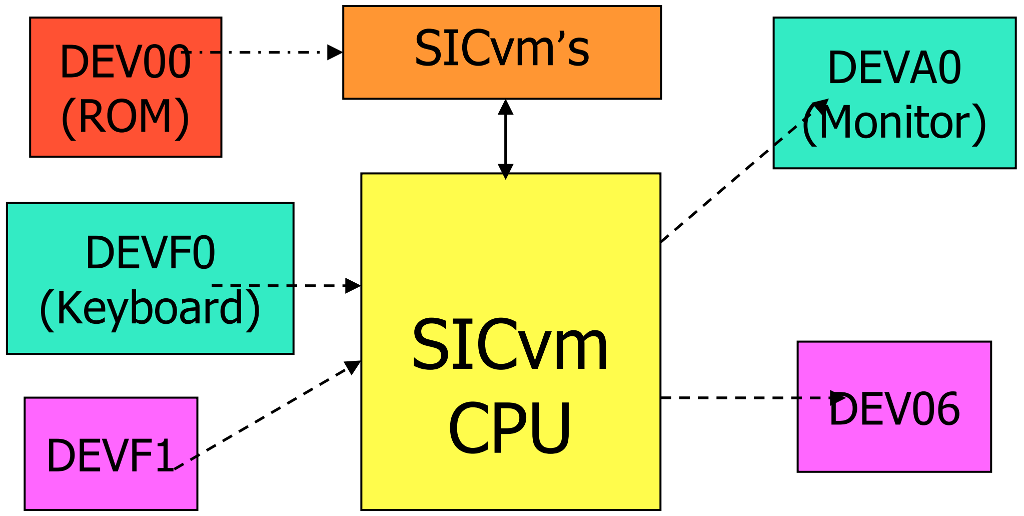
Lab Week 1- 24/09/2014
Eng. Muhammad Sami

1. Go to the course website: http://moodle.manalhelal.com/course/view.php?id=5
2. Download the SICVM from Week 0 material
3. Unzip and make the program:
    a. On windows, you will need MingW32 to compile the source, then open DOS prompt, and navigate to where you unzipped SICVM, then type the following command:
        make win32 - to compile for windows
    b. On Mac or linux, open a terminal and navigate to where you unzipped SICVM, then type the following command:
        make linux - to compile for linux
4. Once the program is compiled, go to bin subfolder, run the “assemble.sh” script
5. Run SICVM using the command sicvm (./sicvm on Mac or Linux).
6. The first command to be entered is “i”, to initialize / turn on the machine.
7. Then load the bootstrap loader from the input device “DEV00” (ROM) using the command “r”.

The SICvm Architecture



8. Make the Bootstrapper loads the absolute loader from DEVF2 and places it in the memory starting from 0x00080 to 0x00300, using the "b" command.
9. Load the minimal shell for SIC using the loader we just loaded using the "l" command.
10. Execute the Shell by entering the "e" command. This will change the prompt to the shell in which you can load programs from devices, execute the loaded program then exit.
11. Now load the add program in DEVF4 using the command "l F4".
12. And then execute using the "e" command.
13. And display the registry contents using the "d" command.
14. Then quit using "q" command.

What's happening in the SICVM:

The assemble.sh converts all assembly files from DOS to Unix to be able to assemble them. These programs are either system programs or sample application programs. The system programs are (we will understand them as we go through the course):

- The Bootstrap Loader "boot.asm"
- The Absolute Loader "ldr.asm"
- The SIC Shell "shell.asm"

The applications programs are:

- A Sample Program Demonstrating Standard I/O "stdio.asm"
- A Sample Program Demonstrating a Simple Add Program "add.asm" and assembled in DEVF4 and we loaded and executed it.

On turning "ON" the machine the Memory locations are initialized to 0xFFH and Registers A, X, L are initialized to 0xFFFFFFH. PC is initialized to 0x000000H and SW is initialized to 0x800000. In the current implementation we use only the Condition Code and Running mode values of SW. This is done to verify the integrity of memory, it's the equivalent of a POST in modern computers.

The bootstrap is loaded into SIC's memory after the initializing procedure where in the memory and the Registers of SIC are initialized. The Bootstrap occupies from 0x00000 to 0x00080.

The Bootstrapper loads the absolute loader from DEVF2 and places it in the memory starting from 0x00080 to 0x00300. After this is done the bootstrap program is no longer required in SIC.
A simple add program in SIC assembly language is as follows:

```
ADD  START    3000
     STA      SHELLAD
     LDA      FIRST
     ADD      SECOND
     STA      THREE
     LDL      SHELLAD
     RSUB
.    HLT


FIRST    WORD     1
SECOND   WORD     10
SHELLAD  RESW     1
```

As compared to a high level language such as C:

```
int main () {
      int x = 10, y = 20;
      int z = x + y;
      return 0;
}
```

We will study more about the SIC programming syntax as we go through the course.