MICROINSTRUCCIONES.

La unidad de control es el ***"cerebro"*** del microprocesador. Esta unidad realiza la decodificación de los códigos de operación y de los códigos de función para poder identificar la instrucción que se va a ejecutar y su tipo (Tipo I, Tipo R y Tipo J). Una vez identificada la instrucción, la unidad de control activa o no cada una de las señales de control de todo el microprocesador. De esta manera se forma un código que emite la unidad de control al que llamamos ***microinstrucción***. Por lo tanto, podemos definir a ***las microinstrucciones como los códigos que emite la unidad de control para ejecutar cada instrucción del ensamblador del microprocesador***.

Para entender como se realiza la ejecución de las instrucciones a través del procesador y como se generan las microinstrucciones, analicemos los ejemplos siguientes.

Ejemplo 1. Programa que genera un contador. Observe el código mostrado en la tabla 1.

Instrucciones	UP	DW	WPC	SDMP	SR2	SWD	SHE	DIR	WR	LF	SEXT	SOP1	SOP2	ALUOP	SDMD	WD	SR
LI R0, #1	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
LI R1, #7	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
CICLO: ADD R1, R1, R0	0	0	0	0	0	1	0	0	1	1	0	0	0	0011	0	0	1
SWI R1, 5	0	0	0	0	1	0	0	0	0	0	0	0	0	0000	1	1	0
B CICLO	0	0	1	0	0	0	0	0	0	0	0	0	0	0000	0	0	0

Tabla 1 Microinstrucciones del programa ejemplo 1.

Durante la ejecución de cada instrucción TODAS las señales de control de cada bloque del microprocesador se encuentran por defecto con cero. Lo primero que se realiza es tomar una instrucción de la memoria de programa, eso quiere decir, que se lee la localidad 0 de la memoria y se saca el primer dato de 25 bits guardado ahí. Después la unidad de control decodifica la instrucción para saber que tipo de instrucción acaba de leer de la memoria de programa. Posteriormente la unidad de control activa las correspondientes señales de control del procesador para ejecutar la instrucción.

La ejecución del programa comienza después de un reset o clear. Esto provoca que el Stack Pointer (SP) de la pila comience en cero al igual que todos contadores de programa y registros del procesador. Con el SP en cero el contador que maneja las direcciones de la memoria de programa es PC0. En este ejemplo en particular la primera instrucción a ejecutar es LI R0, #1. Esta instrucción sale de la memoria de programa y el código de operación colocado en los bits 24…20 se va directamente a la unidad de control. El código del registro R0 colocado en los bits 19…16 se coloca en el bus de entrada WRITE REGISTER del archivo de registros. El número 1 colocado en los bits 15…0 se coloca en el bus de entrada WRITE DATA del archivo de registros. La instrucción se ejecuta cuando la unidad de control activa la señal

WR del archivo de registros, esto provoca que en el siguiente flanco de subida de la señal de reloj se cargue el número 1 en el registro R0 y que el valor del PC0 se incremente en uno. La siguiente instrucción LI R1, #7 se ejecuta de la misma manera.

Después de la ejecución de las instrucciones de carga, la siguiente instrucción a ejecutar se encuentra en la dirección 2 de memoria. Esta instrucción es ADD R1, R1, R0. Esta instrucción sale de la memoria de programa y el código de operación colocado en los bits 24…20 se va directamente a la unidad de control. Como se trata de una instrucción tipo R el código de función colocado en los bits 3…0 también se va a la unidad de control. El código del registro R1 colocado en los bits 19…16 se coloca en el bus de entrada WRITE REGISTER del archivo de registros. El código del registro R1 colocado en los bits 15…12 se coloca en el bus de entrada READ REGISTER 1 del archivo de registros. Esto provoca que el número 7 almacenado en R1 pase directamente a la ALU. El código del registro R0 colocado en los bits 11…8 se coloca en el bus de entrada READ REGISTER 2 del archivo de registros. Esto provoca que el número 1 almacenado en R0 pase directamente a la ALU. La instrucción se ejecuta cuando la unidad de control coloca el código “0011” en el bus ALUOP, activa la señal WR del archivo de registros, la señal SR, la señal SWD y la señal LF del registro de banderas **(La señal LF se activa con todas las instrucciones que modifican las banderas de la ALU).** Esto provoca que se realice la suma de los operandos de la ALU y que el resultado de la suma se coloque en el bus de entrada WRITE REGISTER del archivo de registros. Entonces, el resultado de la suma se almacena en el registro R1 en el siguiente flanco de subida de la señal de reloj y que el valor del PC0 se incremente a tres.

