Arquitectura de Copro II

Curso 2007 - 2008
Asignatura: Laboratorio de Computadores
3º Ingeniería en Informática
Departamento de Informática e Ingeniería de Sistemas
Centro Politécnico Superior

Arquitectura de COPRO II

1.- Introducción

La figura 1.1 muestra el esquema general del procesador microprogrrarnable tal como está montado en la placa Copro II:

Figura 1. 1 Esquema de un procesador microprogramado.

La unidad de procesamiento se encarga del cálculo, y se comunica mediante un bus de datos y un bus de direcciones con la memoria principal y los periféricos. La unidad de control tiene como tarea controlar todo el procesamiento por medio de señales de control. Indirectamente, controla también la transferencia de datos entre la memoria principal, los periféricos y la unidad de procesamiento. La unidad de procesamiento afecta a la unidad de control por medio de unas señales de test llamadas CODIGOS DE CONDICION.

Este procesador se denomina microprogramado por el hecho de que ejecuta varias microinstrucciones por cada instrucción del programa de aplicación (a esta última la denominaremos macroinstrucción). Las microinstrucciones son generadas por la unidad de control.

Este sistema se divide, como ya se ha explicado, en tres partes:
- La unidad de control que esta construída en base al secuenciador IIDT39C10.
- La unidad de tratamiento, que se basa en la RALU IDT49C402.
- La macromernoria y los periféricos.

La unidad de control se compone de la micromemoria o WCS (Writable Control Store), que controla todas las partes del procesador por medio de los registros de *pipeline*. Ésta a su vez está

controlada bien por el registro de instrucción, bien por los flags de condición CC que son los que permiten ejecutar los saltos condicionales.

Las direcciones de los registros Ra y Rb de la RALU son definidas por los campos respectivos del registro de instrucción o bien por la microinstrucción.

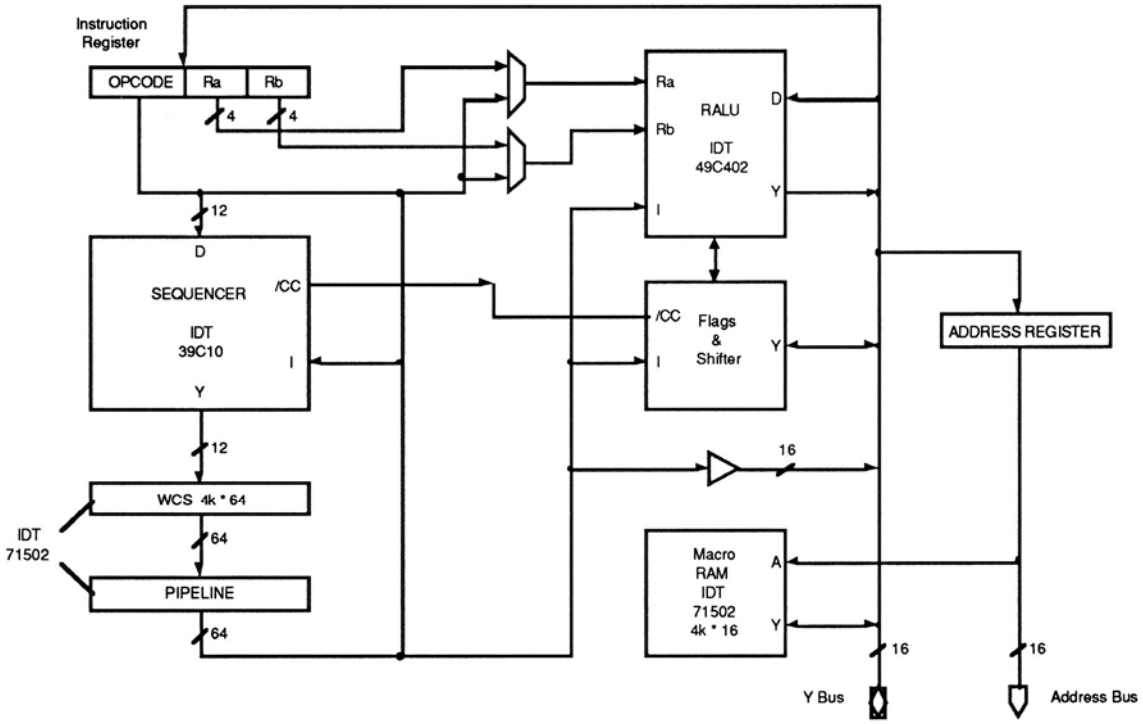


Figura 1.2: Esquema de bloques del procesador

El circuito llamado “FlagShifter” sirve para almacenar los micro y los macroflags, en función de los flags producidos por la RALU, genera los flags de condición y asegura el buen funcionamiento de las operaciones de desplazamiento y de rotación de la RALU.

La macromemoria es accesible mediante el bus de datos bidireccional Y y el bus de direcciones. El bus Y está unido a la entrada directa D y a la salida Y de la RALU, al registro de instrucción y al circuito FlagShifter. El registro de instrucción puede así mismo cargar las macroinstrucciones desde la macromemoria. El FlagShifter puede transferir los flags entre la RALU o la macromemoria. El registro de dirección contiene la dirección actual de la macromemoria y es cargada por la RALU que no es capaz de facilitar directamente la dirección de la macromemoria.

2.- Unidad de control

El papel de la unidad de control es abastecer a los demás elementos del procesador de las señales necesarias para su correcto funcionamiento. La secuencia de las señales de control está asegurada por el registro de *pipeline*, que se carga en cada ciclo de reloj con la próxima microinstrucción proveniente de la memoria.

El principal trabajo del secuenciador es el direccionarniento de la micromemoria. Calcula normalmente la dirección de la proxima microinstruccion a partir de un contador interno denominado

μPC. Para poder efectuar saltos dentro del microcódigo el secuenciador necesita que se le facilite la dirección de salto. Ésta puede provenir bien de la microinstrucción, bien de la PROM de decodificación que produce una dirección de salto en función del código de operación de la macroinstrucción que se encuentre en el registro de instrucción.

