Atividade de Laboratório 6

Objetivo

O objetivo desta atividade é exercitar o uso de instruções aritméticas inteiras e reais e a manipulação de entrada e saída utilizando o conjunto de instruções da arquitetura RISC-V.

Ponto Flutuante

Para expressar computacionalmente números com parte fracionária a maneira mais comum é utilizar a representação chamada ponto flutuante (PF). O padrão IEEE754 é um dos mais utilizados, definindo uma representação binária para valores com ponto-flutuante de utilizando 3 campos: sinal, expoente e mantissa. A quantidade de bits designado a cada campo depende da precisão (simples ou dupla).

Dada uma palavra de 32-bits, a representação de ponto flutuante de precisão simples é mostrada na figura abaixo, com os seguintes campos:

- 1 bit para o sinal (S). 0 para números positivos e 1 para negativos (bit 31)
- 8 bits o expoente (E) (bits 30 a 23)
- 23 bits para mantissa (M) (bits 23 a 0)

O valor decimal do número em ponto flutuante pode ser calculado de acordo com a equação 1, onde b_n é o n-ésimo bit da palavra:

$$N = (-1)^{S} * 2^{E-127} * (1 + \sum_{i=1}^{23} b_{23-i} * 2^{-i}) \qquad (1)$$

Por exemplo, o valor $0x40490FDB_{16}$ possui a seguinte representação binária: $01000000010010010000111111011011_2$, que possui os seguintes valores nos campos da representação de ponto-flutuante: $S=0_{10}$, $E=128_{10}$, e $M=4788187_{10}$. Se utilizarmos esses valores na equação 1, temos (termos onde b_n é 0 não são considerados no somatório):

$$1^{0} * 2^{128-127} * (1 + 2^{-1} + 2^{-4} + 2^{-7} + 2^{-12} + 2^{-13} + 2^{-14} ...) = 3.141592741_{10}$$

No caso do RISC-V há instruções específicas a nível de *hardware* para realizar operações lógicas e aritméticas utilizando ponto flutuante, utilizando a extensão de ponto-flutuante (Extensão F). Para tanto, o RISC-V possui 32 registradores de 32 *bits* para trabalhar com pontos flutuantes (`f0-f31`) separados do banco de registradores inteiros comuns (`x0-x31`). As operações lógicas e aritméticas devem ser realizadas utilizando estes registradores e são muito semelhantes (sintaticamente) as operações inteiras. Algumas delas são mostradas abaixo:

Instrução	Operação realizada
`fadd.s f_rd, f_rs1, f_rs2`	Adiciona o valor do registrador de PF `f_rs1` a `f_rs2` e coloca o resultado no registrador PF `f_rd` (i.e., `f_rd = f_rs1 + f_rs2`)
`fsub.s f_rd, f_rs1, f_rs2`	Subtrai o valor do registrador de PF `f_rs2` de `f_rs1` e coloca o resultado no registrador PF `f_rd` (i.e., `f_rd = f_rs1 - f_rs2`)
`fmul.s f_rd, f_rs1, f_rs2`	Multiplica o valor do registrador de PF `f_rs1` por `f_rs2` e coloca o resultado no registrador PF `f_rd` (i.e., `f_rd = f_rs1 * f_rs2`)
`fdiv.s f_rd, f_rs1, f_rs2`	Divide o valor do registrador de PF `f_rs1` por `f_rs2` e coloca o resultado no registrador PF `f_rd` (i.e., `f_rd = f_rs1 / f_rs2`)
`fcvt.s.w f_rd, rs`	Converte um valor inteiro com sinal (no registrador `rs`) para sua representação em ponto-flutuante e coloca o resultado no registrador de PF `f_rd`
`fcvt.w.s rd, f_rs`	Converte um valor de PF (do registrador `f_rs`) para um inteiro e coloca o resultado em rd. Nota: Apenas a parte inteira é convertida e o valor é arredondado ao inteiro mais próximo
`flw f_rd, offset(rs)`	Carrega o valor de PF da memória, no endereço indicado por `offset(rs)` e coloca no registrador de PF `f_rd`
`fsw f_rs, offset(rd)`	Salva o valor de PF do registrador `f_rs` para a memória, no endereço indicado por `offset(rd)`

