Department of Electrical and Computer Engineering
Rutgers, The State University of New Jersey

Computer Architecture and Assembly Lab
Fall 2018

Lab 1
Introduction, GitHub tutorial, Number representation

Introduction

Grading Policy
Attendance and participation - 10%
Labs - 90%, each lab is 15% of your grade

Submission Policy
We will be using GitHub and Sakai for submissions.
GitHub: Every student will create a **private** repository named RucompArchLab-F18. The TA of each lab will be added as a Collaborator. This GitHub repository will contain one folder for each lab. For every lab there will be a report and/or code submission. **ALL FILES SHOULD BE IN THE GITHUB REPOSITORY.**
Sakai: Sakai will be used for archive reasons. Each student will have to upload a zip file of the GitHub repository for each lab. The submission will have to be before the deadline. **No late submissions will be graded.**

Plagiarism and Cheating
Collaboration and discussion between students is allowed, but **please submit your own code and report**. All submissions are being tested for plagiarism. Copying will be considered cheating and has serious penalties.

GitHub Tutorial

GitHub is one of the most popular code hosting platforms for version control and collaboration.

Tasks:
1. Everyone should create a student GitHub account using their Rutgers email (or change an existing account to student account using the Rutgers email) in order to be able to create unlimited private repositories. We will only need one private repository for the computer architecture lab.
After you created an account with the Rutgers email and verified the email, you can get unlimited private repositories here:
https://education.github.com/pack/join

2. We will be following this official GitHub tutorial to get familiar with GitHub.
https://guides.github.com/activities/hello-world/

3. Everyone will create a private GitHub repository and add (TA GitHub user name) as a collaborator. This guide describes the process in detail https://help.github.com/articles/inviting-collaborators-to-a-personal-repository/

Number Representation

1. Unsigned Integers

If we have an *n*-digit unsigned numeral $d_{n-1}d_{n-2}...d_0$ in radix (or base) r, then the value of that numeral is $\sum_{i=0}^{n-1} r^i d_i$, which is basically saying that instead of a 10's or 100's place we have an r's or r^2's place. For binary, decimal, and hex r equals 2, 10, and 16, respectively.
Just a reminder that in order to write down large number we typically use the IEC prefixing system:
Ki = 2^{10}, Mi = 2^{20}, Gi = 2^{30}, Ti = 2^{40}, Pi = 2^{50}, Ei = 2^{60}, Zi = 2^{70} , Yi = 2^{80} .

1.1 Conversions

a. Convert the following from their initial radix to the other two common radices:
0b10001110, 0xC3BA, 81, 0b100100100, 0xBCA1, 0, 42, 0xBAC4

b. Write the following using IEC prefixes: 2^{14}, 2^{43}, 2^{23}, 2^{58}, 2^{64}, 2^{42}

c. Write the following as powers of 2: 2 Ki, 512 Pi, 256 Ki, 32 Gi, 64 Mi, 8 Ei

2. Signed Integers

Unsigned binary numbers work for natural numbers, but many calculations use negative numbers as well. To deal with this, a number of different schemes have been used to represent signed numbers, but we will focus on two's complement, as it is the standard solution for representing signed integers.

2.1 Two's complement

• Most significant bit has a negative value, all others are positive. So, the value of an n-digit two's complement number can be written as: $\sum_{i=0}^{n-2} 2^i d_i - 2^{n-1} d_n$

• Otherwise exactly the same as unsigned integers.
• A neat trick for flipping the sign of a two's complement number: flip all the bits and add 1.

• Addition is exactly the same as with an unsigned number.
• Only one 0, and it's located at 0b0.

2.2 Exercises

For questions 1 – 3, assume an 8-bit integer and answer each one for the case of a two's complement number and unsigned number, indicating if it cannot be answered with a specific representation.

1. What is the largest integer? The largest integer + 1?

2. How do you represent the numbers 0, 3, and -3?

3. How do you represent 42, -42?

4. What is the largest integer that can be represented by any encoding scheme that only uses 8 bits?

5. Prove that the two's complement inversion trick is valid (i.e. that x and x'+ 1 sum to 0).

6. Explain where each of the three radices shines and why it is preferred over other bases in a given context.

3. Counting

Bitstrings can be used to represent more than just numbers. In fact, we use bitstrings to represent everything inside a computer. And, because we don't want to be wasteful with bits it is important that to remember that n bits can be used to represent 2^n distinct things. For each of the following questions, answer with the minimum number of bits possible.

3.1 Exercises

1. How many bits do we need to represent a variable that can only take on the values 0, π or e?

2. If we need to address 2TiB of memory and we want to address every byte of memory, how long does an address need to be?

3. If the only value a variable can take on is e, how many bits are needed to represent it.