Universidad de Guadalajara

Centro universitario de ciencias exactas e ingeniería división de electrónica y computación

Seminario de solución de problemas de Arquitectura de Computadoras

Reporte Actividad 12

Implementación del ciclo fetch y

Propuesta para el proyecto final

Alumno: Oswaldo Luna Grados

Código: 211718256

Sección: D14

Profesor: J. Ernesto López Arce Delgado

Introducción

Para poder ejecutar un programa en una computadora u otro sistema computacional es el CPU que realiza el ciclo Fetch Decode Execute. Es el periodo de tiempo del CPU para ejecutar una instrucción de lenguaje máquina.

El ciclo se forma de las siguientes etapas:

Traer la instrucción: se recibe la instrucción desde la memoria y se almacena en el registro del CPU para instrucciones.

Decodificar la instrucción: se reconoce el modo de direccionamiento de la instrucción y su ubicación de datos a tratar ya sea de memoria, registro o instrucción directamente.

Los modos de direccionamiento son la forma para indicar la posición de memoria en que se está situado el dato en la instrucción. Existe el inmediato, directo, indirecto y relativo. El inmediato el dato está en la instrucción. El directo la instrucción contiene la dirección de la memoria donde está el dato. Indirecto se refiere a que la dirección no es del dato sino de la posición de memoria que contiene ese dato. El relativo la dirección del dato se consigue sumando la dirección con una magnitud fija contenida en un registro especial.

Carga de Parámetros: Se ejecuta la lectura, cargando todos los datos identificados en el paso anterior.

Ejecutar: Se ejecuta la instrucción ya configurada, realiza la tarea indicada, ya sea una suma, resta, almacenar información, extraer información etc.

Almacenar: Se almacena el resultado obtenido de ejecutar la instrucción, por ejemplo, el resultado de una suma o un índice (Número) como resultado de éxito de almacenar u obtener información de un archivo, entre otros.

Actualizar PC: Esta etapa es la de actualizar el registro PC (Program Counter) que contiene la siguiente dirección a ejecutar.

El ciclo ha tenido algunas alteraciones hoy en día mejorando para que pueda ser multitareas y reduciendo el tiempo de ejecución de instrucciones con concurrencia y paralelismo.

Objetivo

Para la ejecución de una instrucción las instrucciones se sincronizan mediante una señal de reloj. Se usa un contador de programa para establecer la instrucción a leer en la memoria de instrucciones. Este contado se ejecuta cada señal de reloj (sumando 4 por ser una memoria de 32 bits. Es decir que la memoria cada celda tiene un ancho de 8 bits y necesita sumar 4 celdas consecutivas para tener la instrucción completa el cual es recibido por el banco de registros y demás componentes de procesador.

Desarrollo

El código de verilog del contador del programa (PC) es el siguiente imagen, donde

Recibe una señal de reloj para que el valor de entrada sea igual a la salida.

```
1   `timescale 1ns /1ps
2   module PC(
3       input [31:0] dir,
4       input clk,
5       output reg [31:0]dirOut
6   );
7   always@(posedge clk)
8   begin
9       dirOut=dir;
10  end
11  endmodule
```

Imagen 1. Modulo PC (contador del programa).

La entrada y salida es de 32 bits donde se envía cada señal de subida de la señal de reloj.

Para que el valor de entrada del PC cambie, se pone un sumador en la salida del PC de 4 para que el valor avance 4 bits por ciclo. En la imagen 2 se puede observar el módulo de este sumador.

```
1   `timescale 1ns /1ps
2   module add4(
3       input [31:0]in,
4       output reg [31:0]out
5   );
6   always@(*)
7   begin
8       out=in+4;
9   end
10  endmodule
```

Imagen 2. Sumador de 4 bits.

La salida del PC también envía el resultado a la memoria de instrucciones que suma 4 celdas y cada una de 8 bits para hacer la instrucción de 32 bits. Lo que es la razón de por qué avanza el contador de programa de 4 en cuatro celdas.

```
1   `timescale 1ns /1ps
2   module memInstruc(//memoria Asincrona
3       input [31:0]index,
4       output [31:0] outS
5   );
6   reg [31:0] result;
7   reg [7:0]celda[255:0];
8   assign outS=result;
9   always@(*)
10  begin
11      result[31:24]=celda[index];
12      result[23:16]=celda[index +1];
13      result[15:8]=celda[index+2];
14      result[7:0]=celda[index+3];
15  end
16  endmodule
17
```

Imagen 3. Memoria de instrucciones.

Juntando con los componentes de la act 11 termina como la imagen 4.

