Practica 1: Diseño de una computadora basada en acumulador

Arquitectura de Computadoras

Héctor Hugo Huipet Hernández
Maestría en Ciencias de la Computación
ITAM
Ciudad de México, México
hugo.huipet@gmail.com

I. Introducción

En esta práctica se refuerzan los conocimientos vistos en clase sobre el diseño de una computadora basada en acumulador. Se construyó una computadora de un solo registro (acumulador) que responde a cuatro operaciones de control y a 16 operaciones aritmética-lógicas mediante una ALU. Se realizó una simulación utilizando el programa logisim y se comprobó el funcionamiento correcto del diseño mediante un programa

A lo largo de los años han existido distintos tipos de modelos de máquina: de Pila, basadas en acumulador, de Registro memoria y de registro a registro.

También existen distintos números de direcciones para estos modelos:

A. Cero direcciones

– Arquitecturas basadas en stack. Todos los operandos son implícitos

– Relativamente sencillo ensamblador para operaciones aritméticas (Notación polaca inversa)

• Otras operaciones (por ejemplo, manejo de strings)

– Instrucciones muy cortas, programas (ensamblador) muy grandes

– La pila es un cuello de botella. Aún si parte está en el CPU, se puede acceder un elemento a la vez

B. Una dirección

– Arquitecturas basadas en acumulador

Acc ← Acc func Op1

– Código claro y sencillo

– Poco hardware

– Muchos accesos a memoria (o cache)

– Instrucciones cortas, programas relativamente largos

C. Dos o tres direcciones

• Op2 ← Op2 func Op1

• Res ← Op1 func Op2

– Formato de instrucción largo

– Programas más cortos

– Muy eficiente si los operandos son registros

II. Diagrama de Bloques

A continuación se presentan los diagramas de bloques para la computadora basada en acumulador así como los componentes desarrollados para el funcionamiento de la misma, la unidad de control y la unidad aritmética lógica.

A. Unidad Aritmética Lógica.

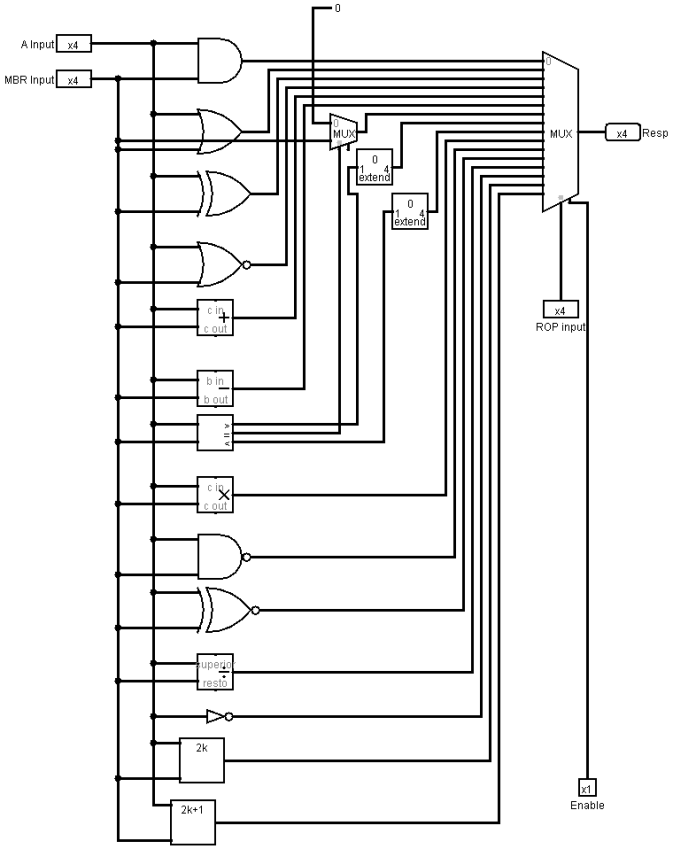

Figura 1: Diagrama de Bloques de la Unidad Aritmética Lógica

La unidad Aritmética Lógica está compuesta por las siguientes 16 operaciones: and, or, xor, nor, suma, resta, comparación modificada, mayor que, menor que, multiplicación, nand, xnor, división, negación detección de paridad par en bits y detección paridad impar en bits.

1) Comparación modificada

Esta operación de comparación se realizó con el motivo de conseguir realizar un programa que implementara un ciclo y pudiera terminar una vez alcanzado un valor, el principio básico es el de una comparación excepto que si son iguales regresa el valor de igualdad y si son diferentes regresa 4 bits en 0.

B. Unidad de Control

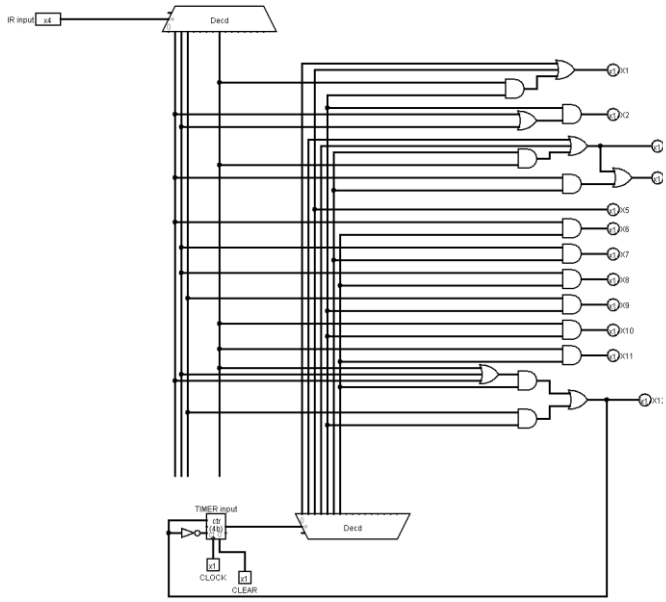

Figura 2: Diagrama de Bloques de la Unidad de Control

Las señales de control fueron calculadas mediante el análisis de las microoperaciones necesarias para cada operación.

