An Introduction to PIC32

Dr. Farahmand
PIC32 Family

Different in terms of IO pins / RAM (data memory) / FLASH (program memory; non-volatile) / Peripherals

32-bit in terms of
   Instructions, register size, Instruction Bus, Data Bus
PIC32 Family

MX6 supports CAN and Ethernet

MZ – Using MIPS – Licensed by MICROCHIP
PIC32 Architecture
Architecture:
<table>
<thead>
<tr>
<th>Product</th>
<th>Pins</th>
<th>MHz</th>
<th>Flash KB</th>
<th>RAM</th>
<th>Temperature Range</th>
<th>Operation Voltage Range</th>
<th>USB</th>
<th>Ethernet</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC32MX110F016B</td>
<td>28</td>
<td>40</td>
<td>16</td>
<td>4096</td>
<td>-40 to 105</td>
<td>2.3V - 3.6V</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>PIC32MX110F016C</td>
<td>36</td>
<td>40</td>
<td>16</td>
<td>4096</td>
<td>-40 to 105</td>
<td>2.3V - 3.6V</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>PIC32MX110F016D</td>
<td>44</td>
<td>40</td>
<td>16</td>
<td>4096</td>
<td>-40 to 105</td>
<td>2.3V - 3.6V</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>PIC32MX120F032B</td>
<td>28</td>
<td>50</td>
<td>32</td>
<td>8192</td>
<td>-40 to 105</td>
<td>2.3V - 3.6V</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>PIC32MX120F032C</td>
<td>36</td>
<td>50</td>
<td>32</td>
<td>8192</td>
<td>-40 to 105</td>
<td>2.3V - 3.6V</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>PIC32MX120F032D</td>
<td>44</td>
<td>50</td>
<td>32</td>
<td>8192</td>
<td>-40 to 105</td>
<td>2.3V - 3.6V</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>PIC32MX120F064H</td>
<td>64</td>
<td>50</td>
<td>64</td>
<td>8192</td>
<td>-40 to 105</td>
<td>2.3V - 3.6V</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>PIC32MX130F064B</td>
<td>28</td>
<td>40</td>
<td>64</td>
<td>16384</td>
<td>-40 to 105</td>
<td>2.3V - 3.6V</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>PIC32MX130F064C</td>
<td>36</td>
<td>40</td>
<td>64</td>
<td>16384</td>
<td>-40 to 105</td>
<td>2.3V - 3.6V</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>PIC32MX130F064D</td>
<td>44</td>
<td>40</td>
<td>64</td>
<td>16384</td>
<td>-40 to 105</td>
<td>2.3V - 3.6V</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

Example

The **PIC32MX795F512L** is powered by a supply voltage in the range 2.3 to 3.6 V and features a max clock frequency of 80 MHz, 512 KB program memory (Flash) 128 KB data memory (RAM) 1610-bit analog-to-digital input lines (multiplexed to a single analog-to-digital converter, or ADC), USB 2.0, Ethernet, two CAN modules, I²C and four SPI synchronous serial communication modules, six UARTs for RS-232 or RS-485 asynchronous serial communication......
Development Boards
Development Boards

• Hardware
  – PIC32 USB Starter Kit – With on-board programmer
  – Development Board – Need PICKIT3
  – Development Board with a Boot Loader – Can be programmed using USB cable

• Software
  – XC32 Compiler
  – MPLABX
```c
#include <plib.h>

// configuration bits are not set by a bootloader, so set here
#pragma config DEBUG   = OFF   // Background Debugger disabled
#pragma config PLLMUL  = MUL_20 // PLL Multiplier: Multiply by 20
#pragma config PLLIDIV = DIV_2  // PLL Input Divider: Divide by 2
#pragma config PLLODIV = DIV_1  // PLL Output Divider: Divide by 1
#pragma config WDTON   = OFF   // WD timer: OFF
#pragma config POSCMOD = HS     // Primary Oscillator Mode: High Speed xtal
#pragma config FNOSC   = PRIPLL // Oscillator Selection: Primary oscillator w/ PLL
#pragma config FBDIV   = DIV_1  // Peripheral Bus Clock: Divide by 1
#pragma config BWP     = OFF    // Boot write protect: OFF
#pragma config ICESEL  = ICS_PGx2 // ICE pins configured on PGx2
#pragma config FOSCSEN = OFF    // Disable second osc to get pins back
#pragma config FSRSSEL = PRIORITY_7 // Shadow Register Set for interrupt priority 7

#define SYS_FREQ 80000000  // 80 million Hz

void delay(void);

int main(void) {

    SYSTEMConfig(SYS_FREQ, SYS_CFG_ALL); // cache on, PBCLK setup, min flash wait
    DDPCONbits.JTAGEN = 0; // Disable JTAG, make pins 4 and 5 of Port A available.
    TRISA = 0xFFCF; // Pins 4 and 5 of Port A are LED1 and LED2. Clear
                   // bits 4/5 to zero, for output. Others are inputs.
    LATAbits.LATA4 = 0; // Turn LED1 on and LED2 off. These pins sink ...
    LATAbits.LATA5 = 1; // ... current on NU32, so "high" = "off."

    while(1) {
        delay();
        LATAINV = 0x0030; // toggle the two lights
    }
    return 0;
}

void delay(void) {
    int j;
    for (j=0; j<1000000; j++) { // number is 1 million
        while(!PORTDbits.RD13); // Pin D13 is the USER switch, low if pressed.
    }
```

