Arquitectura de Computadoras - Práctica 1

Tania Mendoza
Alejandro Coronado

Febrero 2017

1. Especificaciones del Sistema

1.1. Diagrama de Bloques

A continuación se muestra un diagrama con los principales componentes de nuestra computadora

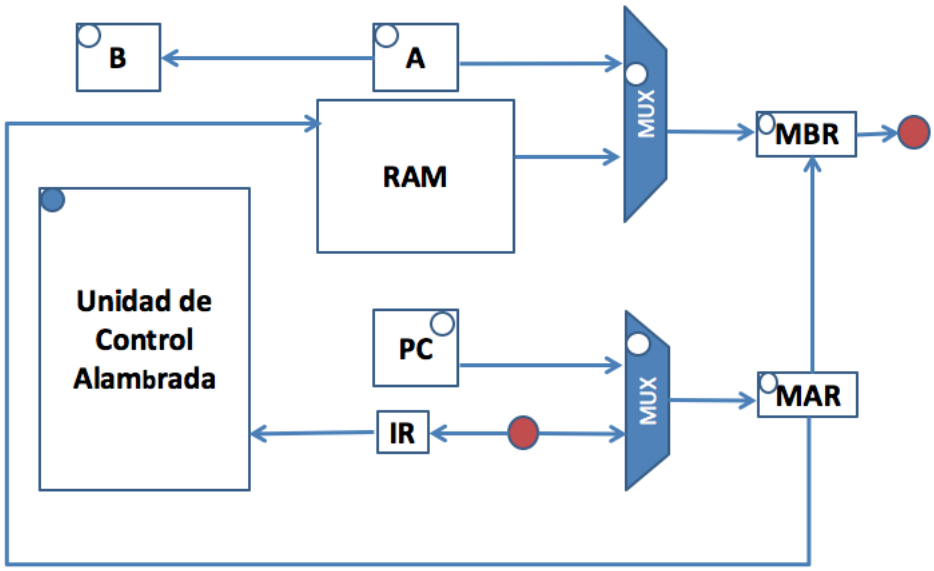

Figura 1: Diagrama de componentes

Este diagrama muestra cómo están conectados los componentes de nuestra computadora. El PC guarda siempre la siguiente dirección de memoria, y la pasa al registro MAR, para que este pueda hacer lecturas de datos. En algunos casos, estos 'datos' son instrucciones, las cuales pasan al MBR y finalmente al IR, el

cual siempre mantiene la instrucción que está siendo ejecutada. El IR es muy importante para la Unidad de Control, ya que envía las señales para decidir en qué instrucción nos encontramos. La Unidad de Control recibe, además, el pulso del reloj. La combinación de estas dos señales de entrada, nos da las señales de control.

Tenemos otros dos registros, el A y el B, utilizados para efectuar operaciones a partir de datos almacenados en sí mismos, o de lecturas hechas a la memoria.

Finalmente, usamos dos multiplexores que ayudan a mantener el control. Por un lado, debemos decidir si MAR lee de PC o del MBR. Y por otro lado, necesitamos saber si MBR lee de A o de la memoria RAM.

1.2. Puntos Críticos

Agregamos varios controles para observar cuál es el flujo en los registros y la memoria. Además, incluimos un botón RESET para regresar todo al estado inicial y facilitar las pruebas.

Sólo incluimos un display de 7 segmentos para reportar el tiempo del reloj, para el resto de los registros usamos el objeto "Probe"de logism.

- T: Control para reportar el tiempo marcado por el reloj.
- PC: Control para reportar el tiempo marcado por la computadora.
- IR: Control para el número de operación guardado en el IR.
- MAR: Control para ver la dirección almacenada en MAR.
- MBR: Información guardada en MBR.
- Registros A,B: Control para ver el valor en estos registros.

