EE3450 Computer Architecture
Project: Find Greatest Common Divisors (GCD)

Due Date: 6/1/2021

1 Objective

In this project, you are asked to implement two classical algorithms to compute the greatest common divisor (GCD) of any two positive integers in C and in MIPS assembly. In addition to coding, you will use MARS as the simulation tool to perform analysis and comparison on these algorithms.

The definition of the GCD of any two nonzero integers is given below:

Definition 1.1 (GCD)**.** Let a, b be nonzero integers. The greatest common divisor (GCD) $\gcd(a, b)$ of a and b is the unique greatest positive integer that divides a and b; that is,

(i) $\gcd(a, b)$ divides a and b,

(ii) If d' divides a and b for some positive integer d', then $d' \leq \gcd(a, b)$.

For example, $\gcd(24, 90) = 6$ and $\gcd(-10, 4) = 2$. To simplify the situation, we only consider the GCD of two positive integers.

We will enumerate three algorithms with different implementation method for you. For each of them, please do the following:

- Use the described method to compute the GCD of any given two positive integers in C.
- Use the described method to compute the GCD of any given two positive integers in MIPS assembly and verify your result with the C version.
- Try to figure out what factors influence the performance of your code (e.g., instruction count) and **give some discussion in the report**. The discussion is not necessarily rigorous. You can give some intuition on your observation.

All your programs need to be explained explicitly and carefully with **comments in source codes**. It will be an important part of grading. **Do not copy your codes into the report** but instead discuss your insight into algorithms or the key idea of your implementation.

After you are done with these algorithms and plots, you must perform analysis and comparison on them with various metrics in your report. For examples: **complexity**, **code size**, **instruction type distribution**, etc.

You might further discuss how an algorithm trade off among these (or any other) metrics. There is no standard template for this report, but you need to **make a conclusion** (e.g., what you have learned or accomplished) in the end of the report. Note that **coding (including comments) and the report will take identical proportion** in grading in this project.

2 Problem Sets

Euclid found the following simple fact to compute the GCD of any two positive integers by subtraction and comparison:

Fact 2.1. *Let a, b be two positive integers.*

(E1) *If $a \neq b$, say, $a > b$, then* $\gcd(a,b) = \gcd(a-b, b)$.

(E2) *If $a = b$, then* $\gcd(a,b) = a$.

For example,

$$\gcd(24,18) \overset{(E1)}{=} \gcd(6,18) \overset{(E1)}{=} \gcd(6,12) \overset{(E1)}{=} \gcd(6,6) \overset{(E2)}{=} 6.$$

Note that this procedure of finding the GCD of any two positive integers will terminate with finite steps; that is, this procedure will terminate in (E2) after repeating (E1) for a finite number of times (**you are encouraged to discuss the reason in the report**). Thus, this procedure is an algorithm, called **Euclid's algorithm**. In the following, you are asked to implement Euclid's algorithm in two ways. You must follow the description of Euclid's algorithm to write your code. For example, **division and taking remainder are not allowed in your code**.

2.1 Problem A: Euclid's Algorithm (Recursive)

In this problem, you are asked to implement Euclid's algorithm by **recursive method**

To solve a problem with recursive method, one recognize a problem as one or many simpler sub-problem(s) and solve the sub-problem(s) to solve the original problem. Recursive method often makes the code elegant. One classical example of using recursive method to solve a problem is to find factorial numbers. We give sample codes in C (fac_rec.c) and in MIPS (fac_rec.asm) of finding factorial numbers by recursive method for your reference.

2.2 Problem B: Euclid's Algorithm (Iterative)

In this problem, you are asked to implement Euclid's algorithm by **iterative method**.

As you learned in the class, the elegance of procedure call (recursive method) comes at a price. For example, you may need to copy the content of argument registers, , some temporary registers, and return address to stack for a non-leaf procedure call. The iterative method will save the day. To solve a problem with iterative method, one utilizes loop structures to solve a problem in one procedure. We also give sample codes of finding factorial numbers by iterative method in C (fac_ite.c) and in MIPS (fac_ite.asm) for your reference.

2.3 Problem C: Binary GCD Algorithm (Recursive)

Since division by a power of 2 can be simply realized by bit shifting in a computer. An improvement can be made on Euclid's algorithm.

Note that we can determine the GCD of any two positive integers by integer factorization. For example, let $a = 24 = 2^3 \cdot 3$ and let $b = 90 = 2 \cdot 3^2 \cdot 5$. Then $\gcd(a,b) = 2^{\min(3,1)} \cdot 3^{\min(1,2)} \cdot 5^{\min(0,1)} = 2 \cdot 3 = 6$; that is, the product of common factors in the factorization of a and b. Divide a and b by 2 iteratively until there is no factor of 2 (i.e., being a odd number) and denote it as a' and b'. Then $a = 2^3 \cdot a'$, $b = 2 \cdot b'$ and $\gcd(a', b') = 3^{\min(1,2)} \cdot 5^{\min(0,1)} = 3$. Therefore, $\gcd(a,b) = 2^{\min(3,1)} \cdot \gcd(a', b')$. We thus reduce the problem of finding the GCD of a and b to the problem of finding the GCD of smaller numbers a' and b'. The discussion above is summarized in the following fact:

Fact 2.2. *Let a and b be two positive integers.*

(BG1) *If a and b are both even, then* $\gcd(a,b) = 2 \cdot \gcd(\frac{a}{2}, \frac{b}{2})$.

(BG2) *If one of a and b is odd, say, b is odd, then* $\gcd(a,b) = \gcd(\frac{a}{2}, b)$.

If $a \neq b$ and a and b are both odd, we use (E1). If $a = b$, we use (E2). Thus, we use (BG1) only when $a \neq b$. We thus have a new procedure to compute the GCD of any two positive integers. For example,

$$\gcd(24,18) \overset{(BG1)}{=} \gcd(12,9) \overset{(BG2)}{=} \gcd(6,9) \overset{(BG2)}{=} \gcd(3,9) \overset{(E1)}{=} \gcd(3,6) \overset{(BG2)}{=} \gcd(3,3) \overset{(E2)}{=} 3.$$

Note that in the example above, it seems that this procedure is worsen than just using Euclid's algorithm. **You may want to discuss in the report that in what kinds of situation this procedure is better than Euclid's algorithm**. This procedure also terminates in finite steps and thus is an algorithm. We call this algorithm **binary GCD algorithm**.

In this problem, you are asked to implement binary GCD algorithm by recursive method. Similarly, please follow the discussion of binary GCD algorithm above for implementation. For example, **divide by 2 must be implemented in bit shifting** and **no division and taking remainder are allowed to use**.

Note that there is no division used in both Euclid's algorithm and binary GCD algorithm. However, there are some algorithms that do use division to compute the GCD of any two positive integer. **Discuss in your report what the potential downsides of using division are**.

3 submission

Naming rules for your codes and report are listed here:

- Three C codes (gcdA.c, gcdB.c, gcdC.c) described in Section 2.
- Three MIPS codes (gcdA.asm, gcdB.asm, gcdC.asm) described in Section 2.
- A pdf report (report.pdf).

Zip these files as **ID_proj.zip**. For example, "109064541_proj.zip".

No credit for following conditions:

- **Not following the naming rules.**
- **Late submission.**
- **Plagiarism (strictly forbidden, including reading other's code. Dicussion is encouraged but please finish the project by yourself).**