Universidad Escuela Colombiana de Ingeniería Julio Garavito
Sistemas Operativos y Plataformas Computacionales
Proyecto

Objetivos

- Implementar en el simulador de hardware la Memoria y la Unidad Aritmética de una version de MARIE de 32 bits

---

1. **Circuitos**: Implementar en el simulador de hardware los siguientes circuitos:

   - **Complemento a 1**: Implementar un circuito en HDL que realice el complemento a 1 de un número de 32 bits. Utilice en la entrada dos vectores de 16 bits (**a1** y **a0**) y en la salida dos vectors de 16 bits (**b1** y **b0**). El archivo del circuito debe llamarse **complement1.hdl**
   - **Incremento de 32 bits**: Implementar un circuito en HDL que realice el incremento en uno de un número de 32 bits. Utilice en la entrada dos vectores de 16 bits (**a1** y **a0**) y en la salida dos vectores de 16 bits (**b1** y **b0**). El archivo del circuito debe llamarse **increment.hdl**
   - **Suma de 32 bits**: Implementar un circuito en HDL que realice la suma de dos numeros de 32 bits. Utilice en la entrada cuatro vectores de 16 bits (**a1**, **a0**, **b1** y **b0**) y en la salida un vector de 16 bits (**c1** y **c0**). El archivo del circuito debe llamarse **add.hdl**
   - **Resta de 32 bits**: Implementar un circuito en HDL que realice la resta de dos numeros de 32 bits usando complemento a 2. Utilice en la entrada cuatro vectores de 16 bits (**a1**, **a0**, **b1** y **b0**) y en la salida dos vector de 16 bits (**c1** y **c0**). El archivo del circuito debe llamarse **subtract.hdl**
   - **Memoria**: Implementar un circuito en HDL para una memoria de 4096 posiciones y 32 bits en cada posicion. Use para la dirección un vector de 12 bits (**a**), use para la entrada dos vectores de 16 bits (**in1** y **in0**), use para la salida dos vectores de 16 bits (**out1** y **out0**) y use un bit (**w**) para indicar si se escribe a la memoria o no. El archivo del circuito debe llamarse **memory.hdl**
   - **Unidad Aritmética**: Implementar un circuito en HDL para la unidad aritmetica. Utilice en la entrada seis vectores de 16 bits (**MEMin1**, **MEMin0**, **MBRin1**, **MBRin0**, **ACin1** y **ACin0**), dos vectores de 12 bits (**MARin**, **PCin**), un vector de 8 bits (**inREG**) y un vector de 4 bits (**inst**).

     Utilice en la salida cuatro vectores de 16 bits (**MBRout1**, **MBRout0**, **ACout1**, **ACout0**), dos vectores de 12 bits (**MARout**, **PCout**) y un vector de 8 bits(**outREG**). Las salidas que no se utilicen en la instrucción a ejecutar deben tomar el valor cero en todos los bits.

     La entrada **inst** contiene el código de la instrucción, **MARin** contiene la dirección en la instrucción, **MBRin** contiene el valor de **Memory**[**MARin**] y **MEMin** contiene el valor de **Memory**[**MBRin**]. La salida **MBRout** contiene el valor que debe ser colocado en la memoria en la dirección **MARout**. En la segunda página se encuentra detallado el **RTN** para cada instrucción. El archivo del circuito debe llamarse **AU.hdl**

   Nota: Para la implementación de cualquiera de los circuitos solo puede utilizar los circuitos **And**, **Or**, **Xor**, **Not**, **Bit**, **HalfAdder** y **FullAdder** de la carpeta BuiltIn y circuitos propios.

inst	**Register Transfer Notation (RTN)**
0000	ACout ← MARin + 1 MARout ← MARin MBRout ← PCin PCout ← MARin + 1
0001	ACout ← MBRin
0010	MARout ← MARin MBRout ← ACin
0011	ACout ← ACin + MBRin
0100	ACout ← ACin - MBRin
0101	ACout ← inREG
0110	outREG ← ACin
0111	

1000	if MARin[11-10] = 00 then
	  if ACin < 0 then PCout = PCin + 1 else PCout = PCin
	else if MARin[11-10] = 01 then
	  if ACin = 0 then PCout = PCin + 1 else PCout = PCin
	else if MARin[11-10] = 10 then
	  if ACin > 0 then PCout = PCin + 1 else PCout = PCin

1001	PCout ← MARin
1010	ACout ← 0
1011	ACout ← ACin + MEMin
1100	PCout ← MBRin
1101	ACout ← MEMin
1110	MARout ← MBRin MBRout ← MEMin
1111	