Cesar Carlos Robles Martínez

Diseño de Microprocesadores

Maestría en Diseño Electrónico

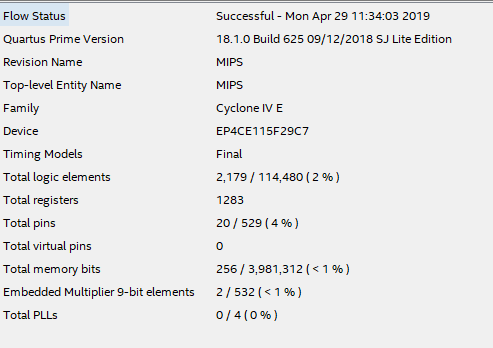
Expediente: 706934

MIPS Multiciclo Factorial

**Practica 01**

Realizar la implementación del procesador MIPS multi-ciclo visto en clase. Este procesador debe ser capaz de ejecutar el programa que implementa el factorial de. El valor de se trasmitirá a través de una terminal serial, se recomienda el uso de docklight, pero puede ser Teraterm o PuTTy. será recibida por el procesador por una UART la cual se tiene que integrar como periférico. Una vez procesado el factorial, el resultado será trasmitido por la UART hacia la terminal en la PC. Note que la UART solo puede transmitir 8 bits a la vez, por lo cual para poder transmitir los 32 bits resultantes se tiene que hacer en paquetes de 8 bits, comenzando por los 8 bits más significativos.

**Resultado de síntesis**



Frecuencia máxima de operación



**Simulación**

A continuación, ser presenta la simulación de las instrucciones del MIPS multiciclo

Instrucción lui:

PC = 400000

Op y Funct correspondientes a la instrucción lui

WB\_Extend

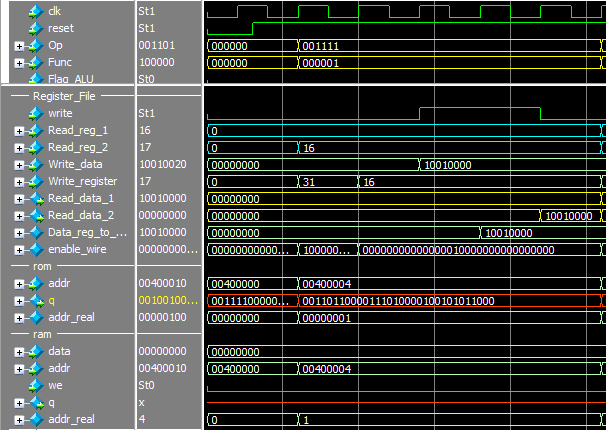
WB

EX

IF

ID

IF



**Registel file:**

**Read\_reg\_2**; Selección de registro a leído = 16

**Write\_data =** Dato a ser escrito

**Write\_register =** número de registro a ser escrito = 16

**Registel file:**

**Read\_data\_2;** Lectura del dato escrito en el register file

**Write;** habilita la escritura en el regirter file = 1

**vv**

Instrucción ori:

PC = 400004

Op y Funct correspondientes a la instrucción lui

WB\_Extend

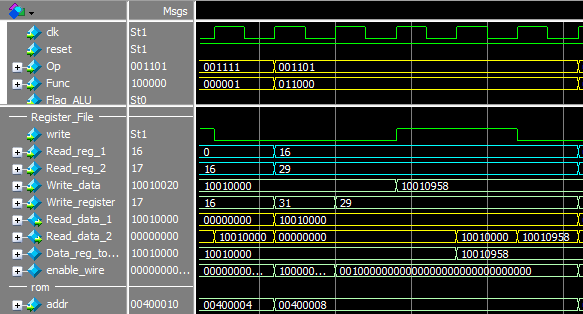
WB

EX

IF

ID

IF



**Registel file:**

**Read\_reg\_2**; Selección de registro a leído = 29

**Write\_data =** Dato a ser escrito

**Write\_register =** número de registro a ser escrito = 29

**Registel file:**

**Read\_data\_2;** Lectura del dato escrito en el register file

**Write;** habilita la escritura en el regirter file = 1

**vv**

Instrucción addiu:

PC = PC = 400008

Op y Funct correspondientes a la instrucción lui

WB\_Extend

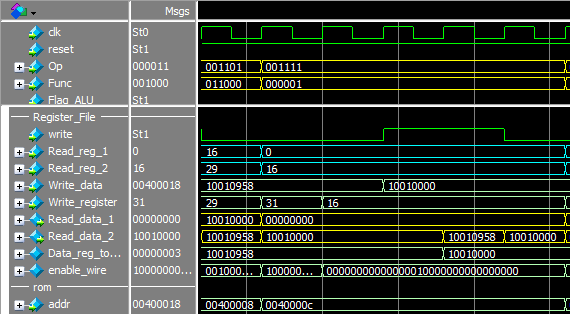
WB

EX

IF

ID

IF



**Registel file:**

**Read\_reg\_2**; Selección de registro a leído = 29

**Write\_data =** Dato a ser escrito

**Write\_register =** número de registro a ser escrito = 29

**Registel file:**

**Read\_data\_2;** Lectura del dato escrito en el register file

**Write;** habilita la escritura en el regirter file = 1

**vv**

Instrucción jal y j:

