #10,    IX      ;   Sumar   ACCA    con     Memoria[IX+10]
INCA            ; Incrementar ACCA
```

La Tabla 1 es el mapa de *opcodes* que muestra todas las instrucciones implementadas en el microprocesador CISCKER, agrupadas en colores por la trama de micro-instrucciones en que se decodifican. Esto permite diferenciar modos de direccionamiento y tipos de instrucciones: Saltos condicionales, manipulación de bits, control de flujo de programa y aquellas con operandos de 8 o 16bits.

La Tabla 2 divide el mapa por la operación que realiza la ALU con la correlación de colores correspndiente a cada operación, incluyendo aquellas que no utilizan la ALU para su ejecucución como el control de bits del CCR o del flujo de programa.

Las instrucciones también pueden ser divididas como muestra la Tabla 3, por los bits que modifican el registro CCR, esta información facilita la descripción de instrucciones de salto condicional dado por las condiciones lógicas presentadas en la Tabla 4.

Tabla 1. Mapa de OpCodes divido por u-instrucciones

	Bit	Branch	Read Modify Write					Register Memory								
	INH	REL	INH	INH	INH	IX	EXT	IX+	IMM	DIR	IX	EXT	IMM	DIR	IX	EXT
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	BHCC	TSX	NEGA	NEGB	NEG	NEG	SUBA	SUBA	SUBA	SUBA	SUBA	SUBB	SUBB	SUBB	SUBB
1	STOP	BHCS	TXS	CLRA	CLRB	CLR	CLR	SBCA	SBCA	SBCA	SBCA	SBCA	SBCB	SBCB	SBCB	SBCB
2	DABNZ	BHI	INS	INCA	INCB	INC	INC	ADDA	ADDA	ADDA	ADDA	ADDA	ADDB	ADDB	ADDB	ADDB
3	DBBNZ	BLS	DES	DECA	DECB	DEC	DEC	ANDA	ANDA	ANDA	ANDA	ANDA	ANDB	ANDB	ANDB	ANDB
4	BRN	BCC	INX	LSRA	LSRB	LSR	LSR	LDA	LDA	LDA	LDA	LDA	LDB	LDB	LDB	LDB
5	BRA	BCS	DEX	RORA	RORB	ROR	ROR	ORAA	ORA	ORA	ORA	ORA	ORB	ORB	ORB	ORB
6	BIL	BNE	PULX	ASRA	ASRB	ASR	ASR	EORA	EORA	EORA	EORA	EORA	EORB	EORB	EORB	EORB
7	BIH	BEQ	PSHX	LSLA	LSLB	LSL	LSL	ADCA	ADCA	ADCA	ADCA	ADCA	ADCB	ADCB	ADCB	ADCB
8	IDIV	BVC	PULA	ROLA	ROLB	ROL	ROL	CMPA	CMPA	CMPA	CMPA	CMPA	CMPB	CMPB	CMPB	CMPB
9	MUL	BVS	PULB	TSTA	TSTB	TST	TST	BITA	BITA	BITA	BITA	BITA	BSR		JSR	JSR
A	CLV	BPL	PSHA	TDX		BRSET	BRSET	STAA		STA	STA	STA		STB	STB	STB
B	SEV	BMI	PSHB			BRCLR	BRCLR		AIS	STS	STS	STS	AIX	STX	STX	STX
C	CLC	BGE	RTS	TBA	TAB	BCLR	BCLR			STD	STD	STD	CPX	CPX	CPX	CPX