Posteriormente, la siguiente instrucción a ejecutar es SWI R1, 5. Esta instrucción sale de la memoria de programa y el código de operación colocado en los bits 24… 20 se va directamente a la unidad de control. El código del registro R1 colocado en los bits 19…16 se coloca en el bus de entrada READ REGISTER 2 del archivo de registros. Esto provoca que el número 8 almacenado en R1 se coloque en el bus Di15…Di0 de la memoria de datos. El número 5 colocado en los bits 15…0 se coloca en el bus de direcciones A15…A0 de la memoria de datos. La instrucción se ejecuta cuando la unidad de control activa la señal WD de la memoria de datos y la señal SDMD**.** Esto provoca que se escriba el número 8 en la dirección de memoria 5 en el siguiente flanco de subida de la señal de reloj y que el valor del PC0 se incremente a cuatro.

Finalmente, la última instrucción a ejecutar es B CICLO. Esta instrucción sale de la memoria de programa y el código de operación colocado en los bits 24…20 se va directamente a la unidad de control. El código colocado en los bits 19…16 no se utiliza en esta instrucción. El número 5 colocado en los bits 15…0 se coloca en el bus de entrada D15…D0 de la pila. La instrucción se ejecuta cuando la unidad de control activa la señal WPC de la pila**.** Esto provoca que se escriba la dirección 2 en PC0 y se de el salto a la etiqueta CICLO.

Ejemplo 2. Programa que obtiene los primeros 12 términos de la serie de Fibonacci. Observe el código mostrado en la tabla 2.

Instrucciones	UP	DW	WPC	SDMP	SR2	SWD	SHE	DIR	WR	LF	SEXT	SOP1	SOP2	ALUOP	SDMD	WD	SR
LI R0, #0	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
LI R1, #1	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
LI R2, #0	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
LI R3, #10	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
SERIE: ADD R4, R0, R1	0	0	0	0	0	1	0	0	1	1	0	0	0	0011	0	0	1
SWI R4, 72	0	0	0	0	1	0	0	0	0	0	0	0	0	0000	1	1	0
ADDI R0, R1,#0	0	0	0	0	0	1	0	0	1	1	0	0	1	0011	0	0	1
ADDI R1, R4,#0	0	0	0	0	0	1	0	0	1	1	0	0	1	0011	0	0	1
ADDI R2, R2,#1	0	0	0	0	0	1	0	0	1	1	0	0	1	0011	0	0	1
BNEI R2, R3, SERIE (semiciclo positivo)	0	0	0	0	1	0	0	0	0	1	0	0	0	0111	0	0	0
Si la condición se cumple (semiciclo negativo)	0	0	1	1	0	0	0	0	0	0	0	1	1	0011	0	0	1
CICLO: NOP	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	0	0	0
B CICLO	0	0	1	0	0	0	0	0	0	0	0	0	0	0000	0	0	0

Tabla 2 Microinstrucciones del programa ejemplo 2.

Ejemplo 3. Programa que genera un contador usando una función. Observe el código mostrado en la tabla 3.

