Number Guessing Game

For this experiment I've created a simple number guessing game. The object of this game is to guess a number between 0 and 1023 (inclusive). Once you've guessed it correctly it is the MPF-1's turn to guess a number in the same range which you have come up with.

If you study the program listing of this experiment you can learn some very common practices in micro processor assembly programming: Decimal to binary conversion, binary to decimal conversion, leading zero suppression, random number generation and successive approximation, to name but a few.

After downloading the software package you can play the mp3 file to the MPF-1 by using file name 0007.

User Guide

The operation of the program is divided into two sections. In the first section it's your turn to try to guess a number between 0 and 1023 (inclusive) which the MPF-1 has picked randomly. After each guess the MPF-1 tells you whether you should try a higher number (indicated by an H following your guess), or whether you should try a lower number (indicated by an L). Once you've guessed the secret number correctly the MPF-1 tells you how many guesses it took you to get it right.
Simply enter your guess and press **GO** to see if you've guessed correctly. If you make a typing error while entering the number you can clear the entire number by pressing **DEL** after which you can restart typing your correct guess.
The program allows you to enter values higher than 1023, which of course will always result in an L shown after your guess. Numbers above 65535 will result in an error because that is the maximum value a 16-bit number can hold.
Once you've guessed the number correctly the display will show "Yes.", followed by the number of tries.

After you've guessed the number correctly you press **GO** to enter the second stage of the program. Now it's your turn to think of a secret number in the range of 0 and 1023, and the MPF-1 will try to guess it. After the MPF-1 shows its guess you tell it whether it should guess higher (by pressing the **+** key), or whether it should guess lower (by pressing the **-** key). When the MPF-1 has guessed your secret number correctly you press the **GO** key and then you'll see how many guesses the MPF-1 needed.
Pressing **GO** once more will restart the program from the beginning.

You can restart the game at any time by pressing **REG** , after which it will be your turn to guess a newly picked secret number again. When it's your turn you may give up and let the MPF-1 guess your secret number by pressing **DATA** .

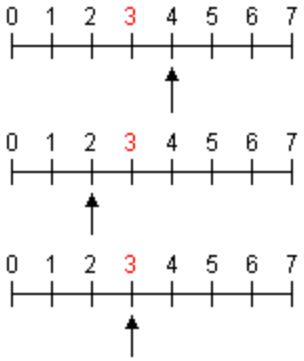
TIP: You can cheat by pressing **PC** when it's your turn to guess a number. The secret number will be displayed as long as you hold the key down.

Operation

There are many things you can learn by studying the source code of this program. I've already mentioned a few, like: Decimal to binary and binary to decimal conversion, leading zero suppression, random number generation and successive approximation.

Maybe the successive approximation deserves a bit more explanation here.

Successive approximation is often used to convert an analog signal to digital. We are not going to convert an analog signal to digital here, however we're going to use a similar technique.



The principle behind it is quite simple. When we want to guess a number in a predefined range we simply split that range in half and make our first guess right in the middle. If we are instructed to guess higher we split the new range (from half way to the top) in the middle again. If we are instructed to guess lower we split the new range (from the bottom to half way) in the middle again. This process is repeated until we are informed that we've found the correct number, or until we can't split the range in two any longer.
In software this is fairly easy, as long as we stick to maximum numbers which are a power of 2. In our program we use 10 to the power of 2 (minus 1) as maximum number, which is a 10 bit number. Our first guess is now simply 1000000000B (which is 512 in decimal). If we should guess higher we leave that "1" bit where it is, if we should guess lower we clear it to "0" again. Then we shift our mask 1 bit to the right and add it to the previous result, this makes our next guess. This continues until the mask becomes 0 or until we've struck gold earlier.
What would be the maximum number of guesses the computer would need to guess any number you come up with in the given range? Is it a surprise that this maximum number is 10 for a 10-bit number?

Well, there is a slight catch. There is one exception, and that is when you want the computer to guess 0. A range with an even number of integers can not be split right in the middle without getting a fraction. Actually with a 10-bit range 511.5 should be our first guess. This is not appropriate, therefore we'll compromise by starting at 512. However this means that it would take 11 guesses only if the target value is 0.
Try to figure this out from the diagram in the picture above. In this 3-bit example you would normally need a maximum of 3 guesses. However it requires 4 guesses to get to 0.

Room For Improvement

As always there are some things you can alter to this program to improve it, or to adapt it to your own ideas. Changing existing programs is a great way of teaching yourself the assembly skills.

- You can alter the program to get different maximum values.
- What would be the easiest way to change the range from 1 to 1024 instead of 0 to 1023?
- You may give some penalty points to the human guesser when he or she cheated!
- When you let the MPF-1 guess a secret number of 0 it will tell you that it needed 10 guesses to get it. However this is wrong, in fact it needed 11 guesses. I didn't bother about that, but it's a good exercise for you to correct this.

Download

Here you can download the file guess.zip (guess.zip), which contains the source listing, the actual plain hex file, a list file and an mp3 file of the program.