**Universidad Católica “Nuestra Señora de la Asunción’’**

**Facultad de Ciencias y Tecnologías**

**Departamento de Informática y Electrónica**

****

**Sistemas Digitales 2**

**Trabajo Práctico 4:**

**Procesador de uso general**

Profesor: Gerónimo Bellassai

Alumno: Esteban Gamarra

Primer Semestre 2021

Asunción – Paraguay

**Índice:**

* Descripción del trabajo…………………………………………………………………
* Proceso de Diseño……………………………………………………………………
  + - Metodología de diseño…………………………………………………
    - Planteamiento del problema……………………………………………
    - Elección de las instrucciones……………………………………………
    - Ruta de Datos…………………………………………………………….
    - Determinación de las señales de control………………………………..
    - Determinación de las señales de estado………………………………….
    - Diseño del ASM y Algoritmo Particular………………………………..
    - Matriz de señales de control……………………………………………..
    - Ecuaciones de control……………………………………………………
    - Datos para el algoritmo particular………………………………………
* Máquina de Control.……………………………………………………………………….
  + - Representación en Circuito………………………………………………
    - Componentes Reales y/o Basados en……………………………………
* Códigos y Simulaciones VHDL.………………………………………………………….
  + - Códigos………………………………………………………………….
    - Capturas de Pantalla……………………………………………………..
    - Conclusiones……………………………………………………………..

**Descripción del trabajo.**

* Diseñar un procesador de uso general de acuerdo a las siguientes especificaciones
* El trabajo es individual
* El diseño tiene dos etapas, una es totalmente funcional y la otra es con la unidad de control estructurada, pudiéndose mantener la ruta de datos en forma funcional.
* Todos los alumnos seleccionarán un “Conjunto de Instrucciones” de **al menos** 16 instrucciones distintas (independientemente de los modos de direccionamiento y operandos). Debe tener **al menos** dos instrucciones de cada grupo.
* Se deben implementar diferentes modos de direccionamiento para los operandos de acuerdo a las especificaciones del trabajo.
* La arquitectura de la Ruta de Datos (RD) de cada trabajo será especificada. (se pueden agregar modificaciones respecto al modelo adjuntado)
* El ASM debe adaptarse usando como ejemplo los modelos estudiados en clase.
* El sistema final debe están funcionando en VHDL con el algoritmo del Trabajo Práctico 2.
* Debe tener una fuente de interrupción HW.

Proceso de diseño

Metodología de diseño.

* Máquina de control: Método de Registro y Decodificador
* Ruta de Datos: Modelo de bus único de tres estados
* N° de Registros de uso general: 4 (R0,R1,R2,R3)
* Operandos en Instrucción: 2 (el primer operando es también el destino)
* Modos de Direccionamiento: 2 por instrucción en 8 instrucciones
* Bus de direcciones: 8 bits
* Bus de datos: 8 bits
* Conjunto de instrucciones: 16 como mínimo
* Conjunto de instrucciones según tipo: 2 como mínimo
* Implementaciones obligatorias: pila, fuente de interrupción HW, bifurcaciones

**Planteamiento del problema**

Nuestro problema principal para el desarrollo de un procesador de uso general con modelo en base a un único bus de tres estados es el de la organización de las micro-instrucciones por cada instrucción en partes, tal que en ningún momento hayan componentes de la ruta de datos que estén ocupando el único bus al mismo tiempo. Esto es, dos componentes quieran escribir sobre el bus al mismo tiempo.

Para el problema del algoritmo en particular, el cual es la Resta BCD de dos números, y debido a que antes se utilizaban una gran cantidad de punteros y registros para almacenar datos únicos y una sola vez por ciclo. Debemos modificar el algoritmo original presentado y hacerle algunas modificaciones, sobre todo por el hecho del modo de utilización de los operandos en cada instrucción, ya que haciendo que el destino se sobre-escriba cada vez que uno realiza una operación, el dato original se pierde. Además a modo de utilizar menos registros auxiliares, los punteros y su representación del anterior trabajo práctico sufrirán modificaciones, los punteros se utilizarán directamente de la memoria y sus direcciones se modificarán, también, en la memoria.

**Elección de instrucciones**

La codificación de nuestro bloque de memoria que será posteriormente decodificado por nuestro Registro de Instrucciones es tal como el que se ve abajo.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BITS | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Código de Operación | | | | | MD |  |  |
| OP1(R1)(Operando 1 y Destino) | | | OP2(R2)(Operando 2) | | |  |  |
| Dirección o Dato | | | | | | | |

Donde se puede apreciar en la parte superior los bits, en la primera fila, el Código de Operación de 5 bits desde el 7 hasta el 3 para representar hasta 32 instrucciones, en el bit siguiente el modo de direccionamiento, esto es, con un bit se puede representar dos modos de direccionamiento por cada instrucción, esto se verá a continuación.

En la segunda fila se pueden ver los operandos, se decidió usar 3 bits para cada operando porque para el algoritmo particular implementaremos un cambio en la ruta de datos, teniendo un total de 6 registros de uso general para el programador en vez de 4. Se decidió hacer este cambio debido a que la implementación del algoritmo particular utilizando solo 4 registros implicaría un uso mucho más complejo de los espacios de la memoria para almacenar datos temporales en cada ciclo, de esta forma también sería el algoritmo uno mucho más largo.

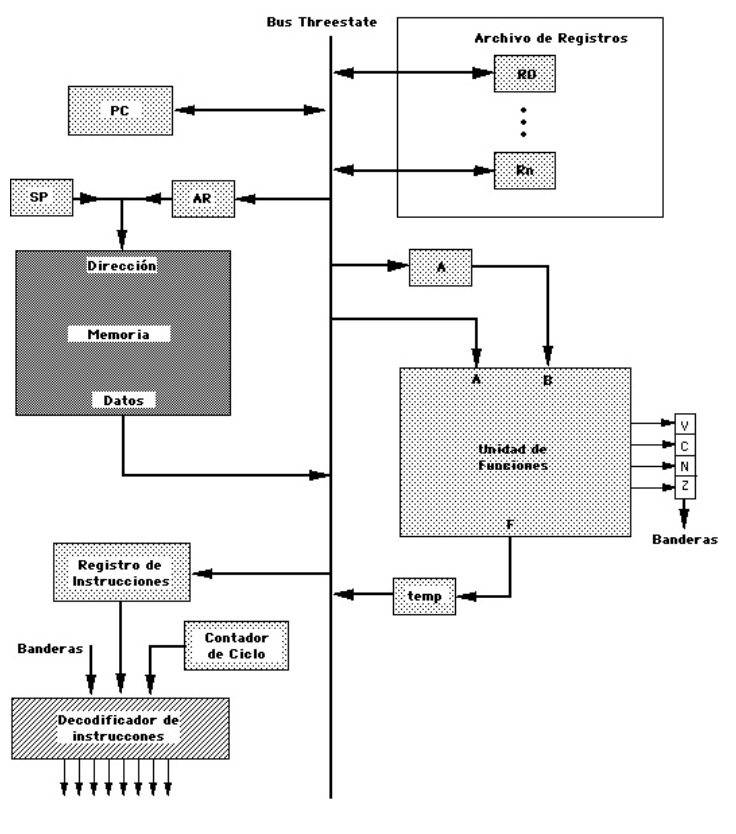
Y por último la Dirección o Dato, dirección o address para los modos de direccionamiento Directo, Indirecto, Relativo o Indexado, y dato para el modo de direccionamiento inmediato. Cabe destacar que si utilizamos el direccionamiento tipo Registro, no hace falta especificar una dirección o dato, y la cantidad de bloques que lee nuestra máquina de control es de solamente dos.

La tabla de las instrucciones elegidas se ve a continuación:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nemotécnico | Significado | Tipo | Código de Operación | Modo de Direccionamiento | |
| 0 | 1 |
| ADD | Suma sin acarreo | Aritmético | 00000 | Registro | Inmediato |
| ADDC | Suma con acarreo | Aritmético | 00001 | Registro | Inmediato |
| CMP | Comparar | Aritmético | 00010 | Registro | Inmediato |
| INC | Incrementar en uno el destino (1) | Aritmético | 00011 | Registro | Directo |
| NOT | Negar | Lógico | 00100 | Registro | Directo |
| AND | AND lógico | Lógico | 00101 | Registro | Inmediato |
| MOVE | Copiar dato a registro | Transferencia | 00110 | Registro | Inmediato |
| BZ | Branch si la bandera es cero | Control | 00111 | Directo | Inmediato |
| RETI | Retorno de interrupción | Control | 01000 | - | - |
| PUSH | Introducir dato a la pila (trabaja con Stack Pointer) | Transferencia | 01001 | Registro | Inmediato |
| POP | Retirar dato de la pila (trabaja con Stack Pointer) | Transferencia | 01010 | Registro | - |
| ST | Copiar el dato de registro a una memoria | Transferencia | 01011 | Indirecto | Directo |
| LD | Copiar el dato de una memoria a un registro | Transferencia | 01100 | Indirecto | Directo |
| CLR | Clear, “cerar” del destino | Transferencia | 01101 | Registro | Reg.Indirecto |
| SHR | Shift Right, desplazamiento de un bit a la derecha del destino | Desplazamiento | 01110 | Registro | Directo |
| SHL | Shift left, desplazamiento de un bit a la izquierda del destino | Desplazamiento | 01111 | Registro | Directo |
| SUB | Resta un número de otro, el resultado es positivo, el minuendo debe ser mayor | Aritmético | 10000 | Registro | Inmediato |
| DEC | Resta dos números, puede dar solo un resultado positivo o cero | Aritmético | 10001 | Registro | Inmediato |
| JMP | Salta a otro lugar de memoria sin preguntar | Control | 10010 | - | Inmediato |

Donde vemos que con un solo bit de Modo de Direccionamiento es suficiente para lo que nos pide el trabajo, también cumplimos con la condición de al menos dos instrucciones de cada tipo independientemente de que se usen o no en nuestro algoritmo particular posteriormente.

**Ruta de Datos**



RB

RC

RAUX

RAUX2

En la ruta de datos vemos el modelo de un bus único de tres estados, al modelo original se le agregaron dos registros extra al archivo de registros, los cuales son registros que el programador puede utilizar, mientras que además se agregaron dos registros bidireccionales RC y RB que son invisibles para el programador pero cumplen un papel vital dentro del desarrollo interno de las instrucciones, estos se colorearon de rojo además del registro A que también resulta invisible para el programador.

También se modificó el bus de datos a la memoria, debe ser bidireccional para que se puedan cargar datos. No es solo salida.

**Determinación de las señales de control**

Para determinar las señales de control hay que analizar la ruta de datos y sus componentes:

**Memoria, R1, R2, R3, R4, RAUX, RAUX2, RA, RB, RC, Banderas, Temp: Registros**

Se encargan de almacenar datos de 8 bits, incluido la memoria la cual es una asociación de 256 registros de 8 bits. El registro de banderas sin embargo solo utiliza 4 bits, porque sólo se necesita de un bit para representar cada bandera en específico, cero, no cero, carry y overflow.

**La función general de cada uno es:**

**R1:** destino y operando primero

**R2:** segundo operando

**R3 y R4:** registros para guardar datos

**RAUX Y RAUX2:** registros extra que el programador puede utilizar

**RA, RB, RC:** registros auxiliares que se utilizan dentro de las micro-instrucciones, invisibles al programador

**Banderas:** de la ALU, se pueden leer (para un Branch) tanto como cargar (según la operación que se realizó y lo modifique)

**Temp:** registro temporal que se utiliza dentro de las micro-instrucciones que trabajan con la ALU, invisible para el programador

**Señales de control:**

**R1, R2, R3, R4, RA, RB, Temp:**

LOAD (LD), CLEAR (CLR), SHIFT RIGHT (SR), SHIFT LEFT (SL), OUTPUT ENABLE (OE) (ESTE ULTIMO ES NECESARIO YA QUE EN EL BUS PRINCIPAL PUEDE HABER SOLO UN DATO A LA VEZ)

**Banderas:**

LOAD (LD), OUTPUT ENABLE (OE)

**Memoria:**

WRITE ENABLE (WE), OUTPUT ENABLE (OE) (MODO POR STACK POINTER Y ADRESS REGISTER)

**Registro de Direcciones, Contador de Programa, Contador de Pilas: Contadores**

INCREASE (INC), DECREASE (DC), LOAD (LD), CLEAR (CLR), OUTPUT ENABLE (OE)

**Determinación de las señales de estado**

La determinación de las señales de estado se logra mediante el análisis de la unidad de control. Estas son el conjunto de instrucciones que tenemos en nuestra cpu y los estados de la misma, los cuales se verán en el ASM más abajo.

**Registro de Instrucciones: Registro**

Es en nuestra máquina de control, la unidad que almacena las instrucciones (Operación y Operandos) sacadas de la memoria. Estas se cargan durante el periodo de búsqueda para luego ejecutarse en el periodo de ejecución.

**Contador de Ciclo: Contador**

Cuenta el estado en que se encuentra todo nuestro sistema, es decir, lleva al tanto en qué punto de ejecución se encuentra nuestra ruta de datos.

**Decodificador de Instrucciones: Decoder**

Es el que se encarga de traducir la información del registro de instrucciones y les da su valor adecuado según el código de operación.

**Señales de estado:**

**ST, Z, ADD, ADDC, CMP, INC, NOT, AND, MOVE, BZ, RETI, PUSH, POP, ST, LD, CLR, SHR, SHL, M.**

**Estados (contador de ciclos):**

**T0, T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15**

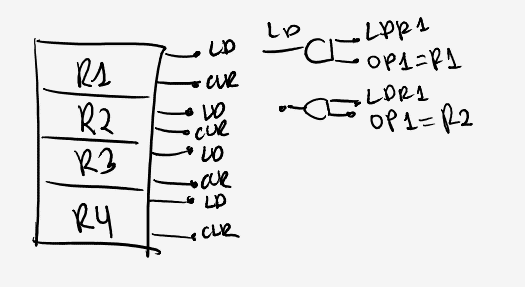
**Diseño del ASM y ASM para el Algoritmo en Particular**

Cuando desarrollamos el ASM R1=OP1 y R2=OP2, esto es porque resultó más fácil la utilización de esta nomenclatura durante el desarrollo.

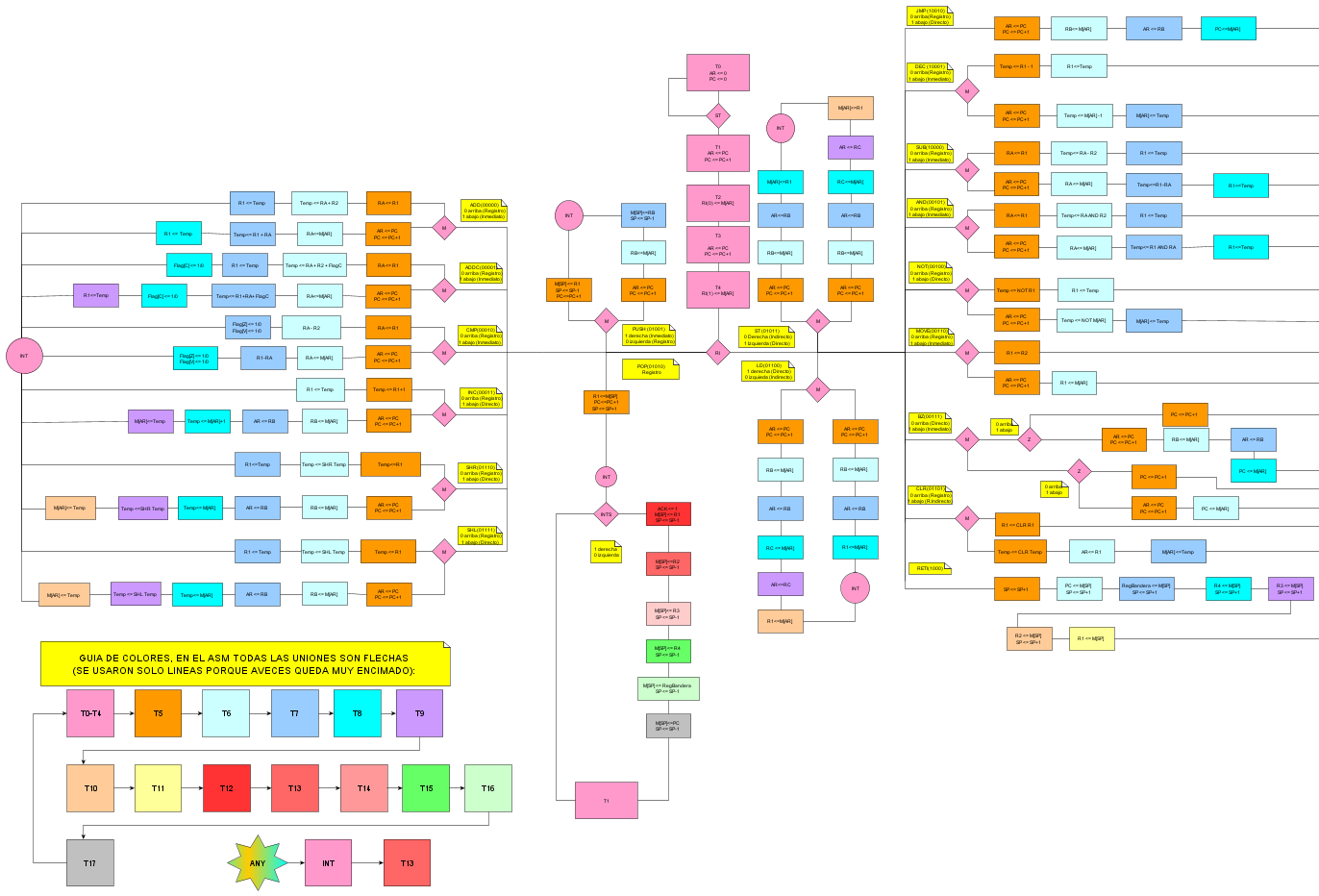
Para la instrucción de retorno de interrupción (RETI) sin embargo, R1 y R2 sí son los registros en sí, no los operadores. Dentro del mismo lazo también aparecen los registros R3 y R4 por lo que debería ser fácil de diferenciar.

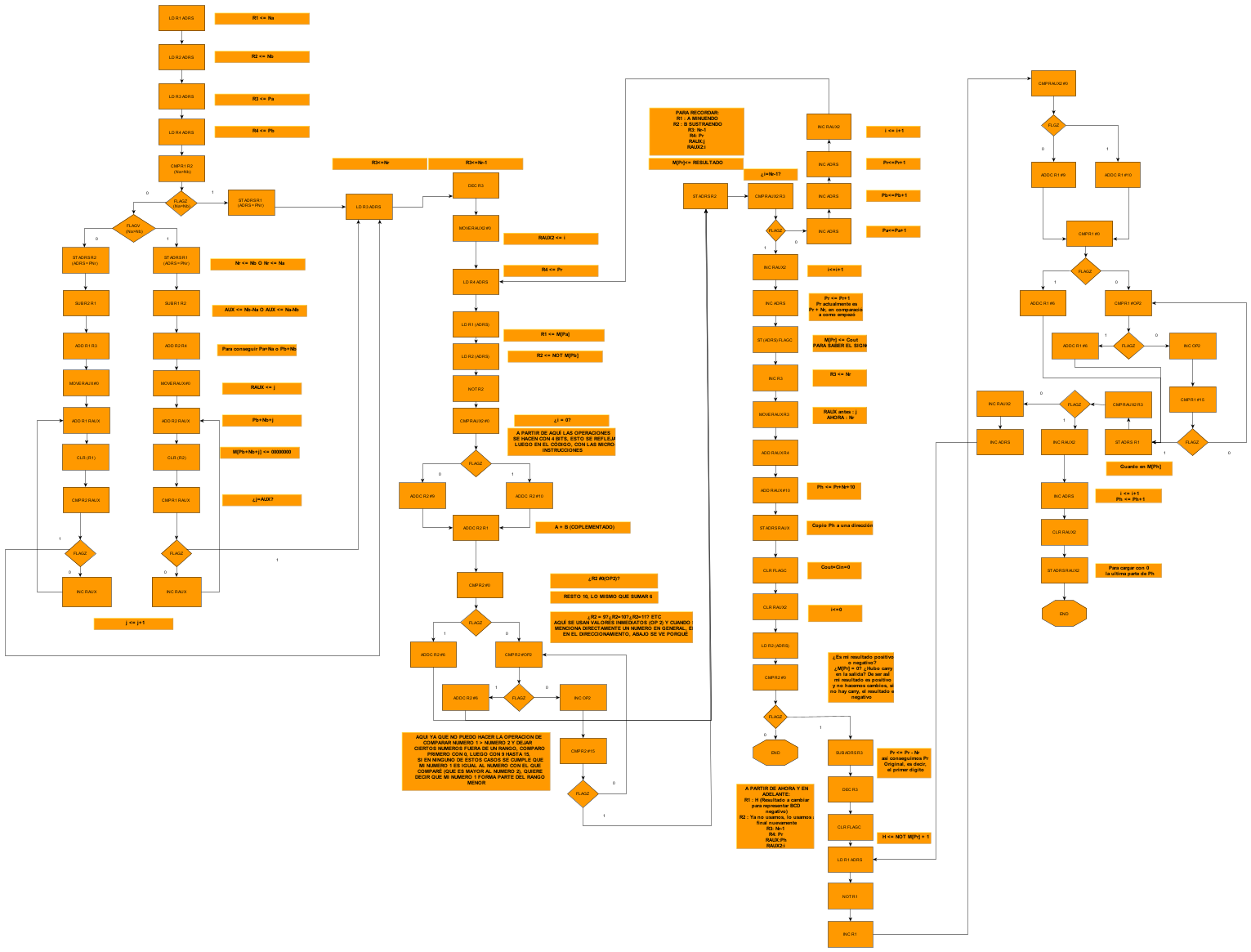
Para el ASM del algoritmo particular, la utilización de cada registro y señal está documentada, se adjunta además el algoritmo original del trabajo anterior para una mejor comprensión.