Instrucciones	UP	DW	WPC	SDMP	SR2	SWD	SHE	DIR	WR	LF	SEXT	SOP1	SOP2	ALUOP	SDMD	WD	SR
LI R0, #1	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
LI R1, #7	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
CICLO: CALL SUMA	1	0	1	0	0	0	0	0	0	0	0	0	0	0000	0	0	0
SWI R1, 5	0	0	0	0	1	0	0	0	0	0	0	0	0	0000	1	1	0
B CICLO	0	0	1	0	0	0	0	0	0	0	0	0	0	0000	0	0	0
SUMA: ADD R1, R1, R0	0	0	0	0	0	1	0	0	1	1	0	0	0	0011	0	0	1
RET	0	1	0	0	0	0	0	0	0	0	0	0	0	0000	0	0	0

Tabla 3 Microinstrucciones del programa ejemplo 3.

AUTOR: VICTOR HUGO GARCÍA ORTEGA

Ejemplo 4. Programa que obtiene el mayor de 3 números diferentes. Observe el código mostrado en la tabla 4.

Instrucciones	UP	DW	WPC	SDMP	SR2	SWD	SHE	DIR	WR	LF	SEXT	SOP1	SOP2	ALUOP	SDMD	WD	SR
LI R0, #23	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
LI R1, #-45	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
LI R1, #165	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
BGTI R0, R1, CR0R2 (semiciclo positivo)	0	0	0	0	1	0	0	0	0	1	0	0	0	0111	0	0	0
Si la condición se cumple (semiciclo negativo)	0	0	1	1	0	0	0	0	0	0	0	1	1	0011	0	0	1
BLTI R1, R2, R2MAY (semiciclo positivo)	0	0	0	0	1	0	0	0	0	1	0	0	0	0111	0	0	0
Si la condición se cumple (semiciclo negativo)	0	0	1	1	0	0	0	0	0	0	0	1	1	0011	0	0	1
SWI R1, 0X20	0	0	0	0	1	0	0	0	0	0	0	0	0	0000	1	1	0
B CICLO	0	0	1	0	0	0	0	0	0	0	0	0	0	0000	0	0	0
CR0R2: BLTI R0, R2, R2MAY (semiciclo positivo)	0	0	0	0	1	0	0	0	0	1	0	0	0	0111	0	0	0
Si la condición se cumple (semiciclo negativo)	0	0	1	1	0	0	0	0	0	0	0	1	1	0011	0	0	1
SWI R0, 0X20	0	0	0	0	1	0	0	0	0	0	0	0	0	0000	1	1	0
B CICLO	0	0	1	0	0	0	0	0	0	0	0	0	0	0000	0	0	0
R2MAY: SWI R2, 0X20	0	0	0	0	1	0	0	0	0	0	0	0	0	0000	1	1	0
CICLO: NOP	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	0	0	0
B CICLO	0	0	1	0	0	0	0	0	0	0	0	0	0	0000	0	0	0

Tabla 4 Microinstrucciones del programa ejemplo 4.

AUTOR: VICTOR HUGO GARCÍA ORTEGA

Ejemplo 5. Programa que calcula el promedio de un arreglo de 4 elementos. Observe el código mostrado en la tabla 5.

Instrucciones	UP	DW	WPC	SDMP	SR2	SWD	SHE	DIR	WR	LF	SEXT	SOP1	SOP2	ALUOP	SDMD	WD	SR
LI R0, #23	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
SWI R0, 10	0	0	0	0	1	0	0	0	0	0	0	0	0	0000	1	1	0
LI R0, #130	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
SWI R0, 11	0	0	0	0	1	0	0	0	0	0	0	0	0	0000	1	1	0
LI R0, #70	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
SWI R0, 12	0	0	0	0	1	0	0	0	0	0	0	0	0	0000	1	1	0
LI R0, #260	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
SWI R0, 13	0	0	0	0	1	0	0	0	0	0	0	0	0	0000	1	1	0
LI R0, #0	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
LI R1, #10	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
LI R2, #0	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
LI R3, #4	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
PROM: LW R4, R1, R2	0	0	0	0	0	1	0	0	1	0	0	0	0	0011	0	0	0
ADD R0, R0, R4	0	0	0	0	0	1	0	0	1	1	0	0	0	0011	0	0	1
ADDI R2, R2, #1	0	0	0	0	0	1	0	0	1	1	0	0	1	0011	0	0	1
BNEI R2, R3, PROM (semiciclo positivo)	0	0	0	0	1	0	0	0	0	1	0	0	0	0111	0	0	0
Si la condición se cumple (semiciclo negativo)	0	0	1	1	0	0	0	0	0	0	0	1	1	0011	0	0	1
SRL R0, R0, #2	0	0	0	0	0	0	1	0	0	0	0	0	0	0000	0	0	0
SWI R0, 20	0	0	0	0	1	0	0	0	0	0	0	0	0	0000	1	1	0
CICLO: NOP	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	0	0	0
B CICLO	0	0	1	0	0	0	0	0	0	0	0	0	0	0000	0	0	0

