PROGRAMADOR UNIVERSITARIO
LICENCIATURA EN INFORMÁTICA
INGENIERÍA EN INFORMÁTICA
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS
Proyecto (primera parte)
Diseño de la MIC-1

Introducción

Es complicado enseñar el nivel de microarquitectura desde el punto de vista de conceptos. Cada ISA (Instruction Set Architecture - Arquitectura de Set de Instrucciones) es único, y como la microarquitectura se diseña según la misma, cada microarquitectura termina siendo única. Lo mejor que podemos hacer es empezar a entender este nivel desde el ejemplo.

Nuestro ejemplo va a ser basado en el que Andrew S. Tanenbaum nos presenta en capitulo 4 de su libro, “Organización de Computadoras – Un Enfoque Estructurado”. En el libro, Tanenbaum presenta a IJVM, Fig. 1, una ISA que es una versión reducida de la de Java Virtual Machine; específicamente, solo trabaja con instrucciones para números enteros.

Hex	Mnemonic	Meaning
0x10	BIPUSH *byte*	Push byte onto stack
0x59	DUP	Copy top word on stack and push onto stack
0xA7	GOTO *offset*	Unconditional branch
0x60	IADD	Pop two words from stack; push their sum
0x7E	IAND	Pop two words from stack; push Boolean AND
0x99	IFEQ *offset*	Pop word from stack and branch if it is zero
0x9B	IFLT *offset*	Pop word from stack and branch if it is less than zero
0x9F	IF_ICMPEQ *offset*	Pop two words from stack; branch if equal
0x84	IINC *varnum const*	Add a constant to a local variable
0x15	ILOAD *varnum*	Push local variable onto stack
0xB6	INVOKEVIRTUAL *disp*	Invoke a method
0x80	IOR	Pop two words from stack; push Boolean OR
0xAC	IRETURN	Return from method with integer value
0x36	ISTORE *varnum*	Pop word from stack and store in local variable
0x64	ISUB	Pop two words from stack; push their difference
0x13	LDC_W *index*	Push constant from constant pool onto stack
0x00	NOP	Do nothing
0x57	POP	Delete word on top of stack
0x5F	SWAP	Swap the two top words on the stack
0xC4	WIDE	Prefix instruction; next instruction has a 16-bit index

Fig. 1: IJVM

Parte 1: Data Path

La primera parte de nuestra microarquitectura es el Data Path, Fig. 2, conocida como la trayectoria de datos. La misma, es la parte que contienen una serie de registros de 32 bits, el ALU, y dos bus de datos: B y C.