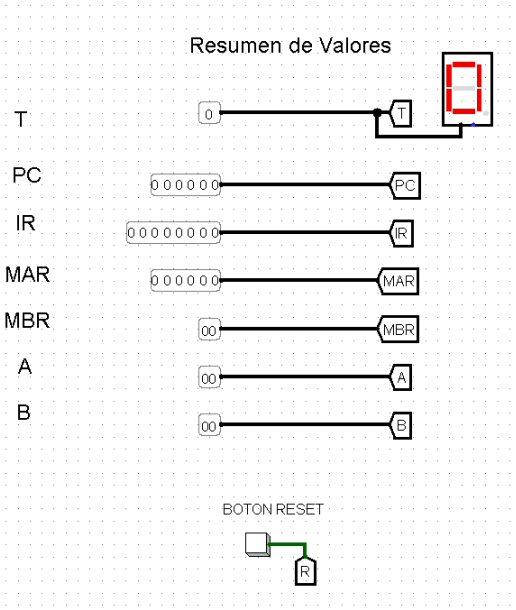

Figura 2: Reporte de Ejecución

1.3. Microoperaciones

A continuación se enlistan todas las microperaciones que hemos implemntado en la computadora:

- FETCH:

  t0 : MAR $\leftarrow$ PC

  t1 : MBR $\leftarrow$ M[MAR] ;PC $\leftarrow$ PC + 1

  t2 : IR $\leftarrow$ MBR

- MOVR:

  t3q0 : B $\leftarrow$ A; T $\leftarrow$ 0

- LD:

  t3q1 : MAR $\leftarrow$ PC

  t4q1 : MBR $\leftarrow$ M[MAR]; PC $\leftarrow$ PC +1

  t5q1 : MAR $\leftarrow$ MBR

  t6q1 : MBR $\leftarrow$ M[MAR]

  t7q1 : A $\leftarrow$ MBR; T $\leftarrow$ 0

- COMPL:

  t3q2 : B ← B'; T ← 0

- INCB:

  t3q3 : B ← B+1; T ← 0

- ADDR:

  t3q4 : B ← A + B; T ← 0

  CLR A:

  t3q5 : A ← 0; T ← 0

- SAVE dir:

  t3q6 : MAR ← PC

  t4q6 : MBR ← M[MAR] mbr tiene a dir; PC ← $PC + 1$

  t5q6 : MAR ← MBR mar tiene a dir

  t6q6 : MBR ← A

  t7q6 : M[MAR] ← MBR; T ← 0

- INCA:

  t3q7 : A ← A+1; T ← 0

- GOTO[dir]:

  t3q8 : MAR ← PC

  t4q8 : MBR ← M[MAR] mbr tiene a dir

  t5q8 : PC ← MBR; T ← 0

- JPN[dir]:

  N ← split(B,0) temp tiene N

  t3q9 : MAR ← PC

  t4q9 : MBR ← M[MAR]; PC ← PC +1

  t5q9N : PC ← MBR; T ← 0

  t5q9N : T ← 0

1.4. Multiplexores Adicionales

Al terminar de construir la computadora fue necesario agregar un multiplexor adicional para realizar las operaciones de suma y complemento. Dependiendo de cual sea la señal enviada por la unidad de control el $Multiplexor_{A,B}$ elegirá si tomar la información de A, del circuito que realiza el complemento a uno ($comp_one$) o del circuito que efectua la suma ($addr$).

