Intel® Distribution for GDB* Reference Sheet

Prerequisites

Set your oneAPI environment variables:

```
$ source <ONEAPI_ROOT>/setvars.sh
```

Add the following kernel parameter (typically done by editing `/etc/default/grub` and then running `update-grub`):

```
i915.debug_eu=1
```

Define the following environment variable:

```
$ export ZET_ENABLE_PROGRAM_DEBUGGING=1
```

Finally, check that your user is a member of the linux group that owns the graphics card.

Auto-Attach

Turn the auto-attach feature off, if desired (e.g. if debugging on CPU or FPGA-emu):

```
$ export INTELGT_AUTO_ATTACH_DISABLE=1
```

Turn the feature on:

```
$ unset INTELGT_AUTO_ATTACH_DISABLE
```

Useful GDB Commands

`help <cmd>`
Print help info about the command `cmd`.

`run [arg1, ... argN]`
Start the program, optionally with arguments.

`break <filename>:<line>`
Define a breakpoint at given source file's specified line.

`info break`
Show the defined breakpoints.

`delete <N>`
Remove the `N`th breakpoint.

`watch <exp>`
Stop when value of the expression `exp` changes.

`step`, `next`
Single-step a source line, stepping into/over func calls.

`continue`
Continue execution.

`print <exp>`
Print value of expression `exp`.

`backtrace`
Show the function call stack.

`up`, `down`
Go one level up/down in the function call stack.

`disassemble`
Disassemble the current function.

`info args/locals`
Show the arguments/local vars of the current function.

`info reg <regname>`
Show contents of the specified register.

`info inferiors`
Display information about the *inferiors*. For GPU offloading, the host process and the GPU devices are each represented by an inferior.

`info devices`
Display information about the *devices*.

Useful GDB Commands (cont'd)

`info threads [-stopped] <ID>`
Display information about threads with id `ID`, including their active SIMD lanes. Omit id to display all threads. Use the `-stopped` flag to limit to stopped threads.

`thread <thread_id>:<lane>`
Switch context to the SIMD lane `lane` of the specified thread. E.g: `thread 2.6:4`

`thread apply <thread_id>:<lane> <cmd>`
Apply command `cmd` to the specified lane of the thread. E.g: '`thread apply 2.3:* print element`' prints `element` for each active lane of thread 2.3. Useful for inspecting vectorized values.

`x /<format> <addr>`
Examine the memory at address `addr` according to `format`. E.g: '`x /i $pc`' shows the instruction pointed by the program counter. '`x /8wd &count`' shows 8 words in decimal format located at the address of `count`.

`set nonstop on/off`
Enable/disable the nonstop mode. This command may *not* be used after the program has started.

`set scheduler-locking on/step/off`
Lock the thread scheduler. Useful to keep the other threads stopped while the current thread is stepping (if set to `step`) or resumed (if `on`) to avoid interference.

`maint jit dump <addr> <filename>`
Save the JIT'ed objfile that contains address `addr` into the file `filename`. Useful for extracting the SYCL kernel when running on the CPU device.

`cond [-force] <N> <exp>`
Define the expression `exp` as the condition for breakpoint `N`. Use the optional `-force` flag to force the condition to be defined even when `exp` is invalid for the current locations of the breakpoint. Useful for defining conditions on breakpoints in JIT-produced code.

Notes

Currently only the Level Zero backend supports debug. Workloads submitted to different devices and/or subdevices can be debugged simultaneously. Only one workload at a time can be debugged on a subdevice. Other workloads submitted to the same subdevice need to wait until the subdevice is free again.

The `ZE_AFFINITY_MASK=`*<device>*`.`*<subdevice>* environment variable can be used to limit the devices/subdevices available to the program.

Links

Get Started Guide
Release Notes
oneAPI Programming Guide