D	SEC	BLT	RTI			BSET	BSET	CPD	LDS	LDS	LDS	LDS	LDX	LDX	LDX	LDX
E	CLI	BGT	WAIT	SBA	CBA	MOV	MOV	SUBD	SUBD	SUBD	SUBD	SUBD	CPD	CPD	CPD	CPD
F	SEI	BLE	SWI	ABA		JMP	JMP	ADDD	ADDD	ADDD	ADDD	ADDD	LDD	LDD	LDD	LDD
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	INH	REL	INH	INH	INH	IX	EXT	IX+	IMM	DIR	IX	EXT	IMM	DIR	IX	EXT

Tabla 2. Mapa de OpCodes divido por operaciones

	Bit	Branch	Read Modify Write					Register Memory								
	INH	REL	INH	INH	INH	IX	EXT	IX+	IMM	DIR	IX	EXT	IMM	DIR	IX	EXT
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	BHCC	TSX	NEGA	NEGB	NEG	NEG	SUBA	SUBA	SUBA	SUBA	SUBA	SUBB	SUBB	SUBB	SUBB
1	STOP	BHCS	TXS	CLRA	CLRB	CLR	CLR	SBCA	SBCA	SBCA	SBCA	SBCA	SBCB	SBCB	SBCB	SBCB
2	DABNZ	BHI	INS	INCA	INCB	INC	INC	ADDA	ADDA	ADDA	ADDA	ADDA	ADDB	ADDB	ADDB	ADDB
3	DBBNZ	BLS	DES	DECA	DECB	DEC	DEC	ANDA	ANDA	ANDA	ANDA	ANDA	ANDB	ANDB	ANDB	ANDB
4	BRN	BCC	INX	LSRA	LSRB	LSR	LSR	LDAA	LDAA	LDAA	LDAA	LDAA	LDAB	LDAB	LDAB	LDAB
5	BRA	BCS	DEX	RORA	RORB	ROR	ROR	ORAA	ORAA	ORAA	ORAA	ORAA	ORAB	ORAB	ORAB	ORAB
6	BIL	BNE	PULX	ASRA	ASRB	ASR	ASR	EORA	EORA	EORA	EORA	EORA	EORB	EORB	EORB	EORB
7	BIH	BEQ	PSHX	LSLA	LSLB	LSL	LSL	ADCA	ADCA	ADCA	ADCA	ADCA	ADCB	ADCB	ADCB	ADCB
8	IDIV	BVC	PULA	ROLA	ROLB	ROL	ROL	CMPA	CMPA	CMPA	CMPA	CMPA	CMPB	CMPB	CMPB	CMPB
9	MUL	BVS	PULB	TSTA	TSTB	TST	TST	BITA	BITA	BITA	BITA	BITA	BSR	JSR	JSR	JSR
A	CLV	BPL	PSHA	TDX		BRSET	BRSET	STAA		STAA	STAA	STAA		STAB	STAB	STAB
B	SEV	BMI	PSHB			BRCLR	BRCLR		AIS	STS	STS	STS	AIX	STX	STX	STX
C	CLC	BGE	RTS	TBA	TAB	BCLR	BCLR			STD	STD	STD	CPX	CPX	CPX	CPX
D	SEC	BLT	RTI			BSET	BSET	SUBD	LDS	LDS	LDS	LDS	LDX	LDX	LDX	LDX
E	CLI	BGT	WAIT	SBA	CBA	MOV	MOV	CPD	SUBD	SUBD	SUBD	SUBD	CPD	CPD	CPD	CPD
F	SEI	BLE	SWI	ABA		JMP	JMP	ADDD	ADDD	ADDD	ADDD	ADDD	LDD	LDD	LDD	LDD
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	INH	REL	INH	INH	INH	IX	EXT	IX+	IMM	DIR	IX	EXT	IMM	DIR	IX	EXT

	Sub		NAND		Sub with carry		Pass	
	Add		NOR		Shift		Control	
	AND		Increment		Neg			
	OR		Decrement		cCear			
	XOR		Add with carry		Don't Care			

Tabla 3. Mapa de OpCodes divido por señales modificadas del CCR

	Bit	Branch	Read Modify Write					Register Memory								
	INH	REL	INH	INH	INH	IX	EXT	IX+	IMM	DIR	IX	EXT	IMM	DIR	IX	EXT