**Ejemplo de la solución para la selección de señal de control utilizando un registro en específico. Todos los registros tienen las mismas patitas y la señal de control cambia según OP1 y OP2. Lo que siempre se mantiene constante es las señales en el periodo de interrupción y retorno**



Para los siguientes gráficos primero presentamos el ASM de Instrucciones y luego presentamos el ASM del Algoritmo Particular, primero el algoritmo original del TP anterior y luego el ASM en sí con las Instrucciones que desarrollamos en el primer ASM.

****

****

**Matriz de micro instrucciones:**

**Instrucciones que utilizan la ALU:**

**ADD (000), ADDC (001), CMP (010), INC (011), AND (100), NOT (101), SUB (110), DEC (111)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Estado | Instrucción | Micro.Inst (0//1) Según direccionamiento | LDPC | CLRPC | INCPC | OEPC | LDRI | LDAR | CLRAR | OEMEM(AR) | OEMEM(SP) | WEMEM(AR) | WEMEM(SP) | INCSP | DCSP | LDR1 | CLR1 | OER1 | LDR2 | OER2 | LDR3 | OER3 | LDR4 | OER4 | LDRA | OEA | LDRB | OEB | LDRC | OEC | LDTEMP | CLRTEMP | SHRTEMP | SHLRTEMP | OETEMP | FUN0 | FUN1 | FUN2 | LDBandera | OEBandera |
| T0 |  | AR<=0 |  |  |  |  |  |  | T0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  |  | PC<=0 |  | T0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T1 |  | AR<=PC |  |  |  | T1 |  | T1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  |  | PC<=PC+1 |  |  | T1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T2 |  | RI(0)<=M[AR] |  |  |  |  | T2 |  |  | T2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T3 |  | AR<=PC |  |  |  | T3 |  | T3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  |  | PC<=PC+1 |  |  | T3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T4 |  | RI(1)<=M[AR] |  |  |  |  | T4 |  |  | T4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T5 | ADD | RA <= R1//AR <=PC,PC<=PC+1 |  |  | M.T5 | M.T5 |  | M.T5 |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | ADDC | RA<=R1//AR<=PC, PC<=PC+1 |  |  | M.T5 | M.T5 |  | M.T5 |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | CMP | RA<=R1//AR<=PC, PC<=PC+1 |  |  | M.T5 | M.T5 |  | M.T5 |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | INC | Temp<=R1+1//AR<=PC,PC<=PC+1 |  |  | M.T5 | M.T5 |  | M.T5 |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  | /M.T5 | /M.T5 | M.T5 | X | X |
|  | NOT | Temp <= NOT R1//AR<=PC,PC<=PC+1 |  |  | M.T5 | M.T5 |  | M.T5 |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  | /M.T5 | M.T5 | /M.T5 | X | X |
|  | AND | RA <= R1//AR <=PC,PC<=PC+1 |  |  | M.T5 | M.T5 |  | M.T5 |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | MOVE | R1<=R2//AR<=PC,PC<=PC+1 |  |  | M.T5 | M.T5 |  | M.T5 |  |  |  |  |  |  |  | /M.T5 |  |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | BZ | **(Z (0//1)):**PC<=PC+1//AR<=PC,PC<=PC+1 |  |  | T5 | Z.T5 |  | Z.T5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | RETI | SP<=SP+1 |  |  |  |  |  |  |  |  |  |  |  | T5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | PUSH | M[SP]<=R1,SP<=SP-1, PC<=PC+1//AR<=PC, PC<=PC+1 |  |  | T5 | M.T5 |  | M.T5 |  |  |  |  | /M.T5 |  | /M.T5 |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | POP | M[SP]<=R1,SP<=SP+1, PC<=PC+1 |  |  | T5 | T5 |  |  |  |  |  |  | T5 | T5 |  |  |  | T5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | ST | AR<=PC,PC<=PC+1 |  |  | T5 | T5 |  | T5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | LD | AR<=PC,PC<=PC+1 |  |  | T5 | T5 |  | T5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | CLR | R1<=CLR R1// Temp<=CLR Temp |  |  |  |  |  |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T5 |  |  |  | X | X | X | X | X |
|  | SHR | Temp<=R1// AR<=PC,PC<=PC+1 |  |  | M.T5 | M.T5 |  | M.T5 |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  | X | X | X | X | X |
|  | SHL | Temp<=R1//AR<=PC,PC<=PC+1 |  |  | M.T5 | M.T5 |  | M.T5 |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  | X | X | X | X | X |
|  | SUB | RA<=R1//AR<=PC, PC<=PC+1 |  |  | M.T5 | M.T5 |  | M.T5 |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | DEC | Temp <= R1-1//AR<=PC,PC<=PC+1 |  |  | M.T5 | M.T5 |  | M.T5 |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  |  |  |  |  |  |  |  |  | /M.T5 |  |  |  |  | /M.T5. | /M.T5 | /M.T5 | X | X |
|  | JMP | AR<=PC, PC<=PC+1 |  |  | T5 | T5 |  | T5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T6 | ADD | Temp<=RA+R2//RA<=M[AR] |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  |  | /M.T6 |  |  |  |  | M.T6 | /M.T6 |  |  |  |  | /M.T6 |  |  |  |  | M.T6 | M.T6 | M.T6 | X | X |
|  | ADDC | Temp<=RA+R2+FLAGC//RA<=M[AR] |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  |  | /M.T6 |  |  |  |  | M.T6 | /M.T6 |  |  |  |  | /M.T6 |  |  |  |  | /M.T6 | M.T6 | M.T6 | X | /M.T6 |
|  | CMP | RA-R2//RA<=M[AR] |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  |  | /M.T6 |  |  |  |  | M.T6 | /M.T6 | M.T6 |  |  |  |  |  |  |  |  | M.T6 | /M.T6 | M.T6 | X | X |
|  | INC | R1<=Temp//RB<=M[AR] |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  | /M.T6 |  |  |  |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  | /M.T6 | X | X | X | X | X |
|  | NOT | R1<=Temp//Temp<=NOT M[AR] |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  | /M.T6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | /M.T6 |  |  |  | /M.T6 | M.T6 | /M.T6 | M.T6 | X | X |
|  | AND | Temp<=RA AND R2//RA <= M[AR] |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  |  | /M.T6 |  |  |  |  | M.T6 | /M.T6 |  |  |  |  | /M.T6 |  |  |  |  | M.T6 | M.T6 | /M.T6 | X | X |
|  | MOVE | NULL//R1<=M[AR] |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | BZ | **(Z (0//1)):** NULL//RB<=M[AR] + PC<=M[AR] | M.T6.Z |  |  |  |  |  |  | /M.T6.Z |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | /M.T6.Z |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | RETI | PC<=M[SP], SP<=SP+1 | T6 |  |  |  |  |  |  |  | T6 |  |  | T6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | PUSH | NULL//RB<=M[AR] |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | ST | RB<=M[AR] |  |  |  |  |  |  |  | T6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T6 |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | LD | RB<=M[AR] |  |  |  |  |  |  |  | T6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T6 |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | CLR | NULL//AR<= R1 |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | SHR | Temp<=SHR Temp// RB<=M[AR] |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  | /M.T6 |  |  | X | X | X | X | X |
|  | SHL | Temp<=SHL Temp// RB<=M[AR] |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  | /M.T6 |  | X | X | X | X | X |
|  | SUB | Temp <= RA - R2 // RA<=M[AR] |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  |  |  |  |  | /M.T6 |  |  |  |  | M.T6 | /M.T6 |  |  |  |  | /M.T6 |  |  |  |  | M.T6 | /M.T6 | /M.T6 | X | X |
|  | DEC | R1 <= Temp // Temp <= M[AR]-1 |  |  |  |  |  |  |  | M.T6 |  |  |  |  |  | /M.T6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T6 |  |  |  | /M.T6 | M.T6 | M.T6 | M.T6 | X | X |
|  | JMP | RB <=M[AR] |  |  |  |  |  |  |  | T6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T6 |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T7 | ADD | R1<=Temp//Temp<=R1+RA |  |  |  |  |  |  |  |  |  |  |  |  |  | /M.T7 |  | M.T7 |  |  |  |  |  |  |  | M.T7 |  |  |  |  | M.T7 |  |  |  | /M.T7 | /M.T7 | /M.T7 | /M.T7 | X | X |
|  | ADDC | R1<=Temp//Temp<=R1+RA+FLAGC |  |  |  |  |  |  |  |  |  |  |  |  |  | /M.T7 |  | M.T7 |  |  |  |  |  |  |  | M.T7 |  |  |  |  | M.T7 |  |  |  | /M.T7 | M.T7 | /M.T7 | /M.T7 | X | M.T7 |
|  | CMP | FLAGZ, FLAGV//R1-RA |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T7 |  |  |  |  |  |  |  | M.T7 |  |  |  |  |  |  |  |  |  | /M.T7 | M.T7 | /M.T7 | /M.T7 | X |
|  | INC | NULL//AR<=RB |  |  |  |  |  | M.T7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T7 |  |  |  |  |  |  |  | X | X | X | X | X |
|  | NOT | NULL//M[AR]<=Temp |  |  |  |  |  |  |  |  |  | M.T7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T7 | X | X | X | X | X |
|  | AND | R1<=Temp//Temp<=R1 AND RA |  |  |  |  |  |  |  |  |  |  |  |  |  | /M.T7 |  |  |  | M.T7 |  |  |  |  |  | M.T7 |  |  |  |  | M.T7 |  |  |  | /M.T7 | /M.T7 | /M.T7 | M.T7 | X | X |
|  | BZ | **(Z (0//1)):** NULL//AR<=RB |  |  |  |  |  | /M.T7.Z |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | /M.T7.Z |  |  |  |  |  |  |  | X | X | X | X | X |
|  | RETI | BANDERA <= M[SP],SP<=SP+1 |  |  |  |  |  |  |  |  | T7 |  |  | T7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | T7 | X |
|  | PUSH | NULL//M[SP]<=RB, SP<=SP-1 |  |  |  |  |  |  |  |  |  |  | M.T7 | M.T7 |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T7 |  |  |  |  |  |  |  | X | X | X | X | X |
|  | ST | AR<=RB |  |  |  |  |  | T7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T7 |  |  |  |  |  |  |  | X | X | X | X | X |
|  | LD | AR<=RB |  |  |  |  |  | T7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T7 |  |  |  |  |  |  |  | X | X | X | X | X |
|  | CLR | NULL//M[AR]<=Temp |  |  |  |  |  |  |  |  |  | M.T7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T7 | X | X | X | X | X |
|  | SHR | R1<=Temp//AR<=RB |  |  |  |  |  | M.T7 |  |  |  |  |  |  |  | /M.T7 |  |  |  |  |  |  |  |  |  |  |  | M.T7 |  |  |  |  |  |  | /M.T7 | X | X | X | X | X |
|  | SHL | R1<=Temp //AR<=RB |  |  |  |  |  | M.T7 |  |  |  |  |  |  |  | /M.T7 |  |  |  |  |  |  |  |  |  |  |  | M.T7 |  |  |  |  |  |  | /M.T7 | X | X | X | X | X |
|  | SUB | R1<=Temp//Temp<=R1-RA |  |  |  |  |  |  |  |  |  |  |  |  |  | /M.T7 |  | M.T7 |  |  |  |  |  |  |  | M.T7 |  |  |  |  | M.T7 |  |  |  | /M.T7 | /M.T7 | M.T7 | M.T7 | X | X |
|  | DEC | NULL//M[AR]<=Temp |  |  |  |  |  |  |  |  |  | M.T7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T7 | X | X | X | X | X |
|  | JMP | AR <= RB |  |  |  |  |  | T7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T7 |  |  |  |  |  |  |  | X | X | X | X | X |
| T8 | ADD | NULL//R1<=Temp |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T8 | X | X | X | X | X |
|  | ADDC | FLAGC |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | T8 | X |
|  | CMP | NULL//FLAGZ,FLAGV |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | M.T8 | X |
|  | INC | NULL//Temp<=M[AR]+1 |  |  |  |  |  |  |  | M.T8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T8 | M.T8 | M.T8 | /M.T8 | X | X |
|  | AND | NULL//R1<=Temp |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T8 | X | X | X | X | X |
|  | BZ | **(Z (0//1)):**NULL//PC<=M[AR] | /M.T8.Z |  |  |  |  |  |  | /M.T8.Z |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | RETI | R4<=M[SP], SP<=SP+1 |  |  |  |  |  |  |  |  | T8 |  |  | T8 |  |  |  |  |  |  |  |  | T8 |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | ST | RC<=M[AR]//M[AR]<=R1 |  |  |  |  |  |  |  | /M.T8 |  | M.T8 |  |  |  |  |  | M.T8 |  |  |  |  |  |  |  |  |  |  | /M.T8 |  |  |  |  |  |  | X | X | X | X | X |
|  | LD | RC<=M[AR]//R1<=M[AR] |  |  |  |  |  |  |  | T8 |  |  |  |  |  | M.T8 |  |  |  |  |  |  |  |  |  |  |  |  | /M.T8 |  |  |  |  |  |  | X | X | X | X | X |
|  | SHR | NULL//Temp<=M[AR] |  |  |  |  |  |  |  | M.T8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T8 |  |  |  |  | X | X | X | X | X |
|  | SHL | NULL//Temp<=M[AR] |  |  |  |  |  |  |  | M.T8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T8 |  |  |  |  | X | X | X | X | X |
|  | SUB | NULL//R1<=Temp |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T8 | X | X | X | X | X |
|  | JMP | PC<=M[AR] | T8 |  |  |  |  |  |  | T8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T9 | ADDC | NULL//R1<=Temp |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T9 | X | X | X | X | X |
|  | INC | NULL//M[AR]<=Temp |  |  |  |  |  |  |  |  |  | M.T9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T9 | X | X | X | X | X |
|  | RETI | R3<=M[SP],SP<=SP+1 |  |  |  |  |  |  |  |  | T9 |  |  | T9 |  |  |  |  |  |  | T9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | ST | AR<=RC//NULL |  |  |  |  |  | /M.T9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | /M.T9 |  |  |  |  |  | X | X | X | X | X |
|  | LD | AR<=RC//NULL |  |  |  |  |  | /M.T9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | /M.T9 |  |  |  |  |  | X | X | X | X | X |
|  | SHR | NULL//Temp<=SHR Temp |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T9 |  |  | X | X | X | X | X |
|  | SHL | NULL//Temp<=SHL Temp |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T9 |  | X | X | X | X | X |
| T10 | SHR | NULL//M[AR]<=Temp |  |  |  |  |  |  |  |  |  | M.T10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T10 | X | X | X | X | X |
|  | SHL | NULL//M[AR]<=Temp |  |  |  |  |  |  |  |  |  | M.T10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M.T10 | X | X | X | X | X |
|  | RETI | R2<=M[SP],SP<=SP+1 |  |  |  |  |  |  |  |  | T10 |  |  | T10 |  |  |  |  | T10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | ST | M[AR]<=R1//NULL |  |  |  |  |  |  |  |  |  | /M.T10 |  |  |  |  |  | /M.T10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
|  | LD | R1<=M[AR]//NULL |  |  |  |  |  |  |  | /M.T10 |  |  |  |  |  | /M.T10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T11 | RETI | R1<=M[SP] |  |  |  |  |  |  |  |  | T11 |  |  |  |  | T11 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T12 |  | **(INTS: 0//1):**NULL//M[SP]<=R1, SP<=SP-1 |  |  |  |  |  |  |  |  |  |  | INTS.T12 |  | INTS.T12 |  |  | INTS.T12 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T13 |  | **(INTS: 0//1):**NULL//M[SP]<=R2, SP<=SP-1 |  |  |  |  |  |  |  |  |  |  | INTS.T13 |  | INTS.T13 |  |  |  |  | INTS.T13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T14 |  | **(INTS: 0//1):**NULL//M[SP]<=R3, SP<=SP-1 |  |  |  |  |  |  |  |  |  |  | INTS.T14 |  | INTS.T14 |  |  |  |  |  |  | INTS.T14 |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T15 |  | **(INTS: 0//1):**NULL//M[SP]<=R4, SP<=SP-1 |  |  |  |  |  |  |  |  |  |  | INTS.T15 |  | INTS.T15 |  |  |  |  |  |  |  |  | INTS.T15 |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |
| T16 |  | **(INTS: 0//1):**NULL//M[SP]<=BANDERA, SP<=SP-1 |  |  |  |  |  |  |  |  |  |  | INTS.T16 |  | INTS.T16 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | INTS.T16 |
| T17 |  | **(INTS: 0//1):**NULL//M[SP]<=PC, SP<=SP-1 |  |  |  | INTS.T17 |  |  |  |  |  |  | INTS.T17 |  | INTS.T17 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X |

**Ecuaciones de control**

**Señales de control sacadas de la tabla de intersección:**

|  |  |  |
| --- | --- | --- |
| **Componente** | **Señal de control** | **Ecuación** |
| **PC** | **LDPC** | **T6.RETI.+/M.BZ.T8.Z+T8.JMP+M.T6.Z.BZ** |
| **PC** | **CLRPC** | **T0** |
| **PC** | **INCPC** | **T1+T3+M.T5(ADD+ADDC+CMP+INC**  **+NOT+AND+MOVE+SHR+SHL+SUB+**  **DEC)+T5(BZ+PUSH+POP+ST+LD+JMP)+** |
| **PC** | **OEPC** | **T1+T3+M.T5(ADD+ADDC+CMP+INC**  **+NOT+AND+MOVE+PUSH+SHR+SHL+SUB+**  **DEC)+T5(ST+LD+JMP+POP)+BZ.T5.Z+INTS.T17** |
| **RI** | **LDRI** | **T2+T4** |
| **AR** | **LDAR** | **T1+T3+M.T5(ADD+ADDC+CMP+INC**  **+NOT+AND+MOVE+ PUSH+SHR+SHL+SUB+**  **DEC)+T5(ST+LD+JMP)+BZ.T5.Z+M.CLR.T6+M.T7(INC+**  **SHR+SHL)+T7(ST+LD+JMP)+/M.BZ.T7.Z+/M.T9(ST+LD)** |
| **AR** | **CLRAR** | **T0** |
| **MEMORIA** | **OEMEM(AR)** | **T2+T4+M.T6(ADD+ADDC+CMP+INC+NOT+AND+**  **MOVE+PUSH+SHR+SHL+SUB+DEC)+T6(ST+LD+JMP)+**  **BZ.T6.Z./M +M.T8(INC+SHR+SHL)+ T8.(JMP+LD)**  **/M.T8.ST+/M.BZ.T8.Z+/M.T10.LD** |
| **MEMORIA** | **OEMEM(SP)** | **RETI(T6+T7+T8+T9+T10+T11)** |
| **MEMORIA** | **WEMEM(AR)** | **M.T7(NOT+CLR+DEC)+M.T8.ST+M.T9.INC+M.T10(SHR**  **+SHL)+/M.T10.ST** |
| **MEMORIA** | **WMEM(SP)** | **/M.T5.PUSH+T5.POP+M.T7.PUSH+INTS(T12+T13+T14+**  **T15+T16+T17)** |
| **SP** | **INCSP** | **RETI.(T5+T6+T7+T8+T9+T10)+POP.T5 +M.T7.PUSH** |
| **SP** | **DCSP** | **/M.T5.PUSH+INTS(T12+T13+T14+T15+T16+T17)** |
| **R1** | **LDR1** | **/M.T5.MOVE+/M.T6(INC+NOT+DEC)+M.T6.MOVE**  **+/M.T7(ADD+ADDC+AND+SHR+SHL+SUB)+**  **M.T8(ADD+AND+LD+SUB)+M.T9(ADDC)+/M.T10.LD**  **+RETI.T11** |
| **R1** | **CLR1** | **/M.CLR.T5** |
| **R1** | **OER1** | **/M.T5(ADD+ADDC+INC+CMP+NOT+AND+PUSH+**  **SHR+SHL+SUB+DEC)+T5.POP+M.T6.CLR+M.T7(ADD+ADDC+CMP+SUB)**  **+M.T8.ST+/M.T10.ST+INTS.T12** |
| **R2** | **LDR2** | **T10.RETI** |
| **R2** | **OER2** | **/M.T5.MOVE+/M.T6(ADD+ADDC+CMP+AND+SUB)**  **+M.T7.AND+INTS.T13** |
| **R3** | **LDR3** | **T9.RETI** |
| **R3** | **OER3** | **INTS.T14** |
| **R4** | **LDR4** | **T8.RETI** |
| **R4** | **OER4** | **INTS.T15** |
| **RA** | **LDRA** | **/M.T5 (ADD+ADDC+CMP+AND+SUB)+M.T6(ADD+ADDC**  **+CMP+AND+SUB)** |
| **RA** | **OEA** | **/M.T6(ADD+ADDC**  **+CMP+AND+SUB)+M.T7(ADD+ADDC+CMP+AND+SUB)** |
| **RB** | **LDRB** | **M.T6(CMP+INC+MOVE+PUSH+SHR+SHL)+T6.(ST+LD+JMP)+BZ./M.T6.Z** |
| **RB** | **OERB** | **M.T7(INC+PUSH+SHR+SHL)+T7(ST+LD+JMP)+/M.BZ.T7.Z** |
| **RC** | **LDRC** | **/M.T8(ST+LD)** |
| **RC** | **OERC** | **/M.T9(ST+LD)** |
| **TEMP** | **LDTEMP** | **/M.T5(INC+NOT+SHR+SHL+DEC) + /M.T6(ADD+ADDC+NOT+AND+SUB)+M.T6(DEC)+**  **M.T7(ADD+ADDC+AND+SUB)+M.T8(SHR+SHL)** |
| **TEMP** | **CLRTEMP** | **M.T5.CLR** |
| **TEMP** | **SHRTEMP** | **TEMP(/M.T6+M.T9)** |
| **TEMP** | **SHLTEMP** | **TEMP(/M.T6+M.T9)** |
| **TEMP** | **OETEMP** | **/M.T6(INC+NOT+DEC)+/M.T7(ADD+ADDC+**  **AND+SHR+SHL+SUB)+M.T7(NOT+CLR+DEC)+**  **M.T8(ADD+INC+AND+SUB)+M.T9(ADDC+INC)+**  **M.T10(SHR+SHL)** |
| **ALU** | **FUN2** | **M.T5.INC+/M.T5.(NOT+DEC)+M.T6.(ADD+ADDC+CMP+NOT+DEC)+**  **/M.T6(AND+SUB)+/M.T7(ADD+ADDC+CMP)+M.T7.(AND+SUB)+/M.T8.INC** |
| **ALU** | **FUN1** | **/M.T5(INC+DEC)+M.T5.(NOT)+M.T6.(ADD+ADDC+AND+DEC)+/M.T6(CMP+NOT+SUB)+/M.T7(ADD+ADDC+AND)+M.T7(CMP+SUB)+M.T8.INC** |
| **ALU** | **FUN0** | **/M.T5.(INC+NOTD+DEC)+M.T6(ADD+CMP+NOT+AND+SUB+DEC)+/M.T6.ADDC+/M.T7.(ADD+CMP+AND+SUB)+M.T7.AADDC+**  **M.T8.INC** |
| **BANDERAS** | **LDBANDERA** | **/M.T7.CMP+T7.RETI+M.T8.CMP+T8.ADDC** |
| **BANDERAS** | **OEBANDERA** | **/M.T6.ADDC+M.T7.ADDC+INTS.T16** |

