ECE 1110
Introduction to Computer Organization & Architecture

Due date: Tuesday September 20th, 2021

In Digital Signal Processing (DSP), a fundamental operation is discrete-time linear convolution. Given an input $x[n]$ of length M, an impulse response $h[n]$ of length L, the output $y[n]$ is computed via equation (1):

$$y[n] = \sum_{m=\max(0,n-L+1)}^{\min(n,M-1)} h[m] * x[n-m] \tag{1}$$

This can be written as a procedure `convolve` in C as follows:

```
void convolve(int M, int L, float *h, float *x, float *y)
{
    int n,m;
    for (n=0; n<L+M, n++)
        for (y[n]=0, m=max(0,n-L+1); m<=min(n,M-1); m++)
            y[n] += h[m] * x[n-m];
}
```

In this implementation, the array arguments `h`, `x` and `y` are pointers to the beginning of the arrays.

1. Complete this procedure using RISC-V assembler. (You will find the `rars` emulator very helpful in this endeavor.) I have provided the code except it is missing `m<=min(n,M-1)` and the body of the loop! Read this code. Understand *every* line. Note that I've used pseudo-instructions, assembler directives and environmental calls. These are described in the online `rars` documentation.

2. How will you test this? (Hint: what do you remember from the continuous time convolution?)

3. What is your total cost in the inner loop?

4. DSP microprocessors include a *multiply-accumulate* instruction. What would such an instruction look like in RISC-V? How will this change the total cost in the inner loop? (be precise)