## I. Learner Objectives:

At the conclusion of this lab, participants should be able to:

- Design and implement subroutines in MIPS32 assembly
- Interface with basic peripheral modules in MIPS32 assembly

## II. Prerequisites:

Before reading this lab, participants should be able to:

- Outline the lab related to ports
- Summarize the unit related to registers

## III. Keywords:

## IV. Pre-lab:

None.

## V. Background:

This lab will introduce you to the wonderful world of MIPS32 assembly. The idea behind the lab is to introduce you to basic low level design and programming. You will be using a switch peripheral module and a LED peripheral module to construct a basic calculator.

As you should already know, the following table lists the instructions for using I/O Port registers. The *n* in the names for the registers may represent A – G. Each of the registers is 32-bits, where each of the bits may be set to read/input or write/output. 1 represents input and 0 represents output. The appropriate TRISn register should be set to indicate whether or not to read or write from/to a bit of the port. If all of the bits in the TRISn register are set for read, then contents from the port will be stored in the corresponding bit position in PORTn. If all of the bits in the TRISn register are set for write, then the contents to write to the port will be stored in the corresponding bit position in LATn. Refer to Table 1. Use the MIPS32 assembly instruction LW (Load Word) to read from a port and SW (Store Word) to write to a port.

| Operation | TRISn | PORTn | LATn |
|---|---|---|---|
| Read (RD) | 1 | Content read from port | N/A |
| Write (WR) | 0 | N/A | Content written to port |

**Table 1: Read and Write Operations**

Table 2 summarizes the pin connections between the Digilent 6-pin 4-slide switch peripheral module (plugged into the JK-01:04 connector) and the Microchip PIC32MX460F512L microcontroller. Note that you may find the port and function for the pin in the Cerebot MX4cK Reference Manual document.

| Cerebot Pin | MCU Port/bit | Switch | On-board LED |
|---|---|---|---|
| JK-01 | RB10 | SW1 | LD1 |
| JK-02 | RB11 | SW2 | LD2 |
| JK-03 | RB12 | SW3 | LD3 |
| JK-04 | RB13 | SW4 | LD4 |

**Table 2: Port to Switch Mapping**

Table 3 summarizes the pin connections between the Digilent 6-pin or 12-pin LED peripheral module (plugged into the JJ connector) and the Microchip PIC32MX460F512L microcontroller. Recall your goal is to map the state (OFF/ON) of a switch to the state (OFF/ON) of a peripheral LED.

| Cerebot Pin | Port/bit | Peripheral LED |
|---|---|---|
| JJ-01 | RB0 | LD0 |
| JJ-02 | RB1 | LD1 |
| JJ-03 | RB2 | LD2 |
| JJ-04 | RB3 | LD3 |

**Table 3: Port to Peripheral LED Mapping**

## VI. Lab:

Before you begin writing C code for the lab make sure you have one end of the USB cable plugged into the Cerebot MX4cK micro-B "DEBUG" port and the other end into one USB-A port on your PC. Also, be sure to plug in the switch module that came with your Digilent board package into the JK-01:04 connector of the Cerebot MX4cK board. Lastly, plug the peripheral LED module into the JJ-01:04 connector. The top pins correspond to 01:06. **NOTE: If you have one of the newer LED modules (i.e. with 8 LEDs), then you will plug the module into JJ-01:12. However, you will only be concerned with 4 of the LEDs.** Turn on power to the board by flipping the switch next to the "DEBUG" port.

In this lab you will create calculator of sorts using MIPS32 assembly. Thus, you will create a different project type than in previous labs. While you are creating a new project with the "Project Wizard" ensure you select the assembly tool as illustrated in Figure 1.