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	BHCC	TSX	NEGA	NEGB	NEG	NEG	SUBA	SUBA	SUBA	SUBA	SUBA	SUBB	SUBB	SUBB	SUBB
1	STOP	BHCS	TXS	CLRA	CLRB	CLR	CLR	SBCA	SBCA	SBCA	SBCA	SBCA	SBCB	SBCB	SBCB	SBCB
2	DABNZ	BHI	INS	INCA	INCB	INC	INC	ADDA	ADDA	ADDA	ADDA	ADDA	ADDB	ADDB	ADDB	ADDB
3	DBBNZ	BLS	DES	DECA	DECB	DEC	DEC	ANDA	ANDA	ANDA	ANDA	ANDA	ANDB	ANDB	ANDB	ANDB
4	BRN	BCC	INX	LSRA	LSRB	LSR	LSR	LDAA	LDAA	LDAA	LDAA	LDAA	LDAB	LDAB	LDAB	LDAB
5	BRA	BCS	DEX	RORA	RORB	ROR	ROR	ORAA	ORAA	ORAA	ORAA	ORAA	ORAB	ORAB	ORAB	ORAB
6	BIL	BNE	PULX	ASRA	ASRB	ASR	ASR	EORA	EORA	EORA	EORA	EORA	EORB	EORB	EORB	EORB
7	BIH	BEQ	PSHX	LSLA	LSLB	LSL	LSL	ADCA	ADCA	ADCA	ADCA	ADCA	ADCB	ADCB	ADCB	ADCB
8	IDIV	BVC	PULA	ROLA	ROLB	ROL	ROL	CMPA	CMPA	CMPA	CMPA	CMPA	CMPB	CMPB	CMPB	CMPB
9	MUL	BVS	PULB	TSTA	TSTB	TST	TST	BITA	BITA	BITA	BITA	BITA	BSR	JSR	JSR	JSR
A	CLV	BPL	PSHA	TDX		BRSET	BRSET	STAA		STAA	STAA	STAA		STAB	STAB	STAB
B	SEV	BMI	PSHB			BRCLR	BRCLR		AIS	STS	STS	STS	AIX	STX	STX	STX
C	CLC	BGE	RTS	TBA	TAB	BCLR	BCLR			STD	STD	STD	CPX	CPX	CPX	CPX
D	SEC	BLT	RTI			BSET	BSET	SUBD	LDS	LDS	LDS	LDS	LDX	LDX	LDX	LDX
E	CLI	BGT	WAIT	SBA	CBA	MOV	MOV	CPD	SUBD	SUBD	SUBD	SUBD	CPD	CPD	CPD	CPD
F	SEI	BLE	SWI	ABA		JMP	JMP	ADDD	ADDD	ADDD	ADDD	ADDD	LDD	LDD	LDD	LDD
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	INH	REL	INH	INH	INH	IX	EXT	IX+	IMM	DIR	IX	EXT	IMM	DIR	IX	EXT

8bits Logic	V,N,Z	set or clr ccr bits	CCR
16bits Arithmetic	V,H,N,Z,C	8 bits Aritmetic	V,N,Z,C
16bits logic	V,N,Z	8 bits INC/DEC	V,N,Z
8bits shift	V,N,Z,C	No actualizan CCR	
don´t care		Branch Bit test	Z,C

Tabla 4. Condiciones de las Instrucciones de salto condicional.

BRN	1'b0	BLS	C|Z==1	BPL	N==0	DABNZ	Z==0
BRA	1'b1	BCC	C==0	BMI	N==1	DBBNZ	Z==0
BIL	I==0	BCS	C==1	BGE	N^V==0	BRSET	Z==0
BIH	I==1	BNE	Z==0	BLT	N^V==1	BRCLR	Z==0
BHCC	H==0	BEQ	Z==1	BGT	Z|N^V==1	BRSET	Z==0
BHCC	H==1	BVC	V==0	BLE	Z|N^V==1	BRCLR	Z==0
BHI	C|Z==0	BVS	V==1				

1.2. RUTA DE DATOS

La Figura 1 muestra la ruta de datos CISCKER, en esta se muestran las unidades lógico-aritmética, de memoria (Von-Neumann), control y de entradas y salidas. Los números corresponden a las señales decritas en la Tabla 5.