PC = 400014

Op y Funct correspondientes a la instrucción lui

WB\_Extend

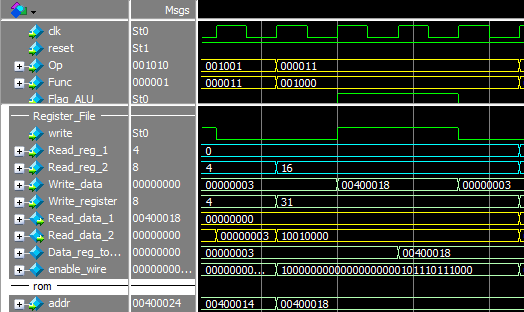
WB

EX

IF

ID

IF



**Registel file:**

**Read\_reg\_2**; Selección de registro a leído = 31

**Write\_data =** Dato a ser escrito

**Write\_register =** para la instrucción es requerido que sea escrito el registro 31

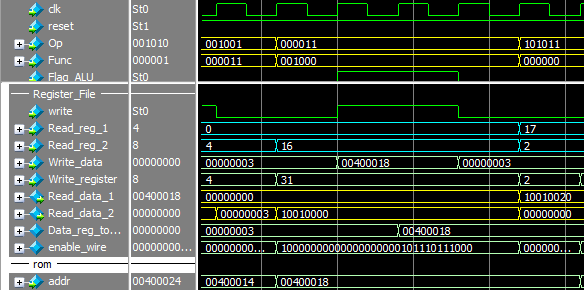
**Registel file:**

**Read\_data\_2;** Dados los problemas de la maquina de estados, no se puede leer el valor que está escrito en el registro 31, por consiguiente, el valor usado de este registro está en cero

**Write;** habilita la escritura en el regirter file = 1

**vv**

El **PC** que es la dirección que recibe la **rom**, es actualizado antesde poder ejecutar la instrucción jal, siendo esta la problemática que se presenta en la máquina de estados al no contar con una condicionante para saber qué tipo de instrucción se ejecutara antes de entrar en la etapa de IF <<*Instructio Fetch>>*



Instrucción sw:

Op y Funct correspondientes a la instrucción lui

WB\_Extend

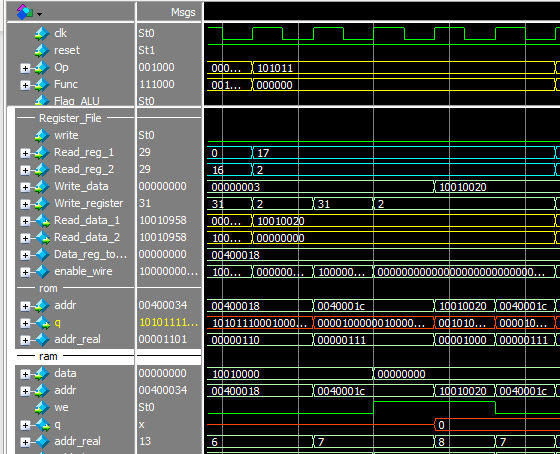
WB

EX

IF

ID

IF



**Memoria RAM:**

**Address\_real;** Es la dirección en la cual se escribirá el dato

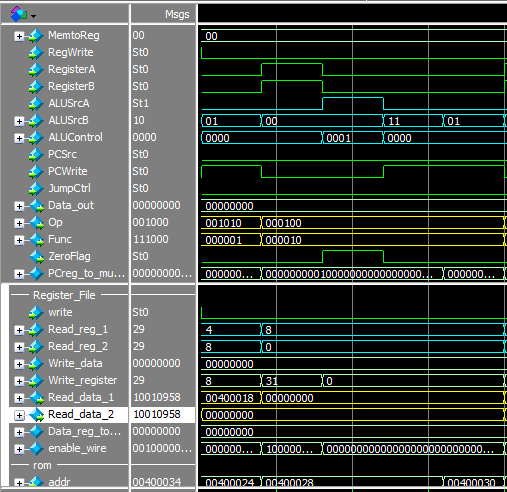
**we;** habilita la escritura en la memoria ram

**addr**: es la dirección proveniente de la ALU, al memento de ser calculada.

**vv**

Instrucción beq:

Op y Funct correspondientes a la instrucción lui



**Write\_registe:** Registro destino

**Read\_data :** El resultado es cero dada la resta realizada por la ALU, cuyo valor será grabado en el registro 0

**Zero\_Flag :** se activa cuando la resta proporciona como resultado un cero 0

A partir de esta instrucción la ejecución se corrompe por los errores de las instrucciones jal y j

**Avances hasta el momento**

Las instrucciones que no se han logrado ejecutar en el procesador son *jal* y *j* dado que en la etapa de IF <<*Instruction Fetch>>* el valor del PC <<*Program Counter>>* ya se ha incrementado, lo cual hace que la instrucción siguiente al *jal* sea ejecutada, por lo tanto, la ejecución de se ve afectada. Esto quiere decir que, se necesita modificar la máquina de estados y para que entre en una condicionante adicional antes del IF, para que el nuevo valor no sea grabado en el PC antes de poder saber si la instrucción es de tipo J y se pueda controlar, qué tipo de instrucción se desea ejecutar. Por otra parte, las instrucciones tipo R e I, sí se ejecutan.

Tomando en cuenta lo antes mencionado, es necesario realizar ajustes en la maquina de estado para que el tiempo coincida entre las etapas, además de reorganizar la microarquitectura de tal manera que estos cambios queden plasmados en ella.

Se pretende continuar trabajando sobre esta practica para hacer que el MIPS logre ejecutar las instrucciones faltantes, asimismo, poder ejecutar el factorial de forma satisfactoria.