Tabla 5 Microinstrucciones del programa ejemplo 6.

Ejemplo 6. Programa que inicializa un arreglo. Observe el código mostrado en la tabla 6.

Instrucciones	UP	DW	WPC	SDMP	SR2	SWD	SHE	DIR	WR	LF	SEXT	SOP1	SOP2	ALUOP	SDMD	WD	SR
LI R0, #0	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
LI R2, #0	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
LI R3, #8	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0
INI: SW R0, 10(R2)	0	0	0	0	1	0	0	0	0	0	1	0	1	0000	0	1	0
ADDI R2, R2, #1	0	0	0	0	0	1	0	0	1	1	0	0	1	0011	0	0	1
BNEI R2, R3, INI (semiciclo positivo)	0	0	0	0	1	0	0	0	0	1	0	0	0	0111	0	0	0
Si la condición se cumple (semiciclo negativo)	0	0	1	1	0	0	0	0	0	0	0	1	1	0011	0	0	1
CICLO: NOP	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	0	0	0
B CICLO	0	0	1	0	0	0	0	0	0	0	0	0	0	0000	0	0	0

Tabla 6 Microinstrucciones del programa ejemplo 6.

Una vez analizado los ejemplos anteriores, las microinstrucciones de cada instrucción las podemos obtener y anotar en la tabla 7. Hay que señalar que la escritura de esta tabla es otra forma de representar las microinstrucciones de cada instrucción del ensamblador.

INSTRUCCIONES DE CARGA Y ALMACENAMIENTO									
Instr.	Ejemplo	Significado	Código de operación						Microinstrucciones
LI	LI Rd, #Slit16	Rd = Slit16	01	Rd	Slit16				WR
LWI	LWI Rd, lit16	Rd = Mem[lit16]	02	Rd	lit16				
LW	LW Rd,lit12(Rt)	Rd = Mem[Rt+lit12]	23	Rd	Rt	lit12			
SWI	SWI Rd, lit16	Mem[lit16] = Rd	03	Rd	lit16				SR2 SDMD WD
SW	SW Rd, lit12(Rt)	Mem[Rt+lit12] = Rd	04	Rd	Rt	lit12			
INSTRUCCIONES ARITMÉTICAS									
ADD	ADD Rd,Rt,Rs	Rd = Rt+Rs	00	Rd	Rt	Rs	S/U	00	SWD WR LF SR ALUOP=0011
SUB	SUB Rd,Rt,Rs	Rd = Rt-Rs	00	Rd	Rt	Rs	S/U	01	
ADDI	ADDI Rd,Rt,#Slit12	Rd = Rt+Slit12	05	Rd	Rt	Slit12			SWD WR LF SR ALUOP=0011 SOP2
SUBI	SUBI Rd,Rt,#Slit12	Rd = Rt-Slit12	06	Rd	Rt	Slit12			
INSTRUCCIONES LÓGICAS									
AND	AND Rd,Rt,R	Rd=Rt&Rs	00	Rd	Rt	Rs	S/U	02	
OR	OR Rd,Rt,Rs	Rd=Rt | Rs	00	Rd	Rt	Rs	S/U	03	
XOR	XOR Rd,Rt,Rs	Rd=Rt ^ Rs	00	Rd	Rt	Rs	S/U	04	
NAND	NAND Rd,Rt,Rs	Rd=~(Rt & Rs)	00	Rd	Rt	Rs	S/U	05	
NOR	NOR Rd,Rt,Rs	Rd=~(Rt | Rs)	00	Rd	Rt	Rs	S/U	06	
XNOR	NOR Rd,Rt,Rs	Rd=~(Rt ^ Rs)	00	Rd	Rt	Rs	S/U	07	
NOT	NOT Rd, Rs	Rd = ~Rs	00	Rd	Rs	Rs	S/U	08	
ANDI	ANDI Rd,Rt,#lit12	Rd=Rt & lit12	07	Rd	Rt	lit12			
ORI	ORI Rd,Rt,#lit12	Rd=Rt | lit12	08	Rd	Rt	lit12			

