Second Assignment in PoP

Jakob H. Schauser, pwn274

September 2021

Contents

1 2i0

As with so many things in life, there are no clear-cut rights or wrongs when it comes to running f# code. The gray area that is code-compilation has multiple possible solutions - each with their own advantages and drawbacks. The three main ways are comprised of:

1.1 Compiling and running with a single command

The first way of bringing f# code to life is via the "fsharpi"-command. This command interprets the f# code in a file and runs it. One might suspect the command "fsharpi" to be the best, as it certainly is the most convenient. But this method of interpretation certainly has some disadvantages. Every time the program has to be run, it will be interpreted anew, making this method slower in the long run as the work done by the intepreter has already been done and is therefore utterly reduntant.

1.2 Compiling and running with two seperate commands

The choice could also fall on first compiling the code before running it on the computer. This is typically done via the "fsharpc" command, which uses dotnet to create an executable .exe file. This file can then, through use of mono, be understood by the computer and runned. The smart thing here is that once the executable is produced, it can easily be run at your heart's content without need for new compiling. The inevitable drawback of this method is, that the compilation and subsequent running of the program takes longer than the direct interpretation as described in the last section.

1.3 Running interactively

Write pretentious stuff here

2 Converting between bases

2.1 From binary to decimal

I use the trusty doubling technique. Given $(10101)_2$, moving from left to right I double the current total and add the next. This gives:

- 1
- 2
- 5
- 10
- 21

Meaning the final answer is $(10101)_2 = (21)_{10}$

2.2 From decimal to binary

Halving technique from $(10)_{10}$:

- 5 - 0
- 2 - 1
- 1 - 0
- 0 - 1

Giving $(10)_{10} = (1010)_2$.

2.3 From binary to hex

asd $(10101)_2$ looking in groups of four from right:
(1)(5) = 15

2.4 From hex to binary

asd opposite in groups of four $(2F)_{16}$: (0010)(1111) = 101111

2.5 From binary to octal

sad same as to hex but groups of three, again using $(10101)_2$: (2)(5) = 25

2.6 From octal to binary

asd Again same as to hex but groups of three, now using $(73)_8$: (111)(011) = 111011