Loyola Marymount University

Department of Electrical Engineering and Computer Science

ARM Assembly Language: Digital Clock

Objectives:

To develop a programmable digital clock using ARM Assembly Language in a Raspberry Pi (RPi) and to practice microcontroller subroutines.

Required Equipment:

HARDWARE:

- Raspberry Pi 4B
- Computer monitor
- USB Keyboard
- USB Mouse
- Video adapter HDMI-VGA
- Power adapter for the Raspberry Pi 3B+
- VGA Cable
- LED (3 units)
- Male-Female Jumper Wire (4 units)
- 220Ω resistor (3 units)
- Jumper Wires

SOFTWARE:

- Geany, GNU Assembler, GNU Project Debugger

Prelab:

(1) Write an ARM assembly language program including comments which implements a clock that is updated and displayed on the terminal every second according to these specifications. The time display is to be in "military" time. For example, display "23:59:03" for 11:59 PM, display "00:01:00" for 12:01 AM, etc. Write the program under the following considerations.

Basic function:

The clock is programmable in that it allows the user to enter the initial time via the keyboard, including hours and minutes only. Seconds do not need to be input. At the begging of the program, a prompt will ask the user to set the clock. Once the user enters the hours and minutes, the clock will start and continuously display the clock on the terminal window.

Details:

The overall clock program must be divided in multiple files or subroutines:

1. The main code will call the “setClock” subroutine to prompt user to set the initial value of the clock. It will then update the clock each second by utilizing another subroutine “delay” which will allow a delay of about one second between clock updates. Finally, once the clock in updated, the main routine will call the subroutine “display” to output the value of the clock on the terminal window.
2. There should be four(4) assembly language files: clock.s, setClock.s, delay.s and display.s

clock.s: calls all other subroutines for setting the initial clock value, update the seconds, minutes, hours every second, and outputs the clock value.
setClock.s: prompts user to input one's minute, ten's minute, one's hour and ten's hour.
delay.s: this subroutine creates a delay loop to provide about 1 second delay between the update of the clock.
display.s: this subroutine shows the value of the clock on the terminal screen every second.

3. You must create a makefile for compilation of these four source files into an executable named "clock" using GNU make. Please read about GNU make online or chapter 3 of the textbook "Raspberry Pi Assembly Language Programming: ARM Processor Coding by Stephen Smith, 2019" available at LMU digital library.

Important: It is not allowed to write the code in C and follow steps of tutorial donothing1 to generate the code in assembly from the code in C. The code needs to be completed in assembly directly. Only the procedures of the tutorial can be used in your solution. It is not allowed to borrow code from anywhere else.

(2) Simulate the code that updates the clock value only. The time delay and input/output parts of your code can be tested in the lab only because the RPi needs to be connected to the computer. Turn in a screenshot of your simulation results demonstrating that the clock transitions correctly from 12:59:59 to 13:00:00, from 13:59:59 to 14:00:00, and from 23:59:59 to 00:00:00. You can use the online simulator CPUlator for this.

(3) Turn in a flow chart of your clock program.

EXPERIMENT

(1) Check the operation of your program with such critical times as 12:59:59 and 23:59:59 by initializing the clock to these times.

(2) Determine the accuracy of your clock by timing it over several minutes with your cellphone's timer.

(3) Obtain the signature of the instructor when finished.

REPORT

(1) Submit the report according to the lab-report guidelines.

(2) Discuss any problems you encountered after downloading your program and how you fixed them. The answers need to be included in the limitations subsection of the report, which is described in the report guidelines.

(3) Attach a copy of your debugged program (with the debugging marked).

(4) What are the factors that might affect the accuracy of your clock? The answers need to be included in the limitations subsection of the report, which is described in the report guidelines.

(6) If your clock were to lose 10 microseconds per second, how much time would it lose per day? How much per year? How accurate would your clock have to be (microseconds per

second) to be accurate to within 1 second per year? The answers need to be included in the limitations subsection of the report, which is described in the report guidelines.

(7) Express all of the values in part (6) in percent and in parts per million. The answers need to be included in the limitations subsection of the report, which is described in the report guidelines.