**Saltos de estados al estado de interrupción (MODO/BANDERA):**

|  |  |
| --- | --- |
| **T5** | **BZ (0/0 Y 1/0), CLR(0), PUSH (0), MOVE (0) , POP** |
| **T6** | **INC (0), DEC (0), NOT (0), MOVE(1), BZ(1/1)** |
| **T7** | **ADD(0), CMP(0), SHR (0), SHL(0), PUSH (1), DEC(1), SUB(0), AND(0), NOT(1), CLR(1)** |
| **T8** | **ADD(1), ADDC(0), CMP(1), LD(1), ST(1), SUB (1), AND (1), BZ(0/1), JMP** |
| **T9** | **ADDC(1), INC(1)** |
| **T10** | **SHR(1), SHL(1), ST(0), LD(0)** |
| **T11** | **RETI** |

**Datos para el algoritmo particular.**

**Qué instrucciones se usan en nuestro algoritmo particular:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N° | Instrucción | Código de Operación | Direccionamiento (0-1) | |
| 0 | ADD | 0000 | Registro | Inmediato |
| 1 | ADDC | 0001 | Registro | Inmediato |
| 2 | CMP | 0010 | Registro | Inmediato |
| 12 | LD | 1100 | Indirecto | Directo |
| 11 | ST | 1011 | Indirecto | Directo |
| 7 | BZ | 0111 | Registro | Inmediato |
| 3 | INC | 0011 | Registro | Directo |
| 4 | NOT | 0100 | Registro | Directo |
| 13 | CLR | 1101 | Registro | Reg.Indirecto |
| 16 | SUB | 10000 | Registro | Inmediato |
| 17 | DEC | 10001 | Registro | Directo |
| 6 | MOVE | 0110 | Registro | Inmediato |
| 18 | JMP | 10010 | - | Inmediato |
| 8 | RETI E INTS (porque hay que implementar máquina de interrupción) | 01000 | - | - |

Entonces eliminamos SHR, SHL, AND, PUSH, POP de la tabla anterior para tener nuestras señales de control para el algoritmo en particular.

|  |  |  |
| --- | --- | --- |
| **Componente** | **Señal de control** | **Ecuación** |
| **PC** | **LDPC** | **T6.RETI+/M.BZ.T8.Z+T8.JMP+M.T6.Z.BZ** |
| **PC** | **CLRPC** | **T0** |
| **PC** | **INCPC** | **T1+T3+M.T5(ADD+ADDC+CMP+INC**  **+NOT +MOVE +SUB+**  **DEC)+T5(BZ+ST+LD+JMP)** |
| **PC** | **OEPC** | **T1+T3+M.T5(ADD+ADDC+CMP+INC**  **+NOT +MOVE+SUB+**  **DEC)+T5(ST+LD+JMP+POP)+BZ.T5.Z+INTS.17** |
| **RI** | **LDRI** | **T2+T4** |
| **AR** | **LDAR** | **T1+T3+M.T5(ADD+ADDC+CMP+INC**  **+NOT +MOVE+SUB+**  **DEC)+T5(ST+LD+JMP)+BZ.T5.Z+M.CLR.T6+M.T7(INC**  **)+T7(ST+LD+JMP)+/M.BZ.T7.Z+/M.T9(ST+LD)** |
| **AR** | **CLRAR** | **T0** |
| **MEMORIA** | **OEMEM(AR)** | **T2+T4+M.T6(ADD+ADDC+CMP+INC+NOT +**  **MOVE+SUB+DEC)+T6(ST+LD+JMP)+**  **BZ.T6.Z./M +M.T8(INC)+ T8.(JMP+LD)+**  **/M.T8.ST+/M.BZ.T8.Z+/M.T10.LD** |
| **MEMORIA** | **OEMEM(SP)** | **RETI(T6+T7+T8+T9+T10+T11)** |
| **MEMORIA** | **WEMEM(AR)** | **M.T7(NOT+CLR+DEC)+M.T8.ST+M.T9.INC+**  **+/M.T10.ST** |
| **MEMORIA** | **WMEM(SP)** | **INTS.(T12+T13+T14+T15+T16+T17)** |
| **SP** | **INCSP** | **RETI.(T5+T6+T7+T8+T9+T10)** |
| **SP** | **DCSP** | **INTS.(T12+T13+T14+T15+T16+T17)** |
| **R1** | **LDR1** | **/M.T5.MOVE+/M.T6(INC+NOT+DEC)+M.T6.MOVE**  **+/M.T7(ADD+ADDC+SUB)+**  **M.T8(ADD +LD+SUB)+M.T9(ADDC)+/M.T10.LD+RETI.T11** |
| **R1** | **CLR1** | **/M.CLR.T5** |
| **R1** | **OER1** | **/M.T5(ADD+ADDC+INC+CMP+NOT+**  **SUB+DEC) +M.T6.CLR+M.T7(ADD+ADDC+CMP+SUB)**  **+M.T8.ST+/M.T10.ST+INTS.T12** |
| **R2** | **LDR2** | **T10.RETI** |
| **R2** | **OER2** | **/M.T5.MOVE+/M.T6(ADD+ADDC+CMP +SUB)+**  **INTS.T13** |
| **R3** | **LDR3** | **T9.RETI** |
| **R3** | **OER3** | **INTS.T14** |
| **R4** | **LDR4** | **T8.RETI** |
| **R4** | **OER4** | **INTS.T15** |
| **RA** | **LDRA** | **/M.T5 (ADD+ADDC+CMP +SUB)+M.T6(ADD+ADDC**  **+CMP +SUB)** |
| **RA** | **OEA** | **/M.T6(ADD+ADDC**  **+CMP +SUB)+M.T7(ADD+ADDC+CMP +SUB)** |
| **RB** | **LDRB** | **M.T6(CMP+INC+MOVE)+T6.(ST+LD+JMP)+BZ./M.T6.Z** |
| **RB** | **OERB** | **M.T7(INC)+T7(ST+LD+JMP)+/M.BZ.T7.Z** |
| **RC** | **LDRC** | **/M.T8(ST+LD)** |
| **RC** | **OERC** | **/M.T9(ST+LD)** |
| **TEMP** | **LDTEMP** | **/M.T5(INC+NOT+DEC) + /M.T6(ADD+ADDC+NOT +SUB)+M.T6(DEC)+**  **M.T7(ADD+ADDC +SUB)** |
| **TEMP** | **CLRTEMP** | **M.T5.CLR** |
| **TEMP** | **SHRTEMP** | **-** |
| **TEMP** | **SHLTEMP** | **-** |
| **TEMP** | **OETEMP** | **/M.T6(INC+NOT+DEC)+/M.T7(ADD+ADDC+SUB)+**  **M.T7(NOT+CLR+DEC)+**  **M.T8(ADD+INC+SUB)+M.T9(ADDC+INC)** |
| **ALU** | **FUN2** | **M.T5.INC+/M.T5.(NOT+DEC)+M.T6.(ADD+ADDC+CMP+NOT+DEC)+**  **/M.T6(SUB)+/M.T7(ADD+ADDC+CMP)+M.T7.( SUB)+/M.T8.INC** |
| **ALU** | **FUN1** | **/M.T5(INC+DEC)+M.T5.(NOT)+M.T6.(ADD+ADDC +DEC)+/M.T6(CMP+NOT+SUB)+/M.T7(ADD+ADDC)+M.T7(CMP+SUB)+M.T8.INC** |
| **ALU** | **FUN0** | **/M.T5.(INC+NOT+DEC)+M.T6(ADD+CMP+NOT +SUB+DEC)+/M.T6.ADDC+/M.T7.(ADD+CMP +SUB)+M.T7.ADDC+**  **M.T8.INC** |
| **BANDERAS** | **LDBANDERA** | **/M.T7.CMP+M.T8.CMP+T8.ADDC+T7.RETI** |
| **BANDERAS** | **OEBANDERA** | **/M.T6.ADDC+M.T7.ADDC+INTS.T16** |

|  |  |
| --- | --- |
| **T5** | **BZ (0/0 Y 1/0), CLR(0), MOVE (0)** |
| **T6** | **INC (0), DEC (0), NOT (0), MOVE(1), BZ(1/1)** |
| **T7** | **ADD(0), CMP(0), DEC(1), SUB(0), NOT(1), CLR(1)** |
| **T8** | **ADD(1), ADDC(0), CMP(1), LD(1), ST(1), SUB (1), BZ(0/1), JMP** |
| **T9** | **ADDC(1), INC(1)** |
| **T10** | **ST(0), LD(0)** |
| **T11** | **RETI** |

Y los saltos son directamente al estado inicial para el proceso de búsqueda T1

**Progresión de nuestros registros dentro del algoritmo particular:**

|  |  |
| --- | --- |
| R1 | A (minuendo) => H (resultado negativo a BCD) (YA NO LLEGAMOS A USAR) |
| R2 | B (sustraendo) => 0000 (para un último cambio) |
| R3 | Nr-1 => Nr => Nr-1 |
| R4 | Pr |
| RAUX | j => Nr => Nr + Pr => Nr + Pr + 10 (Ph) (YA NO LLEGAMOS A USAR) |
| RAUX2 | i |

**Máquina de Control**

**Representación en Circuito**

****

Máquina de Control

Para nuestra máquina de control, vemos a la izquierda y abajo, las entradas a nuestro registro de instrucciones, este se conecta al bus principal de nuestra RD, este se encarga de pasar esta información a decodificadores que transforman esta entrada en la señal de Instrucción.

En la parte posterior se puede apreciar el contador de ciclos, que se encarga de dictar en qué estado se encuentra nuestra Instrucción, también se ve cómo se resetea al cumplirse que en el ciclo no hubo ninguna señal de interrupción, para volver al ciclo de búsqueda, hay que mencionar **sin embargo**, el CLEAR va directo al estado T0, en nuestro diseño original este es antes de comenzar el ciclo de búsqueda, por lo que debería de ser T1 en realidad, como se aprecia en nuestro ASM. En el circuito esto puede cambiarse simplemente usando un multiplexor que elija entre TINT y T1 para cargar en el contador y LOAD ocurre cuando se cumple la condición de CLEAR para ir al estado T1 o alguna condición que le haga saltar al estado TINT. No se utilizó este método porque complica mucho el diseño y queda muy encimado, pero siempre es importante tener en cuenta cómo funciona.

Por otra parte también debemos de definir cuando ocurre un salto de estado, para estados que van del actual a uno mucho superior en vez de incrementar naturalmente, esto se puede ver a la izquierda en el ejemplo de salto T5 -> TINT.

ESTAN JUNTOS PARA RECONOCER 5 BITS CÓDIGO DE OPERACIÓN

Por último abajo a la derecha, vemos algunas señales de control y como estas dependen de las señales que se procesan dentro de la Máquina de Control, también se puede ver en el ejemplo de salto. Aquí las señales de estado son la Instrucción, el Modo de Direccionamiento, INTS y las Banderas, pues todas ellas trabajan para modelar internamente la Máquina de Control, y vinieron de la Ruta de Datos a través del Bus de Datos.

Solo se muestran algunos ejemplos de saltos y señales de control puesto que la Máquina de Control que veremos a continuación es la que se usará para el modelo estructurado, en base a las Instrucciones que necesita nuestro algoritmo particular únicamente.



Algunas Señales de Control

Necesitamos para dictar qué registros del archivo de registros utilizarán la configuración de Operador 1,2,3 Y 4.

Para obtener las señales de control correspondientes

EJEMPLO SALTO T5 -> TINT

ESTAN JUNTOS PARA RECONOCER 5 BITS CÓDIGO DE OPERACIÓN



Máquina de Control

Aquí vemos a la izquierda, la máquina de control ajustada a las señales de control dependiente únicamente de las instrucciones requeridas por el algoritmo en particular, a la izquierda se ven todos los combinatorios que habilitan el salto al estado de interrupción. Así mismo hay que resaltar nuevamente que el clear es realmente salto al estado T1 no T0.

Abajo, vemos las señales de control para cada componente en particular.

