ANDES
The Quiet Giant in RISC-V

RISC-V Virtual Hackathon

Softmax Challenge

- Time to Hack Away

Agenda

- Environment and Getting Started

Environment Intro

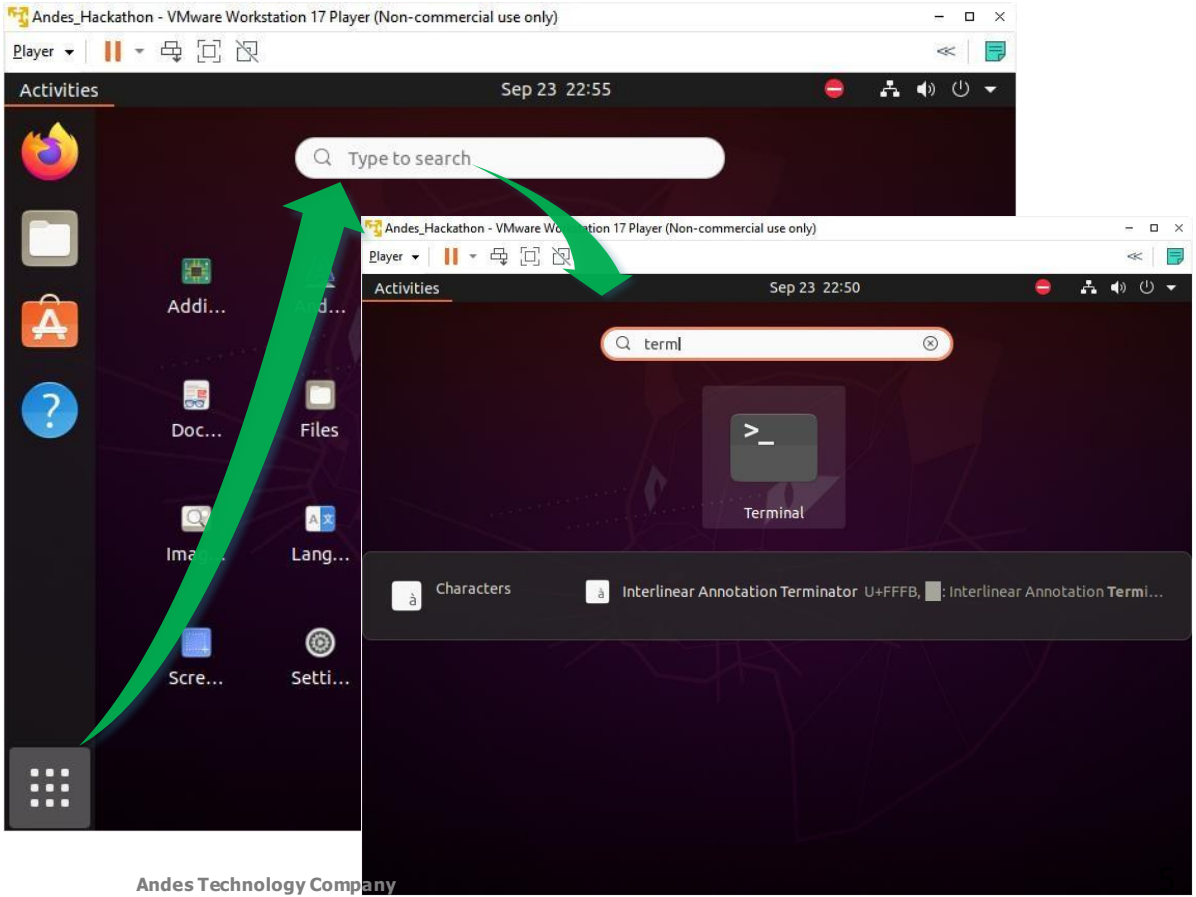
- OS: Ubuntu 20.04.6 LTS. A Linux OS.
- AndeSight: v5.3.3. AndeSight is the IDE tool for C code development. GNU Toolchains are included.
- COPILOT: v7.2.0. COPILOT is a tool to create custom instructions (ACE)
  - Integrated in the AndeSight
- AndesClarity: CPU Pipeline visualizer tool
  - Integrated in the AndeSight
- AX45MPV: An Andes 8-stage pipeline dual-issue core with a vector processor
- ACE: Andes Custom Extension
- Sim: CPU simulator

Tool Location

- AndeSight: /home/ubuntu/Andestech/AndeSight_STD_v533/ide/AndeSight.
- COPILOT: /home/ubuntu/Andestech/AndeSight_STD_v533/COPILOT/bin/copilot

