Implementation:

L1 bypass was already implemented in gpgpusim, with the condition of empty L1 cache, configured to skip L1, etc. So the implementation of this project is adding the condition of L1 bypass - if it's a load instruction and the address matches the range, the cache operation is set to global cache, and thus bypass L1. The additional condition and counter implementations are shown as below.

* shader.cc:

```
diff --git a/src/gpgpu-sim/shader.cc b/src/gpgpu-sim/shader.cc
index c6e7b8f..94cd180 100644
--- a/src/gpgpu-sim/shader.cc
+++ b/src/gpgpu-sim/shader.cc
@@ -52,6 +52,8 @@
 #define MAX(a, b) (((a) > (b)) ? (a) : (b))
 #define MIN(a, b) (((a) < (b)) ? (a) : (b))

+int g_L1D_bypass_count = 0;
+
 mem_fetch *shader_core_mem_fetch_allocator::alloc(
     new_addr_type addr, mem_access_type type, unsigned size, bool wr,
     unsigned long long cycle) const {
@@ -2039,6 +2041,13 @@ bool ldst_unit::memory_cycle(warp_inst_t &inst,
   const mem_access_t &access = inst.accessq_back();

   bool bypassL1D = false;
+
+  if(access.get_addr()>0xc0000000 && access.get_addr()<0xc00fffff && inst.is_load()){
+    //printf("P3-L1D bypassed, Addr: %x, \n",access.get_addr());
+    inst.cache_op = CACHE_GLOBAL;
+    g_L1D_bypass_count++;
+  }
+
   if (CACHE_GLOBAL == inst.cache_op || (m_L1D == NULL)) {
     bypassL1D = true;
   } else if (inst.space.is_global()) {  // global memory access
@@ -2046,6 +2055,7 @@ bool ldst_unit::memory_cycle(warp_inst_t &inst,
     if (m_core->get_config()->gmem_skip_L1D && (CACHE_L1 != inst.cache_op))
       bypassL1D = true;
   }
+
   if (bypassL1D) {
     // bypass L1 cache
     unsigned control_size =
```

Other modification to the code to dump out the counter are shown as below:

* shader.h

```
diff --git a/src/gpgpu-sim/shader.h b/src/gpgpu-sim/shader.h
index 6481790..a3a4989 100644
--- a/src/gpgpu-sim/shader.h
+++ b/src/gpgpu-sim/shader.h
@@ -70,6 +70,8 @@

 #define WRITE_MASK_SIZE 8

+extern int g_L1D_bypass_count;
+
 class gpgpu_context;

 enum exec_unit_type_t {
```

gpgpusim_entrypoint.cc

```
diff --git a/src/gpgpusim_entrypoint.cc b/src/gpgpusim_entrypoint.cc
index f4287d8..0c7d6a7 100644
--- a/src/gpgpusim_entrypoint.cc
+++ b/src/gpgpusim_entrypoint.cc
@@ -37,6 +37,7 @@
 #include "gpgpu-sim/icnt_wrapper.h"
 #include "option_parser.h"
 #include "stream_manager.h"
+#include "gpgpu-sim/shader.h"

 #define MAX(a, b) (((a) > (b)) ? (a) : (b))

@@ -266,6 +267,9 @@ void gpgpu_context::print_simulation_time() {
   printf("gpgpu_simulation_rate = %u (cycle/sec)\n", cycles_per_sec);
   printf("gpgpu_silicon_slowdown = %ux\n",
          the_gpgpusim->g_the_gpu->shader_clock() * 1000 / cycles_per_sec);
+
+  printf("bypassed load instructions: %d\n\n", g_L1D_bypass_count);
+  g_L1D_bypass_count = 0;
   fflush(stdout);
 }
```

Verification:

The original code will have 0 load bypass in all kernel, and the load-bypass modified version yields a very different number, shown as below.

note: gpu configuration SM7_TITANV is used for both simulation.

```
root@5264f688abda:~/ece786_project3/BFS# grep "bypassed load instructions:" original.log
bypassed load instructions: 0
bypassed load instructions: 0
bypassed load instructions: 0
bypassed load instructions: 0
bypassed load instructions: 0
bypassed load instructions: 0
bypassed load instructions: 0
bypassed load instructions: 0
bypassed load instructions: 0
bypassed load instructions: 0
```

```
root@5264f688abda:~/ece786_project3/BFS# grep "bypassed load instructions:" modified.log
bypassed load instructions: 8
bypassed load instructions: 137
bypassed load instructions: 800
bypassed load instructions: 4921
bypassed load instructions: 28224
bypassed load instructions: 147873
bypassed load instructions: 424113
bypassed load instructions: 62755
bypassed load instructions: 2435
bypassed load instructions: 15
```