PC

AR

Señales de Control





R3

R4

RI

SP

RC

Memoria



TEMP

BANDERA

BANDERA



R2

R1



RA

RB

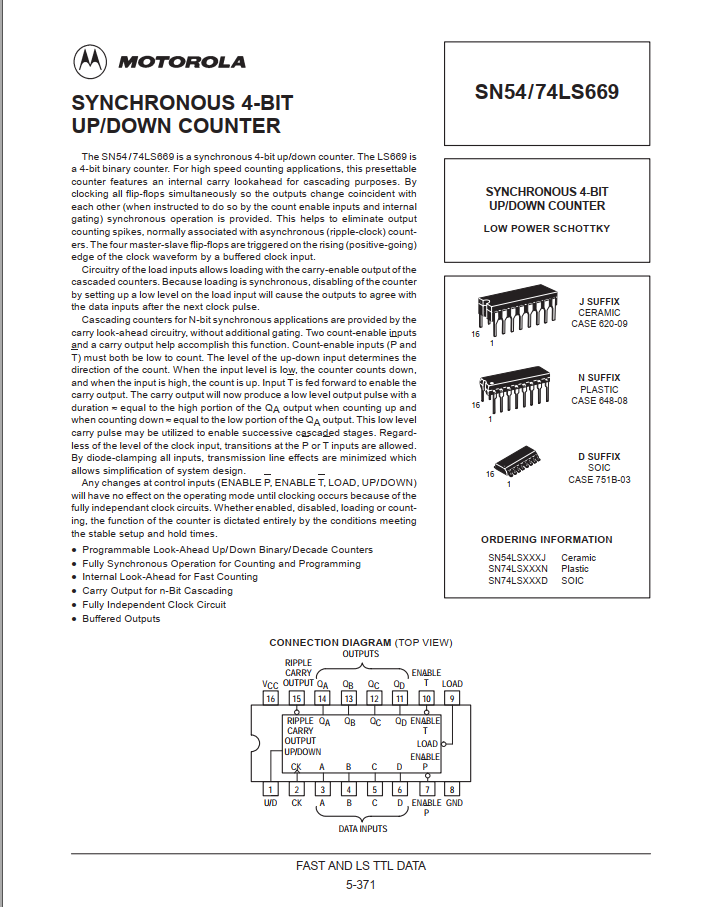




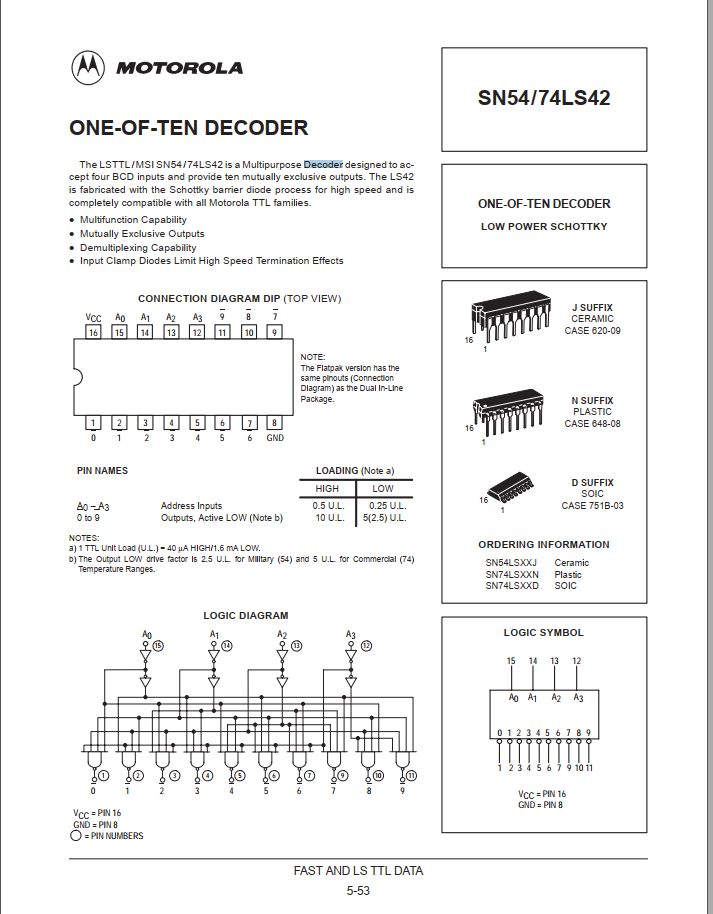
ALU

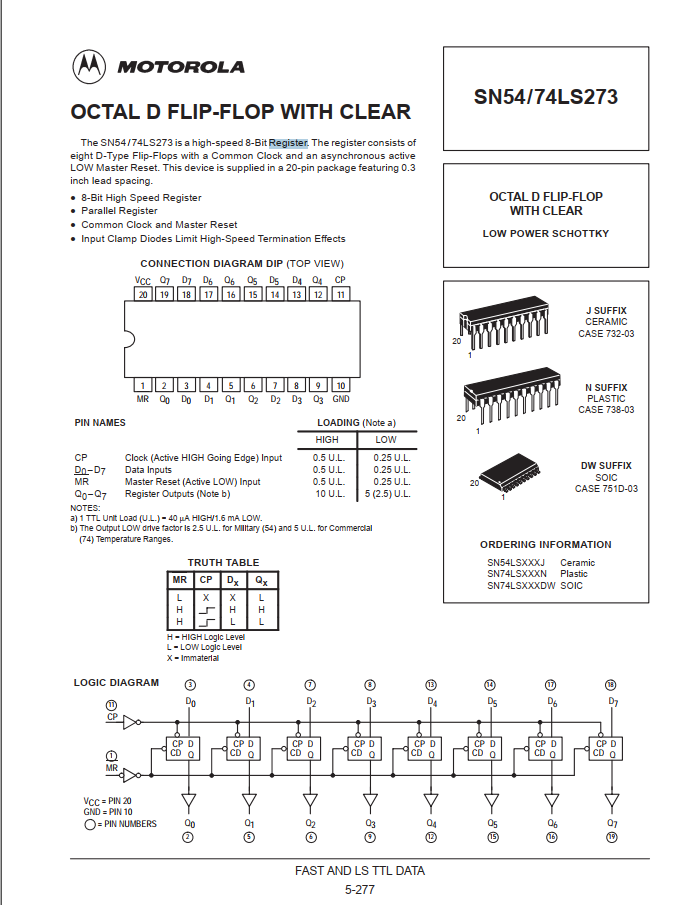
**Componentes Reales y/o Basados en:**

**Contador (PC, AR y Contador de Ciclos)**

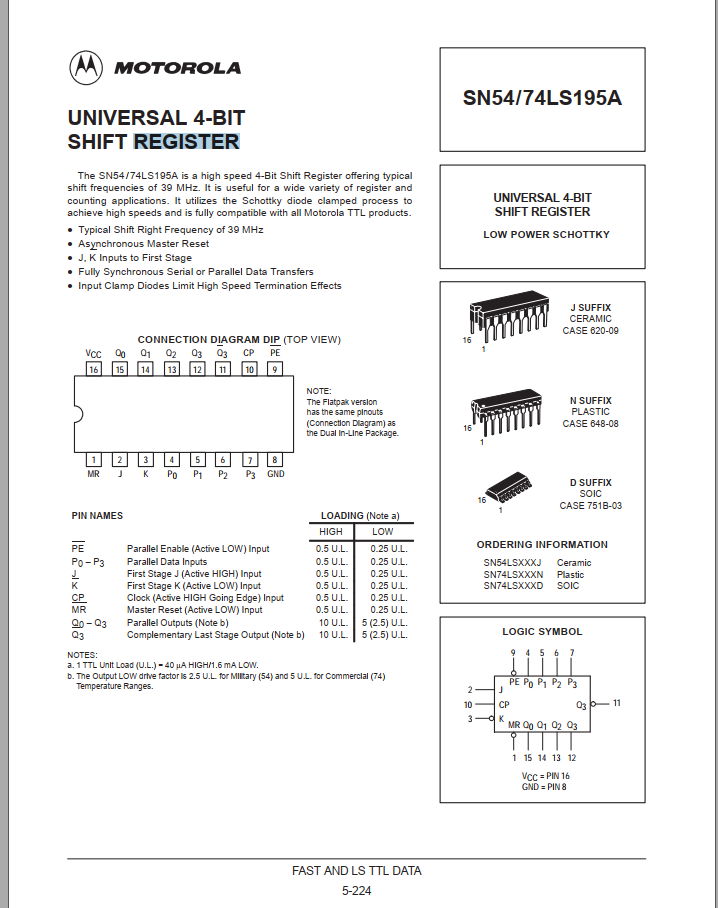


**Decoder**

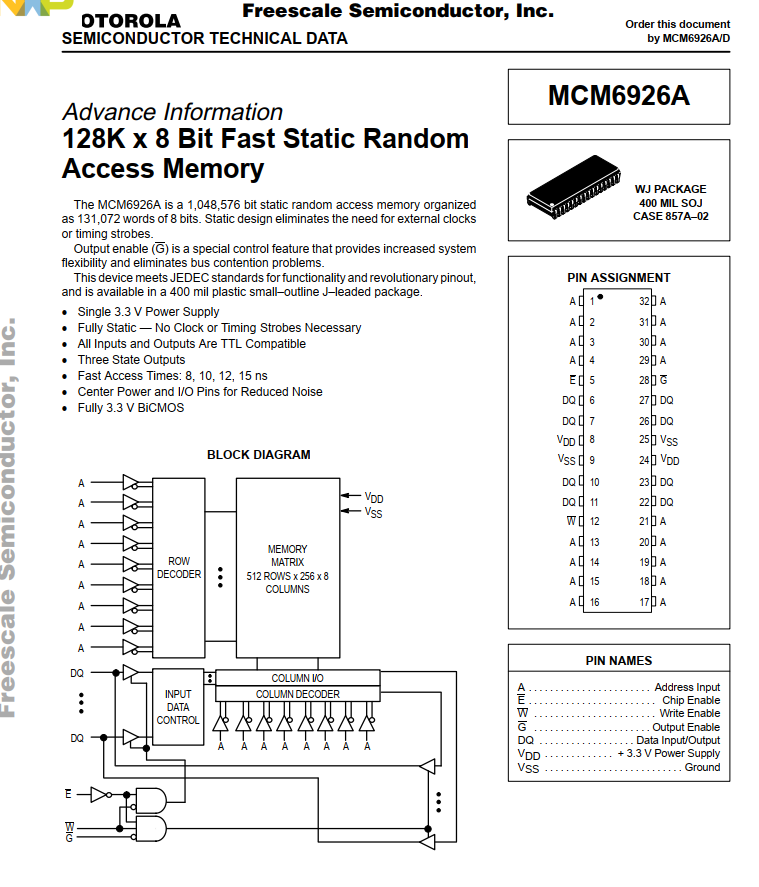


**Registro de Instrucciones**

**Registros del Archivo de Registros, Temp y Banderas**



**Memoria**



**Códigos y Simulaciones VHDL**

**Funcional**

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.NUMERIC\_STD.ALL;

ENTITY RESTABCD\_CPU\_GENERAL IS

PORT(

CLK: IN STD\_LOGIC; --CLOCK

INTS: IN STD\_LOGIC; --INTERRUPT

START: IN STD\_LOGIC --START

);

END RESTABCD\_CPU\_GENERAL;

ARCHITECTURE FUNCIONAL OF RESTABCD\_CPU\_GENERAL IS

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--PARA EL ALGORITMO EN PARTICULAR SE SIGUIÓ UTILIZANDO EL ALGORITMO ORIGINAL DE

--4 BITS, PARA ESTO SIMPLEMENTE CAMBIO, LA FORMA EN QUE IDENTIFICO LA BANDERA DE CARRY

--SIEMPRE SERÁ EL QUINTO BIT INDEPENDIENTEMENTE QUE SE TRABAJE CON 8 BITS EN TODO MOMENTO,

--LA RAZÓN POR LA QUE TRABAJAMOS TODO CON 8 BITS AHORA ES DEBIDO A LA FORMA EN ESPECÍFICO

--QUE TIENE NUESTRA RUTA DE DATOS Y LA MEMORIA, YA QUE TODOS LOS COMPONENTES QUE INTERACTUAN

--CON EL BUS DE DATOS PRINCIPAL LO HACEN CON 8 BITS

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--PARA EL CASO QUE QUERRAMOS HACER UNA SUMA DE 8 BITS CON ACARREO EN EL NOVENO BIT, DEBEMOS CAMBIAR

--NUESTRA FUNCIÓN CARRY Y SU BANDERA A:

--AUX3(8 BITS): OP1

--AUX4(8 BITS): OP2

--AUX5(9 BITS): OP1+OP2

--FLAGCARRY = AUX5(9)

--BANDERA [2(POSICIÓN 3, CARRY)] = FLAGC

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--PARA NUESTRO ALGORITMO USAMOS REGISTROS EXTRA UTILIZABLES POR EL PROGRAMADOR RAUX Y RAUX2

--Y DOS INVICIBLES PARA EL QUE SON RA, RB , RC

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--PARA RECORDAR BANDERA(0-3) ES: Z-NZ-C-V EN ESE ORDEN

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*---

--TM ES UN ESTADO INTERMEDIO PARA LA ASIGNACIÓN DE LOS OPERANDOS (NO FORMA PARTE DEL ASM NI EL DISEÑO EN GENERAL)

--PERO QUEDA IMPLÍCITO, SIN EMBARGO, EN EL CÓDIGO, HAY QUE ESPECIFICAR, ESTO SE EXPLICA EN EN LA PARTE DE DISEÑO DE ASM DE NUESTRO PDF

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*---

--EL ALGORITMO NO HACE EL ÚLTIMO CAMBIO DE NÚMERO NEGATIVO EN COMPLEMENTO A BCD LEÍBLE POR LA PERSONA

--SIN EMBARGO SE PUEDE SEGUIR VIENDO EL NÚMERO NEGATIVO PERO EN COMPLEMENTO

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*---

--REGISTRO DE INSTRUCCIONES

SIGNAL OPCODE: STD\_LOGIC\_VECTOR (4 DOWNTO 0);

SIGNAL RI0, RI1: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL ADRS: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL MODE: STD\_LOGIC;

SIGNAL OP1F,OP2F: STD\_LOGIC\_VECTOR(2 DOWNTO 0);

SIGNAL OP1, OP2, BANDEROTA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

--SEÑALES DE LAS MICRO INSTRUCCIONES Y RD

SIGNAL AR,PC,SP: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL R1,R2,R3,R4,RA,RB,RC,TEMP, RAUX, RAUX2: STD\_LOGIC\_VECTOR (7 DOWNTO 0) := "00000000";

SIGNAL BANDERA: STD\_LOGIC\_VECTOR (3 DOWNTO 0);

SIGNAL FLAGC, FLAGZ, FLAGV: INTEGER;

--ESTADOS

TYPE ESTADO IS (T0,T1,T2,T3,T4,TM,TN,T5,T6,T7,T8,T9,T10,T11,TINT,T12,T13,T14,T15,T16,T17);

SIGNAL EST: ESTADO;

SIGNAL ENDF: STD\_LOGIC:='0';

TYPE OPERACION IS (ADD,ADDC,CMP,INC,NOTT,ANDD, MOVE, BZ, RETI, PUSH, POP, ST, LD, CLR, SHRR, SHLL, SUB, DEC, JMP);

SIGNAL OPERATION: OPERACION;

--MEMORIA

TYPE VECTOR\_ARRAY IS ARRAY (0 to 255) OF STD\_LOGIC\_VECTOR(7 DOWNTO 0);

SIGNAL M: VECTOR\_ARRAY :=(

--- ADDRESSES:

---Pa = 232

---Na = 233

---Pb = 234

---Nb = 235

---Pr = 236

---Nr = 237

---Ph = 238

"01100100", --0 LD M=1

"00000000", --1 R1

"11101001", --2 Na

"01100100", --3 LD M=1

"00100000", --4 R2

"11101011", --5 Nb

"01100100", --6 LD M=1

"01000000", --7 R3

"11101000", --8 Pa

"01100100", --9 LD M=1

"01100000", --10 R4

"11101010", --11 Pb

"00010000", --12 CMP M=0

"00000100", --13 R1,R2

"00111100", --14 BRANCH Z (SI ES UNO SALTO) M=1

"00000000", --15 -

"01000100", --16 -68 --DESTINO 1

"00111100", --17 BRANCH Z (SI ES UNO SALTO) M=1

"00000000", --18 -

"00101100", --19 -44

"01011100", --20 ST M=1 -- SI V=0

"00000100", --21 R2

"11101101", --22 Nr

"10000000", --23 SUB M=0

"00000100", --24 R1,R2

"00000000", --25 ADD M=0

"00001100", --26 R1,R3

"00110100", --27 MOVE M=1

"10000000", --28 RAUX

"00000000", --29 '0'

"00000000", --30 ADD M=0

"00010000", --31 R1,RAUX

"01101100", --32 CLR M=1

"00000000", --33 R1

"00010000", --34 CMP M=0

"00110000", --35 R2,RAUX

"00111100", --36 BRANCH Z M=1

"00000000", --37 -

"01000111", --38 71 -- DESTINO 2

"00011000", --39 INC M=0

"10000000", --40 RAUX

"10010100", --41 JMP

"11100100", --42 -

"00011110", --43 30

"01011100", --44 ST M=1 --DESTINO BRANCH V SI ES 1

"00000000", --45 R1

"11101101", --46 Nr

"10000000", --47 SUB M=0

"00000100", --48 R1,R2

"00000000", --49 ADD

"00101100", --50 R2,R4

"00110100", --51 MOVE M=1

"10000000", --52 RAUX

"00000000", --53 '0'

"00000000", --54 ADD M=0

"11000100", --55 R2,RAUX

"01101100", --56 CLR M=1

"00100000", --57 R2

"00010000", --58 CMP M=0

"00010000", --59 R1 RAUX

"00111100", --60 BRANCH Z M=1

"01001000", --61 -

"01000111", --62 71 - DESTINO 2

"00011000", --63 INC M=0

"10000000", --64 RAUX

"10010100", --65 JMP

"00000000", --66 -

"00110111", --67 55

"01011100", --68 ST M=1 --DESTINO 1

"00000000", --69 R1

"11101101", --70 Nr

"01100100", --71 LD M=1 --DESTINO 2

"01000000", --72 R3

"11101101", --73 Nr

"10001000", --74 DEC M=0

"01000000", --75 R3

"00110100", --76 MOVE M=1

"10100000", --77 RAUX2

"00000000", --78 '0'

"01100100", --79 LD M=1

"01100000", --80 R4

"11101100", --81 Pr

"01100000", --82 LD M=0

"00000000", --83 R1

"11101000", --84 M[Pa]

"01100000", --85 LD M=0

"00100000", --86 R2

"11101010", --87 M[Pb]

"00100100", --88 NOT M=1

"00100000", --89 R2

"00010100", --90 CMP M=1

"10100000", --91 RAUX2

"00000000", --92 '0'

"00111100", --93 BRANCH Z M=1

"00000000", --94 -

"01100011", --95-- 99 -- DESTINO 3

"00001100", --96 ADDC M=1

"00100000", --97 R2

"00001001", --98 '9'

"00001100", --99 ADDC M=1

"00100000", --100 R2

"00001010", --101 '10'

"00010100", --102 CMP M=1

"00100000", --103 R2

"00000000", --104 '0'

"00111100", --105 BRANCH Z M=1

"00000000", --106 -

"01111110", --107- 126-- DESTINO 4

"00010100", --108 CMP M=1

"00100000", --109 R2

"00001001", --110 '9' "#OP2" --EMPIEZO EN 9 VOY HASTA 15

"00111100", --111 BRANCH Z M=1

"00000000", --112 -

"01111110", --113 -126- DESTINO 4

"00011100", --114 INC OP2 M=1

"00000000", --115 -

"01101110", --116 \*110\* "OP2"

"00010100", --117 CMP M=1

"00100000", --118 R2

"00001111", --119 '15'

"00111100", --120 BRANCH Z M=1

"00000000", --121 -

"10000001", --122 -129-DESTINO 5

"10010100", --123 JMP

"00000000", --124 -

"01101100", --125 108

"00001100", --126- ADDC M=1 --DESTINO 4

"00100000", --127 R2

"00000110", --128 '6'

"01011100", --129 ST M=1--DESTINO 5

"00100000", --130 R2

"11101100", --131 Pr

"00010000", --132 CMP M=0

"10101000", --133 RAUX2,R3

"00111100", --134 BRANCH Z M=1

"00000000", --135 -

"10010111", --136- 151 --DESTINO 6

"00011100", --137 INC M=1

"11001010", --138 -

"11101000", --139 Pa

"00011100", --140 INC M=1

"00000000", --141 -

"11101010", --142 Pb

"00011100", --143 INC M=1

"00000000", --144 -

"11101100", --145 Pr

"00011000", --146 INC M=0

"10100000", --147 RAUX2

"10010100", --148 JMP

"00000000", --149 -

"01001111", --150 -79

"00011000", --151 INC M=0 --DESTINO 6

"10100000", --152- RAUX2

"00011100", --153 INC M=1

"00000000", --154 -

"11101100", --155 Pr

"01011000", --156 ST M=0

"11000000", --157 FLAGC

"11101100", --158 Pr

"00011000", --159 INC M=0

"01000000", --160 R3

"00110000", --161 MOVE M=0

"10001000", --162 RAUX, R3

"00000000", --163 ADD M=0

"10001100", --164 RAUX, R4

"00000100", --165 ADD M=1

"10000000", --166 RAUX

"00001010", --167 '10'

"01011100", --168 ST M=1

"10000000", --169 RAUX

"11101110", --170 Ph

"01101000", --171 CLR M=0

"11000000", --172 FLAGC

"01101000", --173 CLR M=0

"10100000", --174 RAUX2

"01100000", --175 LD M=0

"00100000", --176 R2

"11101100", --177 Pr

"00010100", --178 CMP M=1

"00100000", --179 R2

"00000000", --180 '0'

"00111100", --181 BRANCH Z M=1

"00000000", --182 -

"10111000", --183 184--DESTINO 7 (190 PARA CONTINUAR EL ALGORITMO)

"00110000", --184 MOVE M=0

"11111100", --185 OP1,OP2 (111 Y 111 PARA TERMINAR)

"00000000", --186

"00000000", --187

"00000000", --188

"00000000", --189

"10000100", --190-// PARA CONTINUAR EL ALGORITMO: SUB M=1--DESTINO 7 // PARA TERMINAR POR FALTA DE MEMORIA: SE APLICA END

"01000000", --191 R3

"11101100", --192 Pr

"10001000", --193 DEC M=0

"01000000", --194 R3

"01101000", --195 CLR M=0

"11000000", --196 FLAGC

"01100100", --197 LD M=1

"00000000", --198 R1

"11101100", --199 Pr

"00100000", --200 NOT M=0

"00000000", --201 R1

"00011000", --202 INC M=0

"00000000", --203 R1

"00010100", --204 CMP M=1

"10100000", --205 RAUX2

"00000000", --206 '0'

"00111100", --207 BRANCH Z M=1

"00000000", --208 -

"11010101", --209- 213 --DESTINO 8

"00001100", --210 ADDC M=1

"00000000", --211 R1

"00001001", --212 '9'

"00001100", --213-ADDC M=1 --DESTINO 8

"00000000", --214 R1

"00001010", --215 '10'

"00010100", --216 CMP M=1

"00000000", --217 R1

"00000000", --218 '0'

"00111100", --219 BRANCH Z M=1

"00000001", --220 -

"11110000", --221- 240 --DESTINO 9 (YA NO CONCUERDA)

"00010100", --222 CMP M=1

"00000000", --223 R1

"00001001", --224 '9' (#OP2)

"00111100", --225 BRANCH Z M=1

"00000000", --226 -

"11110000", --227 - 240--DESTINO 9 (YA NO CONCUERDA)

"00011100", --228 INC OP2 M=1

"00000000", --229 -

"11100000", --230 224 "OP2"

"11001101", --231 -- NO ALCANZA LA MEMORIA PARA CONTINUAR

"11110000", --232 Pa (240)

"00000010", --233 Na (2)

"11110101", --234 Pb (245)

"00000011", --235 Nb (3)

"11111011", --236 Pr (251)

"00000000", --237 Nr (3) (DEBE CAMBIAR SOLO)

"00000000", --238 Ph (YA NO USAMOS)

"00000000", --239

"00000011", --240 3 M[Pa]

"00000010", --241 2

"00000000", --242

"00000000", --243

"00000000", --244

"00001000", --245 8 M[Pb]

"00000100", --246 4

"00000001", --247 1

"00000000", --248

"00000000", --249

"00000000", --250

"00000000", --251 Resultado 1 M(Pr)

"00000000", --252 Resultado 2

"00000000", --253 Resultado 3

"00000000", --254 Resultado 4

"00000000" --255 -- MAS ABAJO PR+NR+10 HUBIERA HABIDO PH Y EL RESULTADO TRANSFORMADO DE COMPLEMENTO A BCD EN LENGUAJE HUMANO

);BEGIN

PROCESS (CLK)

VARIABLE AUX1,AUX2,AUX3, AUX4, AUX5, AUX6: INTEGER; --1 (PC), 2(AR), 3 (SP), 4 (OPERADOR), 5 (OPERADOR), 6 RESULTADO

VARIABLE AUXV1: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

VARIABLE AUXV2: STD\_LOGIC\_VECTOR (3 DOWNTO 0);

BEGIN

IF (CLK' EVENT AND CLK = '1') THEN

IF(ENDF = '0') THEN

IF (OP1F="111" AND OP2F= "111") THEN

ENDF <= '1'; --PARA TERMINAR EL PROCESO PONEMOS AMBOS OPERADORES EN 111 Y 111

END IF;

IF (EST = T0) THEN

AR <= "00000000";

PC <= "00000000";

IF (START = '1') THEN

EST <= T1;

ELSE

EST <= T0;

END IF;

END IF;

-----------------------------------------------------------------

IF (EST = T1) THEN

AR <= PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

EST <= T2;

END IF;

-----------------------------------------------------------------

IF (EST = T2) THEN

AUX2 := TO\_INTEGER(UNSIGNED(AR));

RI0 <= M(AUX2);

EST <= T3;

END IF;

-----------------------------------------------------------------

IF (EST = T3) THEN

AR <= PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

EST <= T4;

END IF;

-----------------------------------------------------------------

IF (EST = T4) THEN

AUX2 := TO\_INTEGER(UNSIGNED(AR));

RI1<=M(AUX2);

EST <= TM;

END IF;

-----------------------------------------------------------------

IF (EST = TM) THEN

OPCODE <= RI0(7 DOWNTO 3);

MODE <= RI0(2);

OP1F <= RI1 (7 DOWNTO 5);

OP2F <= RI1 (4 DOWNTO 2);

ADRS <= RI1 (7 DOWNTO 0);

EST <= TN;

BANDEROTA <= "0000"&BANDERA;

END IF;

------------------------------------------------------------------

IF ( EST = TN) THEN

CASE OP1F IS

WHEN "000" => OP1 <= R1;

WHEN "001" => OP1 <= R2;

WHEN "010" => OP1 <= R3;

WHEN "011" => OP1 <= R4;

WHEN "100" => OP1 <= RAUX;

WHEN "101" => OP1 <= RAUX2;

WHEN "110" => OP1 <= BANDEROTA; --FLAGC

WHEN OTHERS =>

END CASE;