- ✓ MAR ←PC: X1= $t_0+t_2+q_3t_4$
- ✓ MAR ← MBR: X2= $t_4(q_0+q_1)$
- ✓ PC ← PC +1: X3=$t_1+t_3+q_3t_5$
- ✓ MBR ← M: X4=X3+q_0t_5
- ✓ IR ← MBR: X5=t_2
- ✓ A ← MBR: X6=q_0t_6
- ✓ MBR ← A: X7= q_1t_5
- ✓ M ← MBR: X8= q_1t_6
- ✓ PC ← MBR: X9= q_2t_4
- ✓ ROP ← MBR: X10= q_3t_4
- ✓ A ← A TIPO MBR: X11= q_3t_6
- ✓ T ←0: X12= q_2t_4+ $t_6(q_0+q_1+q_3)$

C. Computadora Basada en acumulador

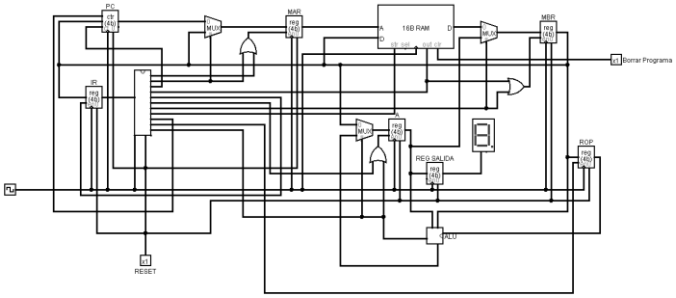

Figura 3: Diagrama de Bloques de la computadora basada en acumulador

Los componentes se armaron de acuerdo a la configuración de una computadora basada en acumulador, es por esto que sólo existe un registro de entrada en la ALU, el cual es el registro A y la otra entrada proviene directamente de la memoria a través del registro MBR.

Las señales de control de la UC se conectan a los dispositivos que intervienen para cada operación y las que comparten uso con otra señal son incluidas mediante compuertas OR en los ENABLE de cada registro y/o como señal de selección en los multiplexores, según sea el caso.

Tanto el registro de salida como el display hexadecimal sólo son una manera de visualizar lo que está sucediendo con el registro A mientras corre el programa.

III. Programa en Lenguaje de Alto Nivel

```
X=8
While X != 12 DO:
      X++;
END
```

Instrucción	Código	Comentario
LEE 14	0001 1110	A← M[14]
OPERA + 1	1000 0100 0001	A ← A + 1
GUARDA 14	0010 1110	M[14] ← A
OPERA = 12	1000 0110 1100	A ←A==* 12
GUARDA 13	0010 1101	M[13] ← A
VETEA 0	0011 0000	PC ←0

Tabla 1: Descripción del código, la última instrucción aunque tiene una dirección se reescribe dependiendo el resultado de la comparación modificada

El programa realiza una acumulación de un valor inicial (en este ejemplo será 8) hasta llegar al valor 12, a partir de ahí el

programa se mantiene en un ciclo infinito hasta que ocurra una interrupción externa.

El código para cargar el programa en la memoria de logisim en hexadecimal es:

1e8412e86c2d3080

IV. Opciones para unidad de control

En el diseño de la unidad de control, las señales de operación dependen básicamente del orden que el diseñador eligió para las microinstrucciones. También si la unidad sólo realiza las compuertas lógicas para las señales de entrada o junto con la decodificación de la señal del registro de instrucciones y el temporizador. Sin embargo en clase se explicó que en sí esta unidad de control es poco flexible por lo que no hay muchas opciones para variar el diseño.

V. ALU de 48 operaciones

En el caso de que se requiera utilizar una ALU de 48 operaciones, se debe tomar en cuenta que algunos de los componentes de la computadora diseñada en esta práctica deben ser modificados. Iniciando con el tamaño de los registros, ya que el tamaño actual de cuatro bits no permite generar la señal para números mayores a 15 en el registro de Operaciones, es por esto que se necesitaría un registro de operaciones de al menos 6 bits.

Esto también implica que la memoria deba ajustarse a 6 bits de datos al menos, o tratar de adaptar la misma memoria para hacer referencia a las operaciones (una operación estaría descrita en dos localidades de memoria) aunque esto puede resultar más complejo y el tamaño no permitiría tener muchas instrucciones.

VI. Tres direcciones

Para poder aprovechar esta tipo de direccionamiento además de tener más registros a la entrada de la ALU o para almacenar resultados (la arquitectura se modificaría), de nuevo tendríamos que aumentar los bits datos y de dirección en la memoria, ya que el tamaño de palabra aumenta y difícilmente se puede escribir un programa que tenga sentido con tres direcciones en una memoria tan pequeña como la de esta práctica. También las operaciones de control deben ser rescritas para tomar en cuenta el nuevo número de direcciones.