Figura 1. Diagrama esquematico de la ruta de datos de la *arqutiectura* CISCKER

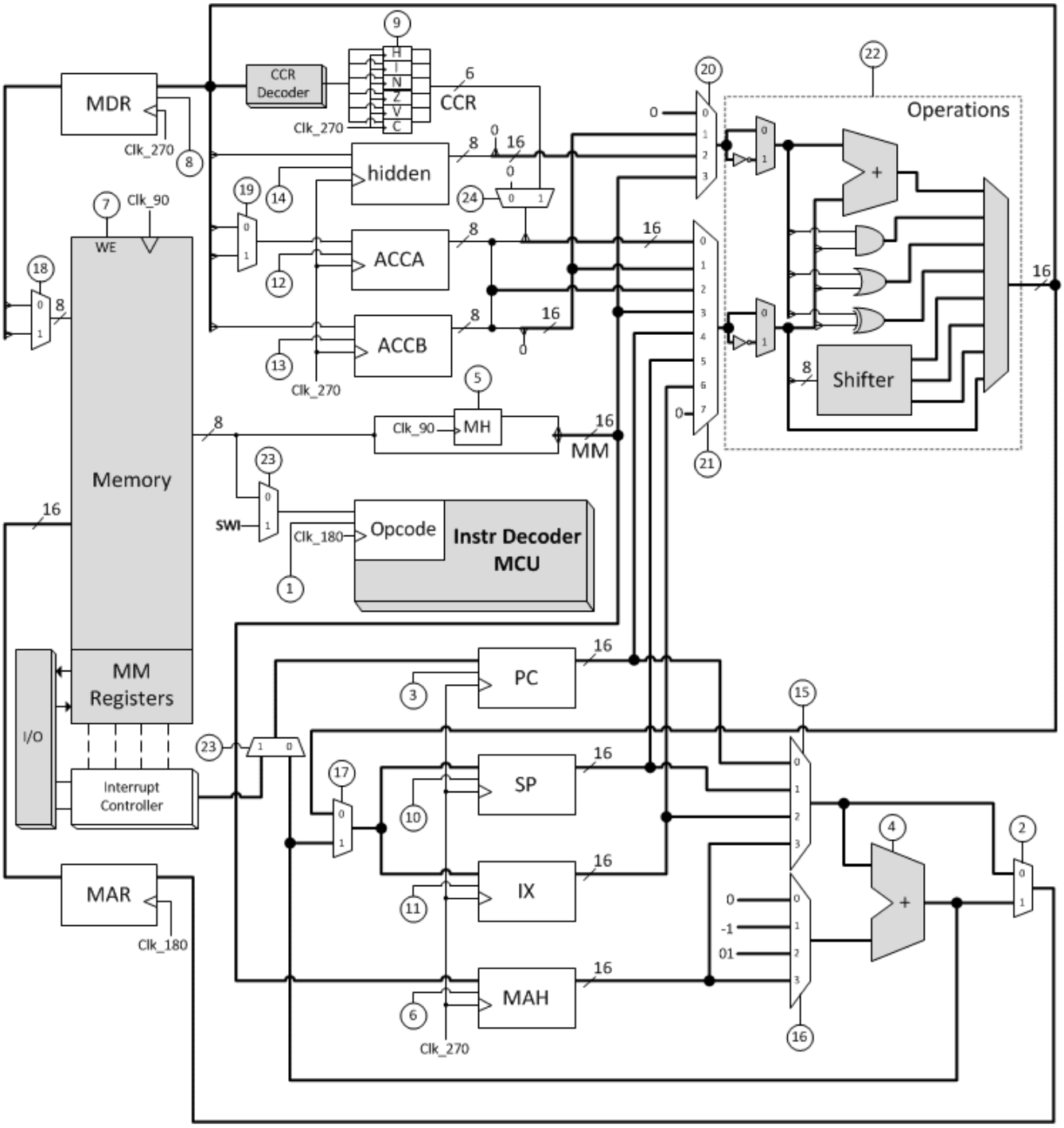


Tabla 5. Señales de control en la ruta de datos CISCKER

Número	Nombre	Descripción
1	opcode_en	Habilitador del registro de opcode
2	pass_adder	Selecciona la fuente del registro MAR
3	pc_en	Habilitador del registro PC
4	adder_cin	Acarreo de entrada al sumador de direcciones
5	h_reset	Reset del registro MH
6	mah_en	Habilitador del registro de memoria para direcciones
7	we	Habilitador de escritura de la memoria
8	mdr_en	Habilitador del registro MDR
9	ccr_en	Habilitador del registro CCR
10	sp_en	Habilitador del registro SP
11	ix_en	Habilitador del registro IX
12	acca_en	Habilitador del ACCA
13	accb_en	Habilitador del ACCB
14	hidden_en	Habilitador del registro HIDDEN
15	adder_src_a[1:0]	Selecciona el operando A del sumador de direcciones
16	adder_src_b[1:0]	Selecciona el operando B del sumador de direcciones
17	adder_alu	Selecciona la fuente de los registros IX,SP
18	wd_src[1:0]	Selecciona el dato que se va a escribir en memoria
19	acca_src	Selecciona la fuente del siguiente valor de ACCA
20	alu_src_a[2:0]	Selecciona el operando A de la ALU
21	alu_src_b[1:0]	Selecciona el operando B de la ALU
22	alu_op[5:0]	Determina la operación de la ALU
23	Interruption	Señal de interrupción
24	wd_ccr	Selecciona CCR como operando de la ALU para pasar a MDR

1.3. UNIDAD DE CONTROL

Las señales de control CISCKER son generadas por la MCU que decodifica los *Opcodes,* leídos de memoria, en las micro-instrucciones descritas en la tabla 6. Cada micro-instrucción es a su vez decodificada generando las señales que se muestran en la tabla 7 para generar las distintas etapas de ejecución de una instrucción..