AUTOR: VICTOR HUGO GARCÍA ORTEGA

XORI	XORI Rd,Rt,#lit12	Rd=Rt ^ lit12	09	Rd	Rt	lit12			
NANDI	NANDI Rd,Rt,#lit12	Rd=~(Rt & lit12)	10	Rd	Rt	lit12			
NORI	NORI Rd,Rt,#lit12	Rd=~(Rt | lit12)	11	Rd	Rt	lit12			
XNORI	XNORI Rd,Rt,#lit12	Rd=~(Rt ^ lit12)	12	Rd	Rt	lit12			
INSTRUCCIONES DE CORRIMIENTO									
SLL	SLL Rd,Rt,#lit4	Rd=Rt<<lit4	00	Rd	Rt	S/U	lit4	09	
SRL	SRL Rd,Rt,#lit4	Rd=Rt>>lit4	00	Rd	Rt	S/U	lit4	10	
INSTRUCCIONES DE SALTOS CONDICIONALES E INCONDICIONALES									
BEQI	BEQI Rd,Rt,Slit12	If(Rd==Rt) goto Slit12 PC = PC + Slit12	13	Rd	Rt	Slit12			
BNEI	BNEI Rd,Rt,Slit12	If(Rd!=Rt) goto Slit12 PC = PC + Slit12	14	Rd	Rt	Slit12			SR2 LF ALUOP=0111 WPC SDMP SOP1 SOP2 SR ALUOP=0011
BLTI	BLTI Rd,Rt,Slit12	If(Rd<Rt) goto Slit12 PC = PC + Slit12	15	Rd	Rt	Slit12			
BLETI	BLETI Rd,Rt,Slit12	If(Rd<=Rt) goto Slit12 PC = PC + Slit12	16	Rd	Rt	Slit12			
BGTI	BGTI Rd,Rt,Slit12	If(Rd>Rt) goto Slit12 PC = PC + Slit12	17	Rd	Rt	Slit12			
BGETI	BGETI Rd,Rt,Slit12	If(Rd>=Rt) goto Slit12 PC = PC + Slit12	18	Rd	Rt	Slit12			
B	B lit16	PC = lit16	19	S/U	lit16				WPC
INSTRUCCIONES DE MANEJO DE SUBRUTINAS									
CALL	CALL #lit16	PC(n+1) = lit16	20	S/U	lit16				UP WPC

AUTOR: VICTOR HUGO GARCÍA ORTEGA

RET	RET	PC = PC(n-1)	21	S/U	S/U	S/U	S/U	S/U	DW
OTRAS INSTRUCCIONES									
NOP	NOP		22	S/U	S/U	S/U	S/U	S/U	

AUTOR: VICTOR HUGO GARCÍA ORTEGA

Memorias de Microcódigo.

Todas las microinstrucciones son almacenadas en memorias, denominadas memoria de Microcódigo. En el ESCOMIPS se tiene dos memorias de este tipo denominadas Memoria de Microcódigo de Operación y Memoria de Microcódigo de Función.