http://www.johnloomis.org/microchip/pic32/resources.html
Introduction to MIPS

• Microprocessors without Interlocked Pipelines Stages
  – MIPS I, II, ...V, 32, 64
• Developed by MIPS Technology
• RISC Instruction Sets
• 32-bit Instructions
  – R-type; I-type, & J-type instructions
• Applications:
  – Routers, Switches, Laser Printers, Sony Station, Nintendo 64, etc.
• Main Competitor is ARM
  – PDAs and Cellphones
MIPS Assembly

- Basic commands:
  - Arithmetic, Data Transfer, Logic, Bit operation, Branch, Jump

<table>
<thead>
<tr>
<th>Add</th>
<th>add</th>
<th>$d = $s + $t</th>
<th>R</th>
<th>0</th>
<th>20_{16}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add unsigned</td>
<td>addu</td>
<td>$d = $s + $t</td>
<td>R</td>
<td>0</td>
<td>21_{16}</td>
</tr>
<tr>
<td>Subtract</td>
<td>sub</td>
<td>$d = $s - $t</td>
<td>R</td>
<td>0</td>
<td>22_{16}</td>
</tr>
<tr>
<td>Subtract unsigned</td>
<td>subu</td>
<td>$d = $s - $t</td>
<td>R</td>
<td>0</td>
<td>23_{16}</td>
</tr>
</tbody>
</table>
MIPS Assembly

• Basic commands:
  – Arithmetic, Data Transfer, Logic, Bit operation, Branch, Jump

<table>
<thead>
<tr>
<th>Command</th>
<th>Format</th>
<th>Description</th>
<th>Opcode</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store word</td>
<td>sw $t, C($s)</td>
<td>Memory[$s + C] = $t</td>
<td>2B16</td>
<td>I</td>
</tr>
<tr>
<td>Store half</td>
<td>sh $t, C($s)</td>
<td>Memory[$s + C] = $t</td>
<td>2916</td>
<td>I</td>
</tr>
<tr>
<td>Store byte</td>
<td>sb $t, C($s)</td>
<td>Memory[$s + C] = $t</td>
<td>2816</td>
<td>I</td>
</tr>
</tbody>
</table>

stores a word into: MEM[$s+C] and the following 3 bytes. The order of the operands is a large source of confusion.
stores the least-significant 16-bit of a register (a halfword) into: MEM[$s+C].
stores the least-significant 8-bit of a register (a byte) into: MEM[$s+C].

Assembly Programming Example

/* leds.S
   Written <date> by <your_name>@hmc.edu
   Test PIC by turning on LEDs */

#include <P32xxxx.h>

#define constants
#define LEDS 0xA5

#define functions
.globl main

# Compiler instructions
.text  # store the code in the main program section of RAM
.set noreorder # do not let the compiler reorganize your code

# Main program
.ent main     # Start function block
main:
    la $t0, TRISD  # Load the address of TRISD into $t0
    add $t1, $0, 0xFF00
    sw $t1, 0($t0)  # TRISD = 0xFF00 (bottom 8 bits outputs)
    add $t1, $zero, LEDS  # $t1 = LEDS (LEDS + 0)

write:  # This is a label you can jump to
    la $t0, PORTD  # Load the address of PORTD into $t0
    sw $t1, 0($t0)  # PORTD = $t1
    j write  # Jump back to write
    nop
.end main  # End function block

http://pages.hmc.edu/jspjut/class/f2014/e155/lab/lab04.pdf