Google Native Client

- Suppose we want to run untrusted code as a module inside of a trusted process
  - Necessary for good performance when untrusted\trusted code call each other a lot
  - Must enable safe sandboxing
  - Must isolated untrusted code even if it is buggy
- Dangers to avoid
  - Resource exhaustion
  - Memory writes
  - Memory reads
  - System calls
  - Forbidden instructions
  - Unconstrained control transfers (jumps)
- Things to allow
  - Reads\Writes to special memory area
  - Make certain calls into trusted code
    - Call gates
- How to enforce
  - A sample check:

  Given  ld $r4, [$r5]

  if $r5 < $r1 or $r2 <= $r5
  abort();

We insert these checks before all load\store instructions, note we must not be able to jump over these checks.

- Loader checks
  - All load/store instructions preceeded by checks
  - No instructions modify $r0, $r1, $r2
  - Jump targets are only to beginning of basic blocks
- To handle jumps native client divides code into 32-byte chunks
  - Ensures every jump is to the beginning of a 32 byte chunk

  if($r4 & 0x1f)
  abort();
  jmp $r4

  or we can do

  and $r4, $r4, 0xfffffffff
  jmp $r4

  but how to constrain jumps to be internal? use this check

  and $r4, $r4, 0x000ffffe0
  or $r4, $r4, $r0
Call Gates

- We insert trusted instructions into the untrusted area which are unrestricted jumps to allowed functions.
- The code will be allowed to jump onto these "trampolines".
- Untrusted code has its own stack/heap/data.
- Trusted code has its own stack/heap/data:
  - It must save stack of untrusted module (i.e. stack pointer).
  - Copy args to trusted stack.
  - Copy result into untrusted stack.
  - Restore state.
  - Return control to untrusted module.
    - Check that return address meets specs.