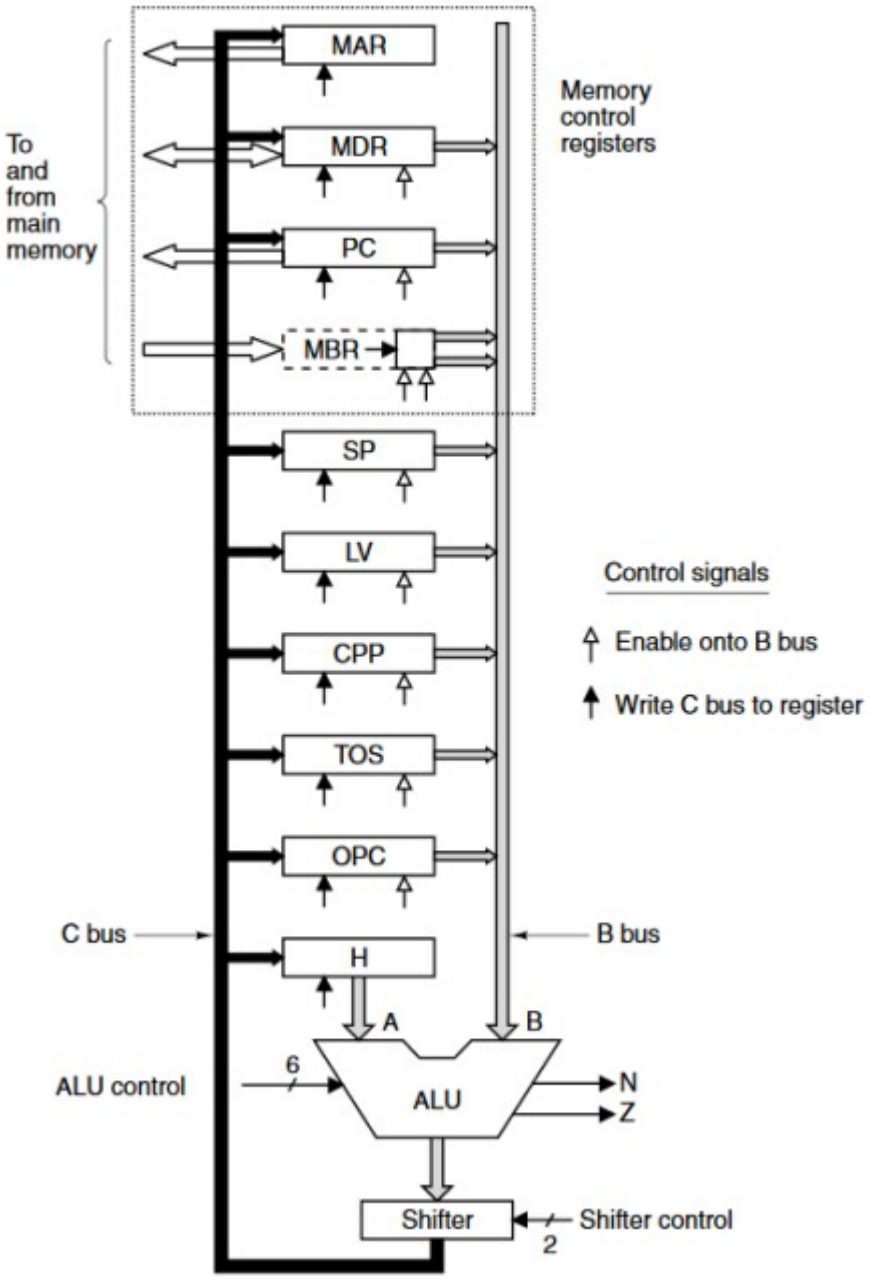

Fig. 2: Data Path del Mic-1.

Cada registro es un registro de 32 bits que tienen dos opciones, Parallel Load y Hold. Los habilitadores de las salidas de los registros serán los mismos que fueron vistos en la sección de memoria, que habilitan su salida.

El ALU, Fig. 3, es el mismo con cual estuvimos trabajando en previos prácticos con 6 señales de control. Su tabla parcial de funcionamiento se puede ver en Fig. 4.

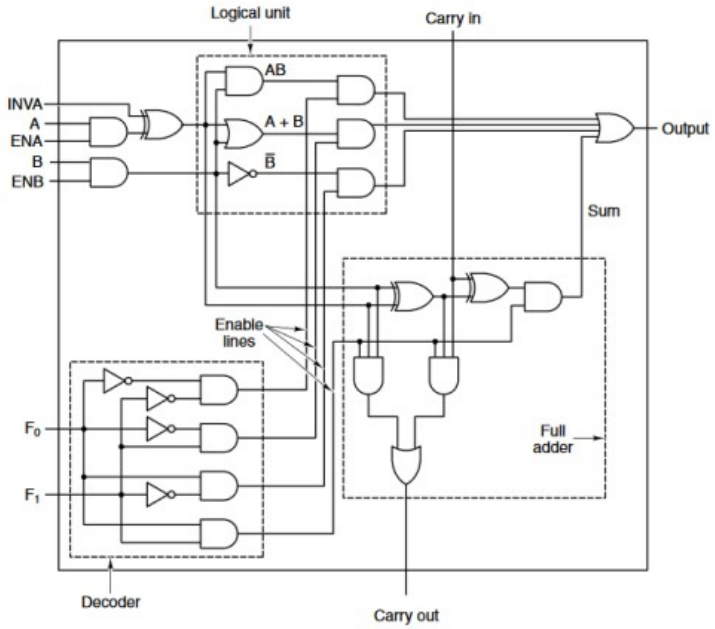

Fig. 3: ALU de 1 bit.

F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1

Fig. 4: Funciones utiles del ALU.

Con respecto a operaciones de memoria, hay dos maneras de comunicarnos con ella:

- A través de los registro MAR y MDR, adonde MAR contienen la dirección en memoria y MDR contienen los datos leídos de memoria o los datos para escribir a memoria.
- A través del registro MBR, adonde se puede leer 8-bits (1 byte) de la memoria, controlado por el PC.