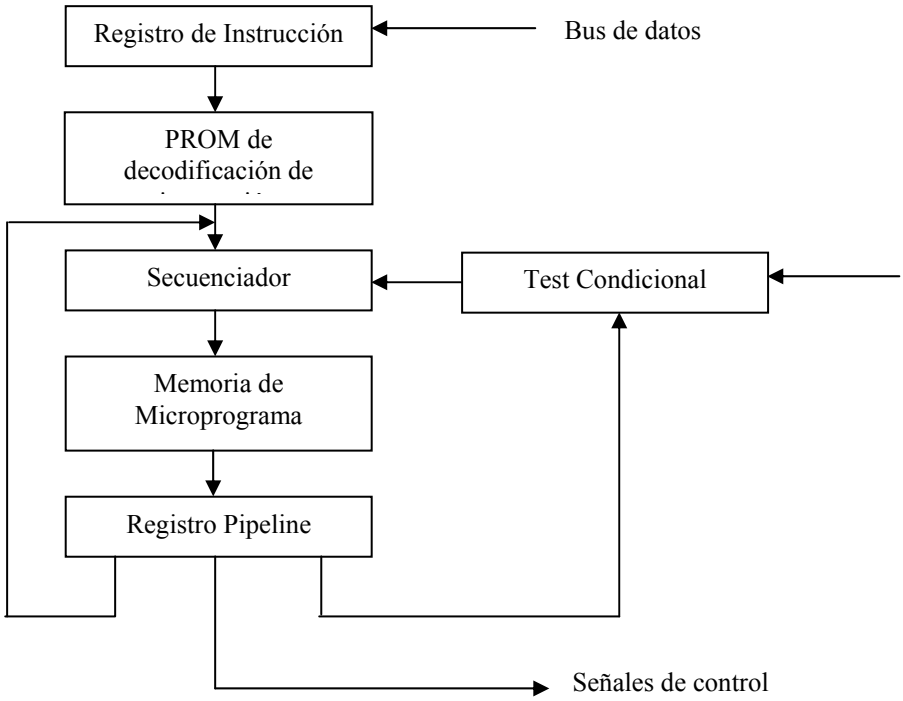


Figura 2.1 unidad de control

2.1. El secuenciador

La figura 2.2 muestra el secuenciador 39C10 de IDT que se incluye en la tarjeta Copro-II. Es un secuenciador que puede direccionar mediante sus 12 bits de direcciones hasta 4K de microinstrucciones. Su juego de instrucciones es compatible con el 2910 de AMD.

El corazón de este secuenciador es un multiplexor de cuatro entradas cuya misión es seleccionar de dónde proviene la siguiente rnicroinstrucción:
- el μPC
- el último dato de pila, para los retornos de subrutina.
- las entradas externas, para los saltos y llamadas a subrutinas.
- el registro/decrementador (R/C) para las instrucciones de bucle.

La figura 2.3 es una lista de las instrucciones del secuenciador. La figura 2.4 nos muestra el empleo de este secuenciador en la tarjeta Copro II y se ve cómo los doce bits de las entradas D, que especifican normalmente la dirección de salto, provienen o bien por completo del pipeline (de la microinstrucción), o bien parcialmente del código de operacion de la instrucción que se halla en el registro de instrucción. El método aplicado es el de asignar el código de operación del registro de

instrucción a los ocho bits de mayor peso de la dirección de salto. Los cuatro bits de menor peso son siempre suministrados por el pipeline, que aporta ceros para las instrucciones de tabla de saltos (Jump Map). La dirección de salto para las tablas de saltos se calcula por la fórmula siguiente:

Dirección de salto = 16 * Código de operación

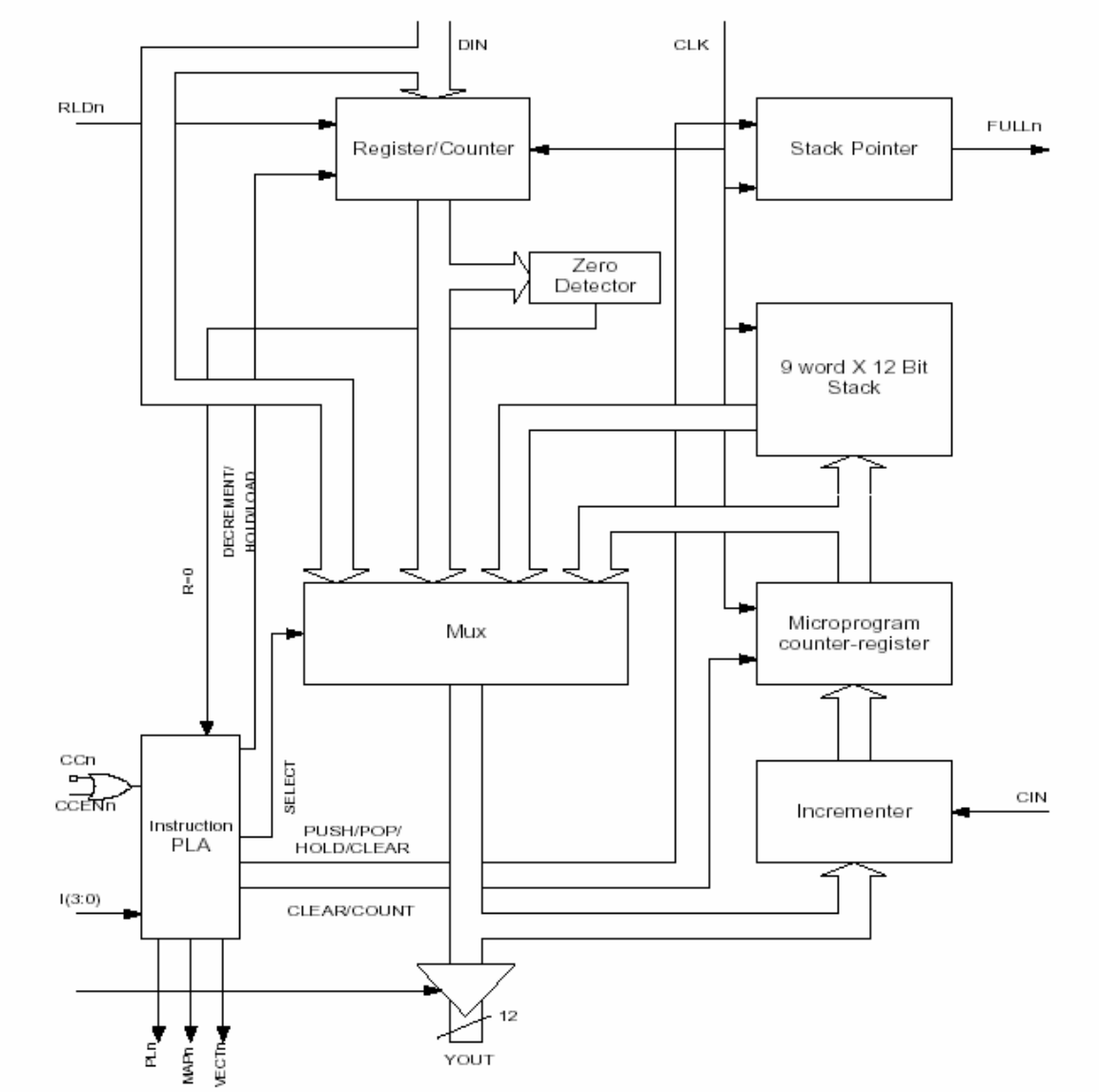


Figura 2.2 Esquema de bloques M secuenciador IDT 39CI 0

Las instrucciones de salto del secuenciador son siempre condicionales y dependen de dos señales /CC y /CCEN. /CCEN es suministrada al secuenciador por el pipeline (bit 62). La puesta a uno de este bit haceque la condición de salto sea siempre cierta, consiguiendose así efectuar saltos incondicionales. La señal /CC que viene del FlagShifter sólo se tiene en cuenta cuando /CCEN está a cero. La condición de salto depende en este caso del test que efectúe el FlagShifter.
La señal /FULL indica que la pila interna del secuenciador está llena. Sólamente se emplea para detectar posibles errores de la tarjeta.
La señal /VECT indica a la instrucción CJV (Conditional Jump Vector) la selección de un tercer origen de dirección. Como no existe tal tercer origen de dirección, la señal /VECT y la instrucción CJV no se emplean.

HEX I3-I0	MNEMONIC	NAME
0	JZ	JUMP ZERO
1	CJS	COND JSB PL
2	JMAP	JUMP MAP
3	CJP	COND JUMP PL
4	PUSH	PUSH/COND LD CNTR
5	JSRP	COND JSB R/PL
6	CJV	COND JUMP VECTOR
7	JRP	COND JUMP R/PL
8	RFCT	REPEAT LOOP, CNTR * 0
9	RPCT	REPEAT PL, CNTR * 0
A	CRTN	COND RTN
B	CJPP	COND JUMP PL & POP
C	LDCT	LD CNTR & CONTINUE
D	LOOP	TEST END LOOP
E	CONT	CONTINUE
F	TWB	THREE-WAY BRANCH

Figura 2.3 Instrucciones del secuenciador.

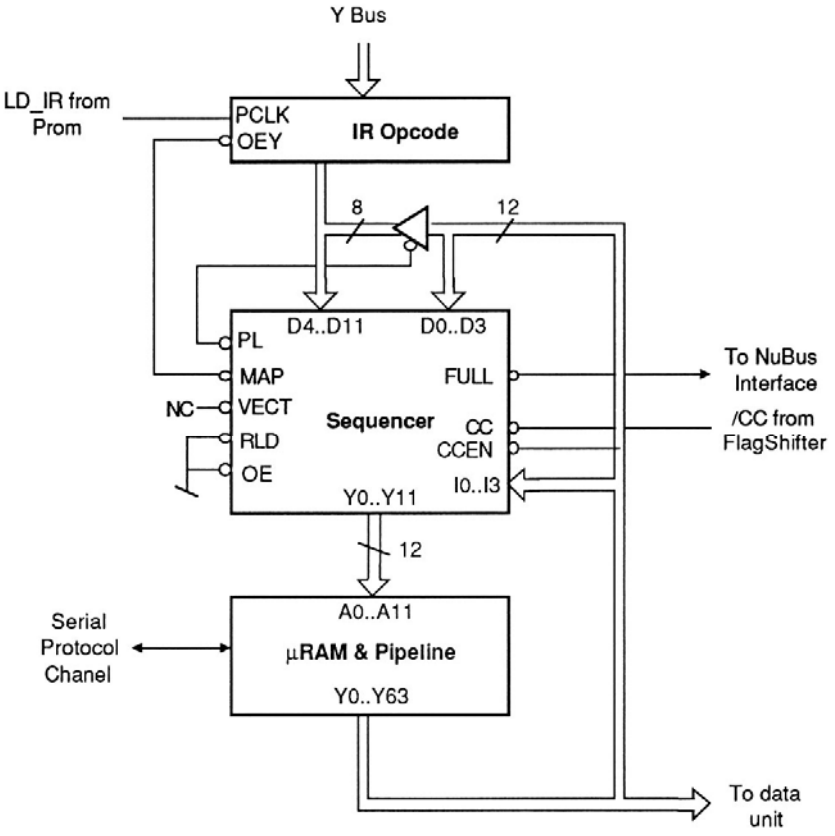


Figura 2.4 Conexión del secuenciador en la tajeta Copro II

La figura 2.5 nos muestra finalmente los campos de la microinstrucción que dirigen la unidad de control o secuenciador. Se aprecia que el campo de la dirección de salto se solapa con el campo empleado para almacenar las constantes a cargar en la RALU. En consecuencia **no se pueden efectuar al mismo tiempo saltos y carga de constantes en la RALU**.

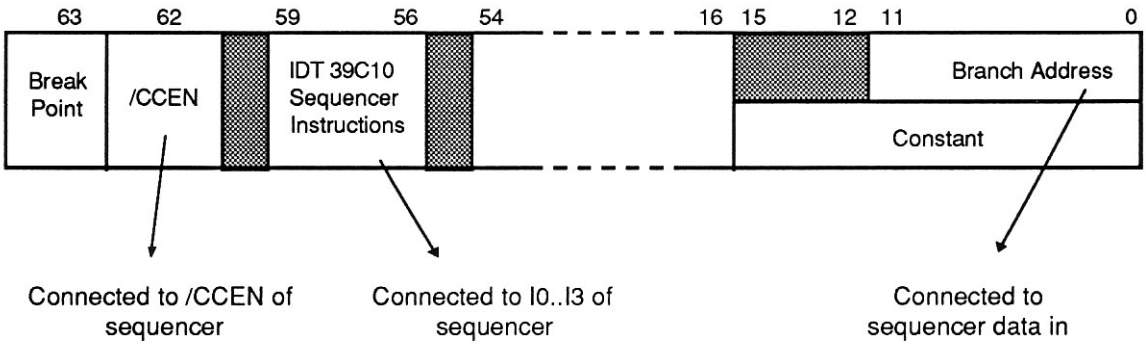


Figura 2.5 Campos de la mícroinstrucción del secuenciador.

