ELE406 FALL2022 LAB1

Rhody Computer System Programming

Overview:

The complete ISA and programmers' reference manual of the Rhody CPU are found in Appendix A of the lecture notes. The Rhody computer system memory map is in Appendix B.1. The second exercise is to familiarize with the configurable assembler: Caspr. We will be using Caspr throughout this semester for the assembly language programming. The ELE406 laboratory exercises will cover both the hardware and the software aspects of the Rhody computer system. At the assembly language programming level, the software is directly tied to the hardware. A proper assembly program is impossible without an intimate knowledge of the underlying hardware, i.e. the CPU and the computer system.

Exercise #1: Basic Rhody Computer System

1. Restore "Rhody_System.qar", compile and then program the DE1-SoC board. Record the compilation summary in the lab report.
2. The initialization file of the program ROM is the "Rhody_CLI". This miniature operating system has fifteen commands. See Appendix B.5 for more details of these commands. Make sure to set SW(9:8) = "11" so you may see both the text and the graphic videos.
3. Open "Rhody_System.vhd". This VHDL describes the entire Rhody system; the equivalent of the "motherboard". Beginning from line 109, the three versions of Rhody CPU are described using the VHDL component instantiation. Since we can only have one CPU, two unused CPUs are commented out. This is equivalent to having an "IC socket" on the "motherboard" so we can swap the CPU at any time.
   - ❖ Go to "output_files" directory/folder to find "Rhody_System.sof". Copy this file and place it in a special folder that you can easily locate later. Rename the file as "Rhody_System_basic.sof".
   - ❖ Go back to "Rhody_System.vhd", comment out "Rhody_CPU" and uncomment "Rhody_CPU_pipe". Re-compile and then re-program the FPGA,

verify that “paint” still work properly. Go to “output_files”, copy “Rhody_System.sof” and save it to the same folder as above. Rename the file as “Rhoy_System_pipe.sof”.

- ❖ Repeat the same procedure as above for “Rhody_CPU_pipe_BP”. Rename the .sof file as “Rhoy_System_pipe_BP.sof”. *This is the version pre-stored in DE1-SoC; “BP” stands for branch prediction.

4. Three ways to put the Rhody computer system on the DE1-SoC board:
   - ❖ The DE1-SoC board in the lab has already been loaded with the Rhody System (Rhody CPU with instruction pipeline with branch prediction) as the power-up configuration.
   - ❖ Use Quartus Prime and open the “programmer”; select the appropriate .sof file for download onto FPGA.
   - ❖ Use Quartus Prime to open/restore the archived “Rhody_System.qar”, compile and then program the DE1-SoC board.

Exercise #2: Reconfigurable Assembler: Caspr

1. Unzip “caspr.zip” and place it in its own directory/folder on the desktop or at a place of convenience. **Do not put it in a folder called “lab1” since we will be using it for all the labs.**
   - For Linux: Open a “Terminal” in Linux and change the working directory to caspr. Use the Linux command “chmod 777 caspr” to modify the privilege mode of “caspr” so it is now executable under Linux.
   - For Windows: The other executable “caspr.exe” is for MS Windows. Use “Command Prompt” under “Windows System” to run caspr.exe in a CLI (command line interpreter) environment.
2. It is easier if all the programs “.asm” files are placed in the same folder as the “caspr”. For the example, place “Hello.asm” program in the caspr folder, then use “caspr” to generate “Hello.hex”. In Linux, the command is “./caspr Hello.asm”.
3. Use the “In-System Memory Content Editor” in Quartus Prime to download “Hello.hex” to the “PROG” ROM. Of course, Rhody computer system must be on the FPGA already. Verify that the message is printed on the MTL2 screen. You will have to push key(0) to emulate a power-up start since the program is changed.

