Memory access checker for Soclib

The Memory checker tool is a software memory access debugger for SoClib.

Overview

The Memory checker is able to perform several analysis on memory accesses performed by the running software to track software bugs. Access to uninitialized or freed data and stack overflow are examples of reported behaviors. It acts as the ?valgrind memory checker tool found on some UNIX.

Implementation

Like the Gdb Server, the Memory checker contains no processor-specific code and can be used to manage any Soclib processor model using the generic [Component/Iss2Api Iss2 interface]. It is implemented as an Iss wrapper class.

When the Memory checker is in use, it traces all events between the processor Iss model and the SoCLib platform. The running operating system must be instrumented slightly to let the Memory checker be aware of valid stack ranges and allocation ranges. The ?MutekH operating system is working with the Memory checker.

What is being checked

All memory accesses are monitored and checked for read to non previously initialized (written) words.

Context and stacks related checks:

- The stack pointer register must stay in range given by the operating system for each software context in use.
- The frame pointer register (if any) must stay in range given by the operating system for each software context in use.
- Contexts stack ranges can not overlap (checked on context creation).
- Stack range must be in allocated memory at context creation (as soon as allocation checks are enabled).
- The stack memory is marked as non-initialized when a new execution context is created.
- The stack memory is always considered as non-initialized below the stack pointer.
- Memory r/w accesses in stack can not occur below the stack pointer.

Memory allocation and region checks:

- Write accesses can not occur in read-only preloaded sections.
- Preloaded sections are marked as uninitialized when appropriate.
- Memory is marked as uninitialized on `malloc()` invocation.
- Memory is marked as uninitialized on `free()` invocation.
- Memory r/w accesses can not occur in freed memory.
- Allocation are only allowed in free memory.

Suspicious memory access reporting

Suspicious memory accesses produce a message on simulator `stdout` stream. This simulation is not stopped though.

An exception can be reported to an optional `GdbServer` module to stop processor execution when a suspicious memory access happened. This enables further analysis of buggy software. When using the Memory checker with the `GdbServer`, the Memory checker must wrap the processor directly and must be wrapped in the GdbServer.

Usage

Adding Memory checker support to your platform

Adding the GdbServer to your topcell is easy. First include the header:

```
#include "iss_memchecker.h"
```

Then call the init function with mapping table and loader parameters and replace processor instantiation:

```
   // Without Memory checker
// soclib::caba::VciXcacheWrapper<soclib::common::Mips32ElIss> cpu0("cpu0", 0, maptab, IntTab(0)

   // With Memory checker
   soclib::common::Memchecker<soclib::common::Mips32ElIss>::init(maptab, loader);
   soclib::caba::VciXcacheWrapper<soclib::common::IssMemchecker<soclib::common::Mips32ElIss> > c
```

Finally do not forget to update the platform description file:

```
Uses('vci_xcache_wrapper', iss_t = 'common:iss_memchecker', iss_memchecker_t = 'common:mips32el'
```

Using an instrumented operating system

The running operating system must communicate with the Memory checker to report information about context creation (stack range), and memory-allocator operations. This is done through read/write operations in specific memory locations which are intercepted by the Memory checker and not forwarded to the rest of the platform.

Currently the only known supported operating system is ?MutekH with Mips32 processor. Other processors are partially supported, only memory allocation checks are performed. To use the memory checker with MutekH, simply add the `CONFIG_SOCLIB_MEMCHECK` configuration token to your configuration file.

Note:

- An instrumented operating system can not be used without the ISS Memory checker module as memory accesses won't be intercepted and may cause bus error or side effects.
- The default base address for the register bank of the memory checker is 0x00004200. This address can be changed but must stay close to 0 to fit on some processor instruction immediate field. You should consider this if you already have components at these addresses.
- The Memory checker registers bank is protected by a magic value and is unlikely to be modified by an other running software.

Output example

With the following code using an unitialized stack variable to set a global:

```
int foo;

void _main(void*unused)
{
  int bar;
  foo = bar;
  // ...
}
```

We get the following output:

```
cache0 error:
 access to uninitialized word
 at PC=[@0x60100564: (_main + 0x8)]
    SP=[@0x62207fb8: (context_stack + 0x7f94)]
    last Dreq: <DataReq mode MODE_KERNEL   valid type DATA_READ @ 0x62207fc8 wdata 0 be 0xf> [@0
```