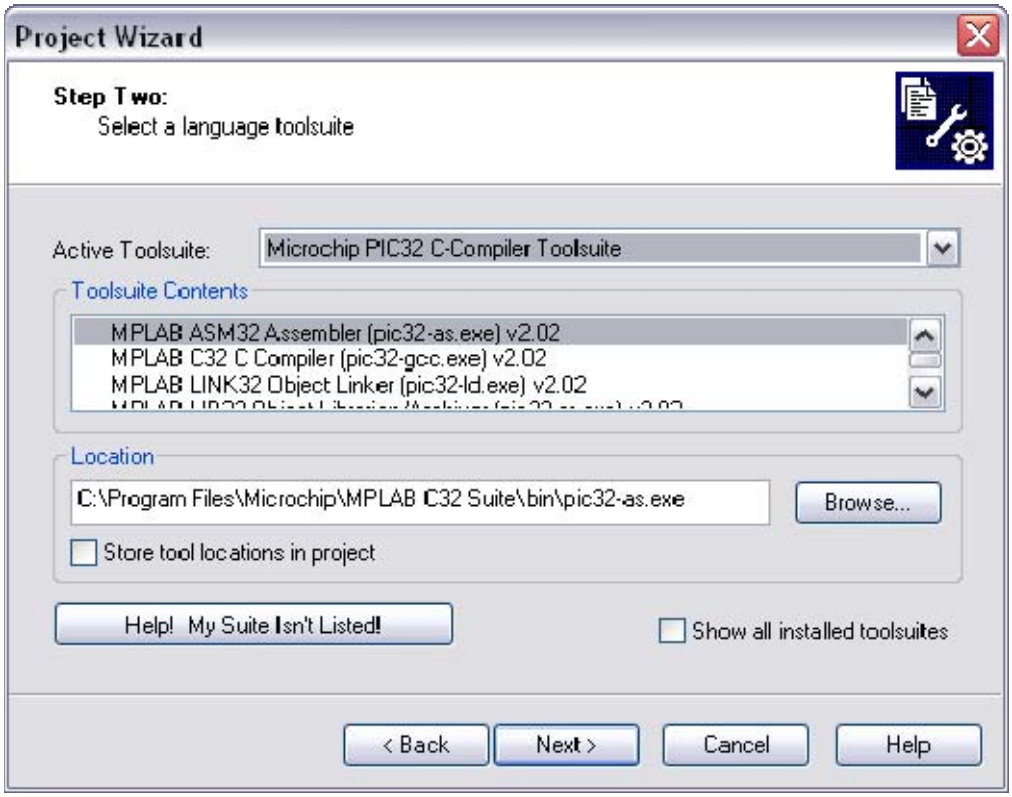



**Figure 1: ASM32 Assembler**

Also, when you create a source file for the project, save it with a .s extension. You will NOT be able to use any C preprocessor directives with this tool suite and file type. You are required to create MIPS32 subroutines to help partition tasks.

In this lab you will be performing calculations like the following, using the switch module:

Select OPERATOR (with SW1 and SW2), according to the table below; if after a couple of seconds no switches are flipped, then assume Addition
Select OPERAND1 (with SW3 and SW4)
Optional: Select OPERAND2 (with SW3 and SW4), only required for binary operations

You will use the switch module to select an operation to perform along with the operands to use in the calculation. The following table (Table 4) demonstrates the state of the switches corresponding to operations.

| SW1 | SW2 | Operation |
|---|---|---|
| 0 | 0 | Addition |
| 0 | 1 | Subtraction |
| 1 | 0 | Shift left by 1 |

| | | |
|---|---|---|
| 1 | 1 | Shift right by 1 |

**Table 4: Selecting Operations**

The following table (Table 5) shows the registers that correspond to the states of switches.

| SW3 | SW4 | Register |
|---|---|---|
| 0 | 0 | t0 |
| 0 | 1 | t1 |
| 1 | 0 | t2 |
| 1 | 1 | t3 |

**Table 5: Selecting Operands**

Thus, if SW1 and SW2 are both 0, and SW3 and SW4 transition from 00 to 01, then an addition will be performed on the registers t0 and t1. Make sure that the contents in the corresponding registers are displayed on the peripheral LED modules as you select them. Also, once the operation has been performed display the result on the peripheral LEDs. Note that anything larger than 4-bit arithmetic can not be accommodated by the peripheral LEDs (since only 4 LEDs are present).

Also, note that addition and subtraction are binary operations requiring two operands and that shift left and shift right by 1 bit are unary operations requiring just one operand.

The following process should be followed for operating the switches:

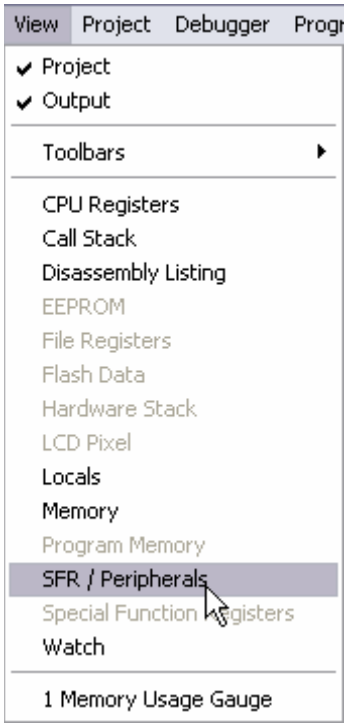
1. Select an operation with SW1 and SW2
2. Pause for some brief amount of time
3. Select an operand with SW3 and SW4
4. Pause for some brief amount of time
5. Display operand on peripheral LEDs for at least 2 seconds (use delay loop)
6. Repeat steps 2 and 3, once, for a binary operation
7. Display the result of the operation on the peripheral LEDs for at least 2 seconds (use delay loop)

Have fun, be creative, and ask your TA questions!!!

**VII. Questions:**

1. Most likely your solution required the use of the stack. Explain why you needed the stack.
2. We discussed the general memory map for the PIC32MX4 MCU in lecture. View the SFR window and provide the virtual addresses for TRISB,

PORTB, and LATB. What are their offsets from the beginning of the SFR section of memory? Recall you can find the SFR window as shown below.




3. Why is it necessary to write assembly programs, instead of C programs, in some cases? Explain.