2.2 La micromemoria y el pipeline.

La micromernoria que sirve de soporte al microcódigo está implementada con circuitos del tipo 71502 de IDT (U11...U14). Este circuito es una memoria rápida de 4K de palabras de 16 bits y posee además registros de pipeline a su salida. Se consigue de esta manera construir los registros de pipeline y la micromemoria de 4K de palabras de 64 bits con tan solo cuatro circuitos, lo que facilita la concepción del procesador (ver fig. 2.6).

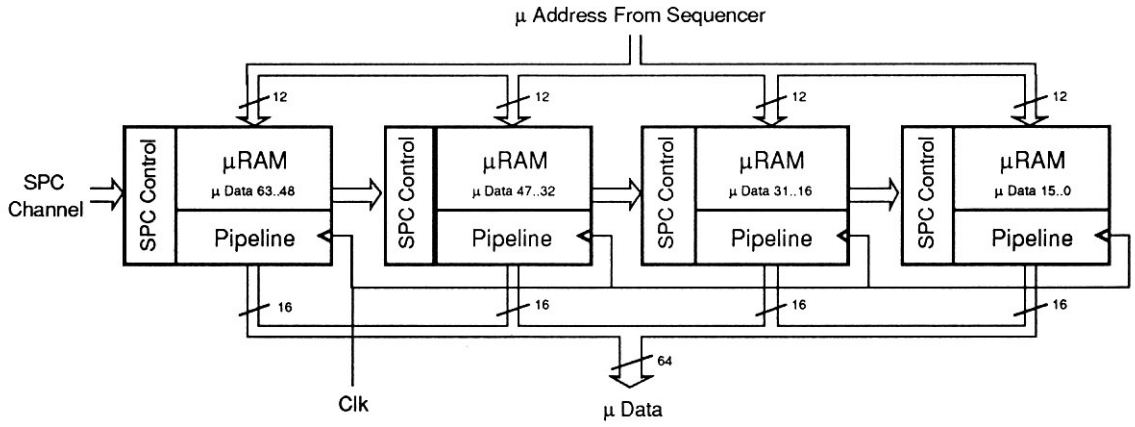


Figura 2.6 La micromemoria y el pipeline

3.- Unidad de Procesamiento

3.1.- La RALU

La RALU es el verdadero corazón de todo el procesador microprogramable. Está implementada con un solo circuito de 16 bits del tipo 49C402 de IDT cuyo esquema de bloques se muestra en la figura 3.1.

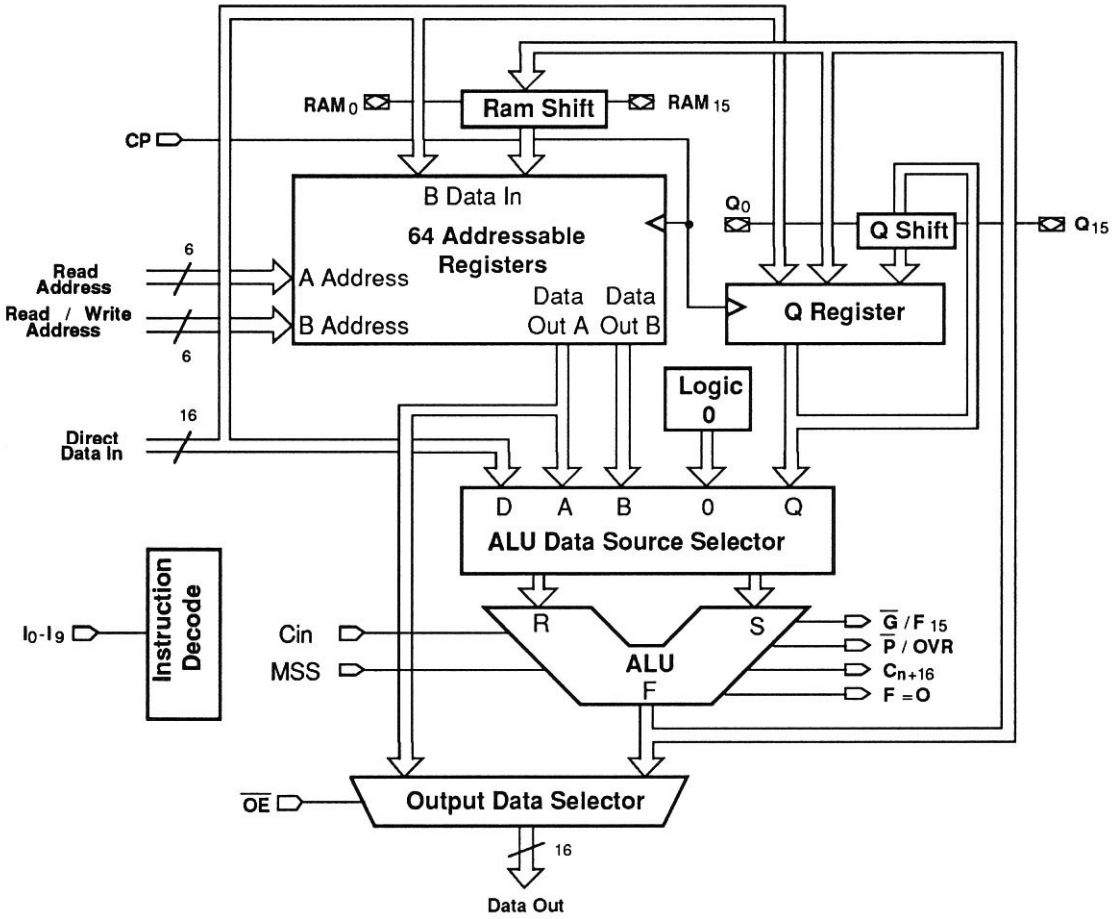


Figura 3.1 Esquema de bloques de la RALU

La RALU 49C402 es una RALU de dos direcciones lo que permite instrucciones del siguiente tipo:

B := A operación B;

donde A y B son uno de los 64 registros intemos de la RALU que son direccionados por las señales A0 ... A5 y B0 ... B5.

