Lab 1: Getting Started with MPLAB and the Cerebot MX3cK

Michael Mentele

Partner: Jonathan Quisenberry

EE 234\_2

9/4/12

**Introduction:**

In this lab we explored the basics of the Cerebot MX4ck board and the accompanying MPLAB software. We downloaded the software and created a project where we loaded some pre-written code that tied pushbuttons 1 and 2 to LEDs on the board. We then analyzed the code and became familiar with the interface for MPLAB.

**Flowchart and Process Description:**

The flowchart below illustrates the algorithm that is being applied for our initial application and familiarization with the PIC32 microcontroller and the Digilent Cerebot MX4cK board.

|  |  |
| --- | --- |
| C:\Users\cmentele\Desktop\EE234\Labs\Lab1\Lab1_FlowChart.jpg | As you can see after you begin running the code and setting up ports A and B you enter an ‘infinite’ loop where the program gets the pushbutton states and then writes those states to the LED’s.  We could use breakpoints somewhere in that loop, perhaps before it gets the pushbutton states to specify time or an exit parameter so that the program is more versatile. |

**Questions:**

1. void setup\_LEDs (void)

{

PORTSetPinsDigitalOut (IOPORT\_B, BIT\_10 | BIT\_11 | BIT\_12 | BIT\_13);

}

For this function it enables the bits 10-13 in Port B and explicitly states that they are outputs.

1. MIPS assembly instructions ADDU is an arithmetic operation that adds two registers and stores it in a third destination register. The purpose is to add 32-bit integers together with the format ADDU rd, rs, rt. For GPR[rd] <- GPR[rs] + GPR[rt] the two 32-bit word value in GPR rt is added to the 32-bit value in GPR rs and the 32-bit arithmetic result is stored in GPR rd and there is no exception for overflow for this instruction.
2. The Cerebot MX4cK reference manual indicates that PORTB is connected to the LEDs. This makes sense because it ties different parts of the board together. Whereas the data sheet for the PIC32 concerns itself only with the local I/O of that microcontroller.
3. The general process followed when an instruction is executed is fetch, decode, and execute. First it takes an instruction from Program Flash Memory based on your flash controller and communicates on a 128-bit bus to the prefetch module which communicates on a 32-bit bus to the bus matrix.

Then it splits the instruction(s) into opcode (operation(s) and operand(s)). It might then wait for an input or continue based on the states of your inputs from PORTS A-G or peripherals. It will then execute on the CPU core and output to your ports and store useful data in Data RAM.

Data is stored on the ‘Data RAM’ memory block and programs are stored on the ‘128-bit wide Program Flash Memory’ memory block.

1. You need 359 bytes of Flash memory and 0 for data memory for the given program.
2. The ALU is uses different configurations of logic gates to perform arithmetic operations. It can do things like add and multiply or make bit comparisons.

\*\*EXPLORATION USING THE WEB\*\* the bus matrix is a high-speed switch and establishes a means of communication between the other modules. It runs at the same speed as the CPU whereas the peripheral bus may run at a different clock speed. The pre-fetch module looks ahead and stores 128 bits of instructions to provide high performance even if the CPU is running faster than the Flash memory speed.

**Conclusion:**

This lab has demonstrated how to open, compile, disassemble, and debug a project that you create. It has also facilitated familiarity with the MPLAB GUI and the file types contained within a respective projects. In addition it has allowed one to gain understanding of the different components found on the Cerebot board as well as a brief insight into the coding aspect of it.