ECE 3710 – Instruction to ALU Opcode Table

Authors: Adrian Sucahyo, Kenneth Gordon, Bryant Watson, and Inhyup Lee

Instruction	ALU Instruction Type	Instruction Opcode	ALU Opcode	Carry In Bit
ADD	STATIC	0101	0000	0
ADDI	STATIC	0101	0000	0
ADDU	STATIC	0110	0001	0
ADDUI	STATIC	0110	0001	0
ADDC	STATIC	0111	0000	1
ADDCI	STATIC	0111	0000	1
SUB	STATIC	1001	0010	0
SUBI	STATIC	1001	0010	0
SUBC	STATIC	1010	0010	1
SUBCI	STATIC	1010	0010	1
CMP	STATIC	1011	0100	0
CMPI	STATIC	1011	0100	0
AND	STATIC	0001	0101	0
ANDI	STATIC	0001	0101	0
OR	STATIC	0010	0110	0
ORI	STATIC	0010	0110	0
XOR	STATIC	0011	0111	0
XORI	STATIC	0011	0111	0
LSH	SHIFT	0100	1000	0
LSHI	SHIFT	0100	1000	0
ASHU	SHIFT	0110	1000	1
ASHUI	SHIFT	0110	1000	1

Notes:

ALU Instruction Types –

- STATIC Type instructions in the ALU represent Arithmetic and Logical operations between the inputs A and B. This means it describes both register and immediate instructions.
- SHIFT Type instructions in the ALU represent Shifting operations (using a shift register), shifting the value in A by B according to the instructions. This means it describes both register and immediate instructions.

Instruction Opcode – The instruction opcode is the binary word that needs to be routed directly from the current instruction being executed from the data path. A separate ALU control module is responsible for decoding the instruction opcode into its corresponding ALU opcode.

ALU Opcode – The ALU opcode is the binary word that determines the overall operation of the ALU module, and is different from the instruction opcodes, due to the potential for duplicated functionality in the ALU across similar instructions (ie ADD, ADDI, and ADDC are ADD opcode instructions, with slightly different datapath routing and control bits).

Carry In Bit – The carry in bit is used for instructions where a carry in bit is necessary. It is also used to determine the shifted in bit when doing arithmetic or logical right shifting.