Memoria de Microcódigo de Operación.

Esta memoria contiene todas las microinstrucciones de las instrucciones que NO son Tipo R, es decir, aquellas instrucciones cuyo código de operación es diferente de cero. En esta memoria el código de operación ubicado en los bits 24 a 20 de la instrucción, I[24-20], de la instrucción especifica la dirección de la memoria donde se encuentra guardada la microinstrucción de dicha instrucción. Asi por ejemplo, para la instrucción LI que tiene el código de operación 01, quiere decir que en la dirección de memoria 01 se encuentra la microinstrucción de la instrucción LI.

Por lo tanto, dado que se tienen 5 bits en el campo del código de operación y se tiene 20 líneas de control en el microprocesador, la organización de la memoria de Microcódgo de Operación es de 32 x 20.

Cabe hacer notar, que todas las instrucciones que tienen código de operación cero son instrucciones Tipo R y las microinstrucciones de estas instrucciones no se pueden colocar al mismo tiempo en la dirección cero de la Memoria de Microcódigo de Operación. Por lo que las microinstrucciones de estas instrucciones se colocan en la Memoria de Microcódigo de Función. **Sin embargo, esta dirección de memoria es aprovechada para colocar la primer microinstrucción de las instrucciones de brinco condicional.** La segunda microinstrucción de las instrucciones de brinco condicional se coloca en la dirección de memoria que corresponde con el código de operación de cada instrucción de brinco condicional.

La Memoria de Microcódigo de Operación está definida de la siguiente forma:

Dir	UP	DW	WPC	SDMP	SR2	SWD	SHE	DIR	WR	LF	SEXT	SOP1	SOP2	ALUOP	SDMD	WD	SR	Inst
0	0	0	0	0	1	0	0	0	0	1	0	0	0	0111	0	0	0	Bcond
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0000	0	0	0	LI
2	0	0	0	0	0	1	0	0	1	0	0	0	0	0000	1	0	0	LWI
3	0	0	0	0	1	0	0	0	0	0	0	0	0	0000	1	1	0	SWI
...																		
13	0	0	1	1	0	0	0	0	0	0	0	1	1	0011	0	0	1	BEQI
14	0	0	1	1	0	0	0	0	0	0	0	1	1	0011	0	0	1	BNEI
15	0	0	1	1	0	0	0	0	0	0	0	1	1	0011	0	0	1	BLTI
...																		

AUTOR: VICTOR HUGO GARCÍA ORTEGA

Memoria de Microcódigo de Función.

Esta memoria contiene todas las microinstrucciones de las instrucciones que son Tipo R, es decir, aquellas instrucciones cuyo código de operación es cero. En esta memoria el código de función ubicado en los bits 3 a 0 de la instrucción, I[3-0], de la instrucción especifica la dirección de la memoria donde se encuentra guardada la microinstrucción de dicha instrucción. Asi por ejemplo, para la instrucción SUB que tiene el código de operación 00 y código de función 01, quiere decir que en la dirección de memoria 01 se encuentra la microinstrucción de la instrucción SUB.

Por lo tanto, dado que se tienen 4 bits en el campo del código de función y se tiene 20 líneas de control en el microprocesador, la organización de la memoria de Microcódgo de Función es de 16 x 20.

La Memoria de Microcódigo de Función está definida de la siguiente forma:

Dir	UP	DW	WPC	SDMP	SR2	SWD	SHE	DIR	WR	LF	SEXT	SOP1	SOP2	ALUOP	SDMD	WD	SR	Inst
0	0	0	0	0	0	1	0	0	1	1	0	0	0	0011	0	0	1	ADD
1	0	0	0	0	0	1	0	0	1	1	0	0	0	0111	0	0	1	SUB
2	0	0	0	0	0	1	0	0	1	1	0	0	0	0000	0	0	1	AND
...																		

Carta ASM de la unidad de Control.