La figura 3.2 muestra el direccionamiento de los registros en la taijeta Copro II. Se aprecia que los dos bits de mayor peso de las direcciones vienen siempre de la microinstrucción, mientras que los cuatro bits menos significativos son multiplexados entre el registro de instrucción y el microcódigo. Esto se debe a que el registro de instrucción sólo tiene dos campos de cuatro bits cada uno para direccionar los operandos pues los ocho bits restantes (de mayor peso) son el código de la instrucción (fig 3.8). Con esta solución se consigue de todos modos acceder a todos los registros de la RALU en caso necesario.

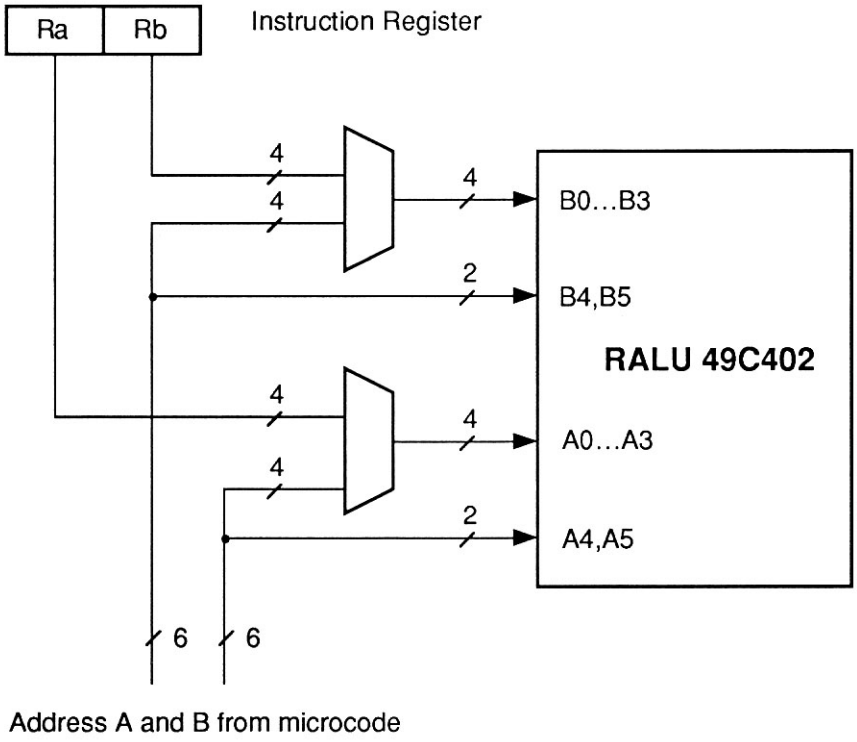
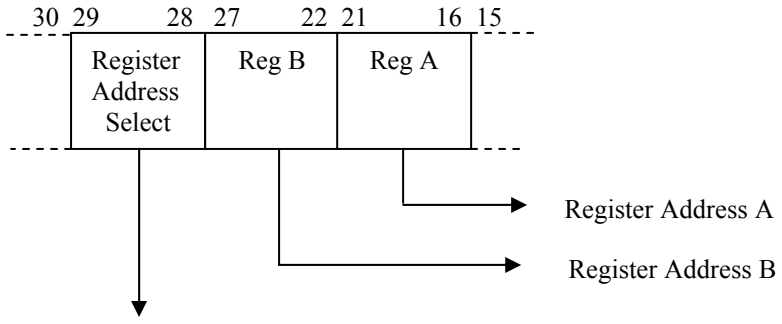


Figura 3.2 Direccionamiento, de los registros de la P.ALU

Los dos multiplexores son controlados por dos bits de la microinstrucción y la figura 3.3 nos muestra de qué manera se codifican.



HEX 11-10	REGISTER ADDRESS A	REGISTER ADDRESS B
0	PL	PL
1	IR	PL
2	PL	IR
3	IR	IR

PL = Pipeline IR = Instruction Register

Figura 3.3 Origen de las direcciones de los registros de la RALU

La instrucción de la RALU está definida por los diez bits de instrucción I0 ... I9 y se divide en tres campos:

El primer campo (I0 ... I2) selecciona las fuentes para la ALU; la figura 3.5 muestra el segundo campo (I3 ... I5) que elige la operación aritmética de la ALU:

HEX I2-I0	ALU SOURCE OPERANDS		MNEMONIC
	R	S	
0	A	Q	AQ
1	A	B	AB
2	0	Q	ZQ
3	0	B	ZB
4	0	A	ZA
5	D	A	DA
6	D	Q	DQ
7	D	0	DZ

Figura 3.4 ALU, control de fuente

HEX I5-I3	ALU FUNCTION	MNEMONIC
0	ADD	R plus S
1	SUBR	S minus R
2	SUBS	R minus S
3	OR	RORS
4	AND	RANDS
5	NOTRS	NOT R AND S
6	EXOR	R EX-OR S
7	EXNOR	R EX-NOR S

Figura 3.5 ALU, control de función.

La figura 3.6 muestra finalmente el tercero de los campos (I6...I9) que define lo que tiene que escribirse en el registro Rb o en el registro Q. Hay algunas instrucciones que permiten la carga simultánea de ambos. Esto es posible gracias a las nuevas instrucciones de las que se habló antes. La instrucción 3 p.e. carga el resultado F de la .ALU en el registro Rb y en el registro Q con las entradas directas (Direct Data In).

