Design Notes

Jeremy Case, Xinhua Fan, Teodor Georgiev, Julie Yu

For Test Program 1:

The design for test program 1 works by keeping track of 21 user inputs, and comparing the first 20 inputs to calculate which of them was closest to the 21st input. This happens essentially in 5 steps, all of which will be broken down in more detail:

Step 1 works by taking user input via the keyboard, and storing it into memory. User inputs must be parsed one char at a time, so the program takes each char, adds the ASCii value of that char to a tally, and stores that tally to memory. Furthermore, the program pre-loads 10 memory positions with the numbers 0-9. The program bumps each char input against these 10 memory positions with a loop. If a char matches one of these memory location values, the program knows to tally the char and add to memory.

Step 2 was mentioned above: after ensuring a char is valid, the program adds the char to a tally. In effect, this ensures a user input of say 12,345 gets parsed as chars 5,4,3,2,1 (our keyboard buffer works "backwards") and placed to memory only to get tallied up again back to their original value.

Step 3 essentially tallies the general register 2 each user input. When this general register reaches a value of 21, the program places the final user input into memory much as step 2 did, but then goes to a special loop of the program.

Step 4 works by looping through each of the first 20 user inputs (conveniently placed into memory) against the last user input. This is done by copying the user input to another general register and then subtracting from that register the current iteration's memory location (iterating through 20 memory positions). When a value gets a lower value after subtraction than was currently in memory, the program puts this new value into memory. In addition, the program places the input used to determine this value into a register.

Step 5 works simply by outputting whatever value was left standing in the aforementioned register containing the closest user input from step 4.

For Cache:

We implemented the cache as a layer on top of memory. In the past, the CPU contained a reference to the Memory object. Now it contains a reference to a Cache object, which internally references Memory. This is all as per the professor's lecture. The cache implements a write-through policy, so whenever a value is changed in the cache, it is written out to memory.