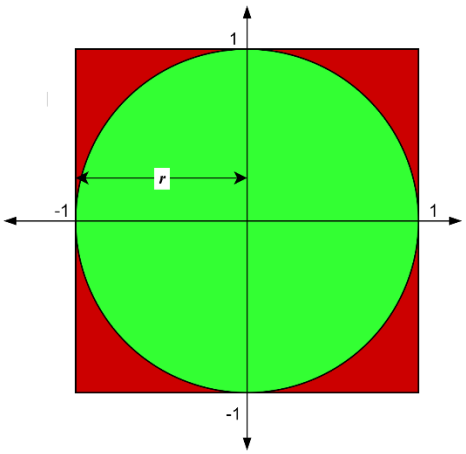
King Fahd University of Petroleum and Minerals
Computer Engineering Department
College of Computing and Mathematics

COE 301: Computer Organization
Assignment 3, Term 251
Pi Estimation using Simulation

Overview:

The number π (pi) is one of the most important constants in mathematics, representing the ratio of a circle's circumference to its diameter. While π can be derived analytically using geometry or calculus, it can also be estimated through simulation methods, particularly using random sampling—an approach known as the **Monte Carlo method.**

The basic idea behind this simulation is to relate π to the area of a circle. Imagine a square of side length 2 that fully contains a circle of radius 1, both centered at the origin (See the figure below).



The area of the square is $(2r)^2 = 4r^2 = 4$, and the area of the circle is $\pi r^2 = \pi(1^2) = \pi$. Thus, the ratio of the circle's area to the square's area is $\frac{area\_of\_square}{area\_of\_circle} = \pi/4$.

To estimate π, we can randomly generate points, with coordinates (x, y) chosen uniformly distributed. Each point is tested to see if it lies inside the square by checking ($-1 \le x \le 1$ and $-1 \le y \le 1$) and if it lies inside the circle by checking whether ($x^2 + y^2 \le 1$). The fraction of points that fall inside the circle approximates the ratio of the areas:

$$\frac{Number\_of\_points\_inside\_circle}{Number\_of\_points\_inside\_square} \approx \frac{\pi}{4} \Longrightarrow \pi \approx 4 \times \frac{Number\_of\_points\_inside\_circle}{Number\_of\_points\_inside\_square}$$

i.e. Multiplying this ratio by 4 gives an estimate of π.

As more random points are generated, the estimate becomes more accurate due to the law of large numbers.

Tasks

You are required to implement the following functions:

1. **`count_points:`** this function takes one input (n) which is the number of points considered in the simulation. It should enter a loop to generate n random numbers by calling **`generate_random`** function that takes no argument and return the x-coordinate in **`$v0`** register and y-coordinate in **`$v1`** register, **where both coordinates are single-precision floating-point numbers**. The function then will check if the generated point is inside the square or not, and it will also check if the point is inside the circle or not. The function should return the total number of points inside the square in **`$v0`** and the total number of points inside the circle in **`$v1`**. The pseudocode of the function is given below:

```
# Function: count_points
# Input: n  → number of random points to generate
# Output: $v0 → total number of points inside the square
#         $v1 → total number of points inside the circle
function count_points(n):
    # Initialize counters
    square_count ← 0
    circle_count ← 0

    # Loop n times
    for i from 1 to n do:
        call generate_random()
        x ← value in $v0   # x-coordinate
        y ← value in $v1   # y-coordinate

        # Check if point is inside the square [-1, 1] × [-1, 1]
        if (x >= -1.0 and x <= 1.0) and (y >= -1.0 and y <= 1.0) then:
            square_count ← square_count + 1
        end if
        # Check if point is inside the unit circle (x² + y² ≤ 1)
        if (x * x + y * y) ≤ 1.0 then:
            circle_count ← circle_count + 1
        end if
    end for
    # Return results
    $v0 ← square_count
    $v1 ← circle_count
    return
```

2. **`estimate_pi:`** this function takes one input (n) which is the number of points considered in the estimation. It should call **`count_points`** function to get the number of points inside the square\circle. It then uses the outcome of the function to estimate the value of pi as described above. The returned value of pi should be a **double-precision floating point number** stored in registers [**$v1:$v0] (i.e. $v0** holds the least significant 32-bits of the value)

Submission Guidelines:

- The due date is **Saturday Nov. 22, 2025 end of the day**.
- The assignment can be solved individually or collaboratively in groups of two students. Both team members should be from the same lecture section and should be declared in Gradescope during submission.
- The submitted file should be renamed as "**assign3.asm**"
- Submit your assignment through Gradescope. Email submissions are not accepted.
- The autograder in Gradescope is designed to give you feedback and help you in debugging. Passing the autograder tests does not guarantee a full mark in the assignment as the tests are not comprehensive and do not test all cases. It is your responsibility to make sure that the submitted code is correct.
- Late submission is not acceptable.
- Using any AI tool for coding or debugging is not allowed.
- The submitted code should include the implementation of the functions listed in the tasks section without testing code. The autograder will call your functions from and pass parameters as described. The structure of the submitted file should be as follows

```
.globl count_points
.globl estimate_pi

count_points:
# function implementation
      Jr $ra

estimate_pi:
# function implementation
      Jr $ra
```