Department of Computer Engineering
Digital Hardware Systems
CpE 3202 - Computer Organization and Architecture

Laboratory Exercise #1
“The Arithmetic and Logic Unit (ALU)”

Instruction: Based on the top level 8-bit ALU design on Fig. 1, construct the ALU using C that can perform operation based on the arithmetic concepts discussed in Unit II “Computer Arithmetic”. Refer to the specifications below. Please note that the ALU will be modified as necessary in the future.

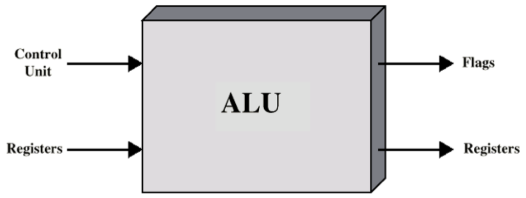


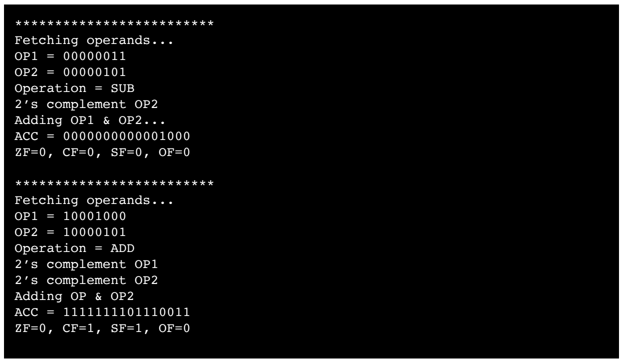
Figure 1. ALU Top Level

1) Function prototype: int ALU(unsigned char operand1, unsigned char operand2, unsigned char control_signals)

   The variable “control_signals” is an 8-bit control lines from the control unit. This signify the type of operation to be performed by the ALU. Refer to the following of the values of the control_signals and the corresponding operation:

   0x01 - addition
   0x02 - subtraction
   0x03 - multiplication
   0x04 - AND
   0x05 - OR
   0x06 - NOT
   0x07 - XOR
   0x08 - shift right (logical)
   0x09 - shift left (logical)

2) Accumulator (ACC) is 8 bits wide but must be declared as 16-bit (unsigned int) to check carry condition.
3) Arithmetic functions:
   a. Addition
   b. Subtraction (addition via 2’s complement)
   c. Multiplication (use Booth’s algorithm)
4) Logic functions:
   a. AND
   b. OR
   c. NOT (second operand will be 0x00)
   d. XOR
   e. shift right (logical)
   f. shift left (logical)
5) All operations must consider **signed numbers** (only for arithmetic operations).
6) Set flags *carry* (C), *zero* (Z), *overflow* (OF) and *sign flag* (SF) and ACC as <u>global</u> unsigned character variables.
7) Test the ALU by writing a code in *main()* that will ask the operands and operation. Echo the process of the calculation on the display. For example, the operation “3 - 5” (00000011_2 - 00000101_2) will be executed by calling the function *ALU(0x03,0x05,0x02)* in which ‘3’ and ‘5’ are the first and second operand respectively while 0x02 refers to the subtraction operation. The operation shall be echoed on the screen like:



```
*************************
Fetching operands...
OP1 = 00000011
OP2 = 00000101
Operation = SUB
2’s complement OP2
Adding OP1 & OP2...
ACC = 0000000000001000
ZF=0, CF=0, SF=0, OF=0

*************************
Fetching operands...
OP1 = 10001000
OP2 = 10000101
Operation = ADD
2’s complement OP1
2’s complement OP2
Adding OP & OP2
ACC = 1111111101110011
ZF=0, CF=1, SF=1, OF=0
```

Figure 2. Screen echo/output.

The echo provides the step by step operation as described in the arithmetic algorithms. This will ensure that the arithmetic algorithm has been performed. The example in Figure 2, calling the ALU function was done twice in the main( ) function.

```
void main(void)
{
   ALU(0x03,0x05,0x02); // 00000011 - 00000101 (subtract)
   ALU(0x88,0x85,0x01); // 10001000 + 10000101 (add)
   ALU(0xC0,0x02,0x03); // 11000000 * 00000010 (multiply)
}
```

8. Create the following functions:

   `unsigned char twosComp(unsigned data` – function to 2's complement a number

   `unsigned char setFlags(unsigned int ACC)` – function to set the zero, overflow, sign and carry flags

   `void printBin(int data, unsigned char data_width)` – print binary characters of data

9. Add additional function calls to the main( ) function to include other operations. Make sure that all operations are are tested.

10. Save your work as "ALU.c". Submit in Canvas on before due date. *Note: before submitting, make sure your source code does not have any errors.*

Assessment

Criteria	Excellent (10 pts)	Satisfactory (8.5 pts)	Marginal (7.5 pts)	Not Acceptable (5 pts)	Not delivered (0 pt)
Logic	ALU logic is 100% correct.	ALU logic has minor issues.	There are several issues in the ALU logic.	ALU logic is incorrect.	
Emulation	Emulation of the Control Unit is very close to the actual.	Emulation of the Control Unit is slightly close to the actual.	Emulation of the Control Unit is ver far from the actual.	Emulation of the Control Unit is not demonstrated.	
Coding	Coding is neat, systematic, logical and followed accepted coding standards.	Coding is logical and somewhat followed some coding standards.	Coding is logical followed little coding standards.	Coding is a mess and did not follow any coding standards.	

Copyright Information

Change log:

Date	Version	Author	Changes
September 6, 2019	1.2.1	Van B. Patiluna	Original laboratory guide (CpE 415N).
February 8, 2021	1.0	Van B. Patiluna	- Added shift left and right. - Removed unnecessary functions required