Para poder ejecutar las microinstrucciones de las memorias de Microcódigo de Operación y Función debemos elaborar una carta ASM que permita decodificar e identificar el tipo de instrucción para determinar si es TIPO R o no a fin de elegir la memoria de Microcódigo de Operación o la de Función. También debemos identificar cuando se trate de las instrucciones de brinco condicional, esto para generar la microinstrucción en el nivel en alto de la señal de reloj, y determinar si su condición se cumple o no, esto para generar la segunda microinstrucción en el nivel en bajo de la señal de reloj.

La carta ASM de la unidad de control se muestra en la figura 1.

AUTOR: VICTOR HUGO GARCÍA ORTEGA

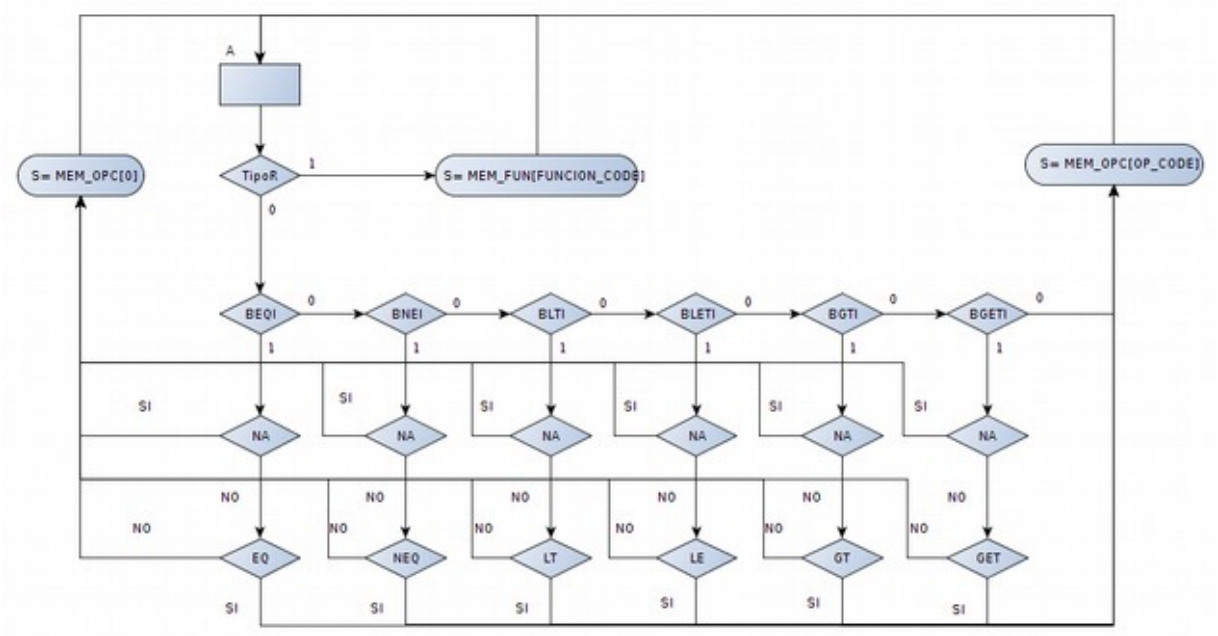
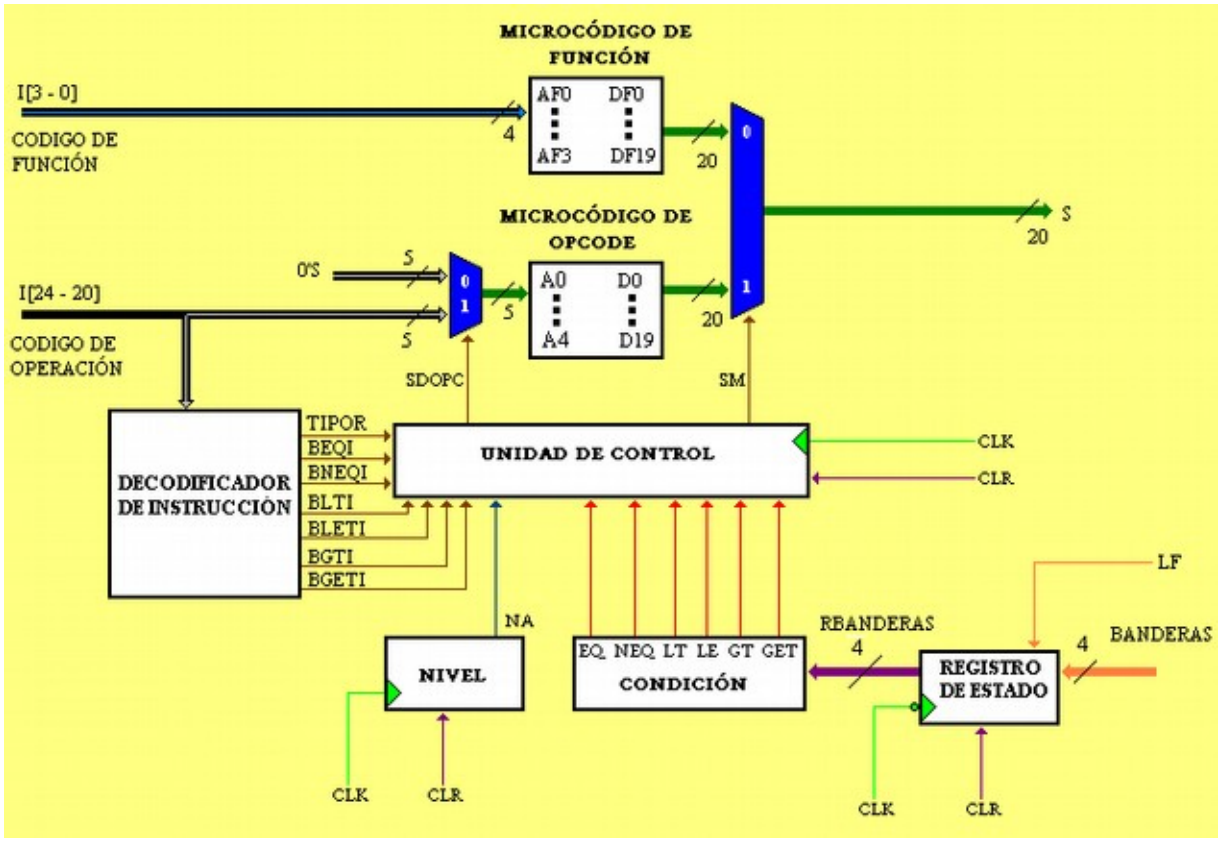


Figura 1: Carta ASM de la unidad de Control

A partir de esta carta ASM se obtiene la ruta de datos de la unidad de control, la cual se muestra en la figura 2.



AUTOR: VICTOR HUGO GARCÍA ORTEGA

Cada bloque de la ruta de datos de la unidad de control se explica a continuación:

Memoria de microcódigo de Operación. Esta memoria contiene las microinstrucciones de las instrucciones cuyo código de operación no es TIPO R.

Memoria de microcódigo de Función. Esta memoria contiene las microinstrucciones de las instrucciones cuyo código de operación es TIPO R.

Decodificador de instrucción. Este bloque determina si se trata de una instrucción TIPO R o es alguna instrucción de brinco condicional. Esto se hace al verificar el código de operación de la instrucción.

Nivel. Este bloque activa su salida NA en nivel alto cuando el reloj, clk, esta en el semiciclo positivo.

Condición. Este bloque determina si la condición de una instrucción de brinco condicional se cumple. Esto se hace con el valor de las banderas generadas en la ALU.

Registro de estado. Este registro guarda el valor de las banderas de estado generadas por la ALU.

Unidad de control. Este bloque contiene el autómata de control de la carta ASM de la unidad de control del ESCOMIPS.