4. A quick summary of how to operate the Rhody computer system:
    - Rhody computer system hardware on FPGA.
    - Use any text editor (e.g. Notepad in Windows) to write ".asm" program; make sure "header" is included.
    - Use "caspr" to generate ".hex" file.
    - Use "In-System Memory Content Editor" to download the ".hex" file.
    - Press key(0) to perform a power-up reset to restart (cold-start) the system.

Assignment: Calling Rhody system library functions

❖ Refer to Lecture notes Ch2.6 for how to use the library functions.

❖ Refer to Lecture notes Ch2.2 for I/O device memory addresses for Timer#0.

1. Write a program call **"draw.asm"**:
    a. Set print format to decimal.
    b. Call the subroutine "wait". This subroutine will wait until user touches the MTL2 screen. Append the subroutine "wait", as shown below, at the end of the main program. Also, this subroutine needs a variable called "oldx", which should be defined in the user data memory, e.g. ".define oldx 0x0801"
    c. Clear graphics screen. Reset timer#0 to 0.
    d. Call system function "circle" to draw a circle at (200, 200) with a radius of 150 and color = 0xC0.
    e. Read timer#0. Print runtime using system function "printn". Record this number in the lab report as the "runtime of system function "circle"".
    f. Call subroutine "wait" to wait for the user to touch MTL2.
    g. Clear graphics screen. Reset timer#0 to 0;
    h. Call system function "scircle" to draw a solid circle at (200, 200) with a radius of 150 and color = 0xF0.
    i. Read timer#0. Print runtime using system function "printn". Record this number in the lab report as the "runtime of system function "scircle"".
    j. Go to b.

2. Program the DE1-SoC board with a different .sof, as prepared earlier in exercise #1. Repeat Step 1 twice for the pipelined version and the pipelined + branch prediction version, respectively.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;wait for touch screen sensor
;Accept next touch only if TX1 changes!
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
wait: ldl   r0, 0
      ldh   r0, 0x10
dly:  adi   r0, 0xFFFF
      jnz   dly
      ldm   r0, trdy
      cmpi  r0, 1
      jz    wait
wait0:
      ldm   r0, trdy
      cmpi  r0, 0
      jz    wait0
      ldm   r0, tcnt
      cmpi  r0, 1
      jnz   wait
      ldm   r0, tx1
      ldm   r1, oldx
      cmp   r0, r1
      jz    wait        ;no respond to old touch
      stm   oldx, r0    ;update oldx
      ret
```

Lab 1 Report:

From Exercise #1:

1. Compilation report for the Rhody computer system (basic CPU):

Logic utilization (in ALMs)	___________ / 32,070 (_______ %)
Total block memory bits	___________ / 4,065,280 (_______%)
Total DSP Blocks	___________ / 87 (_______%)

2. Compilation report for the Rhody computer system (pipelined CPU):

Logic utilization (in ALMs)	___________ / 32,070 (_______ %)
Total block memory bits	___________ / 4,065,280 (_______%)
Total DSP Blocks	___________ / 87 (_______%)

3. Compilation report for the Rhody computer system (pipelined CPU with BP):

Logic utilization (in ALMs)	___________ / 32,070 (_______ %)
Total block memory bits	___________ / 4,065,280 (_______%)
Total DSP Blocks	___________ / 87 (_______%)

From Assignments:

1. Using Rhody basic CPU (Rhody_System_basic.sof)
    - ❖ Runtime of system function "circle": ________________ µs
    - ❖ Runtime of system function "scircle": ________________ µs
2. Using Rhody CPU with pipeline (Rhody_System_pipe.sof)
    - ❖ Runtime of system function "circle": ________________ µs
    - ❖ Runtime of system function "scircle": ________________ µs
3. Using Rhody CPU with pipeline and branch prediction (Rhody_System_pipe_BP.sof)
    - ❖ Runtime of system function "circle": ________________ µs
    - ❖ Runtime of system function "scircle": ________________ µs

Grade: _____________

Assignment Verified by: ____________________________________