EE 231 Prelab 4

Arithmetic Logic Unit

The heart of every computer is an Arithmetic Logic Unit (ALU). This is the part of the computer which performs arithmetic operations on numbers, e.g. addition, subtraction, etc. In this lab you will use the Verilog language to implement an ALU having 10 functions. Use of the case structure will make this job easy.

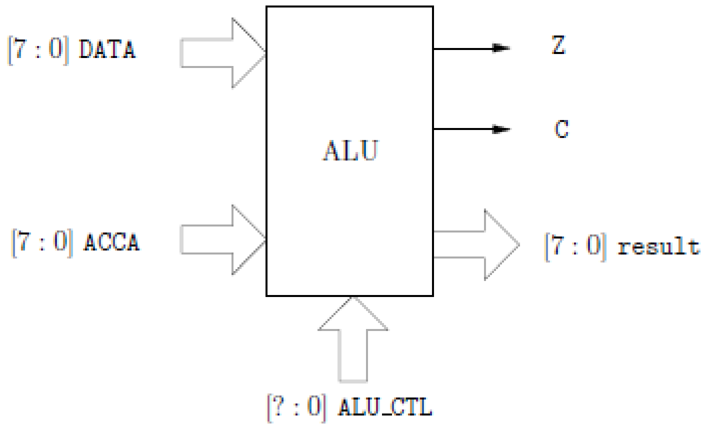
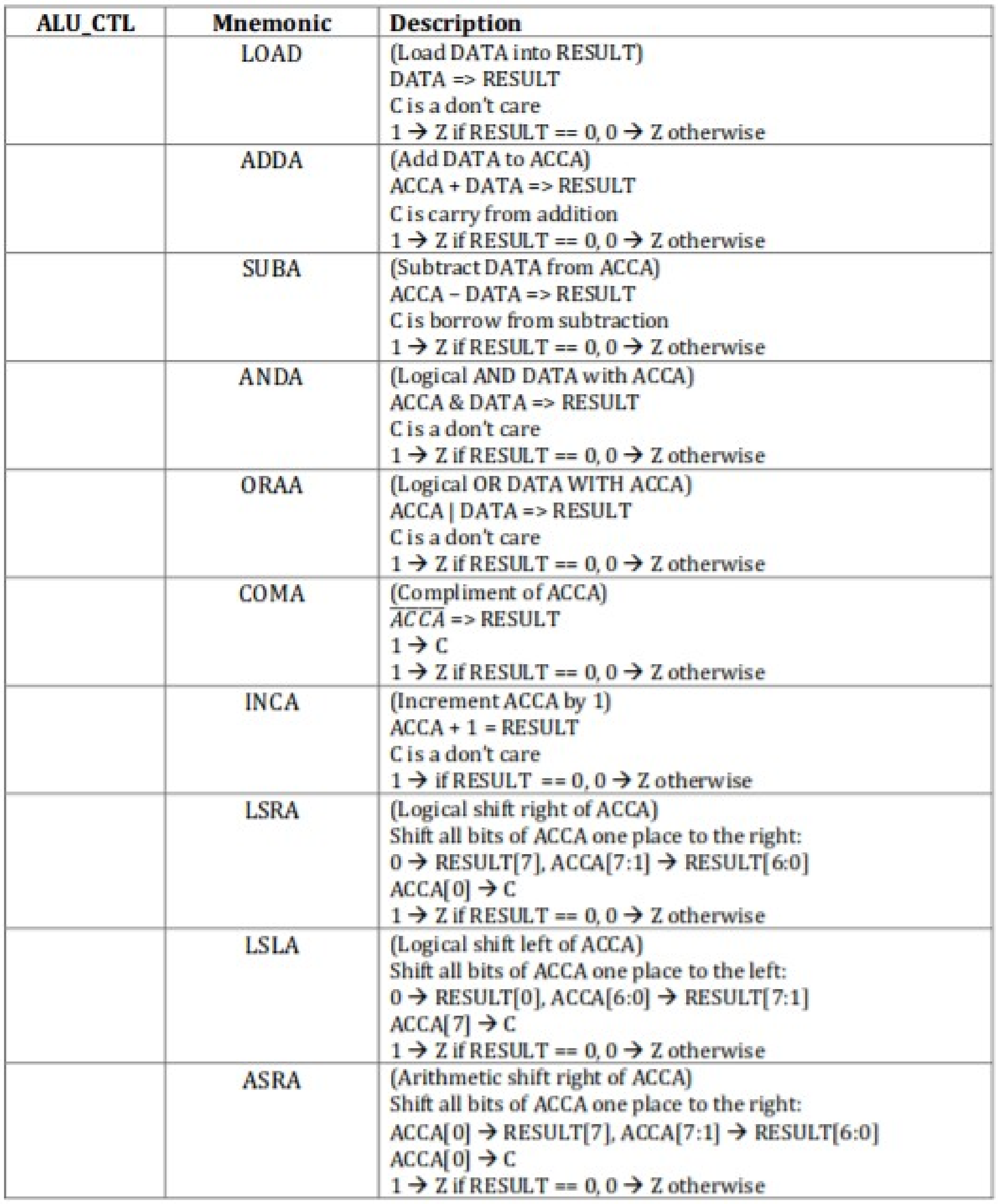


