CS-382 Computer Architecture and Organization Fall 2022

Project 1 · Advanced Assembly Programming

Lecturer: Shudong Hao *Date: See Canvas*

Before starting this project, it's a good idea to review Appendix B.4 to get familiar with floating point number registers and instructions — they are very similar to what we have used but also different.

1 Description

Create an ARMv8 program that implements the **bisection method** for finding a root of a function $f(x)$, which will be restricted to a polynomial in this project.

The bisection method is a root-finding method that applies to any continuous function for which one knows two values with opposite signs. The method consists of repeatedly bisecting the interval defined by these values and then selecting the subinterval in which the function changes sign, and therefore must contain a root. See .

The bisection method is applicable for numerically solving the equation $f(x) = 0$ for the real variable x, where f is a continuous function defined on an interval $[a, b]$ and where $f(a)$ and $f(b)$ have opposite signs. In this case a and b are said to bracket a root since, by the intermediate value theorem, the continuous function f must have at least one root in the interval (a, b).

At each step the method divides the interval in two parts/halves by computing the midpoint $c = (a + b)/2$ of the interval and the value of the function $f(c)$ at that point. Unless c is itself a root (which is very unlikely, but possible) there are now only two possibilities: either $f(a)$ and $f(c)$ have opposite signs and bracket a root, or $f(c)$ and $f(b)$ have opposite signs and bracket a root. The method selects the subinterval that is guaranteed to be a bracket as the new interval to be used in the next step. In this way an interval that contains a zero of f is reduced in width by 50% at each step. The process is continued until $f(c)$ is approximately equal to 0, within a tolerance τ specified by the user. That is, the process terminates when $|f(c)| < \tau$.

In this project, we will limit $f(x)$ to be a polynomial, i.e. $f(x)$ will be of the form:

$$f(x) = \sum_{i=0}^{N} p_i x^i$$

where p_i are the coefficients and N is the degree of the polynomial.

You can use the following function to develop your code, but it will be tested with different inputs:

$$f(x) = 0.2 + 3.1x - 0.3x^2 + 1.9x^3 + 0.2x^4$$

with $a = -1$, $b = 1$ and $\tau = 0.01$.

Hints

- There is no need for an absolute value function to test if $|f(c)| < \tau$.
- There is also no need for a power function. All exponents are integers.
- Insert 0 coefficients for missing terms of the polynomial, such as p_1 in $f(x) = 5.3 + 2.9x^2 - 3.1x^3$. The data segment should be:

```
coeff:
    .double 5.3, 0.0, 2.9, -3.1
N:
    .dword 3
```

2 Requirements

(1) Your code **must** include procedure calls.

(2) The degree and coefficients of the polynomial should be declared in the data segment as a double word and several double precision floating point numbers, respectively.

(3) A user-specified tolerance will also be specified in double precision in the data segment.

(4) Two values, a and b for which $f(a)$ and $f(b)$ have opposite signs will also be specified in double precision in the data segment.

(5) Your code must print the root that was found as well as the corresponding value of the function.

(6) **If your code does not assemble, you will earn 0 credit. If your program only meets some of the requirements, make sure that there are no errors to earn partial credit.**

3 Deliverable

Submit on Canvas a **zip** file containing:

(1) a **pdf** file describing what you did including fragments of the code as needed,

(2) a file with the source code named `bisection.s` (not executables, object files *etc.*).