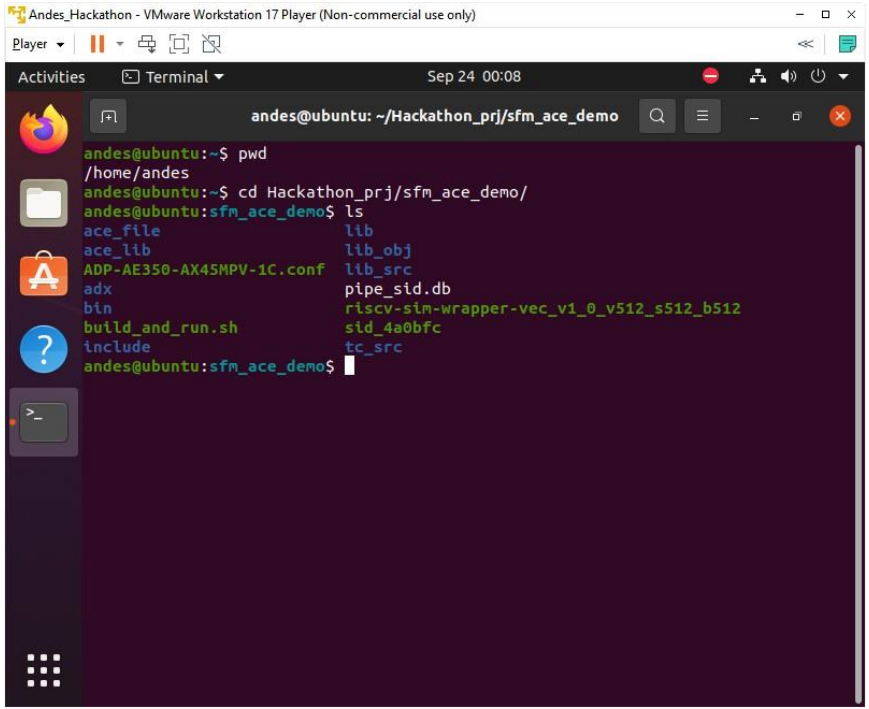
Open a Terminal

- Click “  ”
- Type “terminal” in the search box
- Click “Terminal” in the search and hit enter key



Hackathon Project

- The Hackathon project root is at **/home/ubuntu/Hackathon_prj/sfm_ace_demo** folder



Background and Challenge

- **Softmax in Neural Networks:**
  - Softmax is a very common activation function for many networks including Transformers, CNNs, and RNNs.
  - It is challenging and not well implemented by general-purpose CPUs and GPUs because it is a very specific computation for machine learning.
  - It is an imperative function in neural network performance especially for LLMs which are seeing a huge increase in model size and sequence length and softmax function usage scales with these parameters.
- **Your challenge:**

  Implement a softmax function on an Andes RISC-V vector processor while optimizing for accuracy and performance.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

σ = softmax

$\vec{z}$ = input vector

e^{z_i} = standard exponential function for input vector

K = number of classes in the multi-class classifier

e^{z_j} = standard exponential function for output vector

e^{z_j} = standard exponential function for output vector

Code Structure

- Main code: **tc_src/t_softmax_f32.c**
- Library source: **lib_src/riscv_nn_softmax_f32.c**
- ACE source file: **ace_file/exp.ace**

Code Flow

- The main() function in the tc_src/t_softmax_f32.c invokes riscv_nn_softmax_f32() function in the lib_src/riscv_nn_softmax_f32.c
- The riscv_nn_softmax_f32() invokes ace_exp_f32m8() intrinsic function.
- The ace_exp_f32m8() function is a custom instruction/function, i.e. exp, created by the COPILOT tool. The input of the COPILOT is ace_file/exp.ace which describes the custom instruction names, input/output operands, and the behavior model.
- The exp instruction is implemented with the following equation, in which the accuracy is low.
  - $f(x) = e^x = 2^{23} * \left(\frac{x}{\ln(2)} + 127 - C\right)$
- For the custom instruction generation, please run the build_ace.sh in the ace_file folder. For more details, please refer to Andes_Custom_Extension_Programmer's_Manual.pdf

What can you do?

- Increase the accuracy of the exponential function or use your own algorithm
- Use LLVM auto-vectorization to convert scalar operations into vector operations
- Use Vector Instrinsics to create your own vectorized implementation
- Create your own custom extension for implementing the operation of the exponential function or the softmax function

- **HAVE FUN!!**

Solution Criteria and Judging

- Judging will be based on the following:
  - Accuracy of your Softmax as measured by SNR (higher is better)
  - Performance of your Softmax, as measured by cycle counts (lower is better)
  - Creativity of solution
  - For those who implement custom instructions, consideration of HW implementability.

Hackathon Feedback

- Let us know what you thought about the hackathon here:
  - https://forms.gle/74aV3VDri16RQ6ws7

ANDES
The Quiet Giant in RISC-V

Thank You!