HEX 19 - 16	RAM FUCTION		Q FUNCTION		MNEMONIC
	SHIFT	LOAD	SHIFT	LOAD	
8	N	N	N	F -> 0	OREG
9	N	N	N	N	NOP
A	N	F -> B	N	N	RAMA
B	N	F->B	N	N	RAMF
C	DOWN	F/2 -> B	DOWN	Q/2 -> 0	RAMQD
D	DOWN	F/2 -> B	N	N	RAMD
E	UP	2F -> B	UP	Q/2 -> 0	RAMQU
F	UP	2F -> B	N	N	RAMU
0	N	D->B	N	F->Q	DFF
1	N	D -> 8	N	F->Q	DFA
2	N	F -> B	N	D->Q	FDF
3	N	F -> B	N	D->Q	FDA
4	N	N	DOWN	Q/2 -> 0	XQDF
5	N	D -> B	N	N	DXF
6	N	N	UP	2Q -> Q	XQUF
7	N	N	N	D->Q	XDF

Figura 3.6 ALU, control del destino.

La RALU emplea dos buses de 16 bits para comumicarse con los demás elementos: el bus D de entrada y el bus Y de salida. Sobre la taijeta Copro II estos buses están interconectados para formar un solo bus bidireccional de datos Y que está conectado además a la macromemoria, al FlagShifter, al registro de instrucción, al bus externo, al registro de dirección y a los triestados del driver (vease figura 3.7).

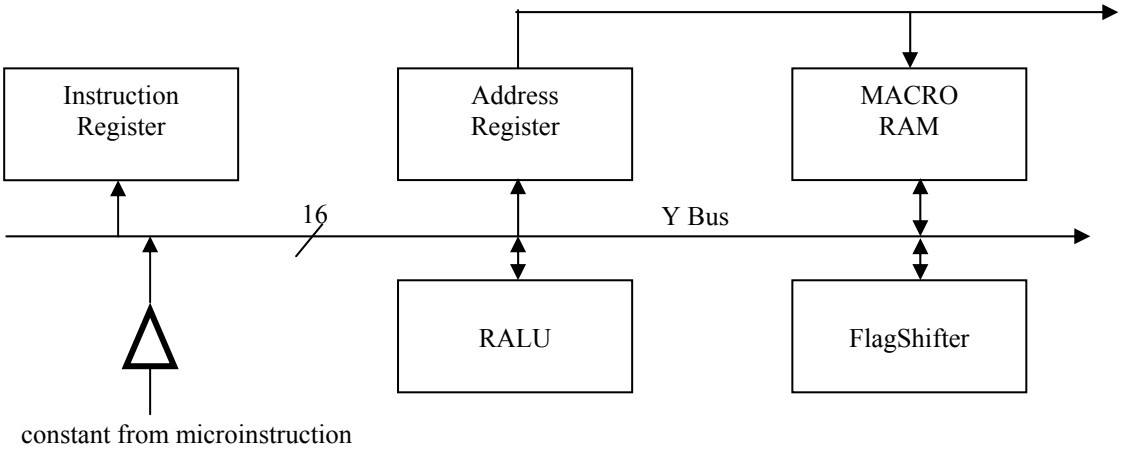


Figura 3.7 El bus Y

3.2 La macromemoria y el registro de instrucción

El tarnaflo de la macromemoria es de 4K de palabras de 16 bits. La macromemoria ocupa las direcciones comprendidas entre 0 y 0FFF hexadecimal. Dado que los registros de direccionamiento son de 16 bits, queda todavía mucho espacio del mapa de memoria sin asignar, que podría ser utilizado para ampliar la memoria física del sistema, o para implementar otras funciones (periféricos mapeados en memoria, ROM...).

El registro de instrucción tiene un tamaño de 16 bits y se carga desde la macromemoria. Se compone de tres partes:

-Los bits 0..3 que especifican la dirección Rb de la RALU.

-Los bits 4..7 que especifican la dirección Ra de la RALU.

-Los bits 8.. 15 que especifican el código de operación le la instrucción cargada.

Recordemos que la dirección de salto que corresponde a un código de operación se calcula de la forma siguiente: **Dirección de salto = 16 * Código de operación**.

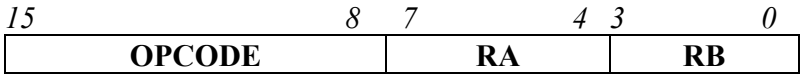


Figura 3.8 Formato del registro de instrucción

3.3- El control del bus Y

El control del bus Y, es decir, las señales 'Output Enable', señales de carga de registro (p. e. el registro de dirección) y las señales generales como /MEM o /WRITE, se efectúa por medio de un GAL 16V8. De esta manera se evitan conflictos en el bus Y porque las combinaciones de estas señales que provocarían cortocircuitos son inhabilitadas por la programación vertical. La programación de transferencias de datos por medio del bus Y se simplifican en gran medida.

La figura 3.9 muestra esquemáficamente de que forma se generan las señales de control del bus Y. Las cuatro señales provenientes de la microinstrucción seleccionan la transferencia.. 'MADDR15', que es el bit de mayor peso del registro de dirección, sirve para distinguir los accesos a la macromemoria interna y externa.

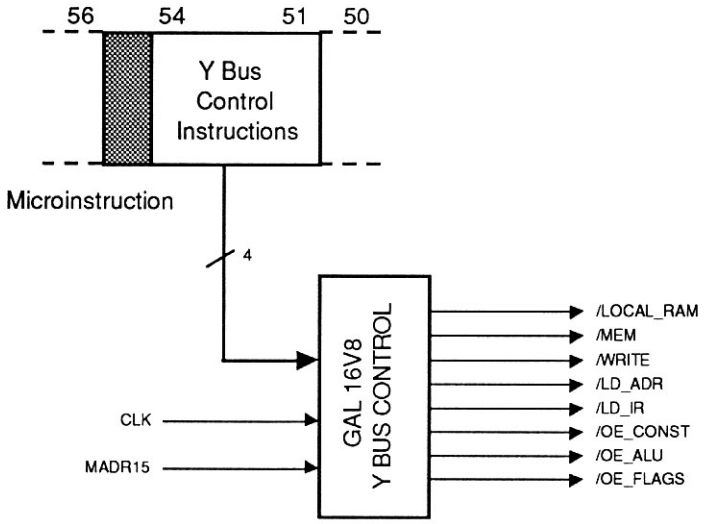


Figura 3.9 El control del bus Y

La figura 3.10 muestra finalmente la lista de posibles transferencias. Se aprecia que no se han implementado todas las posibilidades. La instrucción 2 p.e. corresponde a una búsqueda de instrucción (*intruction fetch*) mientras que la instrucción B sirve para salvaguardar el contenido de los micro o de los macroflags.