Tabla 6. Micro-Instrucciones CISCKER

	Nombre	Código	Descripción
p	program fetch	00	Lectura del siguiente byte de memoria.
r	read byte	01	Lectura de un byte de memoria.
d	Dummy	10	Direccionamiento del siguiente byte de lectura.
w	Write	11	Escritura de byte en memoria.

Tabla 7. Decodificación de las micro-instrucciones CISCKER

	µ-Instrucción	pass_adder	pc_en	adder_cin	h_reset	mah_en	we	mdr_en
00	**p**	1	1	1	*	1	0	1
01	**r**	1	*	1	0	0	0	1
10	**d**	*	0	0	1	0	0	1
11	**w**	1	*	1	0	0	1	0

1.3.1. Bits de control en las u-instrucciones:

Adicional al código de cada micro-instrucción existen cuatro bits de control que modifican funciones especiales que se deben realizar durante su ejecución:

lookahead: en esta u-instrucción se registra el opcode (depende del modo de direccionamiento)

final: determina cual es la última u-instrucción.

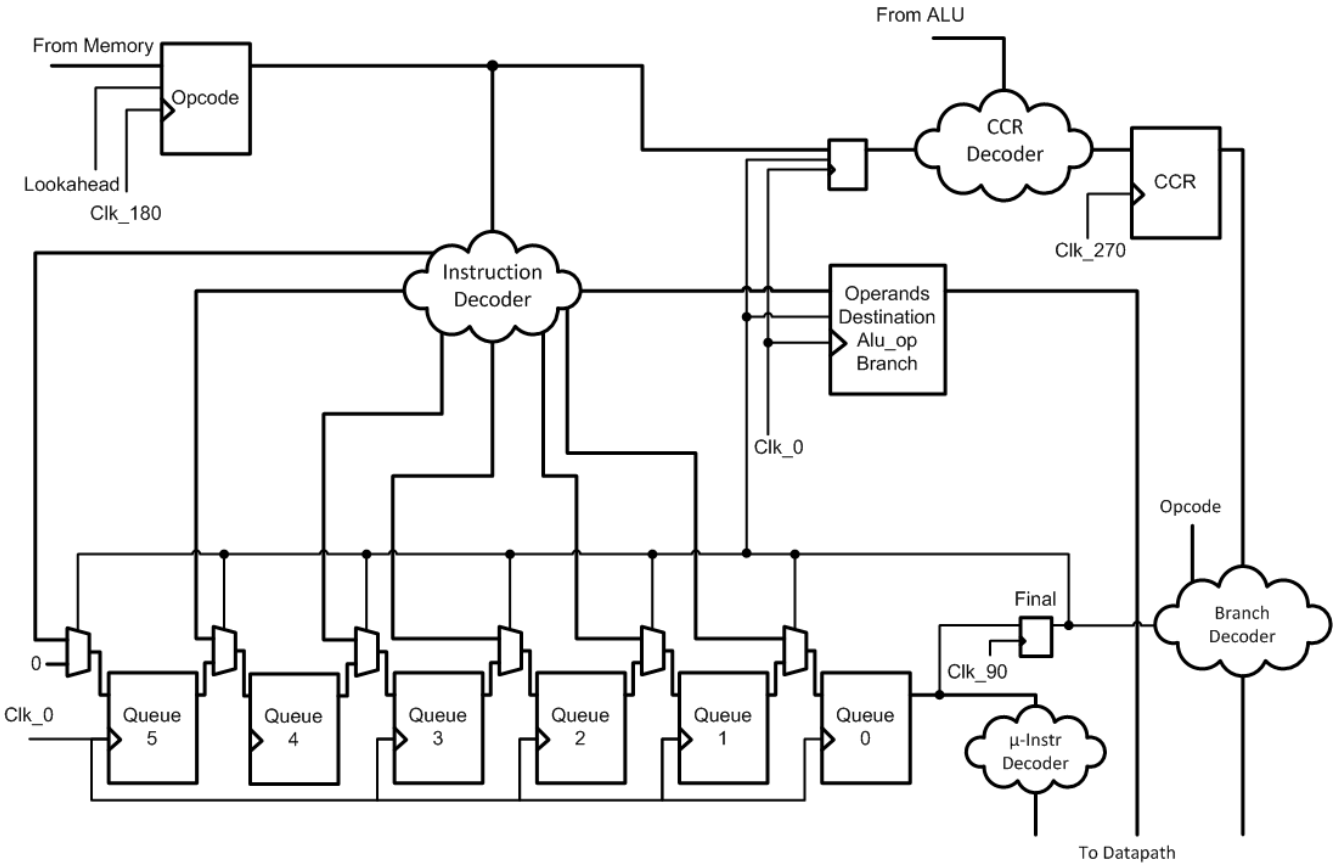
operation: en esta u-instrucción se registra el resultado de la operación y el CCR.

complement: Bit de complemento a la u-instrucción. Determina dos funciones diferentes de una misma u-instrucción. Este bit configura los campos que contienen un asterisco en la tabla 7, permitiendo una mayor versatilidad de las tramas decodificadas.

La Figura 2 es una representación de la MCU CISCKER, en la cual se encuentran los decodificadores de instrucción, CCR, salto condicional y de micro-instrucción. El decodificador de instrucción tiene además el decodificador de la operación de la ALU, el cual retorna los operandos, operación y destino de asignación.

