- CPSC 275: Introduction to Computer Systems

CPSC 275: Introduction to Computer Systems

Fall 2025

- Syllabus
- Schedule
- Resources
- Upload
- Solution

Assignment 4: Computing Digital Roots

Due 5:00 p.m., Monday, October 27

> **IMPORTANT!** This is an individual assignment. You may discuss broad issues of interpretation, understanding, and general approaches to a solution. However, the development of a specific solution or program code must be your own work. The assignment is expected to be entirely your own, designed and coded by you alone. If you need assistance, please consult your instructor or the TAs. Be sure to read the specific policies outlined in the **Academic Honesty in Computing** section.

Problem

The *digital root*, also called the repeated digit sum or persistent sum, is determined by repeatedly adding the digits of a number until a single-digit value is obtained. It can be recursively defined as follows:

1. **dr**(n) = n if $n < 10$.
2. **dr**(n) = **dr**(sumOfDigits(n)) if $n \geq 10$.

For example, the digital root of 4789 can be computed as follows:

```
dr(4789) = 4 + 7 + 8 + 9 = 28
dr(28) = 2 + 8 = 10
dr(10) = 1 + 0 = 1
```

Thus, **dr**(4789) = 1.

The digital root is often used to quickly verify arithmetic calculations and detect simple errors in *checksum* and *hash* algorithms to verify data integrity or detect transmission errors. Also, in cryptography, digital roots are used to strengthen encryption and ensure secure key generation.

Write a recursive function `dr` in IA-32 assembly that given a positive number, n, returns **dr**(n). Using this function, write an assembly program named `dr.s` in which the `main` prompts the user for a positive number and prints its digital root. Compile your program with:

```
$ gcc -m32 -o dr dr.s
```

Run it with:

```
$ ./dr
Enter a positive number: 4789
The digital root of 4789 is 1.
```

Programming Notes

- You may use **only** the IA-32 instructions listed here.
- The recursive definition provided has a closed-form solution: **dr**(n) = 1 + (n – 1) mod 9. However, you are *not* allowed to use this closed-form expression to solve this exercise. Instead, you must implement the function recursively.
- Your program must have the following structure:

```
# program header comment
#
-- definitions such as strings
   .globl main
main:
   pushl %ebp
   movl  %esp,%ebp
   .
   .
   .
   leave
   ret

# function header comment
#
func1:
   .
   .
```

Documentation and Style

Your program should start with a header comment that includes its name, author, purpose, and last modified date. The code section must be structured into four columns separated by tabs: labels, opcodes, operands, and comments. Be sure to annotate your code to explain the function of each instruction.

Handin

Submit your assembly code file (`dr.s`) to the course website.

- **Welcome: Sean**
  - LogOut

Trinity College
HARTFORD CONNECTICUT