Figure 1: ALU Block Diagram

The ALU that you will build (see Figure 1) will perform 10 functions on 8-bit inputs (see Table 1). Please make sure you use the same variable name as the ones used in this lab. Don't make your own. The ALU will generate an 8-bit result (result), a one bit carry (C), and a one bit zero-bit (Z). To select which of the 10 functions to implement you will use ALU_CTL as the selection lines.

1.Prelab

1.1.Fill out Table 1. **How many bits should ALU_CTL be?**
1.2.Write code to implement the ALU.



ALU_CTL	Mnemonic	Description
	LOAD	(Load DATA into RESULT) DATA => RESULT C is a don't care 1 → Z if RESULT == 0, 0 → Z otherwise
	ADDA	(Add DATA to ACCA) ACCA + DATA => RESULT C is carry from addition 1 → Z if RESULT == 0, 0 → Z otherwise
	SUBA	(Subtract DATA from ACCA) ACCA – DATA => RESULT C is borrow from subtraction 1 → Z if RESULT == 0, 0 → Z otherwise
	ANDA	(Logical AND DATA with ACCA) ACCA & DATA => RESULT C is a don't care 1 → Z if RESULT == 0, 0 → Z otherwise
	ORAA	(Logical OR DATA WITH ACCA) ACCA | DATA => RESULT C is a don't care 1 → Z if RESULT == 0, 0 → Z otherwise
	COMA	(Compliment of ACCA) $\overline{ACCA}$ => RESULT 1 → C 1 → Z if RESULT == 0, 0 → Z otherwise
	INCA	(Increment ACCA by 1) ACCA + 1 = RESULT C is a don't care 1 → if RESULT == 0, 0 → Z otherwise
	LSRA	(Logical shift right of ACCA) Shift all bits of ACCA one place to the right: 0 → RESULT[7], ACCA[7:1] → RESULT[6:0] ACCA[0] → C 1 → Z if RESULT == 0, 0 → Z otherwise
	LSLA	(Logical shift left of ACCA) Shift all bits of ACCA one place to the left: 0 → RESULT[0], ACCA[6:0] → RESULT[7:1] ACCA[7] → C 1 → Z if RESULT == 0, 0 → Z otherwise
	ASRA	(Arithmetic shift right of ACCA) Shift all bits of ACCA one place to the right: ACCA[0] → RESULT[7], ACCA[7:1] → RESULT[6:0] ACCA[0] → C 1 → Z if RESULT == 0, 0 → Z otherwise

Table 1: ALU functions