Case study 1: The 8051 Architecture

# Objective

To self-study the architecture of an 8-bit microcontroller, including general purpose input/outputs and memory organization.

# Numerical Base Conversion and 2’s Complement

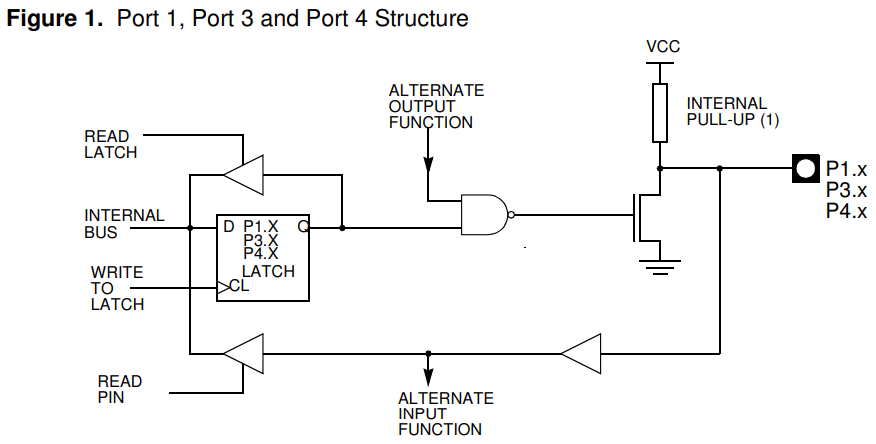
# Procedure

[INTEL’s 8051](https://en.wikipedia.org/wiki/Intel_8051) is an 8-bit microcontroller architecture developed in 1981 and still found in the marked today. One of the currently available microcontrollers based on the 8051 architecture is [ATMEL’s AT89C51](https://en.wikipedia.org/wiki/Atmel_AT89_series), which datasheet can be [found online](http://ww1.microchip.com/downloads/en/devicedoc/doc4383.pdf). In this case study, we will go over some of the 8051’s features using the AT89C51 device. Notice that some other devices that share similar architecture can be also found in the market.

According to the schematic diagram

## General Purpose Input/Output

According to Figure 1 (page 8) of the AT89C51AC3 microcontroller datasheet, ports 1, 3 and 4 share similar structure, which is shown below:



To drive a logic level (0 or 1) as an output of ports 1, 3 and 4, some conditions must be accomplished, as follows (see pin description table on pages 5-7 on the datasheet for further reference):

1. The signal to output is held by INTERNAL BUS and should follow the path towards P1.x, P3.x or P4.x
2. To this to happen, first the LATCH must drive the signal in its input, D, (connected to INTERNAL BUS) towards Q with a clock pulse (CL)
3. Once the INTERNAL BUS signal is transferred to the LATCH output, Q, it is used as one of two inputs of a NAND gate.
4. The second input of the NAND gate is a signal called ALTERNATE OUTPUT FUNCTION. According to the truth table of a NAND gate, if the value of this signal is 0, the output is set to 1 (highlighted in grey in the table below); however, if the value of ALTERNATE OUTPUT FUNCTION is 1, the output of the gate is the inverted value of Q, so is the inverted value of INTERNAL BUS.

|  |  |  |
| --- | --- | --- |
| Q (from flip-flop) | ALTERNATE OUTPUT FUNCTION | OUT |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