CASE OP2F IS

WHEN "000" => OP2 <= R1;

WHEN "001" => OP2 <= R2;

WHEN "010" => OP2 <= R3;

WHEN "011" => OP2 <= R4;

WHEN "100" => OP2 <= RAUX;

WHEN "101" => OP2 <= RAUX2;

WHEN "110" => OP2 <= BANDEROTA; --FLAGC

WHEN OTHERS =>

END CASE;

CASE OPCODE IS

WHEN "00000" => OPERATION <= ADD;

WHEN "00001" => OPERATION <= ADDC;

WHEN "00010" => OPERATION <= CMP;

WHEN "00011" => OPERATION <= INC;

WHEN "00100" => OPERATION <= NOTT;

WHEN "00101" => OPERATION <= ANDD;

WHEN "00110" => OPERATION <= MOVE;

WHEN "00111" => OPERATION <= BZ;

WHEN "01000" => OPERATION <= RETI;

WHEN "01001" => OPERATION <= PUSH;

WHEN "01010" => OPERATION <= POP;

WHEN "01011" => OPERATION <= ST;

WHEN "01100" => OPERATION <= LD;

WHEN "01101" => OPERATION <= CLR;

WHEN "01110" => OPERATION <= SHRR;

WHEN "01111" => OPERATION <= SHLL;

WHEN "10000" => OPERATION <= SUB;

WHEN "10001" => OPERATION <= DEC;

WHEN "10010" => OPERATION <= JMP;

WHEN OTHERS =>

END CASE;

EST <= T5;

END IF;

-----------------------------------------------------------------

IF (EST = T5) THEN

IF (OPCODE = "00000" OR OPCODE = "00001" OR OPCODE = "00010" OR OPCODE = "00101") THEN --ADD & ADDC & CMP & AND

IF MODE = '0' THEN

RA <= OP1;

END IF;

IF MODE = '1' THEN

AR <=PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

END IF;

IF (OPCODE = "00011") THEN --INC

IF MODE = '0' THEN

AUX4 := TO\_INTEGER(UNSIGNED(OP1));

AUX4 := AUX4+1;

TEMP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX4,8));

END IF;

IF MODE = '1' THEN

AR<=PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

END IF;

IF (OPCODE = "00100") THEN --NOT

IF MODE = '0' THEN

TEMP <= NOT OP1;

END IF;

IF MODE = '1' THEN

AR<=PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

END IF;

IF (OPCODE = "00110") THEN --MOVE

IF MODE = '0' THEN

OP1<=OP2;

END IF;

IF MODE = '1' THEN

AR<=PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

END IF;

IF (OPCODE = "00111") THEN --BZ

IF BANDERA(0) = '0' THEN

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

ELSE

AR<=PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

END IF;

IF (OPCODE = "01000") THEN --RETI

AUX3 := TO\_INTEGER(UNSIGNED(SP));

AUX3 := AUX3+1;

SP <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

IF (OPCODE = "01001") THEN --PUSH

IF MODE = '0' THEN

M(TO\_INTEGER(UNSIGNED(SP)))<=OP1;

AUX3 := TO\_INTEGER(UNSIGNED(SP));

AUX3 := AUX3-1;

SP <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

IF MODE = '1' THEN

AR<=PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

END IF;

IF (OPCODE = "01010") THEN --POP

M(TO\_INTEGER(UNSIGNED(SP))) <= OP1;

AUX3 := TO\_INTEGER(UNSIGNED(SP));

AUX3 := AUX3+1;

SP <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

IF (OPCODE = "01011" OR OPCODE ="01100") THEN --ST & LD

AR<=PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

IF (OPCODE = "01101") THEN --CLR

IF MODE = '0' THEN

OP1<="00000000";

END IF;

IF MODE = '1' THEN

Temp <= "00000000";

END IF;

END IF;

IF (OPCODE = "01110" OR OPCODE = "01111") THEN --SHR & SHL

IF MODE = '0' THEN

TEMP<=OP1;

END IF;

IF MODE ='1' THEN

AR<=PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

END IF;

IF (OPCODE = "10000") THEN --SUB

IF MODE = '0' THEN

RA<=OP1;

END IF;

IF MODE = '1' THEN

AR<=PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

END IF;

IF (OPCODE = "10001") THEN --DEC

IF MODE = '0' THEN

AUX4 := TO\_INTEGER(UNSIGNED(OP1));

AUX4 := AUX4-1;

TEMP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX4,8));

END IF;

IF MODE = '1' THEN

AR<=PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

END IF;

IF (OPCODE = "10010") THEN --JMP

AR<=PC;

AUX1 := TO\_INTEGER(UNSIGNED(PC));

AUX1 := AUX1+1;

PC <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

------REDIRECCIONAMIENTO----------------------------------------------

IF ((OPCODE = "01001" AND MODE='0') OR --PUSH (0)

(OPCODE = "00110" AND MODE='0') OR --MOVE(0)

(OPCODE ="00111" AND BANDERA(0)='0') --BZ(Z=0)

OR (OPCODE="01101" AND MODE='0') OR --CLR(0)

(OPCODE = "01010")) THEN --POP(0)

EST <= TINT;

ELSE

EST <= T6;

END IF;

END IF;

----------------------------------------------------------------------

IF (EST = T6) THEN

IF (OPCODE = "00000" ) THEN --ADD

IF MODE = '0' THEN

AUX4 := TO\_INTEGER(UNSIGNED(RA));

AUX5 := TO\_INTEGER(UNSIGNED(OP2));

AUX6 := AUX4+AUX5;

TEMP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX6,8));

END IF;

IF MODE = '1' THEN

RA<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "00001" ) THEN --ADDC

IF MODE = '0' THEN

AUX4 := TO\_INTEGER(UNSIGNED(RA));

AUX5 := TO\_INTEGER(UNSIGNED(OP2));

AUX6 := AUX4+AUX5+FLAGC;

TEMP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX6,8));

END IF;

IF MODE = '1' THEN

RA<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "00010") THEN --CMP

IF MODE = '0' THEN

AUX4 := TO\_INTEGER(UNSIGNED(RA));

AUX5 := TO\_INTEGER(UNSIGNED(OP2));

FLAGZ<=AUX4-AUX5; -- RESTA

END IF;

IF MODE = '1' THEN

RB<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "00011") THEN --INC

IF MODE = '0' THEN

OP1<=TEMP;

END IF;

IF MODE = '1' THEN

RB<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "00100") THEN --NOT

IF MODE = '0' THEN

OP1<=TEMP;

END IF;

IF MODE = '1' THEN

TEMP <= NOT M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "00101") THEN --AND

IF MODE = '0' THEN

TEMP<=RA AND R2;

END IF;

IF MODE = '1' THEN

RA<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "00110") THEN --MOVE

IF MODE = '1' THEN

OP1<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "00111") THEN --BZ

IF (MODE = '0' AND BANDERA(0)='1') THEN

RB<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

IF (MODE = '1' AND BANDERA(0)='1') THEN

PC<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "01000") THEN --RETI

PC<=M(TO\_INTEGER(UNSIGNED(AR)));

AUX3 := TO\_INTEGER(UNSIGNED(SP));

AUX3 := AUX3 +1;

SP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

IF (OPCODE = "01001") THEN --PUSH

IF MODE = '1' THEN

RB<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "01011" OR OPCODE ="01100") THEN --ST & LD

RB<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

IF (OPCODE = "01101") THEN --CLR

IF MODE = '1' THEN

AR<=OP1;

END IF;

END IF;

IF (OPCODE = "01110") THEN --SHR

IF MODE = '0' THEN

TEMP <= '0' & TEMP (7 DOWNTO 1);

END IF;

IF MODE = '1' THEN

RB<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "01111") THEN --SHL

IF MODE = '0' THEN

TEMP <= TEMP (6 DOWNTO 0) & '0';

END IF;

IF MODE = '1' THEN

RB<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "10000") THEN --SUB

IF MODE = '0' THEN

AUX4:= TO\_INTEGER(UNSIGNED(RA));

AUX5:= TO\_INTEGER(UNSIGNED(OP2));

AUX6:= AUX4-AUX5;

TEMP <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX6,8));

END IF;

IF MODE = '1' THEN

RA <= M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "10001") THEN --DEC

IF MODE ='0' THEN

OP1<=Temp;

END IF;

IF MODE = '1' THEN

AUX4:= TO\_INTEGER(UNSIGNED(M(TO\_INTEGER(UNSIGNED(AR)))));

AUX4:= AUX4-1;

Temp <=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX4,8));

END IF;

END IF;

IF (OPCODE ="10010") THEN --JMP

RB <= M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

-----------REDIRECCIONAMIENTO----------------------------------------------

IF ((OPCODE = "10001" AND MODE ='0')OR --DEC (0)

(OPCODE="00100" AND MODE='1') OR --MOVE(1)

(OPCODE="00011" AND MODE='0') OR --INC(0)

(OPCODE="00100" AND MODE='0') OR --NOT(0)

(OPCODE="01010") OR --POP

(OPCODE = "00111" AND MODE='1' AND BANDERA(0)='1') OR --BZ(1/1)

(OPCODE = "00011" AND MODE='0')) THEN

EST <= TINT;

ELSE

EST <= T7;

END IF;

END IF;

---------------------------------------------------------------------------

IF (EST = T7) THEN

IF (OPCODE = "00000") THEN --ADD

IF MODE = '0' THEN

OP1<=TEMP;

END IF;

IF MODE = '1' THEN

AUX4:= TO\_INTEGER(UNSIGNED(RA));

AUX5:= TO\_INTEGER(UNSIGNED(OP2));

AUX6:= AUX4+AUX5;

TEMP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX6,8));

END IF;

END IF;

IF (OPCODE = "00001") THEN --ADDC

IF MODE = '0' THEN

OP1<=TEMP;

END IF;

IF MODE = '1' THEN

AUX4:= TO\_INTEGER(UNSIGNED(RA));

AUX5:= TO\_INTEGER(UNSIGNED(OP2));

AUX6:= AUX4+AUX5+FLAGC;

TEMP <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX6,8));

END IF;

END IF;

IF (OPCODE = "00010") THEN --CMP

IF MODE = '0' THEN

IF FLAGZ = 0 THEN

BANDERA(0) <= '1';

BANDERA(1) <= '0';

ELSE

IF FLAGZ > 1 THEN

BANDERA(3)<='1';

END IF;

BANDERA(0) <= '0';

BANDERA(1) <= '1';

END IF;

END IF;

IF MODE = '1' THEN

AUX4:= TO\_INTEGER(UNSIGNED(OP1));

AUX5:= TO\_INTEGER(UNSIGNED(RA));

FLAGZ<=AUX4-AUX5;

END IF;

END IF;

IF (OPCODE = "00011") THEN --INC

IF MODE ='1' THEN

AR<=RB;

END IF;

END IF;

IF (OPCODE = "00100") THEN --NOT

IF MODE = '1' THEN

M (TO\_INTEGER(UNSIGNED(AR))) <= TEMP;

END IF;

END IF;

IF (OPCODE = "00101") THEN --AND

IF MODE = '0' THEN

OP1 <= TEMP;

END IF;

IF MODE = '1' THEN

TEMP<=RA AND R2;

END IF;

END IF;

IF (OPCODE = "00111") THEN --BZ

IF (MODE = '0' AND BANDERA(0)='1') THEN

AR<=RB;

END IF;

END IF;

IF (OPCODE = "01000") THEN --RETI

AUXV1 := M(TO\_INTEGER(UNSIGNED(SP)));

AUXV2 := AUXV1 (3 DOWNTO 0);

BANDERA <= AUXV2;

AUX3:= TO\_INTEGER(UNSIGNED(SP));

AUX3:= AUX3+1;

SP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

IF (OPCODE = "01001" OR OPCODE ="01100") THEN --PUSH

IF MODE = '1' THEN

M(TO\_INTEGER(UNSIGNED(SP)))<=RB;

AUX3:= TO\_INTEGER(UNSIGNED(SP));

AUX3:= AUX3-1;

SP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

END IF;

IF (OPCODE = "01011") THEN --ST & LD

AR<=RB;

END IF;

IF (OPCODE = "01110" OR OPCODE = "01111") THEN --SHR&SHL

IF MODE = '0' THEN

OP1<=TEMP;

END IF;

IF MODE = '1' THEN

AR <=RB;

END IF;

END IF;

IF (OPCODE = "01101") THEN --CLR

IF MODE = '1' THEN

M(TO\_INTEGER(UNSIGNED(AR)))<=TEMP;

END IF;

END IF;

IF (OPCODE = "10000") THEN --SUB

IF MODE = '0' THEN

OP1<=TEMP;

END IF;

IF MODE = '1' THEN

AUX4:= TO\_INTEGER(UNSIGNED(OP1));

AUX5:= TO\_INTEGER(UNSIGNED(RA));

AUX6:= AUX4+AUX5+FLAGC;

TEMP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX6,8));

END IF;

END IF;

IF (OPCODE = "10001") THEN --DEC

IF MODE = '1'THEN

M(TO\_INTEGER(UNSIGNED(AR)))<=TEMP;

END IF;

END IF;

IF (OPCODE = "10010") THEN --JMP

AR <= RB;

END IF;

----------REDIRECCIONAMIENTO----------------------------------------------

IF ((OPCODE = "00000" AND MODE = '0') --ADD(0)

OR (OPCODE = "00010" AND MODE = '0') OR --CMP(0)

(OPCODE = "01110" AND MODE = '0') OR --SHR(0)

(OPCODE = "01111" AND MODE = '0') OR --SHL(0)

(OPCODE = "01001" AND MODE = '0') OR --PUSH(0)

(OPCODE = "10001" AND MODE = '1') OR --DEC(1)

(OPCODE = "10000" AND MODE = '0') OR --SUB(0)

(OPCODE = "00101" AND MODE = '0') OR --AND (0)

(OPCODE = "00100" AND MODE = '1') OR --NOT (1)

(OPCODE = "01101" AND MODE = '1')) THEN --CLR(1)

EST <= TINT;

ELSE

EST <= T8;

END IF;

END IF;

--------------------------------------------------------------------------

IF (EST = T8) THEN

IF (OPCODE = "00000") THEN --ADD

IF MODE = '1' THEN

R1<=TEMP;

END IF;

END IF;

IF (OPCODE = "00001") THEN --ADDC

IF (TEMP (4)='1') THEN

FLAGC <= 1; --BANDERA DE CARRY

BANDERA(2) <= '1';

END IF;

END IF;

IF (OPCODE = "00010") THEN --CMP

IF MODE = '1' THEN

IF FLAGZ = 0 THEN

BANDERA (0) <= '1';

BANDERA (1) <= '0';

ELSE

FLAGV <= FLAGZ;

BANDERA (0) <= '0';

BANDERA (1) <= '1';

BANDERA (3) <= '1';

END IF;

END IF;

END IF;

IF (OPCODE = "00011") THEN --INC

IF MODE = '1' THEN

AUX4:= TO\_INTEGER(UNSIGNED(M(TO\_INTEGER(UNSIGNED(AR)))));

AUX4:= AUX4-1;

TEMP <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX4, 8));

END IF;

END IF;

IF (OPCODE = "00101") THEN --AND

IF MODE = '1' THEN

OP1<=TEMP;

END IF;

END IF;

IF (OPCODE = "00111") THEN --BZ

IF (MODE = '0' AND BANDERA(0)='1') THEN

PC<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "01000") THEN --RETI AQUÍ USAMOS REGISTROS ESTATICOS EN VEZ DE OPERADORES

R4<=M(TO\_INTEGER(UNSIGNED(SP)));

AUX3:= TO\_INTEGER(UNSIGNED(SP));

AUX3:= AUX3+1;

SP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

IF (OPCODE = "01011" ) THEN --ST

IF MODE ='0' THEN

RC<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

IF MODE = '1' THEN

M(TO\_INTEGER(UNSIGNED(AR)))<=OP1;

END IF;

END IF;

IF (OPCODE = "01100") THEN --LD

IF MODE ='0' THEN

RC<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

IF MODE = '1' THEN

OP1<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "01110" OR OPCODE = "01111") THEN --SHR & SHL

IF MODE = '1' THEN

TEMP<=M(TO\_INTEGER(UNSIGNED(AR))) ;

END IF;

END IF;

IF (OPCODE = "10000") THEN --SUB

IF MODE = '1' THEN

OP1<=TEMP;

END IF;

END IF;

----------REDIRECCIONAMIENTO----------------------------------------------

IF ((OPCODE = "00000" AND MODE='1') OR --ADD(1)

(OPCODE ="00001" AND MODE = '0') OR --ADDC(0)

(OPCODE="00010" AND MODE ='1')OR --CMP(1)

(OPCODE ="01011" AND MODE='1') OR --ST(1)

(OPCODE ="01100" AND MODE= '1') OR --LD(1)

(OPCODE = "00101" AND MODE='1') OR --AND(1)

(OPCODE = "00111" AND MODE='0' AND BANDERA(0)='1')OR --BZ(0/1)

(OPCODE = "10010"))THEN --JMP

EST <= TINT;

ELSE

EST <= T9;

END IF;

END IF;

--------------------------------------------------------------------------

IF (EST = T9) THEN

IF (OPCODE = "00001") THEN --ADDC

IF MODE='1' THEN

OP1<=TEMP;

END IF;

END IF;

IF (OPCODE = "00011")THEN --INC

IF MODE='1' THEN

M (TO\_INTEGER(UNSIGNED(AR)))<=TEMP;

END IF;

END IF;

IF (OPCODE ="01000") THEN --RETI

R3<=M(TO\_INTEGER(UNSIGNED(SP)));

AUX3:= TO\_INTEGER(UNSIGNED(SP));

AUX3:= AUX3+1;

SP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

IF (OPCODE ="01011" OR OPCODE ="01100") THEN --ST&LD

IF MODE='0' THEN

AR<=RC;

END IF;

END IF;

IF (OPCODE ="01110") THEN --SHR

IF MODE='1' THEN

TEMP <= '0' & TEMP (7 DOWNTO 1);

END IF;

END IF;

IF (OPCODE ="01110") THEN --SHR

IF MODE='1' THEN

TEMP<= TEMP(6 DOWNTO 0)& '0';

END IF;

END IF;

IF (OPCODE = "10010") THEN --JMP

PC <= M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

----------REDIRECCIONAMIENTO----------------------------------------------

IF ((OPCODE = "00001" AND MODE='1') OR --ADDC(1)

(OPCODE ="00011" AND MODE='1') --INC(1)

) THEN

EST <= TINT;

ELSE

EST <= T10;

END IF;

END IF;

--------------------------------------------------------------------------

IF (EST = T10) THEN

IF (OPCODE ="01000") THEN --RETI

R2<=M(TO\_INTEGER(UNSIGNED(SP)));

AUX3:= TO\_INTEGER(UNSIGNED(SP));

AUX3:= AUX3+1;

SP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

IF (OPCODE ="01011") THEN --ST

IF MODE = '0' THEN

M(TO\_INTEGER(UNSIGNED(AR)))<=OP1;

END IF;

END IF;

IF (OPCODE ="01100") THEN --LD

IF MODE = '0' THEN

OP1<=M(TO\_INTEGER(UNSIGNED(AR)));

END IF;

END IF;

IF (OPCODE = "01111" OR OPCODE = "01110") THEN --SHR&SHL

IF MODE ='0' THEN

M(TO\_INTEGER(UNSIGNED(AR))) <= TEMP;

END IF;

END IF;

----------REDIRECCIONAMIENTO----------------------------------------------

IF ((OPCODE ="01110" AND MODE='1') OR --SHR(1)

(OPCODE="01111" AND MODE='1') OR --SHL(1)

(OPCODE="01011" AND MODE='1') OR --ST(1)

(OPCODE="01100" AND MODE='1'))THEN --LD(1)

EST <= TINT;

ELSE

EST <=T11;

END IF;

END IF;

--------------------------------------------------------------------------

IF (EST = T11) THEN

IF (OPCODE ="01000") THEN --RETI

R1 <=M(TO\_INTEGER(UNSIGNED(SP)));

EST <= TINT;

END IF;

END IF;

--------------------------------------------------------------------------

IF (EST = TINT) THEN

IF (INTS ='1') THEN

EST <= T12;

ELSE

EST <= T1;

END IF;

CASE OP1F IS

WHEN "000" => R1 <= OP1;

WHEN "001" => R2 <= OP1;

WHEN "010" => R3 <= OP1;

WHEN "011" => R4 <= OP1;

WHEN "100" => RAUX <= OP1;

WHEN "101" => RAUX2 <= OP1;

WHEN "110" => BANDEROTA <= OP1; --FLAGC

WHEN OTHERS =>

END CASE;

CASE OP2F IS

WHEN "000" => R1 <= OP2;

WHEN "001" => R2 <= OP2;

WHEN "010" => R3 <= OP2;

WHEN "011" => R4 <= OP2;

WHEN "100" => RAUX <= OP2;

WHEN "101" => RAUX2 <= OP2;

WHEN "110" => BANDEROTA <= OP2; --FLAGC

WHEN OTHERS =>

END CASE;

END IF;

--------------------------------------------------------------------------

IF (EST = T12) THEN

M(TO\_INTEGER(UNSIGNED(SP)))<=R1;