El registro de corrimeinto de seis iteraciones, llamado *Queue*, almacena y permite la decodificación secuencial de las micro-instrucciones. La señal *Final* indica que se debe cargar una nueva trama de micro-instrucciones cuando termine la ejecución actual teniendo en cuenta que el *Opcode* de la nueva instrucción ya habría sido cargado en el registro gracias a la señal *Lookahead*. En el caso del decodificador del CCR es necesario que el nuevo *Opcode* esté durante la ejecucución de la instrucción, no antes o después como ocurre con la señal *Lookahead (a lo cual debe su nombre)* es por esto que se añade un registro que es habilitado por la señal *Final*.

Figura 2. MCU CISCKER

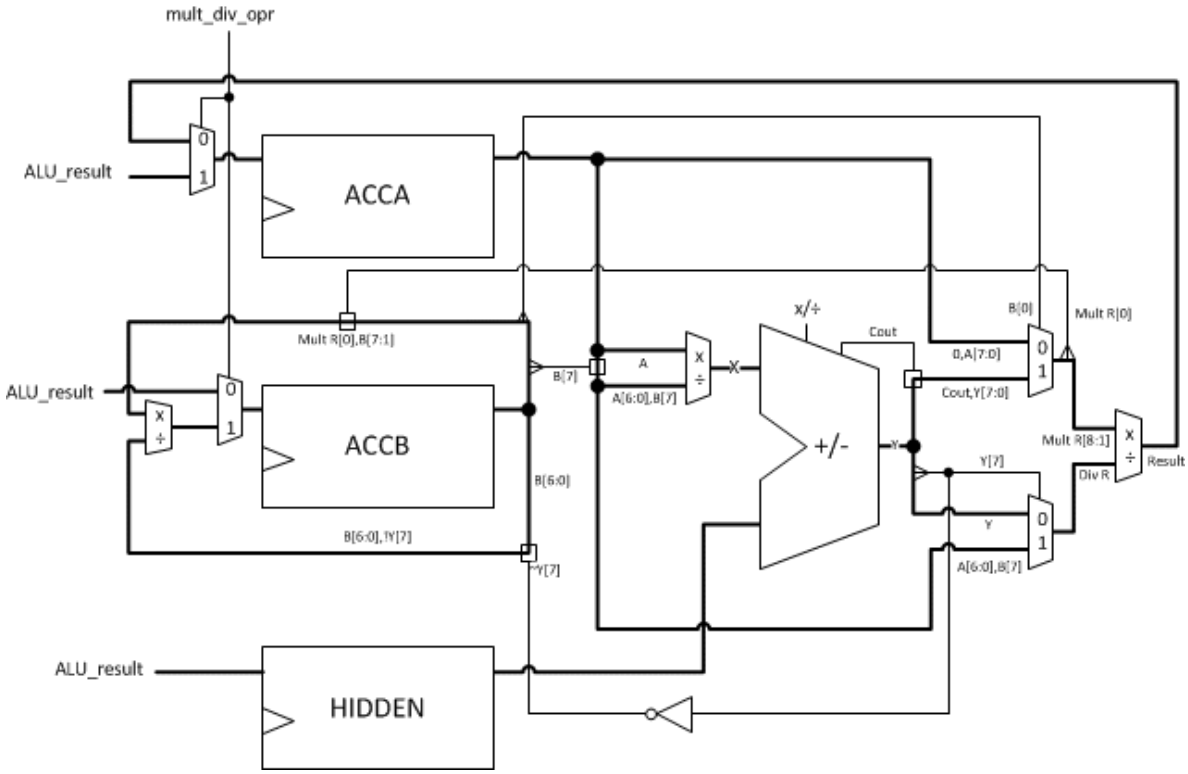


La función de instrucciones iterativas de la MCU CISCKER hace uso de los mismos registros *Queue,* y toma un conteo de la señal *Final* como referencia para conocer cuando y cual iteración ha sido completada. Esto le permite al

decodificador cambiar sus salidas sin necesidad de que se haya leído un nuevo *Opcode*. De esta forma es posible ejecutar instrucciones como multiplicación y división, o las de retorno y entrada a la rutina de interrupción.

El circuito que permite la ejecución de instrucciones de multiplicación y división por iteraciones está representado en el diagrama esquemático de la figura 3. La primera trama de micro-instrucciones se transladar el cociente o el multiplicador, dependiendo de la instrucción, desde el acumulador B al registro *Hidden*, para así poder iniciar la ejecución de las iteraciones que darán como resultado un valor de 16 bits en multiplicación y el cociente y el dividendo de 8 bits en el acumulador D.

