Program Assignment 3A: Load Bypass

Brian Park

North Carolina State University, Computer Engineering 786

March 2023

1 Load Bypass Implementation

I implemented the load bypass by modifying some of the code already existing in `ldst_unit`, specifically, `ldst_unit::cycle()` and `ldst_unit::memory_cycle()`. You can review the changes at the bottom of this report. It seems that `ldst_unit::cycle()` is where memory accesses are generated and `ldst_unit::memory_cycle()` is where memory unit actually executes the memory instructions, as indicated by the hints. In both functions, I modified the conditions to set the boolean flag `bypassL1D` to true if the address of the memory request lies between `0xc0000000-0xc00fffff`.

The results can be reproduced as follows (with `SM6_TITANX` configuration), where each line represents the bypasses cache instructions per kernel:

```
cd ~/ISPASS/BFS
./ispass-2009-BFS graph65536.txt > gpuout.txt
cat gpuout.txt | grep "bypassed load instructions: "

bypassed load instructions: 52
bypassed load instructions: 326
bypassed load instructions: 1926
bypassed load instructions: 11769
bypassed load instructions: 68266
bypassed load instructions: 337901
bypassed load instructions: 965138
bypassed load instructions: 1090953
bypassed load instructions: 1095823
bypassed load instructions: 1095853
```

For a correct implementation, it was hinted that the L1 data cache accesses statistics would change. We see that is the case. Because we let load instructions in the address range of `0xc0000000-0xc00fffff` bypass the L1 data cache, intuitively there should be less accesses, which also seems to be the case. I noticed that BFS also has some accesses above `0xc00fffff` when debugging, so that also confirms the difference.

Before:

```
cat gpuout_before.txt | grep "L1D_total_cache_accesses"
>>>
L1D_total_cache_accesses = 2116
L1D_total_cache_accesses = 4596
L1D_total_cache_accesses = 9307
L1D_total_cache_accesses = 27763
L1D_total_cache_accesses = 120908
L1D_total_cache_accesses = 508961
L1D_total_cache_accesses = 1230577
L1D_total_cache_accesses = 1478462
L1D_total_cache_accesses = 1488124
L1D_total_cache_accesses = 1490210
```

After:

```
cat gpuout_after.txt | grep "L1D_total_cache_accesses"
>>>
L1D_total_cache_accesses = 2090
L1D_total_cache_accesses = 4433
L1D_total_cache_accesses = 8344
L1D_total_cache_accesses = 21879
L1D_total_cache_accesses = 86800
L1D_total_cache_accesses = 353055
L1D_total_cache_accesses = 875477
L1D_total_cache_accesses = 1060607
L1D_total_cache_accesses = 1067834
L1D_total_cache_accesses = 1069905
```

How does this affect the miss rate? Shown below are the statistics before and after.
Before:

```
cat gpuout_before.txt | grep "L1D_total_cache_miss_rate"
>>>
L1D_total_cache_miss_rate = 0.9849
L1D_total_cache_miss_rate = 0.9508
L1D_total_cache_miss_rate = 0.8465
L1D_total_cache_miss_rate = 0.6784
L1D_total_cache_miss_rate = 0.5801
L1D_total_cache_miss_rate = 0.5364
L1D_total_cache_miss_rate = 0.4957
L1D_total_cache_miss_rate = 0.5018
L1D_total_cache_miss_rate = 0.5032
L1D_total_cache_miss_rate = 0.5039
```

After:

```
cat gpuout_after.txt | grep "L1D_total_cache_miss_rate"
>>>
L1D_total_cache_miss_rate = 0.9962
L1D_total_cache_miss_rate = 0.9824
L1D_total_cache_miss_rate = 0.9335
L1D_total_cache_miss_rate = 0.8364
L1D_total_cache_miss_rate = 0.7701
L1D_total_cache_miss_rate = 0.7261
L1D_total_cache_miss_rate = 0.6438
L1D_total_cache_miss_rate = 0.6353
L1D_total_cache_miss_rate = 0.6363
L1D_total_cache_miss_rate = 0.6370
```

For reference, this is the full diff of the code that was changed:

```
diff --git a/src/gpgpu-sim/shader.cc b/src/gpgpu-sim/shader.cc
index c6e7b8f..7e0bbbc 100644
--- a/src/gpgpu-sim/shader.cc
+++ b/src/gpgpu-sim/shader.cc
@@ -651,6 +651,9 @@ void shader_core_stats::print(FILE *fout) const {
               gpu_stall_shd_mem_breakdown[L_MEM_ST]
                                          [DATA_PORT_STALL]);  // data port stall
                                                               // at data cache
+
+
+  fprintf(fout, "bypassed load instructions: %d\n", gpgpu_n_bypassL1D);
   // fprintf(fout, "gpgpu_stall_shd_mem[g_mem_ld][mshr_rc] = %d\n",
   // gpu_stall_shd_mem_breakdown[G_MEM_LD][MSHR_RC_FAIL]); fprintf(fout,
   // "gpgpu_stall_shd_mem[g_mem_ld][icnt_rc] = %d\n",
@@ -1807,6 +1810,7 @@ mem_stage_stall_type ldst_unit::process_cache_access(
   mem_stage_stall_type result = NO_RC_FAIL;
   bool write_sent = was_write_sent(events);
   bool read_sent = was_read_sent(events);
+
   if (write_sent) {
     unsigned inc_ack = (m_config->m_L1D_config.get_mshr_type() == SECTOR_ASSOC)
                            ? (mf->get_data_size() / SECTOR_SIZE)
@@ -2039,6 +2043,7 @@ bool ldst_unit::memory_cycle(warp_inst_t &inst,
   const mem_access_t &access = inst.accessq_back();

   bool bypassL1D = false;
+
   if (CACHE_GLOBAL == inst.cache_op || (m_L1D == NULL)) {
     bypassL1D = true;
   } else if (inst.space.is_global()) {  // global memory access
@@ -2046,6 +2051,11 @@ 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 ((access.get_addr() >= 0xc0000000) && (access.get_addr() <= 0xc00fffff)) {
+      bypassL1D = true;
+      m_stats->gpgpu_n_bypassL1D++;
+  }
+
   if (bypassL1D) {
     // bypass L1 cache
     unsigned control_size =
@@ -2584,8 +2594,9 @@ void ldst_unit::cycle() {
                                        // on load miss only

         bool bypassL1D = false;
-        if (CACHE_GLOBAL == mf->get_inst().cache_op || (m_L1D == NULL)) {
+        if (CACHE_GLOBAL == mf->get_inst().cache_op || (m_L1D == NULL) || (mf->get_addr() >= 0xc0000000
&& mf->get_addr() <= 0xc00fffff)) {
           bypassL1D = true;
+          m_stats->gpgpu_n_bypassL1D++;
         } else if (mf->get_access_type() == GLOBAL_ACC_R ||
                    mf->get_access_type() ==
                        GLOBAL_ACC_W) {  // global memory access
diff --git a/src/gpgpu-sim/shader.h b/src/gpgpu-sim/shader.h
index 6481790..bd69665 100644
```

```
--- a/src/gpgpu-sim/shader.h
+++ b/src/gpgpu-sim/shader.h
@@ -1689,6 +1689,8 @@ struct shader_core_stats_pod {
   unsigned *gpgpu_n_shmem_bank_access;
   long *n_simt_to_mem;  // Interconnect power stats
   long *n_mem_to_simt;
+
+  unsigned gpgpu_n_bypassL1D;
 };

 class shader_core_stats : public shader_core_stats_pod {
```