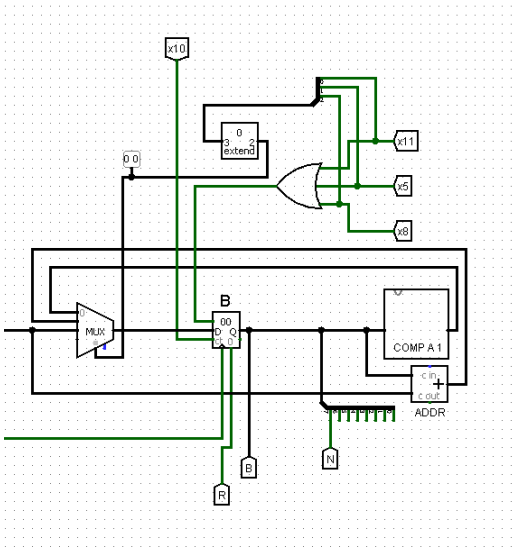

Figura 3: Multiplexor adicional

1.5. Unidad de Control Alambrada

La unidad de Control alambrada (circuito XDecoder dentro de nuestra computadora) esta diseñado para recibir la información de 19 canales (8 canales para las señales del pulso del reloj y 11 canales para las instrucciones almacenadas en IR). Dependiendo de los valores de entrada la unidad de control alambrada seleccionara una linea de salida para ejecutar una operación especifica.

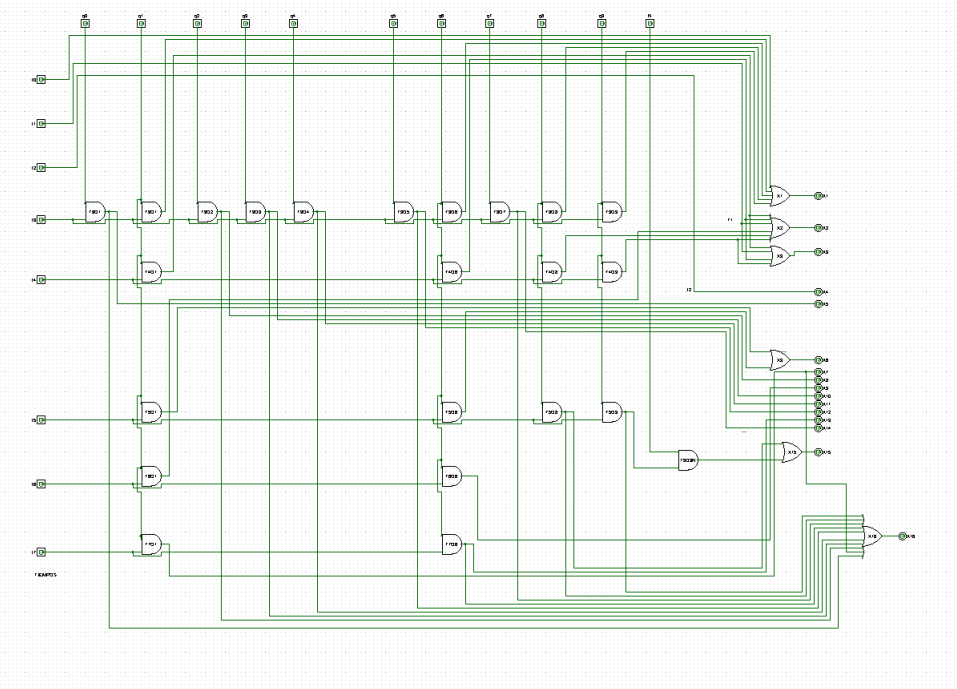

Figura 4: Unidad de Control Alambrada

Estas son las señales de control que especifican cuales son las entradas necesarias para ejecutar cada operación:

- MAR ← PC : X1 = t0 + t3q1 + t3q6 + t3q8 + t3q9
- MBR ← M[MAR] : X2 = t1 + t4q1 + t6q1 + t4q6 + t4q8 + t4q9
- PC ← PC + 1 : X3 = t1 + t4q1 + t4q6 + t4q9
- IR ← MBR : X4 = t2
- B ← A : X5 = t3q0
- MAR ← MBR : X6 = t5q1 + t5q6
- A ← MBR : X7 = t7q1
- B ← B': X8 = t3q2
- MBR ← A : X9 = t6q6

- B ← B+1: X10 = t3q3
- B ← A + B: X11 = t3q4
- A ← 0: X12 = t3q5
- M[MAR] ← MBR : X13 = t7q6
- A ← A+1: X14 = t3q7
- PC ← MBR : X15 = t5q8 + t5q9N
- T ← 0 X16 = t3q0 + t7q1 + t3q2 + t3q3 + t3q4 + t3q5 + t7q6 + t3q7 + t5q8 + t5q9
- T ← 0 X16 = t3q0 + t7q1 + t3q2 + t3q3 + t3q4 + t3q5 + t7q6 + t3q7 + t5q8 + t5q9