Figura 3. Diagrama esquemático del modulo de multiplicación y división



1.4. UNIDAD DE MEMORIA

En la figura 4 está ilustrada la separación virtual que se hace de la unidad de memoria CISCKER. En esta se indican también los registros mapeados en memoria.

Las direcciones 7FF8 a 7FFF se encuentran reservadas para los vectores de interrupción. Estos valores deben ser modificados en el caso de habilitar una interrupción para que señale la ubicación de la RSI correspondiente.

7FF6 y 7FF7 vector de la interrupción SWI

7FF8 y 7FF9 vector de la interrupción externa 1

7FFA y 7FFB vector de la interrupción externa 2

7FFC y 7FFD vector de la interrupción por temporizador 1

7FFE y 7FFF vector de la interrupción por temporizador 2

Tabla 4. Distribución de memoria

RAM 0-00FF
Program 0100-X
Data X-X
Stack X-7FF7
Reserved 7FF8-7FFF
MM Registers 8000 Interruption Control 8001 Timer Control 8002 Timer1 Period 8003 Timer1 Period 8004 Timer2 Period 8005 Timer2 Period 8006 Output Port 1 8007 Output Port 2 8008 Input Port 1 8009 Input Port 2 800A Timer1 800B Timer1 800C Timer 2 800D Timer 2
Invalid 800E-FFFF

1.4.1. Registros Mapeados en Memoria (Registros MM)

Los registros MM de la arquitectura CISCKER se describen en la Tabla 6. La función de los bits de los registros de control de interrupciones y temporizadores se encuentra en la Tabla 7. Las direcciones de estos registros comienzan en el valor hexadecimal 8000 donde el bit más significativo los selecciona para escritura y lectura. Es posible agregar nuevos registros MM por encima de la dirección 800D, pero la lectura y escritura de estas direcciones vacías son inválidas.