AUX3:= TO\_INTEGER(UNSIGNED(SP));

AUX3:= AUX3-1;

SP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

--------------------------------------------------------------------------

IF (EST = T13) THEN

M(TO\_INTEGER(UNSIGNED(SP)))<=R2;

AUX3:= TO\_INTEGER(UNSIGNED(SP));

AUX3:= AUX3-1;

SP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

--------------------------------------------------------------------------

IF (EST = T14) THEN

M(TO\_INTEGER(UNSIGNED(SP)))<=R3;

AUX3:= TO\_INTEGER(UNSIGNED(SP));

AUX3:= AUX3-1;

SP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

--------------------------------------------------------------------------

IF (EST = T15) THEN

M(TO\_INTEGER(UNSIGNED(SP)))<=R4;

AUX3:= TO\_INTEGER(UNSIGNED(SP));

AUX3:= AUX3-1;

SP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

--------------------------------------------------------------------------

IF (EST = T16) THEN

M(TO\_INTEGER(UNSIGNED(SP)))<= "0000" & BANDERA (3 DOWNTO 0);

AUX3:= TO\_INTEGER(UNSIGNED(SP));

AUX3:= AUX3-1;

SP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

--------------------------------------------------------------------------

IF (EST = T17) THEN

M(TO\_INTEGER(UNSIGNED(SP)))<=PC;

AUX3:= TO\_INTEGER(UNSIGNED(SP));

AUX3:= AUX3-1;

SP<=STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

END IF;

END IF;

END PROCESS;

END FUNCIONAL;

**Funcional Testbench**

library IEEE;

use IEEE.Std\_logic\_1164.all;

use IEEE.Numeric\_Std.all;

entity RESTABCD\_CPU\_GENERAL\_tb is

end;

architecture bench of RESTABCD\_CPU\_GENERAL\_tb is

component RESTABCD\_CPU\_GENERAL

PORT(

CLK: IN STD\_LOGIC;

INTS: IN STD\_LOGIC;

START: IN STD\_LOGIC

);

end component;

signal CLK: STD\_LOGIC;

signal INTS: STD\_LOGIC;

signal START: STD\_LOGIC ;

constant clock\_period: time := 10 ns;

BEGIN

uut: RESTABCD\_CPU\_GENERAL port map ( CLK => CLK,

INTS => INTS,

START => START );

CLOCK: process

BEGIN

CLK <= '0';

WAIT FOR CLOCK\_PERIOD;

CLK <= '1';

WAIT FOR CLOCK\_PERIOD;

END PROCESS;

START <= '1';

INTS <= '0'; --'1' AFTER 935 NS, '0' AFTER 1035 NS;

END;

**Estructural**

**Decoder Registro de Instrucciones 0**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

USE ieee.numeric\_std.all;

ENTITY DECODER IS --74\_LS\_42 MODIFICADO

PORT (

ENABLE\_DEC: IN STD\_LOGIC;

IND: IN STD\_LOGIC\_VECTOR(7 downto 0);

ADD,ADDC,CMP,INC,NOTT,MOVE,BZ,RETI: OUT STD\_LOGIC;

ST,LD,CLR,SHRR,SHLL,SUB,DEC,JMP: OUT STD\_LOGIC;

MODE: OUT STD\_LOGIC;

ADRS: OUT STD\_LOGIC;

OP1: OUT STD\_LOGIC\_VECTOR(2 downto 0);

OP2: OUT STD\_LOGIC\_VECTOR (2 downto 0)

);

END DECODER;

ARCHITECTURE ESTRUCTURAL OF DECODER IS

BEGIN

PROCESS (ENABLE\_DEC, IND)

VARIABLE LECTURA: STD\_LOGIC\_VECTOR(4 DOWNTO 0);

VARIABLE MODO: STD\_LOGIC;

BEGIN

LECTURA := IND (7 DOWNTO 3);

MODO := IND (2);

IF LECTURA = "00000" THEN

ADD <= '1';

END IF;

IF LECTURA = "00001" THEN

ADDC <= '1';

END IF;

IF LECTURA = "00010" THEN

CMP <= '1';

END IF;

IF LECTURA = "00011" THEN

INC <= '1';

END IF;

IF LECTURA = "00100" THEN

NOTT <= '1';

END IF;

IF LECTURA = "00110" THEN

MOVE <= '1';

END IF;

IF LECTURA = "00111" THEN

BZ <= '1';

END IF;

IF LECTURA = "01000" THEN

RETI <= '1';

END IF;

IF LECTURA = "01011" THEN

ST <= '1';

END IF;

IF LECTURA = "01100" THEN

LD <= '1';

END IF;

IF LECTURA = "01101" THEN

CLR <= '1';

END IF;

IF LECTURA = "01110" THEN

SHRR <= '1';

END IF;

IF LECTURA = "01111" THEN

SHLL <= '1';

END IF;

IF LECTURA = "10000" THEN

SUB <= '1';

END IF;

IF LECTURA = "10001" THEN

DEC <= '1';

END IF;

IF LECTURA = "10001" THEN

JMP <= '1';

END IF;

IF MODO = '0' THEN

MODE <= '0';

END IF;

IF MODO ='1' THEN

MODE <= '1';

END IF;

END PROCESS;

END ESTRUCTURAL;

**Decoder Registro de Instrucciones 1**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

USE ieee.numeric\_std.all;

ENTITY DECODER2 IS --74\_LS\_42 MODIFICADO

PORT (

ENABLE\_DEC: IN STD\_LOGIC;

IND: IN STD\_LOGIC\_VECTOR(7 downto 0);

OP1: OUT STD\_LOGIC\_VECTOR(2 downto 0);

OP2: OUT STD\_LOGIC\_VECTOR (2 downto 0)

);

END DECODER2;

ARCHITECTURE ESTRUCTURAL OF DECODER2 IS

BEGIN

PROCESS (ENABLE\_DEC, IND)

VARIABLE IND1: STD\_LOGIC\_VECTOR(2 DOWNTO 0);

VARIABLE IND2: STD\_LOGIC\_VECTOR(2 DOWNTO 0);

BEGIN

IND1 :=IND (7 DOWNTO 5);

IND2 :=IND (4 DOWNTO 2);

OP1 <= IND1;

OP2 <= IND2;

END PROCESS;

END ESTRUCTURAL;

**Decoder Contador de Ciclos**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

USE ieee.numeric\_std.all;

ENTITY DECODER3 IS --74\_LS\_42 MODIFICADO

PORT (

ENABLE\_DEC: IN STD\_LOGIC;

IND: IN STD\_LOGIC\_VECTOR(7 downto 0);

E0, E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11, E12, E13, E14, E15, E16, E17, EINT: OUT STD\_LOGIC

);

END DECODER3;

ARCHITECTURE ESTRUCTURAL OF DECODER3 IS

BEGIN

PROCESS (ENABLE\_DEC, IND)

BEGIN

IF IND = "00000000" THEN

E0 <= '1';

END IF;

IF IND = "00000001" THEN

E1 <= '1';

END IF;

IF IND = "00000010" THEN

E2 <= '1';

END IF;

IF IND = "00000011" THEN

E3 <= '1';

END IF;

IF IND = "00000100" THEN

E4 <= '1';

END IF;

IF IND = "00000101" THEN

E5 <= '1';

END IF;

IF IND = "00000110" THEN

E6 <= '1';

END IF;

IF IND = "00000111" THEN

E7 <= '1';

END IF;

IF IND = "00001000" THEN

E8 <= '1';

END IF;

IF IND = "00001001" THEN

E9 <= '1';

END IF;

IF IND = "00001010" THEN

E10 <= '1';

END IF;

IF IND = "00001011" THEN

E11 <= '1';

END IF;

IF IND = "00001100" THEN --12 BINARIO

EINT <= '1';

END IF;

IF IND = "00001101" THEN

E12 <= '1';

END IF;

IF IND = "00001110" THEN

E13 <= '1';

END IF;

IF IND = "00001111" THEN

E14 <= '1';

END IF;

IF IND = "00010000" THEN

E15 <= '1';

END IF;

IF IND = "00010001" THEN

E16 <= '1';

END IF;

IF IND = "00010010" THEN

E17 <= '1';

END IF;

END PROCESS;

END ESTRUCTURAL;

**Registro de Instrucciones**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

USE ieee.numeric\_std.all;

ENTITY REGISTROINS IS -- --74\_LS\_273

PORT

(

clk : IN STD\_LOGIC;

lod : IN STD\_LOGIC;

D : IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

Q : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0)

);

END REGISTROINS;

ARCHITECTURE ESTRUCTURAL OF REGISTROINS IS

BEGIN

PROCESS(clk)

BEGIN

IF(clk'EVENT AND clk='1')THEN

IF (lod = '1')THEN

Q <= D;

END IF;

END IF;

END PROCESS;

END ESTRUCTURAL;

**Contador de Ciclos**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

USE ieee.numeric\_std.all;

ENTITY CONTADORCICLOS IS --74\_LS\_669

PORT

(

LDCount: IN STD\_LOGIC\_VECTOR (7 downto 0);

LDEn: IN STD\_LOGIC;

clk : IN STD\_LOGIC; --Reloj

INcount : IN STD\_LOGIC; --enable count

Rcount: IN STD\_LOGIC; --Reset count

Q : BUFFER INTEGER;

QBIT: OUT STD\_LOGIC\_VECTOR (7 downto 0)

);

END CONTADORCICLOS;

ARCHITECTURE ESTRUCTURAL OF CONTADORCICLOS IS

BEGIN

PROCESS(clk)

BEGIN

IF(clk'EVENT AND clk='1')THEN

IF((INcount = '1') AND (Rcount = '0'))THEN --incremento natural

Q <= Q + 1;

END IF;

IF (Rcount = '1') THEN --reseteo del contador

Q <= 0;

END IF;

IF ((LDEn='1') AND (Rcount = '0')) THEN --Carga del contador

Q <= to\_integer(unsigned(LDCount));

END IF;

QBIT<= std\_logic\_vector(to\_unsigned(Q,8));

END IF;

END PROCESS;

END ESTRUCTURAL;

**Código Principal**

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.NUMERIC\_STD.ALL;

LIBRARY WORK;

USE WORK.LIBRERIA\_TPFINAL.ALL;

ENTITY RESTABCD\_CPU IS

PORT(

CLK: IN STD\_LOGIC; --CLOCK

START: IN STD\_LOGIC; --START

INTS: IN STD\_lOGIC; --INTERRUPT

ENDD: BUFFER STD\_LOGIC --END

);

END RESTABCD\_CPU;

ARCHITECTURE ESTRUCTURAL OF RESTABCD\_CPU IS

--SEÑALES GLOBALES

SIGNAL BUS\_TS: STD\_LOGIC\_VECTOR(7 DOWNTO 0):= "00000000";

--SEÑALES DE LA UNIDAD DE CONTROL

SIGNAL E0,E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11, E12, E13, E14, E15, E16, E17, EINT: STD\_LOGIC := '0';

SIGNAL ADD, ADDC, CMP, INC, NOTT, MOVE, BZ, ST, LD, CLR, SUB, DEC, JMP, RETI: STD\_LOGIC :='0';

SIGNAL M: STD\_LOGIC;

SIGNAL ADRS: STD\_LOGIC\_VECTOR(7 DOWNTO 0);

SIGNAL OP1, OP2: STD\_LOGIC\_VECTOR(2 DOWNTO 0);

SIGNAL LD\_RI0, LD\_RI1: STD\_LOGIC;

SIGNAL DI0, DI1, DI2, QI0, QI1: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL OP1\_LD, OP1\_CLR, OP1\_OE: STD\_LOGIC; ---ESTE USAMOS PARA R1 EN NUESTRA TABLA DE SEÑALES DE CONTROL, R1=DESTINO=OP1 =/= REGISTRO ESTÁTICO 1

SIGNAL OP2\_LD, OP2\_OE: STD\_LOGIC; --ESTE USAMOS PARA R2 EN NUESTRA TABLA DE SEÑALES DE CONTROL, R2=OPERANDO=OP2 =/= REGISTRO ESTÁTICO 2

SIGNAL OUT\_COUNT\_CICLOS, LD\_COUNT\_CICLOS: STD\_LOGIC\_VECTOR(7 DOWNTO 0);

SIGNAL LD\_EN\_CICLOS: STD\_LOGIC;

--SEÑALES DE LA RUTA DE DATOS

--R1

SIGNAL R1\_DATA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL R1\_OP1: STD\_LOGIC;

SIGNAL R1\_CLR: STD\_LOGIC;

SIGNAL R1\_OUT\_ENABLE: STD\_LOGIC;

SIGNAL R1\_LOAD\_ENABLE: STD\_LOGIC;

--R2

SIGNAL R2\_DATA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL R2\_OP1: STD\_LOGIC;

SIGNAL R2\_CLR: STD\_LOGIC;

SIGNAL R2\_OUT\_ENABLE: STD\_LOGIC;

SIGNAL R2\_LOAD\_ENABLE: STD\_LOGIC;

--R3

SIGNAL R3\_DATA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL R3\_OP1: STD\_LOGIC;

SIGNAL R3\_CLR: STD\_LOGIC;

SIGNAL R3\_OUT\_ENABLE: STD\_LOGIC;

SIGNAL R3\_LOAD\_ENABLE: STD\_LOGIC;

--R4

SIGNAL R4\_DATA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL R4\_OP1: STD\_LOGIC;

SIGNAL R4\_CLR: STD\_LOGIC;

SIGNAL R4\_OUT\_ENABLE: STD\_LOGIC;

SIGNAL R4\_LOAD\_ENABLE: STD\_LOGIC;

--RAUX

SIGNAL RAUX\_DATA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL RAUX\_OP1: STD\_LOGIC;

SIGNAL RAUX\_CLR: STD\_LOGIC;

SIGNAL RAUX\_OUT\_ENABLE: STD\_LOGIC;

SIGNAL RAUX\_LOAD\_ENABLE: STD\_LOGIC;

--RAUX2

SIGNAL RAUX2\_DATA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL RAUX2\_OP1: STD\_LOGIC;

SIGNAL RAUX2\_CLR: STD\_LOGIC;

SIGNAL RAUX2\_OUT\_ENABLE: STD\_LOGIC;

SIGNAL RAUX2\_LOAD\_ENABLE: STD\_LOGIC;

--RA

SIGNAL RA\_DATA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL RA\_LOAD\_ENABLE: STD\_LOGIC;

SIGNAL RA\_OUT\_ENABLE: STD\_LOGIC;

SIGNAL RA\_OUT: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

--RB

SIGNAL RB\_DATA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL RB\_OP1: STD\_LOGIC;

SIGNAL RB\_CLR: STD\_LOGIC;

SIGNAL RB\_OUT\_ENABLE: STD\_LOGIC;

SIGNAL RB\_LOAD\_ENABLE: STD\_LOGIC;

--RC

SIGNAL RC\_DATA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL RC\_OP1: STD\_LOGIC;

SIGNAL RC\_CLR: STD\_LOGIC;

SIGNAL RC\_OUT\_ENABLE: STD\_LOGIC;

SIGNAL RC\_LOAD\_ENABLE: STD\_LOGIC;

--TEMP

SIGNAL TEMP\_DATA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL TEMP\_OUT\_ENABLE: STD\_LOGIC;

SIGNAL TEMP\_LOAD\_ENABLE: STD\_LOGIC;

SIGNAL TEMP\_CLR: STD\_LOGIC;

--BANDERAS

SIGNAL BANDERA\_DATA: STD\_LOGIC\_VECTOR (3 DOWNTO 0);

SIGNAL Z,NZ,C,V: STD\_LOGIC;

SIGNAL BANDERA\_LOAD:STD\_LOGIC;

SIGNAL BANDERA\_OUT: STD\_LOGIC;

-- Z,NZ,C,V -> 0000 EN ESE ORDEN

--ALU

SIGNAL FUN: STD\_LOGIC\_VECTOR (2 DOWNTO 0);

SIGNAL ALU\_OUT: STD\_LOGIC\_VECTOR(7 DOWNTO 0);

--PC

SIGNAL PC\_DATA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL OE\_PC: STD\_LOGIC; --OUTPC

SIGNAL INC\_PC: STD\_LOGIC; --PCINC

SIGNAL LD\_PC: STD\_LOGIC; --LDPC

SIGNAL CLR\_PC: STD\_LOGIC; --CLRPC

--AR

SIGNAL LD\_AR: STD\_LOGIC;

SIGNAL CLR\_AR: STD\_LOGIC;

SIGNAL AR\_DATA: STD\_LOGIC\_VECTOR(7 DOWNTO 0);

--SP

SIGNAL INC\_SP: STD\_LOGIC;

SIGNAL DEC\_SP: STD\_LOGIC;

SIGNAL SP\_DATA: STD\_LOGIC\_VECTOR (7 DOWNTO 0);

--MEMORIA

SIGNAL MEM\_OE\_SP: STD\_LOGIC;

SIGNAL MEM\_OE\_AR: STD\_LOGIC;

SIGNAL MEM\_WE\_SP: STD\_LOGIC;

SIGNAL MEM\_WE\_AR: STD\_LOGIC;

TYPE VECTOR\_ARRAY IS ARRAY (0 to 255) OF STD\_LOGIC\_VECTOR(7 DOWNTO 0);

SIGNAL MEMORIA: VECTOR\_ARRAY :=(

--- ADDRESSES:

---Pa = 232

---Na = 233

---Pb = 234

---Nb = 235

---Pr = 236

---Nr = 237

---Ph = 238

"01100100", --0 LD M=1

"00000000", --1 R1

"11101001", --2 Na

"01100100", --3 LD M=1

"00100000", --4 R2

"11101011", --5 Nb

"01100100", --6 LD M=1

"01000000", --7 R3

"11101000", --8 Pa

"01100100", --9 LD M=1

"01100000", --10 R4

"11101010", --11 Pb

"00010000", --12 CMP M=0

"00000100", --13 R1,R2

"00111100", --14 BRANCH Z (SI ES UNO SALTO) M=1

"00000000", --15 -

"01000100", --16 -68 --DESTINO 1

"00111100", --17 BRANCH Z (SI ES UNO SALTO) M=1

"00000000", --18 -

"00101100", --19 -44

"01011100", --20 ST M=1 -- SI V=0

"00000100", --21 R2

"11101101", --22 Nr

"10000000", --23 SUB M=0

"00000100", --24 R1,R2

"00000000", --25 ADD M=0

"00001100", --26 R1,R3

"00110100", --27 MOVE M=1

"10000000", --28 RAUX

"00000000", --29 '0'

"00000000", --30 ADD M=0

"00010000", --31 R1,RAUX

"01101100", --32 CLR M=1

"00000000", --33 R1

"00010000", --34 CMP M=0

"00110000", --35 R2,RAUX

"00111100", --36 BRANCH Z M=1

"00000000", --37 -

"01000111", --38 71 -- DESTINO 2

"00011000", --39 INC M=0

"10000000", --40 RAUX

"10010100", --41 JMP

"11100100", --42 -

"00011110", --43 30

"01011100", --44 ST M=1 --DESTINO BRANCH V SI ES 1

"00000000", --45 R1

"11101101", --46 Nr

"10000000", --47 SUB M=0

"00000100", --48 R1,R2

"00000000", --49 ADD

"00101100", --50 R2,R4

"00110100", --51 MOVE M=1

"10000000", --52 RAUX

"00000000", --53 '0'

"00000000", --54 ADD M=0

"11000100", --55 R2,RAUX

"01101100", --56 CLR M=1

"00100000", --57 R2

"00010000", --58 CMP M=0

"00010000", --59 R1 RAUX

"00111100", --60 BRANCH Z M=1

"01001000", --61 -

"01000111", --62 71 - DESTINO 2

"00011000", --63 INC M=0

"10000000", --64 RAUX

"10010100", --65 JMP

"00000000", --66 -

"00110111", --67 55

"01011100", --68 ST M=1 --DESTINO 1

"00000000", --69 R1

"11101101", --70 Nr

"01100100", --71 LD M=1 --DESTINO 2

"01000000", --72 R3

"11101101", --73 Nr

"10001000", --74 DEC M=0

"01000000", --75 R3

"00110100", --76 MOVE M=1

"10100000", --77 RAUX2

"00000000", --78 '0'

"01100100", --79 LD M=1

"01100000", --80 R4

"11101100", --81 Pr

"01100000", --82 LD M=0

"00000000", --83 R1

"11101000", --84 M[Pa]

"01100000", --85 LD M=0

"00100000", --86 R2

"11101010", --87 M[Pb]

"00100100", --88 NOT M=1

"00100000", --89 R2

"00010100", --90 CMP M=1

"10100000", --91 RAUX2

"00000000", --92 '0'

"00111100", --93 BRANCH Z M=1

"00000000", --94 -

"01100011", --95-- 99 -- DESTINO 3

"00001100", --96 ADDC M=1

"00100000", --97 R2

"00001001", --98 '9'

"00001100", --99 ADDC M=1

"00100000", --100 R2

"00001010", --101 '10'

"00010100", --102 CMP M=1

"00100000", --103 R2

"00000000", --104 '0'

