Laboratory 5: PIPPIN – Assembly Programming

Assembly Programming

LOD #2	= 0001 0100 0000 0010
ADD #3	= 0001 0000 0000 0011
STO Y	= 0000 0101 1000 0010
HLT	= 0000 1111 0000 0000

Originally, computers were programmed in their binary “Machine” language. Even after human readable programming languages were introduced, some computers (e.g. Digital Equipment Corporation’s PDP8) still needed to be “boot strapped” by entering a sequence of binary instructions into their program memory using switches on the console. This allowed the computer to read binary instructions from a peripheral (Punched tape or later, Magnetic tape) and execute a user’s program.

An “Assembly” language makes programing a machine easier by using simple, human readable names for each of the supported executable instructions. More sophisticated “Assemblers” also allow “Macros” that can expand into useful sequences of machine instructions. The output of a Macro Assembler is a “Relocatable Binary” file. The process of building an executable binary file then “Binds” the addresses in the set of Relocatable files to take into account the actual sizes and location of each instruction in the set of files being “Linked” together.

Today most programmers use “Higher level” programming languages whose compilers translate a source file to run on a given machine. Some of these compilers just translate down to an assembly program which then must be run through the “Assembler” and “Linker” to produce the executable binary file.

PIPPIN

The PIPPIN CPU Simulator

PIPPIN was written a few years ago to help students visualize the machine “Fetch-Execute” cycle. It is a Java Applet that runs in the Java interpreter “Sandbox” (since PIPPIN is not a “signed” applet, the file access mechanisms are blocked and you will need to manually enter your programs and use the “PrintScreen” functions to save images of your programs.) that you start by loading:

cpusim.html

Into your Java enabled Internet Browser.

An Introduction to Using PIPPEN from the original Authors is:

The+PIPPIN+Machine.pdf

And a User Manual is:

PippinGuide.html

An Example Assembly Program

I wrote and tested the following PIPPIN assembly program which calculates 2^n for a range of values (1, 2, 4, … , 2^n) for you to use as an example in your effort to write and test your own assembly program.

Address	Instruction	Comments	
0	LOD #1		Initialization
2	STO Z	Initialize Output variable (Z) to 1	
4	LOD #2		
6	STO X	Store 2 into the variable X to use as the exponent	
8	LOD #4		
10	STO Y	Set variable Y to 4; stop at 2^4	
12	NOP		
14	LOD Z	Get the current output value	Generate the next output value
16	MUL X	Multiply by 2	
18	STO Z	Store the new output value	
20	NOP		
22	LOD Y	Get the current count	Loop control logic
24	SUB #1	Reduce Y by 1	
26	STO Y	Store the new count in Y	
28	JMZ 32	Stop by using the halt at location 32 if the count is zero	
30	JMP 14	Return to location 14 for another cycle of the loop	
32	HLT	Stop, we're done, 16 = 2^4 is the value in Z	

Assembly Code Assignment

Write, test and document one of the following problems in PIPPIN assembler. As there are more of you than there are defined problems, some of you will be writing code to do the same task. **I expect independent work!**

1. Distance Travelled

Calculate the final location for a car that starts in position D_0, goes V_0 miles/hour and accelerates at a rate of A_0 Miles/Hours^2. The equation is (if you have forgotten your physics):

$$D = D_0 + V_0*t + ½A*t^2$$

2. Prime Numbers

Calculate the first N prime numbers (1, 2, 3, 5, 7, 11, …)

3. Factorials (N!)

Calculate the first N Factorials (1, 2, 6, 24, 120, 720, …)

4. Perfect Squares (N^2)

Calculate the first N squares (1, 4, 9, 16, 25, 36, …)

5. Binary to Decimal

Given an eight bit binary number, convert it to its decimal equivalent

6. Algebraic Sum

Calculate the sum of the numbers starting at N_1 through N_2 ($N_2 > N_1$ test for validity)

7. Geometric Sum

Calculate the geometric sum: $W = \sum_{n=1}^{K} a^n$ where a > 1 is a small integer

8. N take k

Calculate the number of ways you can remove k objects from a basket of N objects. The equation is:

$$\binom{N}{k} = \frac{N!}{k! * (N-k)!} = \frac{\prod_{(k+1)}^{N} m}{(N-k)!}$$

K must be less than N (test for validity) and since PIPPIN is slow, keep K small.

PIPPIN Laboratory Report

Your report should include:

1. A restatement of the problem your program is computing
2. A commented copy of your PIPPIN program and some screen shots of the PIPPIN system executing your program
3. Discussion of any difficulties you ran into and your solution
4. Your thoughts on the effectiveness of programing in assembly vs a higher level language.