1.5. CONTROLADOR DE INTERRUPCIONES

Consta de dos partes: La unidad de reconocimiento de interrupciones y el algoritmo que se encarga de almacenar los registros de estado del procesador y saltar a la dirección de la RSI. La figura 5 muestra el circuito de reconocimiento de interrupciones, que entrega a la MCU la señal Interruption y a la ruta de dato, la dirección de donde leer la RSI. En la ruta de datos se carga síncronamente SWI en *Opcode* con la señal Interruption.

SWI pone la máscara de Interrupción en 1, así se deshabilitan las interrupciones y se borra la señal *Interruption* durante la RSI.

Tabla 6. Registros Mapeados en Memoria

Dirección (HEX)	Registro
8000	Control de Interrupciones
8001	Control de Temporizadores
8002	Periodo del Temporizador 1 [15:8]
8003	Periodo del Temporizador 1 [7:0]
8004	Periodo del Temporizador 2 [15:8]
8005	Periodo del Temporizador 2 [7:0]
8006	Puerto de Salida 1
8007	Puerto de Salida 2
8008	Puerto de Entrada 1
8009	Puerto de Entrada 1
800A	Temporizador 1 [15:8]
800B	Temporizador 1 [7:0]
800C	Temporizador 2 [15:8]
800D	Temporizador 2 [7:0]

Tabla 7. Registros MM de Control de temporizadores e interrupciones

Control de Interrupciones		**Control de Temporizadores**	
Bit	**Función**	**Bit**	**Función**
7	Timer1 Prescaler H	**7**	Enable External2 IRQ
6	Timer1 Prescaler L	**6**	Enable External1 IRQ
5	Timer2 Prescaler H	**5**	Enable Timer2 IRQ
4	Timer2 Prescaler L	**4**	Enable Timer1 IRQ
3	Timer1 Enable	**3**	Reset External2 IRQ
2	Timer2 Enable	**2**	Reset External1 IRQ
1	Timer1 Reset	**1**	Reset Timer2 IRQ
0	Timer2 Reset	**0**	Reset Timer1 IRQ

Instrucción SWI: En modo de instrucción de ejecución cíclica, se ejecutan 3 micro-instrucciones 3 veces seguidas por una última trama de 4 micro-instrucciones de objetivo distinto. Estas 3 primeras tramas almacenan en el *Stack los* registros de 16 bits (PC),(X) y (CCR,A). El contador de señales *final* escoge el dato a guardar en la pila correspondientes a las Señales (PC), (X) y (CCR,A) en la ALU con la operación "PASS". La señal (CCR,A) se crea reemplazando la parte alta de "0A" por CCR durante toda la ejecución de la instruccion SWI como se ve en la figura de ruta de datos. Una vez cumplidos los 3 ciclos, la nueva decodificación de SWI dará como resultado 4 micro-instrucciones cuyo objetivo es leer de la dirección donde se encuentra el vector de la interrupción correspondiente y saltar a él. Estas direcciones se encuentran físicamente dentro del procesador, solo podrían ser cambiadas por hardware y son unidas a la ruta de datos mediante un multiplexor a la entrada de PC seleccionado mientras (Opcode==SWI).

Tres micro-instrucciones para almacenar registros:
d: Direcciona SP, y Carga MDR con el valor a guardar (PC, X, CCR, A)
w: Escribe en la memoria el valor L
w: Escribe en la memoria el Valor H y carga PC y MAR con el valor de dirección de la interrupción, esto se hace una vez por cada ciclo de u-instrucciones. (Loop)

Cuatro micro-instrucciones de salto al vector de interrupción:
p: Lee la dirección de interrupción que está en MAR, obteniendo la parte alta del vector de interrupción y la dirección se aumenta para guardarla en PC (El multiplexor de entrada a PC vuelve a su estado normal).
p: En el nuevo valor de dirección en MAR se lee la parte baja del vector de interrupción, completando su valor y guardándolo en MAH.
d: Direcciona MAH para guardar el vector de interrupción en MAR.
r: Lee el Opcode que está en el vector de interrupción, siendo esta la primera instrucción del servicio de interrupción, y deja PC y MAR en el valor siguiente para continuar la ejecución normal del programa.

Figura 5. Controlador de interrupciones CISCKER