O código de exemplo abaixo mostra o como pode ser realizado o cálculo da conversão de um ângulo x, em graus, para radianos, utilizando a equação 2.

$$x = \frac{x*pi}{180} \qquad (2)$$

```
.text
# Carrega o inteiro 45 para o registrador a0
li a0, 45
# Converte o inteiro 45 para representação em ponto flutuante e
coloca no registrador fa0 (45.0)
fcvt.s.w fa0, a0
# Carrega o endereço contendo o valor de pi
la a0, .float_pi
# Carrega a palavra contida no endereço de a0 para o registrador fa1
flw fa1, 0(a0)
# Multiplica 45 por 3.1415 e armazena o resultado em fa0
fmul.s fa0, fa0, fa1
# Carrega o valor inteiro 180 para o registrador a0
li a0, 180
# Converte o inteiro 180 para PF e coloca no registrador fa1 (180.0)
fcvt.s.w fa1, a0
# Divide 45*pi por 180 e armazena o resultado em fa0
fdiv.s fa0, fa0, fa2

.align 4
.data
  .float_pi:
     # Valor de pi (3.1415...) na representação de ponto-flutuante
     .word 0x40490fdb
```

Atividade

Neste laboratório, você deve fazer um programa que calcula uma aproximação para a função seno(x) utilizando a Série de Taylor, e imprimir o resultado na saída, utilizando como base o arquivo disponível em modelo.s.

Uma série de Taylor é uma representação aproximada de uma função como uma soma infinita de termos que são calculados a partir dos valores das derivadas da função em um único ponto. Uma aproximação para a função seno(x) pode ser obtida com a equação 3, dado um real x representando um ângulo, em radianos.

$$sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} * x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots \quad \forall x \qquad (3)$$

Para isso você deverá modificar o arquivo modelo.s deve ser alterado em dois pontos:

- No trecho *** INSIRA AQUI SEU CÓDIGO PARA CÁLCULO DA SERIE *** (linha 27) você deve implementar a série de taylor com aproximação de 10 elementos, isto é, n=10 para a equação 3. No início do trecho onde o código deve ser inserido, o valor de x (em radianos) já estará no registrador de PF `f0`. Sua série deve deixar o valor final do somatório no registrador `f0`.

- No trecho ** *INSIRA AQUI SEU CÓDIGO PARA IMPRESSÃO* ** (linha 47) você deve inserir um código para imprimir o valor resultante da série na tela. No ponto onde o seu código será inserido, você já receberá três valores inteiros, vindos nos registradores `a0, a1 e a2`, sendo que:
  - `a0` conterá o valor zero, indicando que o número resultante da série é positivo e diferente de zero caso seja negativo.
  - `a1` conterá um número inteiro representando parte inteira do número resultante da série;
  - `a2` conterá um número inteiro representando parte fracionária do número resultante da série, truncada na 3ª casa decimal (inteiro de 0 a 999).

  Por exemplo, se o número resultante da soma (em `f0`) for 3.141592, os registradores terão os seguintes valores inteiros nesta parte do código `a0=0, a1=3, a2=141`. Se o número resultante da soma (em `f0`) for 0.017, os registradores terão os seguintes valores inteiros nesta parte do código `a0=0, a1=0, a2=17.`

O resultado final deve ser impresso na tela com a seguinte formatação, com letras maiúsculas e um espaço após “SENO:”:

SENO: `[+|-][1 CARACTERE PARTE INT].[3 CARACTERES PARTE FRAC]\n`

Por exemplo, dado que o ângulo de entrada para o seu programa é 45, após realizar a soma a saída deve ser:

SENO: `+0.707`

Ou, dado que o ângulo é 359, a saída deve ser:

SENO: `-0.017`

Infraestrutura

ATENÇÃO: Para depurar seu código no simulador, utilize o navegador do Google Chrome, a depuração pode não funcionar em outros navegadores.