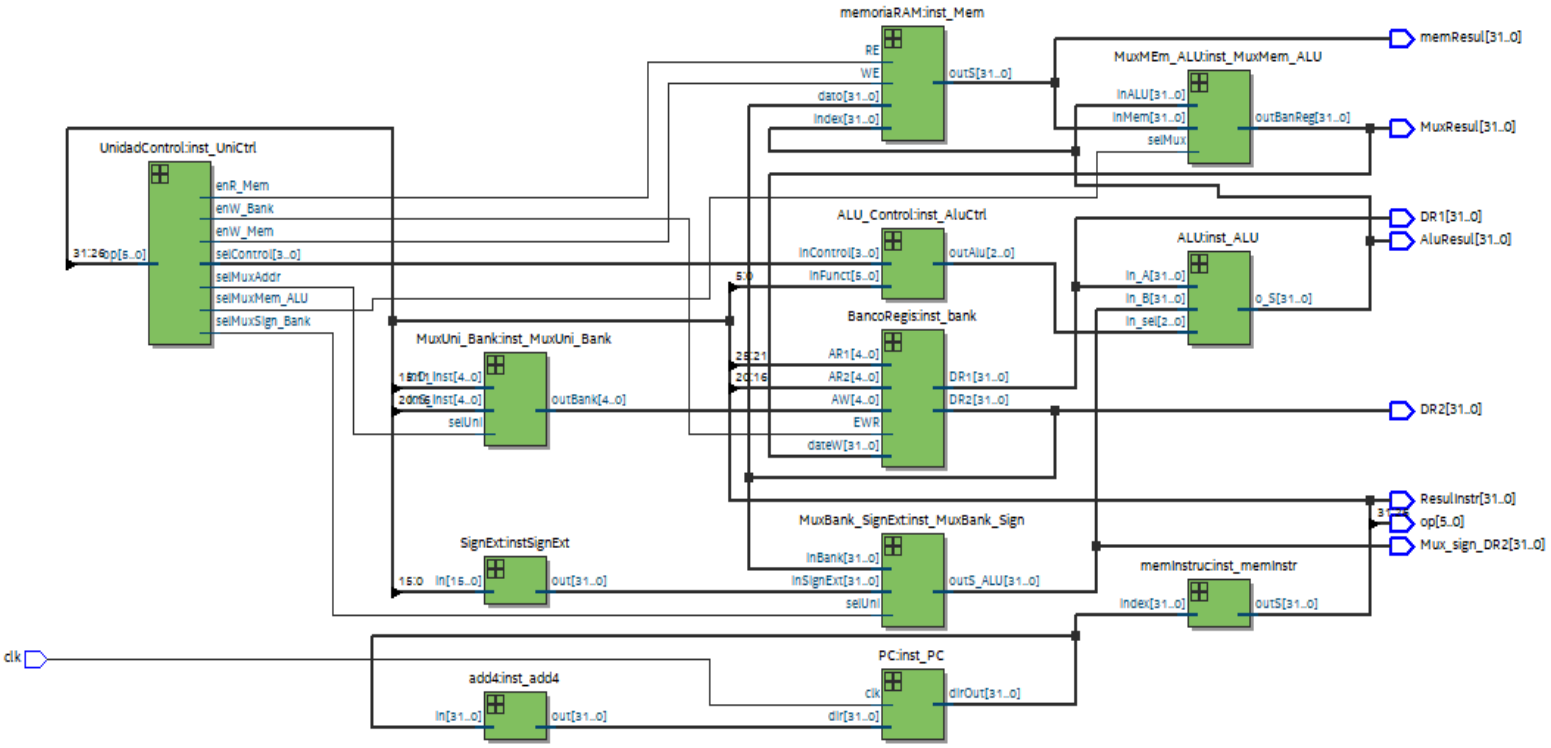

Imagen 4. Diagrama digital de componentes del topLevel.

Propuesta de programa a implementar en el proyecto final

El programa a implementar es un cifrado vigenère la cual hay un mensaje y una llave a usar para concatenar y dar un nuevo mensaje simétrico. Cada letra del abecedario tiene un valor numérico y consiste en hacer una suma, letra por letra del mensaje con la llave, cuando llegamos al límite se empieza la llave otras vez hasta terminar con el ultimo carácter del mensaje.

Los símbolos en el mensaje son las letras del alfabeto donde k= $\{k_0,\ k_1,\ldots,\ k_{D-1}\}$ donde D es la longitud. La siguiente forma expresa la transformación lineal de cifrado:

$$E(k_i) = (k_i + X_i) mod\ L$$

Donde k_i es la letra en el texto a cifrar en la posición de i, X_i es el carácter de la llave en la misma posición de carácter del texto a cifrar. Y L es el tamaño del alfabeto. Y para descifrar se resta el valor de la llave del mensaje encriptado.

Referencias

http://arquitecturadecomputadorasheiner.blogspot.com/2017/10/ciclo-fetch-decode-execute.html

https://es.scribd.com/doc/30420235/Ciclo-Fetch#fullscreen&from_embed

Singh, Simon (1999). "Chapter 2: Le Chiffre Indéchiffrable". The Code Book. Anchor Book, Random House. ISBN 0-385-49532-3.

R. Morelli, R. Morelli, Historical Cryptography: The Vigenere Cipher, Trinity College Hartford, Connecticut

https://es.wikipedia.org/wiki/Cifrado_de_Vigen%C3%A8re

http://www.dma.fi.upm.es/recursos/aplicaciones/matematica_discreta/web/aritmetica_modular/polialfabeto.html