"00111100", --105 BRANCH Z M=1

"00000000", --106 -

"01111110", --107- 126-- DESTINO 4

"00010100", --108 CMP M=1

"00100000", --109 R2

"00001001", --110 '9' "#OP2" --EMPIEZO EN 9 VOY HASTA 15

"00111100", --111 BRANCH Z M=1

"00000000", --112 -

"01111110", --113 -126- DESTINO 4

"00011100", --114 INC OP2 M=1

"00000000", --115 -

"01101110", --116 \*110\* "OP2"

"00010100", --117 CMP M=1

"00100000", --118 R2

"00001111", --119 '15'

"00111100", --120 BRANCH Z M=1

"00000000", --121 -

"10000001", --122 -129-DESTINO 5

"10010100", --123 JMP

"00000000", --124 -

"01101100", --125 108

"00001100", --126- ADDC M=1 --DESTINO 4

"00100000", --127 R2

"00000110", --128 '6'

"01011100", --129 ST M=1--DESTINO 5

"00100000", --130 R2

"11101100", --131 Pr

"00010000", --132 CMP M=0

"10101000", --133 RAUX2,R3

"00111100", --134 BRANCH Z M=1

"00000000", --135 -

"10010111", --136- 151 --DESTINO 6

"00011100", --137 INC M=1

"11001010", --138 -

"11101000", --139 Pa

"00011100", --140 INC M=1

"00000000", --141 -

"11101010", --142 Pb

"00011100", --143 INC M=1

"00000000", --144 -

"11101100", --145 Pr

"00011000", --146 INC M=0

"10100000", --147 RAUX2

"10010100", --148 JMP

"00000000", --149 -

"01001111", --150 -79

"00011000", --151 INC M=0 --DESTINO 6

"10100000", --152- RAUX2

"00011100", --153 INC M=1

"00000000", --154 -

"11101100", --155 Pr

"01011000", --156 ST M=0

"11000000", --157 FLAGC

"11101100", --158 Pr

"00011000", --159 INC M=0

"01000000", --160 R3

"00110000", --161 MOVE M=0

"10001000", --162 RAUX, R3

"00000000", --163 ADD M=0

"10001100", --164 RAUX, R4

"00000100", --165 ADD M=1

"10000000", --166 RAUX

"00001010", --167 '10'

"01011100", --168 ST M=1

"10000000", --169 RAUX

"11101110", --170 Ph

"01101000", --171 CLR M=0

"11000000", --172 FLAGC

"01101000", --173 CLR M=0

"10100000", --174 RAUX2

"01100000", --175 LD M=0

"00100000", --176 R2

"11101100", --177 Pr

"00010100", --178 CMP M=1

"00100000", --179 R2

"00000000", --180 '0'

"00111100", --181 BRANCH Z M=1

"00000000", --182 -

"10111000", --183 184--DESTINO 7 (190 PARA CONTINUAR EL ALGORITMO)

"00110000", --184 MOVE M=0

"11111100", --185 OP1,OP2 (111 Y 111 PARA TERMINAR)

"00000000", --186

"00000000", --187

"00000000", --188

"00000000", --189

"10000100", --190-// PARA CONTINUAR EL ALGORITMO: SUB M=1--DESTINO 7 // PARA TERMINAR POR FALTA DE MEMORIA: SE APLICA END

"01000000", --191 R3

"11101100", --192 Pr

"10001000", --193 DEC M=0

"01000000", --194 R3

"01101000", --195 CLR M=0

"11000000", --196 FLAGC

"01100100", --197 LD M=1

"00000000", --198 R1

"11101100", --199 Pr

"00100000", --200 NOT M=0

"00000000", --201 R1

"00011000", --202 INC M=0

"00000000", --203 R1

"00010100", --204 CMP M=1

"10100000", --205 RAUX2

"00000000", --206 '0'

"00111100", --207 BRANCH Z M=1

"00000000", --208 -

"11010101", --209- 213 --DESTINO 8

"00001100", --210 ADDC M=1

"00000000", --211 R1

"00001001", --212 '9'

"00001100", --213-ADDC M=1 --DESTINO 8

"00000000", --214 R1

"00001010", --215 '10'

"00010100", --216 CMP M=1

"00000000", --217 R1

"00000000", --218 '0'

"00111100", --219 BRANCH Z M=1

"00000001", --220 -

"11110000", --221- 240 --DESTINO 9 (YA NO CONCUERDA)

"00010100", --222 CMP M=1

"00000000", --223 R1

"00001001", --224 '9' (#OP2)

"00111100", --225 BRANCH Z M=1

"00000000", --226 -

"11110000", --227 - 240--DESTINO 9 (YA NO CONCUERDA)

"00011100", --228 INC OP2 M=1

"00000000", --229 -

"11100000", --230 224 "OP2"

"11001101", --231 -- NO ALCANZA LA MEMORIA PARA CONTINUAR

"11110000", --232 Pa (240)

"00000010", --233 Na (2)

"11110101", --234 Pb (245)

"00000011", --235 Nb (3)

"11111011", --236 Pr (251)

"00000000", --237 Nr (3) (DEBE CAMBIAR SOLO)

"00000000", --238 Ph (YA NO USAMOS)

"00000000", --239

"00000011", --240 3 M[Pa]

"00000010", --241 2

"00000000", --242

"00000000", --243

"00000000", --244

"00001000", --245 8 M[Pb]

"00000100", --246 4

"00000001", --247 1

"00000000", --248

"00000000", --249

"00000000", --250

"00000000", --251 Resultado 1 M(Pr)

"00000000", --252 Resultado 2

"00000000", --253 Resultado 3

"00000000", --254 Resultado 4

"00000000" --255 -- MAS ABAJO PR+NR+10 HUBIERA HABIDO PH Y EL RESULTADO TRANSFORMADO DE COMPLEMENTO A BCD EN LENGUAJE HUMANO

);

BEGIN

--CONEXION DE SEÑALES

--SEÑALES DE CONTROL

--PC

V <= BANDERA\_DATA(3);

C <=BANDERA\_DATA(2);

Z <=BANDERA\_DATA(0);

LD\_PC <=(E6 AND RETI)OR(NOT(M) AND BZ AND E8 AND Z)OR(E8 AND JMP) OR (M AND E6 AND Z AND BZ);

CLR\_PC <= E0;

INC\_PC <= E1 OR E3 OR (M AND E5 AND (ADD OR ADDC OR CMP OR INC OR NOTT OR MOVE OR SUB OR DEC)) OR (E5 AND (BZ OR ST OR LD OR JMP));

OE\_PC <= E1 OR E3 OR (M AND E5 AND (ADD OR ADDC OR CMP OR INC OR NOTT OR MOVE OR SUB OR

DEC)) OR (E5 AND (ST OR LD OR JMP)) OR (BZ AND E5 AND Z) OR (INTS AND E17);

--TEMP

TEMP\_LOAD\_ENABLE <= (NOT(M) AND E5 AND (INC OR NOTT OR DEC)) OR (NOT(M) AND E6 AND (ADD OR ADDC

OR NOTT OR SUB)) OR (M AND E6 AND DEC) OR (M AND E7 AND (ADD OR ADDC OR SUB));

TEMP\_CLR <= M AND E5 AND CLR;

TEMP\_OUT\_ENABLE <= (NOT(M) AND E6 AND (INC OR NOTT OR DEC))OR(NOT(M) AND E7 AND (ADD OR ADDC OR SUB)) OR

(M AND E7 AND (NOTT OR CLR OR DEC))OR (M AND E8 AND (ADD OR INC OR SUB)) OR (M AND E9 AND (ADDC OR INC));

--ALU

FUN(2) <= (M AND E5 AND INC) OR (NOT(M) AND E5 AND (NOTT OR DEC)) OR (M AND E6 AND (ADD OR ADDC OR CMP OR NOTT OR DEC)) OR

(NOT(M) AND E6 AND SUB) OR (NOT(M) AND E7 AND (ADD OR ADDC OR CMP)) OR (M AND E7 AND SUB) OR (NOT (M) AND E8 AND INC);

FUN(1) <= (NOT(M) AND E5 AND (INC OR DEC)) OR (M AND E5 AND NOTT) OR (M AND E6 AND(ADD AND ADDC AND DEC)) OR

(NOT(M) AND E6 AND (CMP OR NOTT OR SUB)) OR (NOT(M) AND E7 AND (ADD OR ADDC)) OR (M AND E7 AND (CMP OR SUB)) OR (M AND E8 AND INC);

FUN(0) <= (NOT(M) AND E5 AND (INC OR NOTT OR DEC)) OR (M AND E6 AND (ADD OR CMP OR NOTT OR SUB OR DEC))

OR (NOT(M) AND E6 AND ADDC) OR (NOT(M) AND E7 AND (ADD AND CMP AND SUB)) OR (M AND E7 AND ADDC) OR (M AND E8 AND INC);

--BANDERAS

BANDERA\_LOAD <=(NOT(M) AND E7 AND CMP) OR (M AND E8 AND CMP) OR (E8 AND ADDC) OR (E7 AND RETI);

BANDERA\_OUT <= (NOT(M) AND E6 AND ADDC) OR (M AND E7 AND ADDC) OR (INTS AND E16);

--RI -- AQUÍ HAGO LA DISTINCIÓN DE RI PORQUE CARGO MI DATO EN DOS REGISTRO DIFERENTES DE ACUERDO A MI UNIDAD DE CONTROL

LD\_RI0 <= E2;

LD\_RI1 <= E4;

--AR

LD\_AR <= E1 OR E3 OR (M AND E5 AND (ADD OR ADDC OR CMP OR INC OR NOTT OR MOVE OR SUB OR

DEC)) OR (E5 AND (ST OR LD OR JMP)) OR (BZ AND E5 AND Z) OR (M AND CLR AND E6) OR (M AND E7 AND INC

) OR (E7 AND (ST OR LD OR JMP)) OR (NOT(M) AND BZ AND E7 AND Z) OR (NOT(M) AND E9 AND (ST OR LD));

CLR\_AR <= E0;

--SP

INC\_SP <= RETI AND (E5 OR E6 OR E7 OR E8 OR E9 OR E10);

DEC\_SP <= INTS AND (E12 OR E13 OR E14 OR E15 OR E16 OR E17);

--MEMORIA

MEM\_OE\_AR <= E2 OR E4 OR (M AND E6 AND (ADD OR ADDC OR CMP OR INC OR NOTT OR

MOVE OR SUB OR DEC)) OR (E6 AND (ST OR LD OR JMP)) OR (BZ AND E6 AND Z AND NOT(M)) OR

(M AND E8 AND INC) OR (E8 AND (JMP OR LD)) OR (

NOT(M) AND E8 AND ST) OR (NOT (M) AND BZ AND E8 AND Z) OR (NOT(M) AND E10 AND LD);

MEM\_OE\_SP <= RETI AND (E6 OR E7 OR E8 OR E9 OR E10 OR E11);

MEM\_WE\_AR <= (M AND E7 AND ((NOTT OR CLR OR DEC))) OR (M AND E8 AND ST) OR (M AND E9 AND INC) OR (NOT(M) AND E10 AND ST);

MEM\_WE\_SP <= INTS AND (E12 OR E13 OR E14 OR E15 OR E16 OR E17);

--OP1 \*\*AQUI SEPARO R1 DE OP1, OSEA CUANDO SE HACE UNA INTERRUPCIÓN O LLAMA A RETI, EL REGISTRO 1 ES EL QUE SE DEBE GUARDAR O LEER, LO MISMO PARA OP2

OP1\_LD <= (NOT(M) AND E5 AND MOVE) OR (NOT (M) AND E6 AND(INC OR NOTT OR DEC)) OR (M AND E6 AND MOVE)

OR (NOT(M) AND E7 AND (ADD OR ADDC OR SUB)) OR (M AND E8 AND (ADD OR LD OR SUB))

OR (M AND E9 AND ADDC) OR (NOT(M) AND E10 AND LD) OR (RETI AND E11);

OP1\_CLR <= NOT(M) AND CLR AND E5;

OP1\_OE <= (NOT(M) AND E5 AND (ADD OR ADDC OR INC OR CMP OR NOTT OR

SUB OR DEC)) OR (M AND E6 AND CLR) OR (M AND E7 AND (ADD OR ADDC OR CMP OR SUB))

OR (M AND E8 AND ST) OR (NOT(M) AND E10 AND ST) OR (INTS AND E12);

R1\_OUT\_ENABLE <= (INTS AND E12);

R1\_LOAD\_ENABLE <= (RETI AND E11);

--OP2

OP2\_LD <= E10 AND RETI;

OP2\_OE <= (NOT(M) AND E5 AND MOVE)OR(NOT(M) AND E6 AND(ADD OR ADDC OR CMP OR SUB)) OR (INTS AND E13);

R2\_OUT\_ENABLE <= E10 AND RETI;

R2\_LOAD\_ENABLE <= INTS AND E13;

--R3

R3\_LOAD\_ENABLE <= E9 AND RETI;

R3\_OUT\_ENABLE <= E14 AND INTS;

--R4

R4\_LOAD\_ENABLE <= E10 AND RETI;

R4\_OUT\_ENABLE <= E15 AND INTS;

--RA

RA\_LOAD\_ENABLE<= (NOT(M) AND E5 AND (ADD OR ADDC OR CMP OR SUB)) OR (M AND E6 AND (ADD OR ADDC OR CMP OR SUB));

RA\_OUT\_ENABLE <= (NOT(M) AND E6 AND (ADD OR ADDC OR CMP OR SUB)) OR (M AND E7 AND (ADD OR ADDC OR CMP OR SUB));

--RB

RB\_LOAD\_ENABLE <= (M AND E6 AND (CMP OR INC OR MOVE)) OR (E6 AND (ST OR LD OR JMP)) OR (BZ AND NOT (M) AND E6 AND Z);

RB\_OUT\_ENABLE <= (M AND E7 AND INC) OR (E7 AND (ST OR LD OR JMP)) OR (NOT(M) AND BZ AND E7 AND Z);

--RC

RC\_LOAD\_ENABLE <= NOT(M) AND E8 AND (ST OR LD);

RC\_OUT\_ENABLE <= NOT(M) AND E9 AND (ST OR LD);

--SEÑALES DE LA MÁQUINA DE CONTROL

--DECODERS

DECODER\_RI0: DECODER

PORT MAP (

ENABLE\_DEC => '1',

IND => QI0,

ADD => ADD,

ADDC => ADDC,

CMP => CMP,

INC => INC,

NOTT => NOTT,

MOVE => MOVE,

BZ => BZ,

RETI => RETI,

ST => ST,

LD => LD,

CLR => CLR,

SUB => SUB,

DEC => DEC,

JMP => JMP,

MODE => M

);

DECODER\_RI1: DECODER2

PORT MAP(

ENABLE\_DEC => '1',

IND => QI1,

OP1 => OP1,

OP2 => OP2

);

DECODER\_3: DECODER3

PORT MAP(

ENABLE\_DEC => '1',

IND => OUT\_COUNT\_CICLOS,

E0 => E0,

E1 => E1,

E2 => E2,

E3 => E3,

E4 => E4,

E5 => E5,

E6 => E6,

E7 => E7,

E8 => E8,

E9 => E9,

E10 => E10,

E11 => E11,

E12 => E12,

E13 => E13,

E14 => E14,

E15 => E15,

E16 => E16,

E17 => E17,

EINT => EINT

);

--REGISTROS DE INSTRUCCION

RI0: REGISTROINS

PORT MAP(

clk => CLK,

lod => '1',

D => BUS\_TS,

Q => QI0

);

RI1: REGISTROINS

PORT MAP(

clk => CLK,

lod => '1',

D => BUS\_TS,

Q => QI1

);

CONTADOR\_DE\_CICLOS: CONTADORCICLOS

PORT MAP(

LDCount => LD\_COUNT\_CICLOS,

LDEn => LD\_EN\_CICLOS,

clk => CLK,

INcount => '1',

Rcount => '0',

QBIT => OUT\_COUNT\_CICLOS

);

LD\_EN\_CICLOS <= (E5 AND ((BZ AND NOT(Z)) OR (NOT (M) AND CLR) OR (NOT (M) AND MOVE))) OR

(E6 AND ((NOT(M) AND INC) OR (NOT (M) AND DEC) OR (NOT(M) AND NOTT) OR (M AND MOVE) OR (M AND BZ AND Z)))

OR (E7 AND ((NOT(M) AND (ADD OR CMP OR SUB)) OR (M AND (DEC OR SUB OR NOTT))))

OR (E8 AND ((M AND (ADD OR CMP OR LD OR ST OR SUB)) OR (NOT(M) AND (ADDC OR (BZ AND Z))) OR JMP))

OR (E9 AND M AND (ADDC OR INC)) OR (E10 AND (NOT (M)) AND (ST OR LD)) OR (E11 AND RETI);

PROCESS (CLK)

VARIABLE AUX1, AUX2, AUX3, AUX4: INTEGER;

VARIABLE AUXV: STD\_LOGIC\_VECTOR(7 DOWNTO 0);

VARIABLE AUXB: STD\_LOGIC\_VECTOR(2 DOWNTO 0);

BEGIN

IF (CLK' EVENT AND CLK = '1') THEN

--INTERRUPCIÓN

IF (INTS = '0' AND EINT='1') THEN

LD\_COUNT\_CICLOS <= "00000001";

END IF;

IF (INTS = '1' AND EINT='1') THEN

LD\_COUNT\_CICLOS <= "00001100";

END IF;

--RUTA DE DATOS

--ALU

IF (START = '1' ) THEN

E1 <= '1';

ENDD <= '0';

END IF;

IF (ENDD = '0') THEN

CASE FUN IS

WHEN "000" => --ADD

AUX1:= TO\_INTEGER(UNSIGNED(RA\_OUT));

AUX2:= TO\_INTEGER(UNSIGNED(BUS\_TS));

AUX3:=AUX1+AUX2;

ALU\_OUT <= STD\_lOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

WHEN "001" => --ADDC

AUX1:= TO\_INTEGER(UNSIGNED(RA\_OUT));

AUX2:= TO\_INTEGER(UNSIGNED(BUS\_TS));

AUXB:= "00"&C;

AUX3:= TO\_INTEGER(UNSIGNED(AUXB));

AUX4:= AUX1+AUX2+AUX3;

ALU\_OUT <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX4,8));

IF (C='1')THEN

BANDERA\_LOAD <= '1';

BANDERA\_DATA(2) <= '1'; --FLAGC

END IF;

WHEN "010" => --CMP

IF M='0' THEN --RA-OP2

AUX1:= TO\_INTEGER(UNSIGNED(RA\_OUT));

AUX2:= TO\_INTEGER(UNSIGNED(BUS\_TS));

AUX3:= AUX2-AUX1;

END IF;

IF M='1' THEN --OP1-RA

AUX1:= TO\_INTEGER(UNSIGNED(RA\_OUT));

AUX2:= TO\_INTEGER(UNSIGNED(BUS\_TS));

AUX3:= AUX1-AUX2;

END IF;

IF (AUX3=0) THEN

BANDERA\_DATA(0)<='1'; --FLAGZ

END IF;

IF (AUX3>0) THEN

BANDERA\_DATA(1)<='1'; --FLAGNZ

BANDERA\_DATA(3)<='1'; --FLAGV

END IF;

WHEN "011" => --INC -

IF M='0' THEN --DEL BUS

AUX1 := TO\_INTEGER(UNSIGNED(BUS\_TS));

AUX1 := AUX1-1;

ALU\_OUT <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

IF M='1' THEN --DEL REGISTRO B

AUX1 := TO\_INTEGER(UNSIGNED(RB\_DATA));

AUX1 := AUX1-1;

ALU\_OUT <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

WHEN "101" => --NOT -- AMBOS DEL BUS

AUXV := NOT(BUS\_TS);

ALU\_OUT <= AUXV;

WHEN "110" => --SUB

IF M='0' THEN --RA-OP2

AUX1 := TO\_INTEGER(UNSIGNED(RA\_DATA));

AUX2 := TO\_INTEGER(UNSIGNED(BUS\_TS));

AUX3 := AUX1-AUX2;

ALU\_OUT <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

IF M='1' THEN --OP1-RA

AUX1 := TO\_INTEGER(UNSIGNED(RA\_DATA));

AUX2 := TO\_INTEGER(UNSIGNED(BUS\_TS));

AUX3 := AUX2-AUX1;

ALU\_OUT <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX3,8));

END IF;

WHEN "111" => --DEC

IF M='0' THEN --DEL BUS AMBOS

AUX1 := TO\_INTEGER(UNSIGNED(BUS\_TS));

AUX1 := AUX1-1;

ALU\_OUT <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

WHEN OTHERS => --NO SE USA EN REALIDAD PORQUE ES "AND" PERO PIDE EL COMPILADOR

AUX1:= AUX2;

END CASE;

--MEMORIA

IF (MEM\_WE\_SP = '1') THEN

MEMORIA(TO\_INTEGER(UNSIGNED(SP\_DATA))) <= BUS\_TS;

END IF;

IF (MEM\_WE\_AR = '1') THEN

MEMORIA(TO\_INTEGER(UNSIGNED(AR\_DATA))) <= BUS\_TS;

END IF;

IF (MEM\_OE\_AR = '1') THEN

BUS\_TS <= MEMORIA(TO\_INTEGER(UNSIGNED(AR\_DATA)));

END IF;