1.6. Implementación de computadora

Después de algunas modificaciones el siguiente diagrama muestra la estructura final de nuestra computadora

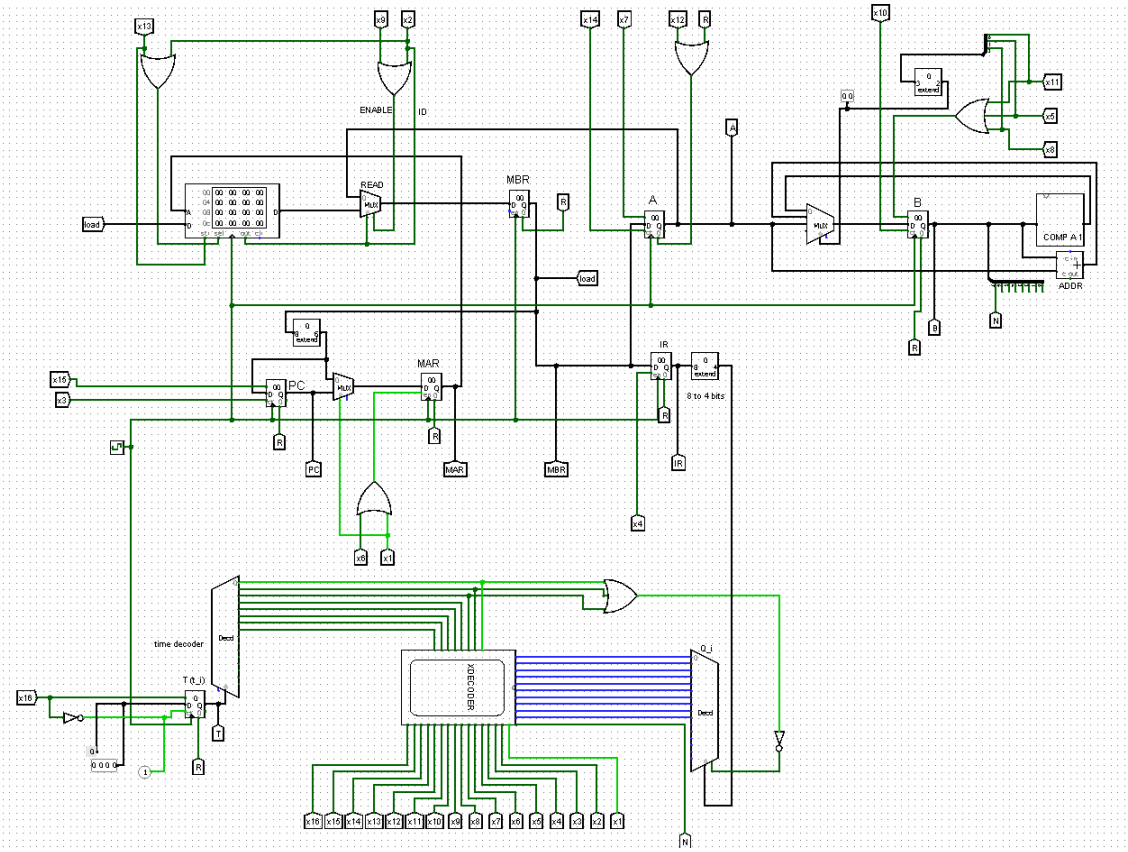

Figura 5: Computadora TANAlEPH 3.0

2. Simulación de código de Alto Nivel

Ejecutar el siguiente segmento de código.