1. In this sense, to allow the value of INTERNAL BUS to pass through the NAND gate, ALTERNATE OUTPUT FUNCTION must be set to 1, at the expense that the value to output is inverted.
2. Following the signal path, the output of the NAND gate is used to turn ON (with logic 1) or OFF (with logic 0) the Field Effect Transistor (FET), as it is connected to its Gate terminal. This means that if the value of INTERNAL BUS is 0, the FET’s Gate logic value is set to 1, turning ON the transistor; and if INTERNAL BUS is 1, the FET’s Gate logic value is 0, turning the transistor OFF.
3. If the FET is OFF, an open circuit can be assumed between the FET’s Source and Drain pins, so the pull-up resistor connected between the FET’s Source and VCC pulls the pin voltage (P1.x, P3.x or P4.x) up to a logic 1 (hence the name, pull-up resistor). In the opposite case, if the FET is ON, a short circuit is assumed between Source and Drain, driving the pin P1.x, P3.x or P4.x value to a logic 0 (by shorting the pin to ground) and the VCC voltage is entirely absorbed by the pull-up resistor. This means, that the value at INTERNAL BUS is inverted again by the pull-up resistor.
4. This way, the value of INTERNAL BUS value is transferred from the internal data bus of the microcontroller to any pin of ports P1, P3 or P4 as soon as ALTERNATE OUTPUT FUNCTION is set to 1, as shown in the table below:

|  |  |  |
| --- | --- | --- |
| INTERNAL BUS | ALTERNATE OUTPUT FUNCTION | P1.x, P3.x or P4.x |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

EXERCISE: On the same fashion, describe the path followed by the value of INTERNAL SIGNAL towards P0.x (see Figure 2 of the datasheet). Complement your description with a truth table that shows the change at P0.x depending on INTERNAL BUS, ADDRESS LOW/DATA and CONTROL signals.

|  |  |  |  |
| --- | --- | --- | --- |
| INTERNAL BUS | ADDRESS LOW/DATA | CONTROL | P0.x |
| 0 | 0 | 0 |  |
| 0 | 0 | 1 |  |
| 0 | 1 | 0 |  |
| 0 | 1 | 1 |  |
| 1 | 0 | 0 |  |
| 1 | 0 | 1 |  |
| 1 | 1 | 0 |  |
| 1 | 1 | 1 |  |

## Memory Organization

The special function registers (SFRs) of the 8051’s architecture are 8-bit memory locations that are used to carry out specific funtions. Each of the SFRs have a unique address in the device’s memory and a name that allows us to easily identify all the different registers while writing programs for the microcontroller. For example, the register Accumulator is located at address 0xE0 and is identified as ACC in the instruction set. The list of SFRs for the AT89C51 microcontroller can be found in pages 11-13, and their corresponding memory map in page 14 of the datasheet.

The accumulator (ACC) is an SFR used to store the result of any operation, either logical (AND, OR, NOT, etc.) or arithmetical (ADD, SUBSTRATION, MULTIPLICATION, etc.) carried out by the Arithmetic/Logic Unit (ALU). Registers P0 through P4 hold the port values that are either read from the outside when working as an input or set to be sent to the outside when working as outputs (which means that the ports are bidirectional). After an ALU operation, we can send out the result through, for example, Port 2, by moving the held by ACC to P2. We can also do the opposite and move the value at P2 (which is set from an external device) into ACC to carry out an ALU operation or store it in data memory.

For some SFRs, each bit has an specific purpose and can be either written or read, depending of such function. For example, the Program Status Register (PSW) at address D0h, holds 8 status bits that are set by the architecture depending upon the result of an ALU operation:

A screenshot of a computer

Description automatically generated with medium confidence

Each bit of PSW will the us something about the result from the operation an ALU has just carried out. The table below shows the meaning of each of these status bits:

Table

Description automatically generated

If the ALU carries out an addition,

We can see the memory mapping of the 8051 architecture by using a simulator such as [EdSim51](https://www.edsim51.com/), which is free to use and runs on Java, thus no installation is required to execute it. Moreover, [free code examples](https://www.edsim51.com/examples.html) are also available for you to test the simulator and study the architecture.

A picture containing graphical user interface

Description automatically generated

Run program “1. Binary Pattern on the Port 1 LED” and execute it. Observe that the values of P1 increase in a binary count. Moreover, this information can also be graphically observed with the LED strip shown. Finally, P1 pins can be monitored as well on the right panel, which indicates P1.0 through P1.7, as well as the rest of the ports.

## Instruction Set

# Deliverable

Record a video explaining the following:

1. Describe the path followed by the value of INTERNAL SIGNAL towards either P2.x (see Figure 3 of the datasheet)