Using GDB to debug The "hello_world" example

Madhav Desai
Department of Electrical Engg.
IIT Powai
Mumbai 400076

August 8, 2021

Recall the hello world example

- Print “hello_world” and stop.
- Single core version (with MMU).
- Multi core version (with MMU).

Blank slide: GDB mechanism to debug a single AJIT thread

Single thread debug

- In shell 1, start the GDB client:

  ```
  %  sparc-linux-gdb hello_world.elf
  ```

- In shell 2, start the C model:

  ```
  %  ajit_C_system_model -m hello_world.mmap.remapped -g
  ```

- Go back to shell 1, and specify the target server.

  ```
  (gdb) target remote:8888
  Remote debugging using :8888
  0x00000000 in _start ()
  (gdb)
  ```

  You will see that the gdb client will stop at the

  `_start`

  routine.

- Now in shell 2, you can control the execution of the program using GDB.

Blank slide: GDB mechanism to simultaneously debug a multiple AJIT thread

Multi thread debug for 4 threads (0,0), (0,1), (1,0), (1,1)

- In shells 1,2,3,4 start GDB clients:

  ```
  prompt>  sparc-linux-gdb hello_world.elf
  ```

- In shell 5, start the C model:

  ```
  prompt>  ajit_C_system_model -m hello_world.mmap.remapp
  ```

  This assigns port 8888 to GDB session for (0,0), port 8889 for (0,1), port 8890 for (1,0) and port 8891 for (1,1).

- Go back to shells 1,2,3,4 and specify the target server.

  ```
  (gdb) target remote:8888
  ```

  etc. Remember to put the port numbers for the GDB clients (in shell 1, use port 8888, in shell 2, use 8889 etc.). You will see that the gdb client will stop at the

  ```
  _start
  ```

  routine in each of the shells 1-4.

- Now in shells 1-4, you can control the execution of the program using GDB on thread (0,0), (0,1), (1,0) and (1,1).