- $X = (X > Y)?0 : X + 1;$

2.1. Código en Ensamblador

Para ejecutar el código en lenguaje ensamblador primero descomponemos la instrucción $X = (X > Y)?0 : X + 1;$ en cada uno de sus componentes.

```
If X>Y:
    return 0
Else:
    return X+1
```

Primero necesitamos cargar los valores que deseamos comparar. Tenemos el valor de X esta en la dirección de memoria 20 y y el valor de Y en la dirección de memoria 21. Se utilizará la instrucción **LD 20** para obtener el valor de X y guardarlo en el registro A. Como se requerirá la información de dos valores se deberá liberar la memoria de A usando las instrucción **MOVR** y guardar el contenido en el Registro B. Posteriormente Cargamos Y en A y se procederá a realizar la comparación.

Dadas las operaciones que puede realizar nuestra computadora debemos idear la forma de efectuar una comparación. Primero se utilizará la función complemento a 1 de B (registro con el valor de X), le sumamos 1 utilizando la función **INC B** para obtener el complemento a 2 y por último usaremos la función de suma (**ADDR**) para sumar el complemento a dos de X y el número guaraddo en el Registro A con el valor de Y. Hasta este punto, tenemos guardado en B el valor de -X+Y

Nos interesa saber si el número guardado en B es positivo o negativo (equivalente a si X es menor o mayor a Y). La operación **JPN** hace uso de esta información, con el bit de signo. Dependiendo del valor del bit más significativo se ejecutarán diferentes operaciones:

Cuando el bit es 0 (X es mayor a Y) entonces cargamos nuevamente el valor de X e incrementamos en uno su valor (**INC A**) para después guardarlo en la memoria(**SAVE**). Cuando el bit sea 1 entonces la diferencia es negativa y debemos eliminar el valor de A igualándolo a 0. El 0 será guardado en memoria. La elección de cuál operación efectuar se hace a partir de guardar las instrucciones para $N = 0$ en las direcciones de memoria consecutivas, y las instrucciones para $N = 1$ en otra dirección de memoria, y efectuar el **JPN** a esa dirección.

A continuación se muestra el código para esta operación el lenguaje ensamblador. Para facilitar la lectura, los números están en sistema decimal.

X está en la dirección 20
Y en la dirección 21

```
LD 20
MOVR
LD 21
COMPL
INCB
ADDR
JPN 15
```

Ejecución $X < Y$ el bit más significativo es 1:

```
CLR A
SAVE 20
```

GOTO 25

Ejecución $X > Y$ el bit más significativo es 0:

LD 20

INC A

SAVE 20

GOTO 15

2.2. Ejecución en computadora

A continuación se presenta una tabla con las instrucciones que deberán estar guardadas en la RAM para ejecutar el codigo de alto nivel.

	Valor en Memoria	Ubicación
Posición 0	01	A=X
Posición 1	14	
Posición 2	00	B=X
Posición 3	01	A=Y
Posición 4	15	
Posición 5	02	B=-X-1
Posición 6	03	B=-X
Posición 7	04	B=-X+Y
Posición 8	09	
Posición 9	16	if -X+Y >0
Posición 10	01	A=X
Posición 11	14	
Posición 12	07	A=X+1
Posición 13	06	X=X+1
Posición 14	14	
Posición 15	08	
Posición 16	$0f$	
Posición 17		
Posición 18		
Posición 19		
Posición 20	X	
Posición 21	Y	
Posición 22	05	A=0
Posición 23	06	X=0
Posición 24	14	
Posición 25	08	
Posición 26	18	

Cuadro 1: Información en RAM para ejecutar código de alto nivel

3. Conclusiones

En esta práctica trabajamos con los componentes principales de una computadora y comprendimos la manera en la que funcionan de manera conjunta para efectuar operaciones. Es importante conocer las conexiones que existen en una computadora porque el funcionamiento depende de la sincronización entre las operaciones, el pulso del reloj y la lógica que sigue la computadora. La parte más retadora del proyecto fue crear las señales de control ya que el diseño de la computadora depende completamente de esta base. Es indispensable seguir un orden para evitar hacer operaciones innecesarias y tener visibilidad sobre lo que esta sucediendo en la computadora. El trabajo también fue de gran ayuda para entender como funciona la memoria y la manera en la que la información se transfiere entre los diferentes registros.