Sign-off sheet

Problem #2 _____

Problem #4 _____

2. You will now modify your program to accept console TTY input/output, such as you will also do on the PDP-8 you will build for this class. Examples of how to poll the terminal are found on pages 354-355 in Prosser and Winkel. Your modified program will accept n from the console as input in decimal (terminated by a carriage return). The answer (the number of the first n odd numbers) will be printed out to the console, followed by a carriage return, as a decimal number. The attached table describes the 7-bit ASCII code. The left-most bit in the byte going to and coming from the terminal is zero. Test your program on the emulator and get this problem signed-off on the attached sign-off sheet, indicating that you have demonstrated a working program to the instructor or TA. For this problem, please also turn in your PAL program and the listing file generated by the assembler.

3. *Using the RIM loader.* In `/user/ee4340/public/rim.pal`, you will find the PAL assembly code for the RIM loader. This routine is toggled into the machine and run to allow programs in RIM format to be quickly read into the PDP-8 through the TTY or paper-tape reader devices. From this PAL program, describe the RIM format. From the point-of-view of the PDP-8, what differences are there between the behavior of the TTY device and the paper-tape reader?

4. In this problem, you will write a simple calculator in PAL which performs decimal addition and subtraction. You program will take TTY input of the form `-23+34-34` or `34-45-78`, followed by a carriage return. The computer will then return the answer in decimal. In the event of an overflow or underflow, print `OF` to the TTY output. Test your program on the emulator and get this problem signed-off on the attached sign-off sheet. For this problem, please also turn in your PAL program and the listing file generated by the assembler.

- A block of comments are provided for each subroutine to provide the same information.
- Comments on groups of statements can provide "high-level" detail on the function performed by the group, while comments on individual lines can describe the contribution of that line.
- Comments on variables and block storage describe contents and usage.

1. Write a program in PAL that computes the sum of the first n odd numbers. Store n is memory location 0002 and place the result in memory location 0003. Begin storing the program at memory location 0200. Use comments to extensively document your PAL program. Assemble your program into machine language by hand. To check your results, you can run the PAL assembler by typing `pal -r foo.pal`, where `foo.pal` is the file name of your PAL program. A listing file will be generated in `foo.lst`, which you can compare with your hand-generated results. You will also notice a file `foo.rim`, which is your program in PDP-8 RIM (Read-In Mode) format. You will work with this in more detail below.

You now need to enter and run your program on the PDP-8 emulator. To bring up the emulator, type `pdp8e`. You will then get an X-windows display that resembles the front-panel of the actual machine. There are two rows of LEDs in the front panel. The first row displays the contents of the memory address register. The second row displays displays the contexts of the facility selected from the options: `STATE`, `STATUS`, `AC`, `MD`, `MQ`, and `BUS`. The most useful of these are `AC`, which displays the contents of the accumulator, and `MD`, which displays the contents of the memory data bus. You will now need to toggle in your program (in much the same way you will do once you have built the real machine). To do this, enter the first address into the switch register (the row of switches on the front panel). Down on the switches is logic '0', while up is logic '1'. Then hit "ADDR" to load the address. Next enter the data to be loaded to this address and hit "DEP" for deposit. The address will automatically increment and you can enter the data for your program sequentially from this point.

Run your program and verify the result in location 0003 for different values of n in 0002.

For this problem, please turn in your tested PAL program and the listing file generated by the assembler.

Columbia University
Department of Electrical Engineering
EE E4340. Homework/Laboratory #1.
Instruction-Set Architecture and Assembly Language Programming of the PDP/8.
Due: January 28, 1999

In this laboratory, you will learn about the PDP-8 instruction set and about PDP-8 Assembly Language (PAL) programming. For reference, please refer to the "Programmer's Reference Manual for the PDP-8" and chapters 7 and 9 of Prosser and Winkel (both handed out in class). Although the emulator we are using is for a PDP-8E, we are assuming that we can employ only the capabilities of the PDP-8I (in particular, we only have 12 bits of address for 4K 12-bit words of memory, no extended arithmetic element, and no group-three microcoded instructions). We will be employing only two I/O devices: console TTY input-output and high-speed paper tape input-output. We will use the latter with the emulator to load programs by mounting a file onto the paper tape device. When we actually implement the PDP-8I in hardware, we will only support the TTY device through a serial port and we will use this device to load software as well.

While the PDP-8 machine is old (1967), working with PAL is not entirely an academic exercise. For example, forty PDP-8 machines still control the signs in the stations at BART (Bay Area Rapid Transit). One of the current PAL assemblers was actually written for use by BART to modify the software for these machines.

For this lab, you will need to use the SUN machines in the VLSI Design Teaching Lab (1212 Mudd). These machines are also available for remote log-in from the PC's in Mudd 1235. Example files are kept in `/user/ee4340/public`. Software is kept in `/user/ee4340/bin`, which should be added to your UNIX search path.

The program `square.pal` gives an example of PAL programming. Please note the extensive use of comments to document the program. This absolutely essential to make assembly code at all understandable by someone else.

- A block of comments at the top of the program identify what the program does and provide detail on inputs and outputs