The University of Arizona
Department of Aerospace and Mechanical Engineering
Mechatronics Laboratory
Instructor: Professor. Eniko T. Enikov, enikov@email.arizona.edu

Laboratory Task Sheet 01

Title: Mathematic Operations

Registers to be learned: STATUS,C

Section A – Addition

Objective: Write a code for adding two binary numbers. Run the program for two cases, so that the result of one is greater than 255, and one is less than 255. Determine the value of STATUS C for each case.

Tasks

1. Make a copy of the P16f84A_Template file and name it TASK01AGroup00. Open the file in MPLAB Software in **Simulation Mode** and use the table below to construct the code.

Suggested Code Structure
Define ByteA, ByteB, and Result as memory files
Start
Call Initialization **Go to Main**
Main Choose a number and move it to ByteA Choose a number and move it to ByteB Move ByteA to Result Add ByteB to Result Do nothing Go to the previous line
Initialization Bank1 Bank0 Return
end

2. Demonstrate the result to the instructor.
3. Upload the code on D2L and save it for yourself.

The University of Arizona
Department of Aerospace and Mechanical Engineering
Mechatronics Laboratory
Instructor: Professor. Eniko T. Enikov, enikov@email.arizona.edu

Section B – Multiplication

Objective: Write a code for multiplying two binary numbers. Run the program so that the result of the multiplication is less than 255.

Tasks

1. Make a copy of the P16f84A_Template file and name it TASK01BGroup00. Open the file in MPLAB Software in **Simulation Mode** and use the table below to construct the code.

Suggested Code Structure
Define Multiplicand, Multiplier, and Result as memory files
Start
Call Initialization **Go to main**
Main Choose a number and move it to Multiplicand Choose a number and move it to Multiplier Call Multiplication Do nothing Go to the previous line
Multiplication Clear Result Move Multiplicand to the Work Register Add Work Register to Result Decrement Multiplier If Multiplier is zero, return If Multiplier is not zero, repeat last three lines Return
Initialization Bank1 Bank0 Return
end

2. Demonstrate the result to the instructor.
3. Upload the code on D2L and save it for yourself.

The University of Arizona
Department of Aerospace and Mechanical Engineering
Mechatronics Laboratory
Instructor: Professor. Eniko T. Enikov, enikov@email.arizona.edu

Section C – Subtraction

Objective: Write a code for subtracting two binary numbers. This code must be able to calculate the absolute value of negative results. Run the program for two cases, so that the result of one is positive and one is negative. Determine the value of STATUS C and BitA in each case.

Tasks

1. Make a copy of the P16f84A_Template file and name it TASK01CGroup00. Open the file in MPLAB Software in **Simulation Mode** and use the table below to construct the code.

Suggested Code Structure
Define ByteA, ByteB, and Result as memory files Define BitA as a memory bit
Start
Call Initialization **Go to Main**
Main Choose a number and move it to ByteA Choose a number and move it to ByteB Call Subtraction Do nothing Go to the previous line
Subtraction Set BitA Move ByteB to the Work Register Subtract the Work Register from ByteA and put the result in the Work Register Move the Work Register to Result Check if the result is negative or positive If it is positive, return If it is negative, clear BitA Make compliment of the Result Increment the Result Return
Initialization Bank1 Bank0 Return
end

2. Demonstrate the result to the instructor.
3. Upload the code on D2L and save it for yourself.

The University of Arizona
Department of Aerospace and Mechanical Engineering
Mechatronics Laboratory
Instructor: Professor. Eniko T. Enikov, enikov@email.arizona.edu

Section D – Division

Objective: Write a code for dividing two binary numbers.

Tasks

1. Make a copy of the P16f84A_Template file and name it TASK01DGroup00. Open the file in MPLAB Software in **Simulation Mode** and use the table below to construct the code.

Suggested Code Structure
Define Quotient, Divisor, and Remainder as memory files
Start
Call Initialization **Go to Main**
Main Choose a number and move it to Remainder Choose a number and move it to Divisor Clear Quotient Call Division Do nothing Go to the previous line
Division Move Divisor to the Work Register Subtract Work Register from Remainder and put the result in the Work Register Check if the result is negative or positive If the result is negative, return If the result is positive, move the Work Register to Remainder Increment Quotient Go to Division
Initialization Bank1 Bank0 Return
end

2. Demonstrate the result to the instructor.
3. Upload the code on D2L and save it for yourself.