HEX I3-I0	Y BUS FUNCTION	HEX I3-I0	Y BUS FUNCTION
0	NOP	7	Const to ALU
1	Mem lo ALU	8	Const to Mem
2	Mem lo IR	9	Const to Flags
3	Mem to Flags	A	Flags to ALU
4	ALU lo Mem	B	Flags to Mem
5	ALU to ADR	C	Flags to ADR
6	ALU to Flags	D - F	Not Defined

Figura 3. 10 Control del bus Y

4.- Flagshifter

4.1.- Funcionamiento general

La figura 4.3 muestra el esquema de bloques del FlagShifler que está implementado en el circuito programable EP900, y que se compone de dos partes funcionalrnente independientes.

La primera se encarga de almacenar los micro- y macrofiags y de proveer al secuenciador de los flags de condición (/CC). Los cuatro microflags y sus respectivos macroflags son normalmente una copia anterior de los cuatro flags Z, N, V y C de la RALU, pero gracias a la conexión con el bus Y se pueden cargar o salvaguardar uno de los conjuntos de flags. Por ejemplo, es posible transferir el contenido de los macroflags a la RALU, modificar uno de los flags y devolver el resultado a los macroflags. Son las señales de instrucción I0..I2 las que seleccionan la transferencia deseada.

Los flags de condición son generados por una función lógica combinatoria. Esta función puede efectuar el test simplemente a uno de los flags o a su complemento o bien comparar números con signo o no. Ejemplos de test típicos son: Micro Zero, Macro Not Carry o A < B.

La segunda parte permite manejar las funciones de desplazamiento y de rotación. La RALU espera que se le especifice el bit 0 para un desplazamiento a la izquierda, e igualmente con el bit 15 para un desplazamiento a la derecha. Para un desplazamiento aritmético a la derecha, el bit 15 tiene p.e. que guardar el signo del número a desplazar (figura 4.1), mientras que el desplazamiento lógico a la derecha demanda siempre que este mismo bit esté a cero (figura 4.2).

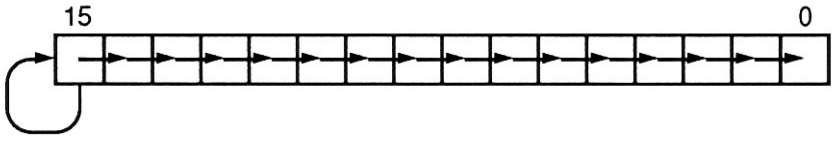


Figura 4.1 Desplazamiento aritmético a la derecha.

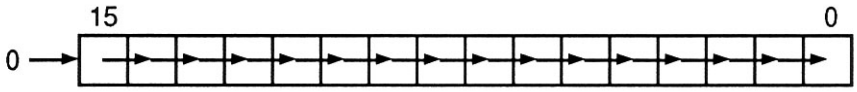


Figura 4.2 Desplazamiento lógico a la derecha.

Los bits de instrucción I8 ... I9 permiten elegir entre cuatro distintos valores futuros del bit 15, ya sea 0, 1 N (bit de signo) o el bit 0. Lo mismo es evidentemente posible para los desplazamicntos a la izquierda invirtiendo los papeles de los bits 0 y 15.

La RALU tiene dos unidades de desplazamiento diferentes. Por una parte la que se encuentra a la entrada de los registros generales, y que emplea las señales RAM0 y RAM15 y por otra parte la que está asociada al registro Q y que emplea las señales Q0 y Q15. Sin embargo no es posible desplazar a la derecha con una y a la izquierda con la otra unidad, sí permite controlar las señales RAM0 y Q0 (respectivamente RAM15 y Q15) así como sus estados triestado simultáneamente.