IF (MEM\_OE\_SP = '1') THEN

BUS\_TS <= MEMORIA(TO\_INTEGER(UNSIGNED(SP\_DATA)));

END IF;

--REGISTROS

CASE OP1 IS

WHEN "000" => --R1

R1\_LOAD\_ENABLE <= OP1\_LD;

R1\_OUT\_ENABLE <= OP1\_OE;

R1\_CLR <= OP1\_CLR;

WHEN "001" => --R2

R2\_LOAD\_ENABLE <= OP1\_LD;

R2\_OUT\_ENABLE <= OP1\_OE;

R2\_CLR <= OP1\_CLR;

WHEN "010" => --R3

R3\_LOAD\_ENABLE <= OP1\_LD;

R3\_OUT\_ENABLE <= OP1\_OE;

R3\_CLR <= OP1\_CLR;

WHEN "011" => --R4

R4\_LOAD\_ENABLE <= OP1\_LD;

R4\_OUT\_ENABLE <= OP1\_OE;

R4\_CLR <= OP1\_CLR;

WHEN "100" => --RAUX

RAUX\_LOAD\_ENABLE <= OP1\_LD;

RAUX\_OUT\_ENABLE <= OP1\_OE;

RAUX\_CLR <= OP1\_CLR;

WHEN "101" => --RAUX2

RAUX2\_LOAD\_ENABLE <= OP1\_LD;

RAUX2\_OUT\_ENABLE <= OP1\_OE;

RAUX2\_CLR <= OP1\_CLR;

WHEN "110" => --BANDERA

BANDERA\_LOAD<= OP1\_LD;

BANDERA\_OUT <= OP1\_OE;

WHEN OTHERS =>

IF (OP2 = "111") THEN

ENDD <= '1';

END IF;

END CASE;

CASE OP2 IS

WHEN "000" => --R1

R1\_LOAD\_ENABLE <= OP2\_LD;

R1\_OUT\_ENABLE <= OP2\_OE;

WHEN "001" => --R2

R2\_LOAD\_ENABLE <= OP2\_LD;

R2\_OUT\_ENABLE <= OP2\_OE;

WHEN "010" => --R3

R3\_LOAD\_ENABLE <= OP2\_LD;

R3\_OUT\_ENABLE <= OP2\_OE;

WHEN "011" => --R4

R4\_LOAD\_ENABLE <= OP2\_LD;

R4\_OUT\_ENABLE <= OP2\_OE;

WHEN "100" => --RAUX

RAUX\_LOAD\_ENABLE <= OP2\_LD;

RAUX\_OUT\_ENABLE <= OP2\_OE;

WHEN "101" => --RAUX2

RAUX2\_LOAD\_ENABLE <= OP2\_LD;

RAUX2\_OUT\_ENABLE <= OP2\_OE;

WHEN "110" => --BANDERA

BANDERA\_LOAD<= OP2\_LD;

BANDERA\_OUT <= OP2\_OE;

WHEN OTHERS =>

IF (OP1 = "111") THEN

ENDD <= '1';

END IF;

END CASE;

--R1

IF (R1\_LOAD\_ENABLE = '1' ) THEN

R1\_DATA <= BUS\_TS;

END IF;

IF (R1\_OUT\_ENABLE = '1') THEN

BUS\_TS <= R1\_DATA;

END IF;

--R2

IF (R2\_LOAD\_ENABLE = '1' ) THEN

R2\_DATA <= BUS\_TS;

END IF;

IF (R2\_OUT\_ENABLE = '1' ) THEN

BUS\_TS <= R2\_DATA;

END IF;

--R3

IF (R3\_LOAD\_ENABLE = '1' ) THEN

R3\_DATA <= BUS\_TS;

END IF;

IF (R3\_OUT\_ENABLE = '1' ) THEN

BUS\_TS <= R3\_DATA;

END IF;

--R4

IF (R4\_LOAD\_ENABLE = '1' ) THEN

R4\_DATA <= BUS\_TS;

END IF;

IF (R4\_OUT\_ENABLE = '1' ) THEN

BUS\_TS <= R4\_DATA;

END IF;

--RAUX

IF (RAUX\_LOAD\_ENABLE = '1' ) THEN

RAUX\_DATA <= BUS\_TS;

END IF;

IF (RAUX\_OUT\_ENABLE = '1' ) THEN

BUS\_TS <= RAUX\_DATA;

END IF;

--RAUX2

IF (RAUX2\_LOAD\_ENABLE = '1' ) THEN

RAUX2\_DATA <= BUS\_TS;

END IF;

IF (RAUX2\_OUT\_ENABLE = '1' ) THEN

BUS\_TS <= RAUX2\_DATA;

END IF;

--TEMP

IF (TEMP\_LOAD\_ENABLE = '1') THEN

TEMP\_DATA <= ALU\_OUT;

END IF;

IF (TEMP\_OUT\_ENABLE = '1') THEN

BUS\_TS <= TEMP\_DATA;

END IF;

IF (TEMP\_CLR = '1') THEN

TEMP\_DATA <= "00000000";

END IF;

--RA

IF (RA\_LOAD\_ENABLE = '1') THEN

RA\_DATA <= BUS\_TS;

END IF;

IF (RA\_OUT\_ENABLE = '1') THEN

RA\_OUT <= RA\_DATA;

END IF;

--RB

IF (RB\_LOAD\_ENABLE = '1') THEN

RB\_DATA <= BUS\_TS;

END IF;

IF (RB\_OUT\_ENABLE = '1') THEN

BUS\_TS <= RB\_DATA;

END IF;

--RC

IF (RC\_LOAD\_ENABLE = '1') THEN

RC\_DATA <= BUS\_TS;

END IF;

IF (RC\_OUT\_ENABLE = '1') THEN

BUS\_TS <= RC\_DATA;

END IF;

--PC

IF (INC\_PC ='1') THEN

AUX1:= TO\_INTEGER(UNSIGNED(PC\_DATA));

AUX1:= AUX1+1;

PC\_DATA <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

IF (CLR\_PC = '1') THEN

PC\_DATA <= "00000000";

END IF;

IF (LD\_PC = '1') THEN

PC\_DATA <= BUS\_TS;

END IF;

IF (OE\_PC = '1') THEN

BUS\_TS <= PC\_DATA;

END IF;

--AR

IF (CLR\_AR = '1') THEN

AR\_DATA <= "00000000";

END IF;

IF (LD\_AR = '1') THEN

AR\_DATA <= BUS\_TS;

END IF;

--MEMORIA

IF (MEM\_OE\_AR = '1') THEN

BUS\_TS <= MEMORIA(TO\_INTEGER(UNSIGNED(AR\_DATA)));

END IF;

IF (MEM\_OE\_SP = '1') THEN

BUS\_TS <= MEMORIA(TO\_INTEGER(UNSIGNED(SP\_DATA)));

END IF;

IF (MEM\_WE\_AR = '1') THEN

MEMORIA(TO\_INTEGER(UNSIGNED(AR\_DATA))) <= BUS\_TS;

END IF;

IF (MEM\_WE\_SP = '1') THEN

MEMORIA(TO\_INTEGER(UNSIGNED(SP\_DATA))) <= BUS\_TS;

END IF;

--SP

IF (INC\_SP = '1') THEN

AUX1:= TO\_INTEGER(UNSIGNED(SP\_DATA));

AUX1:= AUX1+1;

SP\_DATA <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

IF (DEC\_SP = '1') THEN

AUX1:= TO\_INTEGER(UNSIGNED(SP\_DATA));

AUX1:= AUX1-1;

SP\_DATA <= STD\_LOGIC\_VECTOR(TO\_UNSIGNED(AUX1,8));

END IF;

--BANDERAS RECORDANDO QUE TIENE SOLO 4 BITS

IF (BANDERA\_LOAD = '1') THEN

BANDERA\_DATA <= BUS\_TS (3 DOWNTO 0);

END IF;

IF (BANDERA\_OUT = '1') THEN

BUS\_TS <= "0000" & BANDERA\_DATA;

END IF;

END IF;

END IF;

END PROCESS;

END ESTRUCTURAL;

**Testbench Estructural**

library IEEE;

use IEEE.Std\_logic\_1164.all;

use IEEE.Numeric\_Std.all;

entity RESTABCD\_CPU\_tb is

end;

architecture bench of RESTABCD\_CPU\_tb is

component RESTABCD\_CPU

PORT(

CLK: IN STD\_LOGIC;

START: IN STD\_LOGIC;

INTS: IN STD\_lOGIC;

ENDD: BUFFER STD\_LOGIC

);

end component;

signal CLK: STD\_LOGIC;

signal START: STD\_LOGIC;

signal INTS: STD\_lOGIC;

constant clock\_period: time := 10 ns;

begin

uut: RESTABCD\_CPU port map ( CLK => CLK,

START => START,

INTS => INTS

);

CLOCKING:PROCESS

BEGIN

CLK <= '0';

WAIT FOR CLOCK\_PERIOD;

CLK <= '1';

WAIT FOR CLOCK\_PERIOD;

END PROCESS;

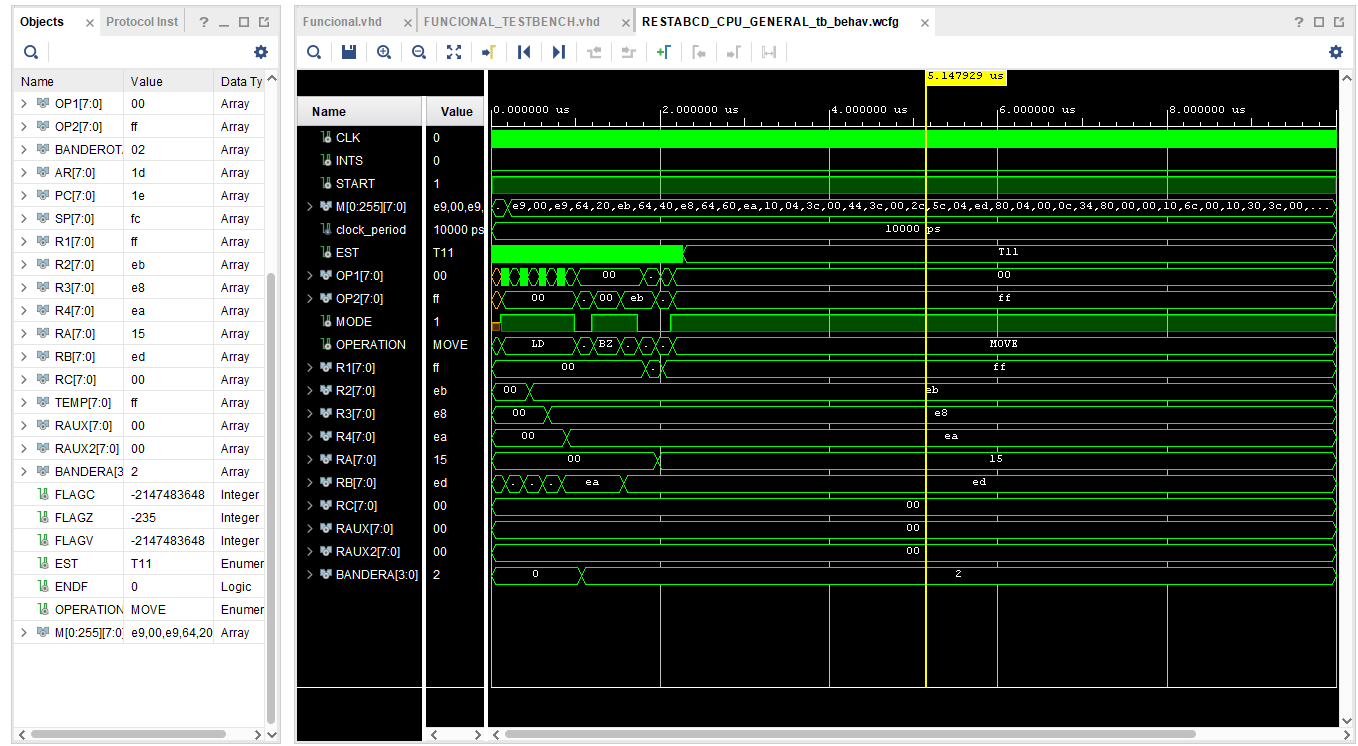
START <= '1';

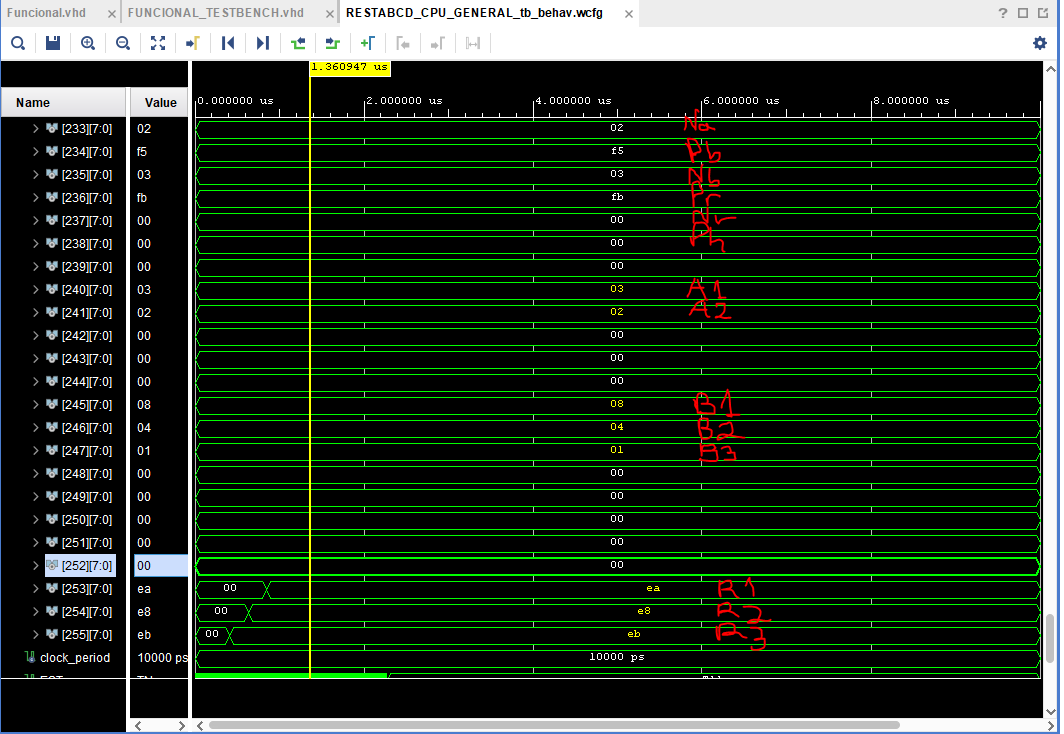
INTS <= '0';

END;

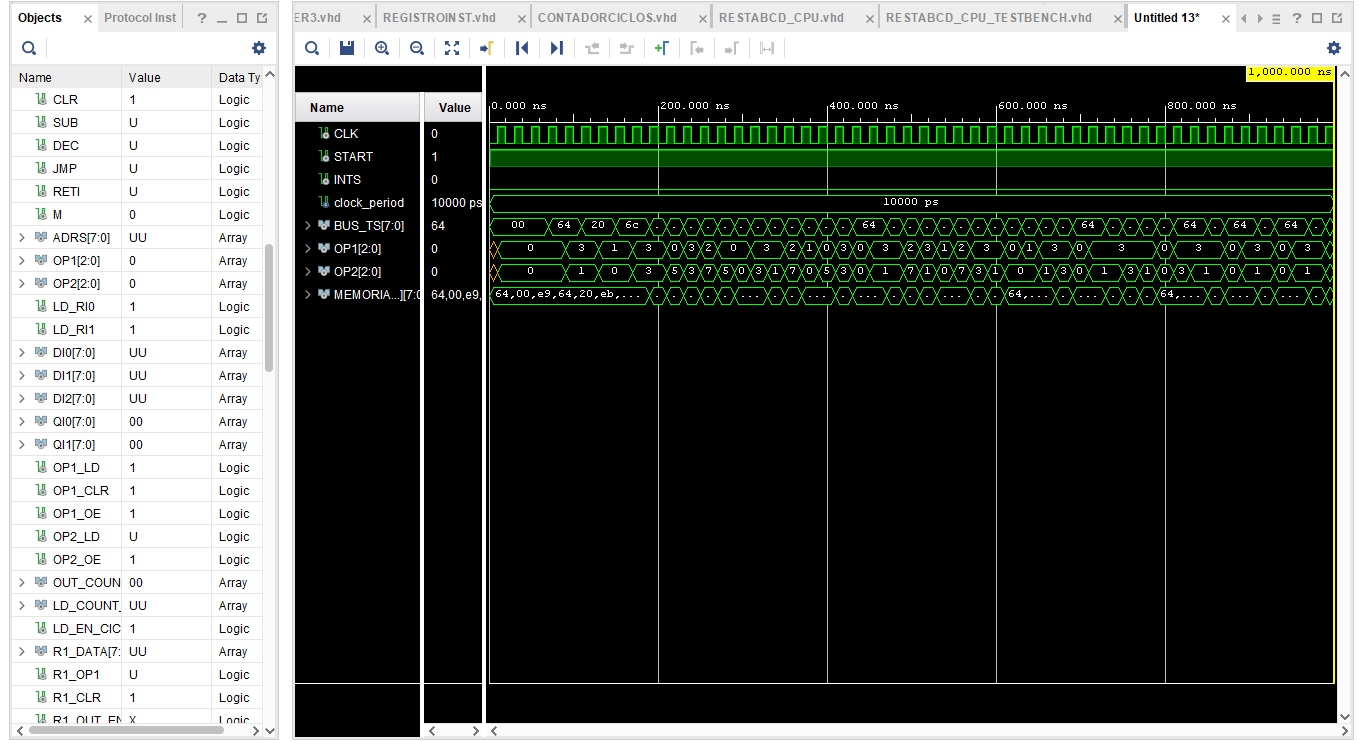
**Capturas**

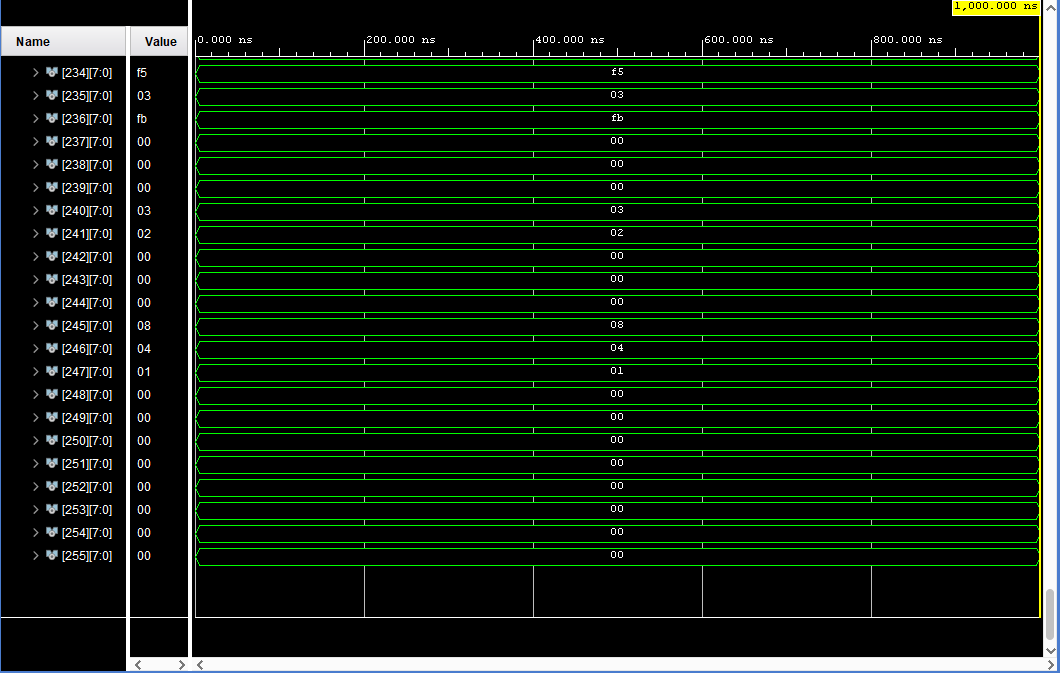
**Funcional.**





**Estructural**





**Conclusiones**

En la captura del algoritmo funcional, se subraya qué elemento es qué señal o variable en el algoritmo particular, en el algoritmo funcional la señal de Nr que es la longitud del resultado no se carga ni modifica. Y el resultado final es incorrecto. Cabe **resaltar** que si bien el resultado está en complemento A10 y no en BCD leíble por lenguaje humano, el resultado es aun así equívoco.

En la captura del algoritmo estructural, el resultado final no termina de cargarse, por lo que quedan en 0 los dígitos del resultado. La razón de este problema pueden ser demasiadas, desde un AND equívoco en alguna señal de control, algún 0 u 1 en el código binario o algún salto de sincronizado respecto a señales retrasadas un período.

Al final no se pudo realizar el algoritmo particular con éxito pero sí se pudieron implementar las instrucciones de la cpu con éxito y estructurar las señales. Se pudo seguir el proceso de diseño, desde la selección de señales, selección de direccionamiento hasta la selección de elementos físicos particular representativos en la ruta de datos.