UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Ciencias Exactas, Físicas y Naturales

TRABAJO PRÁCTICO N° 3
DE ARQUITECTURA DE COMPUTADORAS

Implementación de un procesador monociclo simple sin saltos: BIP

Integrantes:

SORIANO, JUAN. 37.753.154

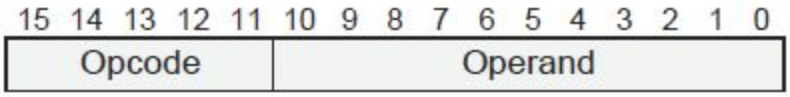
VIGNOLLES, IVAN. 36.407.987

DOCENTE: Rodriguez, Santiago

Introducción y objetivos

En lenguaje de descripción de hardware Verilog, se implementó un procesador simple, sin saltos monociclo: BIP, basado en el paper “A Basic Processor for Teaching Digital Circuits and Systems Design with FPGA” de Pereira y otros.

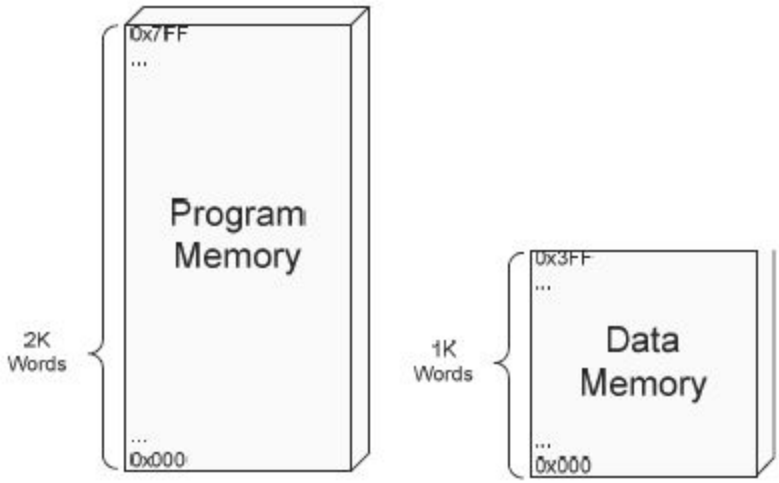
El procesador trabaja con operaciones de 16 bits como sigue:



El conjunto de instrucciones que se implementan es:

Operation	Opcode	Instruction	Data Memory (DM) and Accumulator (ACC) Updating	Program Counter (PC) updating	Affected Flags	BIP Model
Halt	00000	**HLT**		PC ← PC		I, II
Store Variable	00001	**STO** operand	DM[operand] ← ACC	PC ← PC + 1		I, II
Load Variable	00010	**LD** operand	ACC ← DM[operand]	PC ← PC + 1		I, II
Load Immediate	00011	**LDI** operand	ACC ← operand	PC ← PC + 1		I, II
Add Variable	00100	**ADD** operand	ACC ← ACC + DM[operand]	PC ← PC + 1	Z,N	I, II
Add Immediate	00101	**ADDI** operand	ACC ← ACC + DM	PC ← PC + 1	Z,N	I, II
Subtract Variable	00110	**SUB** operand	ACC ← ACC - DM[operand]	PC ← PC + 1	Z,N	I, II
Subtract Immediate	00111	**SUBI** operand	ACC ← ACC - operand	PC ← PC + 1	Z,N	I, II

La memoria está direccionada como se muestra a continuación:



Finalmente, el diagrama de bloques del procesador es como se presenta a continuación:

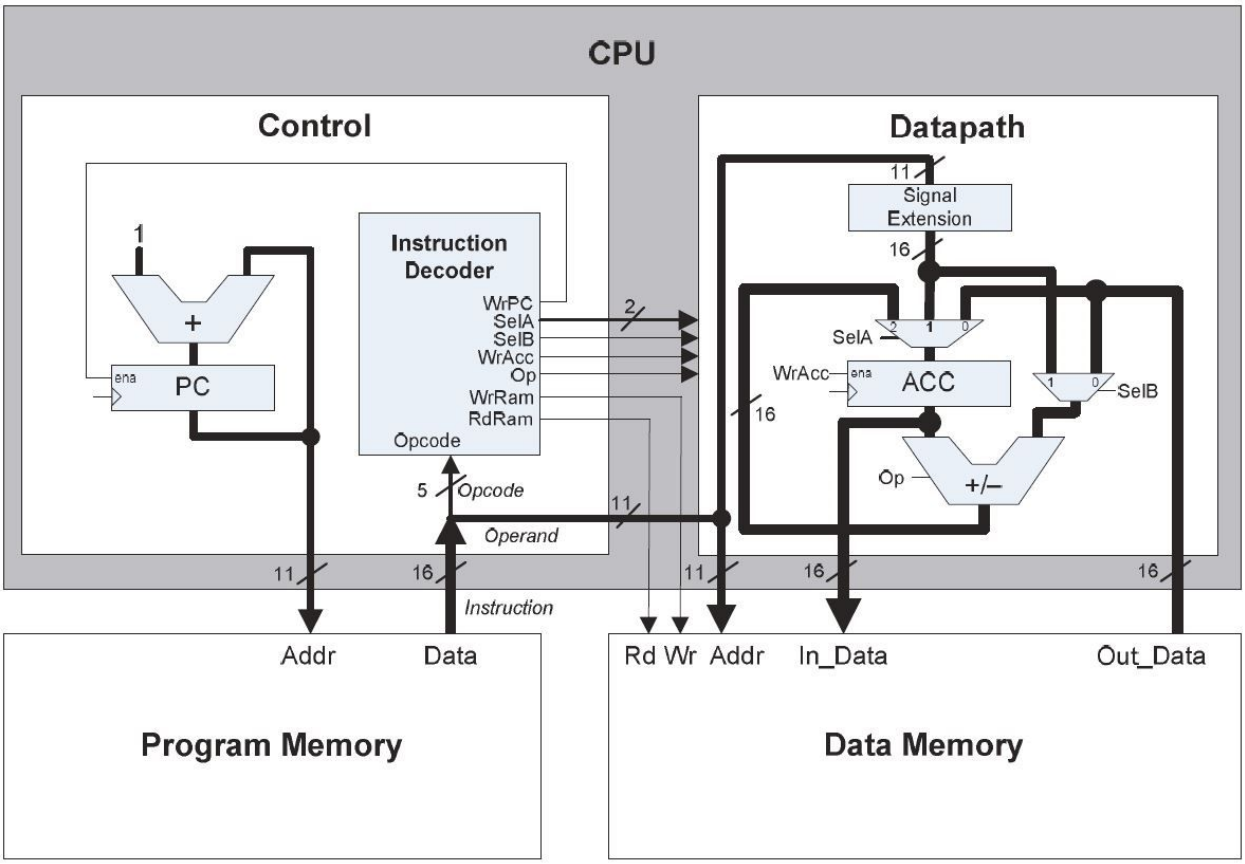


Figure 3. BIP I organization

Una vez implementado el procesador, se debía enviar por el módulo UART implementado en el trabajo práctico 2, un contador con la cantidad de clocks, el código de la instrucción y el valor del registro acumulador.

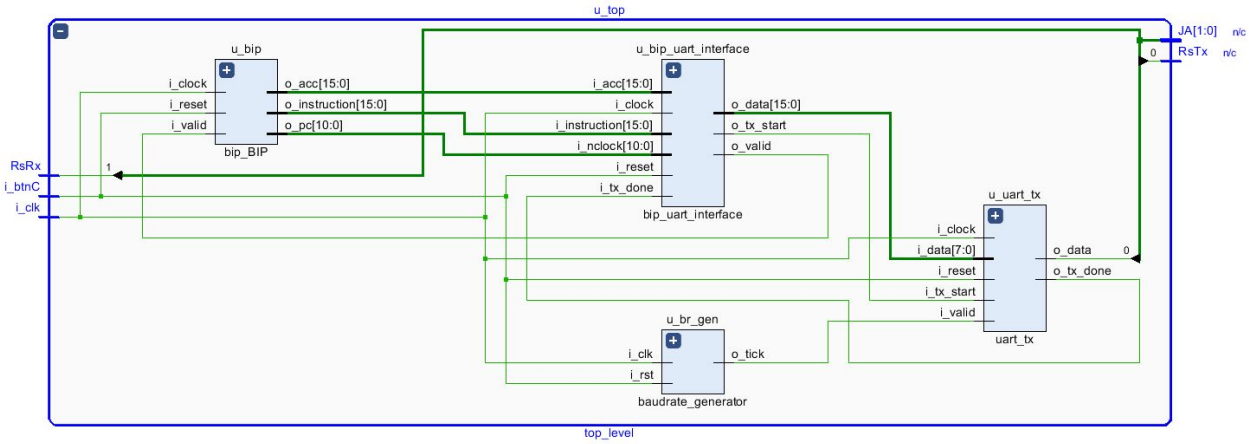
Resultados

Se conectó el BIP con el UART y se corrió una ejecución del programa con las siguientes instrucciones:

```
mem_bank[0] = 16'b00010_000_0000_0001 ; //Load variable 0x01 => ACC=0x01
mem_bank[1] = 16'b00101_000_0000_0010 ; //Add immediate +0x2 => ACC=0x03
mem_bank[2] = 16'b00001_000_0000_0111 ; //Store in 0x7 => ACC=0x03
mem_bank[3] = 16'b00011_000_0000_1000 ; //Load immediate 0x08 => ACC=0x08
mem_bank[4] = 16'b00110_000_0000_0010 ; //Substract variable in 0x02 => ACC=0x06
mem_bank[5] = 16'b00100_000_0000_0010 ; //Add variable in 0x02 => ACC=0x08
mem_bank[6] = 16'b00001_000_0000_1011 ; //Store in 0xB => ACC=0x08
mem_bank[7] = 16'b00011_000_0000_0011 ; //Load immediate 0x03 => ACC=0x03
mem_bank[8] = 16'b00111_000_0000_0011 ; //Substract immediate 0x03 => ACC=0x00
mem_bank[9] = 16'b00000_000_0000_0000 ; // Halt
```

En la interfaz entre el BIP y el UART se obtuvo en el testbench:

En donde se puede ver que la ejecución del programa se lleva a cabo correctamente. A continuación se presenta el diagrama RTL toplevel:



En el siguiente enlace se puede acceder al código RTL:
https://github.com/SorianoJuan/ArqDeComputadoras/tree/TP3

Conclusiones

En este trabajo se incluyeron, de alguna manera, los 2 trabajos anteriores. En este caso la ALU se simplificó y solamente realizaba dos operaciones. La próxima etapa sería implementar un procesador que realice saltos y tenga más cantidad de etapas en el pipeline para poder aprovechar al máximo el ILP.