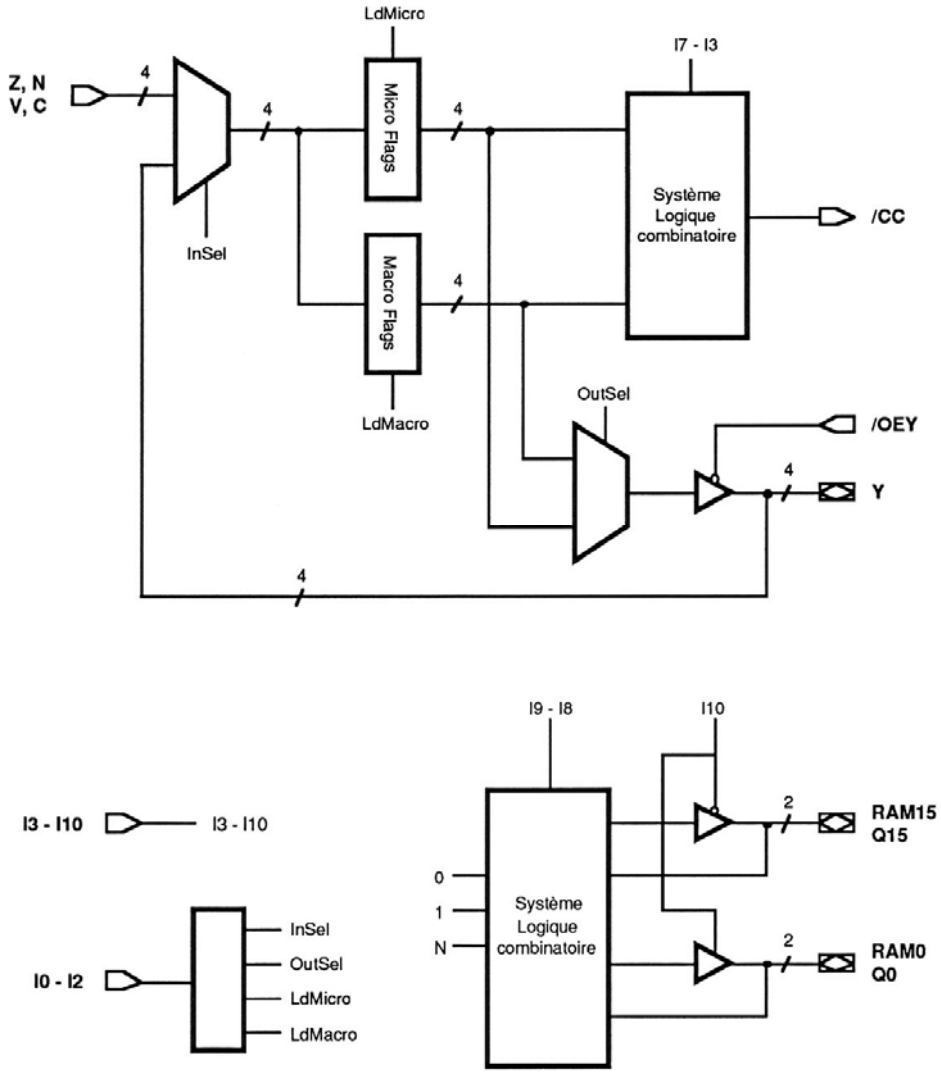


Figura 4.3 Esquema de bloques del FlagShifter

4.2.- Las instrucciones del FlagShifter

4.2.1.- Campos de instrucción del FlagShifter

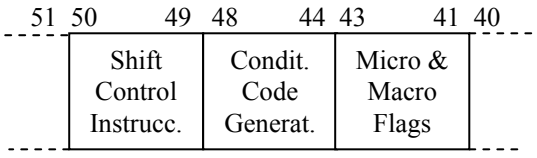


Figura 4.4. Campos de microinstrucción del FlagShifter.

El FlagShifter utiliza tres campos de la microinstrucción. El primero se ocupa de la carga y almacenamiento de los flags, el segundo de la selección de la función de test, y el tercero de controlar las funciones de rotación y desplazamiento de la RALU.

4.2.2.- Instrucciones de carga / almacenamiento de los flags

La figura 4.5 muestra el efecto de los bits de instrucción I0..I2. Las instrucciones 0..3 no tienen efecto alguno sobre el contenido de los dos conjuntos de flags, mientras que las instrucciones 4..7 cargan los micro o los macroflags desde los flags de la RALU o desde los cuatro bits de menor peso del bus Y. Las instrucciones 2 y 3 permiten además transferir el contenido de los micro o macroflags al bus Y.

HEX 12-10	FLAG FUNCTION	MNEMONIC
0		NOP
1		NOP
2	Put Micro to Y Bus	MITOY
3	Put Macro to Y Bus	MaToY
4	Load Micro from Alu	AluTÓMI
5	Load Macro from Alu	AluToMa
6	Load Micro from Y Bus	YTOMI
7	Load Macro from Y Bus	YToMa

Figura 4.5 Instrucciones de carga /almacenamiento de los flags.

4.2.3.- Instrucciones de test

La generación de las banderas de condición que son utilizadas por el secuenciador dependen de las señales de instrucción I3 .. I7 (figura 4.6). Las instrucciones 0 .. 7 verifican uno de los micro o de los macroflags, mientras que las instrucciones 10 .. 17 verifican su complementario lógico. LAs instrucciones 8 .. F y 18 .. 1F efectuan comparaciones aritméticas entre números con signo o sin signo. Los mnemonicos deben leerse de la manera siguiente: p.ej. MaUGE se lee como **Ma**cro **U**nsigned **G**reater **E**qual. **Ma** hace referencia a Macroflags, **Mi** a microflags, etc.

HEX 15-13	TEST FUNCTION	MNEMONIC
0	Micro Zero	MiZ
1	Micro Negativ	MiN
2	Micro Overflow	MiV
3	Micro Carry	MiC
4	Macro Zero	MaZ
5	Macro Negativ	MaN
6	Macro Overflow	MaV
7	Macro Carry	MaC
8	Micro Signed A > B	MiSG
9	Micro Signed A < B	MiSS
A	Micro Unsigned A > B	MiUG
B	Micro Unsigned A < B	MiUS
c	Macro Signed A > B	MaSG
D	Macro Signed A < B	MaSS
E	Macro Unsigned A > B	MaUG
F	Macro Unsigned A < B	MaUS

HEX 15-13	TEST FUNCTION	MNEMONIC
10	Not Micro Zero	NMiZ
11	Not Micro Negativ	NMiN
12	Not Micro Overflow	NMiV
13	Not Micro Carry	NMiC
14	Not Macro Zero	NMaZ
15	Not Macro Negativ	NMaN
16	Not Macro Overflow	NMaV
17	Not Macro Carry	NMaC
18	Micro Sígned A <= B	MiSSE
19	Micro Sígned A >= B	MiSGE
1 A	Micro Unsigned A <= B	MiUSE
1 B	Micro Unsigned A >= B	MiUGE
1 C	Macro Signed A <= B	MaSSE
1 D	Macro Signed A >= B	MaSGE
1 E	Macro Unsigned A <= B	MaUSE
1 F	Macro Unsigned A >= B	MaUGE

Figura 4.6 Instrucciones de test

4.2.4.- Las Instrucciones de desplazamiento / rotación.

La figura 4.7 muestra el comportamiento del FlagShifter en función de los bita de instrucciones I8 ... I10. El bit de instrucción I10 está asociado al bit de instrucción I7 de la RALU, lo que evita posibles conflictos entre las señales RAM0, RAM15, Q0 ó Q15, ya que es el bit de la RALU el que indica el sentido del desplazamiento. Por consiguiente, el sentido de la rotación no se especifica en el campo de la microinstrucción del FlagShifter, sino en el campo de la RALU. Lo que queda a especificar en el campo del FlagShifter por las señales I8 e I9 es lo que se desea inyectar en una de las unidades de desplazamiento. De esta forma un desplazamiento aritmético a la derecha se expresa mediante un código de instrucción 0 (ALU I7 = 1) o bien una rotación a la izquierda por el código 3 (ALU I7 = 0).

ALU I7	HEX I9-I8	RAM 0	RAM 15	Q 0	Q 15
0	0	High Z	0	High Z	0
0	1	High Z	1	Hígh Z	1
0	2	High Z	N	High Z	N
0	3	High Z	RAM0	High Z	Q0
1	0	0	High Z	0	High Z
1	1	1	High Z	1	High Z
1	2	N	High Z	N	High Z
1	3	RAM 15	High Z	Q15	High Z

Figura 4.7 Instrucciones de desplazamiento / rotación.