Para montar seu código em linguagem de montagem, use o comando:

```
clang -g -O0 --target=riscv32 -march=rv32gc -mabi=ilp32d
-mno-relax arquivo_de_entrada.s -c -o arquivo_de_saida.o
```

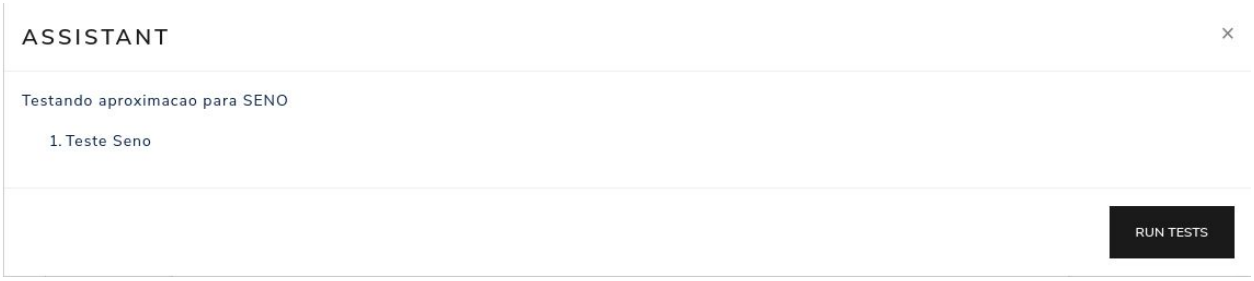
Note o parâmetro na ferramenta clang: *--target=riscv32 -march=rv32gc -mabi=ilp32d.* Esse parâmetro indica que queremos que seja gerado um objeto que não é da arquitetura do seu computador (provavelmente x86_64) mas sim para RISC-V de 32 bits. Após essa etapa, tendo o arquivo objeto em mãos, podemos executar o ligador (*linker*) através do seguinte comando:

```
ld.lld -g arquivo_de_saida.o -o executavel.x
```

Testando seu lab

Do lado direito do simulador há um pequeno olho preto que abre o sistema de testes do laboratório:

Ao clicar nele irá ser aberto uma janela com a opção de executar os testes no código que você fez upload:



Entrega

Você deve submeter APENAS um arquivo denominado raXXXXXX.s (em que XXXXXX é seu RA com 6 dígitos) que contenha o código do modelo base com suas modificações, em linguagem de montagem.

Dicas e Observações

- Os trechos de código antecessores e sucessores ao que deve ser inserido **não deve ser modificado**. O arquivo de modelo deve ser submetido juntamente com suas alterações.
- Cada linha deve ser finalizada com o caractere '`\n`' e deve haver um espaço após a palavra SENO.
- Linhas que não tenham um '`\n`' no final não serão mostradas no simulador.

- O valor de entrada para o ângulo é um inteiro variando de 0 a 360. Durante os testes o valor do ângulo será inserido no endereço denominado pelo label `angle`.
- Não modifique os valores das variáveis na seção .data. Entretanto, você pode inserir mais variáveis nesta seção caso necessário.
- O resultado deverá ter um erro aproximado máximo de 1e3 (0.001). Por exemplo, para o valor 1.0: $0.999 \approx 1.0 \approx 1.001$
- Você pode verificar o estado dos registradores de ponto flutuante no simulador, quando executando seu programa em modo de depuração, com o seguinte comando: `peek f 0`, onde 0 é o número do registrador de PF (0 a 31). O valor exibido em hexadecimal é o valor na representação de PF e sua representação decimal pode ser obtida através da equação 1.
- Preste atenção no tipo dos registradores que você utiliza para fazer os cálculos em seu programa, a fórmula (3) apresenta termos de rápido crescimento, como (2n+1)!, que podem não ser representável em registradores inteiros de 32 bits.
- Para realizar a impressão neste laboratório você deverá utilizar a syscall 64, já utilizada no laboratório 5.
- Para imprimir um valor que está em um registrador, será primeiro necessário converter cada dígito do valor para sua representação Ascii, e depois guardar os valores em ascii na memória.
- Para executar seu código até um determinado rótulo, podemos realizar o seguinte o processo na seção de Debug do simulador:
    - Verifique o endereço do rótulo desejado na lista de símbolos para isso use o seguinte comando:
        - symbols
    - Agora que temos o endereço do rótulo desejado, podemos executar o programa até o ponto do rótulo com o seguinte comando:
        - Until <endereço>