Computer System Design Lab
Design Experiment 4:
Assembly Language Programming

Indian Institute of Technology Tirupati

Oct 22, 2020

1 Objective and Problem Statement:

As a computer system designer, it is essential to be able to see the relationship between software and hardware. Assembly language and machine level code provides an interface between the software and underlying hardware which facilitates high level programming. For a processor the machine and the assembly code are specific to the processor, different processor architectures have their own assembly language and therefore the machine level code.

The objective of this experiment is to explore the HACK hardware-software interaction through a set of assembly programs. A chosen set of problems will be solved using the HACK assembly programming which are then **executed in a CPU emulator to verify the correctness and efficiency of the program**.

As part of this exercise the following set of programs need to be coded and verified.

1. Arithmetic and logic operation
   (a) Write a program to perform arithmetic and logic operations on two numbers as $c = a \ op \ b$ where a and b are stored in register D and A respectively, the result c will be stored back in register D. Replace *op* with $+, -, \&$, and $|$.
   (b) Write a program to preform arithmetic and logic operations on two numbers as $c = a \ op \ b$ where a and b are stored in some memory location in Data Memory, and the result c will be stored back in the memory location. Replace *op* with $+, -, \&$ and $|$.

2. Control operation
   (a) Given a number stored in a memory location, write a program to identify whether the number is even or odd. If the number is even indicate this by storing number 0 in the location identified as EVEN, else if the number is odd indicate this by storing number 1 in the location identified as ODD.
   (b) Code a program to compare two numbers in such that if $a == b$ then $c = 0$, if $a > b$ then $c = 1$, if $a < b$ then $c = 2$. The source and destination operands a, b, and c are memory operands.

3. Iteration and loop
   (a) Code a program to count the total number of elements in a given one dimensional array. The counter number will be stored back in a memory location. This program need to implement the concept of loop/iteration.
   (b) Given two 2-D matrices A and B, write a program to perform $[C]_{n\times m} = [A]_{n\times m} + [B]_{n\times m}$. All three matrices are to be stored in respective memory locations.

4. Function and subroutine call

   The programs has to be coded as function call. The concept of stack needs to implemented to keep track of the local variables and stack pointers etc.

   (a) Write a program to perform modulo operation as $b = mod(a, m)$. Since there is no divider unit in the ALU, the modulo function can be computed using repeated subtraction. Here, the operand a, b, and m are stored in memory.

   (b) Write a program to solve the following equation: $f(x) = f(x-1) + f(x-2)$, $f(0) = 1$ and $f(1) = 1$ where $x \epsilon \{positve\ integer\}$

5. Accessing I/O device

   (a) Code a program which continuously listen to the key stroke and record the ASCII value into the memory location RAM[24576]. Next, the content of location RAM[24576] need to be moved to a specific location in memory such that it can be accessed later on.

   (b) Code a program to display a pattern of alternative black and white pixels in 256×512 display screen of HACK system.

2 Experimental Flow

1. Once the program is coded pass it through assembler and generate corresponding machine code using the assembler tool. The tool can be invoked by Assembler.sh.

2. Next, once the error free code has been generated, the .asm program (or the generated machine code) can be run using the CPUemulator (CPUemulator.sh). The CPUemulator is an emulator tool that emulate the HACK processor that has been designed. The .asm file which is an assembly program file to be loaded to the CPUemulator to get executed. The content of the register and memory can be observed in the screen.

   These tools can be find in the tools directory and can be launched directly from the same directory. You may add the path to .bashrc file to launch the tools from the current working directory.

3. There are certain syntax which at first-look seems to be supported but if the emulator throws an error it means that the syntax is not supported and the alternative set of instruction need to be used.

4. For each program maintain a corresponding output file where you keep the log of execution of the program for different set of inputs. Explore if there is any way the log file can be directly saved from the emulator.

3 Tools:

- Language: The Nand2Tetris HDL and TSL (test scripting language)
  Refer: Appendix A and B of text book.

- Tools: Assembler.sh and CPUemulator.sh
  `https://www.nand2tetris.org/software`

- Machine and OS: x86_64 machines with any distribution of Linux (Ubuntu or CentOS). (Your personal laptop/desktop is sufficient.)

4 Reporting and Evaluation

All the programs will be evaluated based on correctness and robustness (how good the program handles variety of inputs) of the program. Efficient with respect to number of instructions and memory space used will be considered.

Write an one page report in the common report shared doc file. The report should briefly notes the learning and challenges that is involved during experiment.

The experiment will be evaluated on 29th Oct along with viva-voce.