ECE-786 PA-2

Nirmal Kumar Sedhumadhavan

TASK 1

The instruction.cc consists of several functions namely ptx_thread_info, and_impl, or_impl, abs_impl, add_impl, sub_impl, bra_impl, break_impl, call_impl etc. The function ptx_thread_info is mainly used to get the source and destination register values, the operation used, size of the data etc. There are arithmetic (abs_impl, add_impl, sub_impl, etc), logical (and_impl, or_impl, etc), control (bra_impl, break_impl, call_impl, etc) and other instructions that are implemented by the function to perform the necessary operation.

For Task 1, to change the semantics of float(32-bits) point add instruction to power operation the changes is made as shown below (Line 1061 of instructions.cc)

```
case F16_TYPE:
  data.f16 = src1_data.f16 + src2_data.f16;
  break;  // assert(0); break;
case F32_TYPE:
  data.f32 = pow(src1_data.f32, src2_data.f32); // Task 1 Implementation of new semantics
  break;
case F64_TYPE:
case FF64_TYPE:
  data.f64 = src1_data.f64 + src2_data.f64;
  break;
default:
  assert(0);
  break;
```

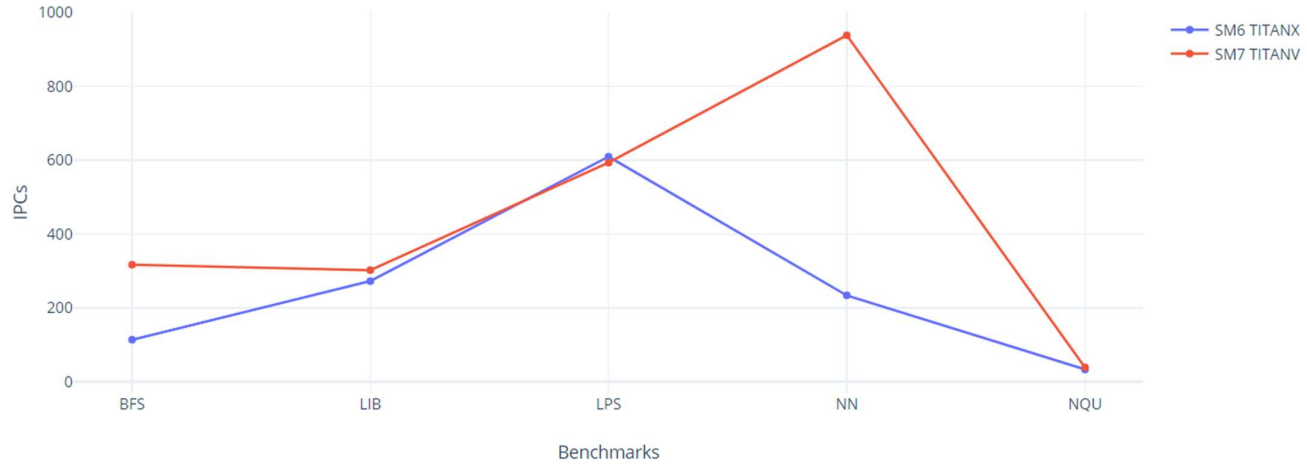
Output of vectorAdd.cu using original semantic

```
gpgpu_simulation_time = 0 days, 0 hrs, 0 min, 1 sec (1 sec)
gpgpu_simulation_rate = 286 (inst/sec)
source vector A: 1.000 2.000 3.000 4.000 5.000 6.000 7.000 8.000 9.000 1.000
source vector B: 2.000 3.000 2.000 3.000 2.000 3.000 2.000 3.000 2.000 3.000
result vector C: 1.000 8.000 9.000 64.000 25.000 216.000 49.000 512.000 81.000 1.000
GPGPU-Sim: *** exit detected ***
```

Output of vectorAdd.cu using modified semantic

```
gpgpu_simulation_time = 0 days, 0 hrs, 0 min, 1 sec (1 sec)
gpgpu_simulation_rate = 286 (inst/sec)
source vector A: 1.000 2.000 3.000 4.000 5.000 6.000 7.000 8.000 9.000 1.000
source vector B: 2.000 3.000 2.000 3.000 2.000 3.000 2.000 3.000 2.000 3.000
result vector C: 3.000 5.000 5.000 7.000 7.000 9.000 9.000 11.000 11.000 4.000
GPGPU-Sim: *** exit detected ***
```

TASK 2



Plot points for the graph

	BFS	LIB	LPS	NN	NQU
SM6_TITANX	113.5762	272.4011	609.5062	233.8858	32.9685
SM7_TITANV	316.8863	302.142	593.1517	938.1788	38.8251

From the graph we can infer that

- LIB, LBS & NQU benchmarks have less difference in their IPCs.
- BFS benchmark have moderate difference in their IPCs.
- NN benchmark have significant difference in their IPCs.

TASK 3

Modified code shown below (line 666, 1350 & 1567 of shader.cc, line 1845 of shader.h).

```
if (pc != pI->pc)
{
  m_stats->branch++; // Task 3 Increment warp counter for branch
  SCHED_DPRINTF(
      "Warp (warp_id %u, dynamic_warp_id %u) control hazard "
      "instruction flush\n",
      (*iter)->get_warp_id(), (*iter)->get_dynamic_warp_id());
  // control hazard
  warp(warp_id).set_next_pc(pc);
  warp(warp_id).ibuffer_flush();
}
```

```
else if (valid)
{
  // this case can happen after a return instruction in diverged warp
  m_stats->divergence++; // Task 3 Increment warp counter for branch with divergence
  SCHED_DPRINTF(
      "Warp (warp_id %u, dynamic_warp_id %u) return from diverged warp "
      "flush\n",
      (*iter)->get_warp_id(), (*iter)->get_dynamic_warp_id());
  warp(warp_id).set_next_pc(pc);
  warp(warp_id).ibuffer_flush();
}
```

```
fprintf(fout, "# of warps excuted conditional branch instrctions and have divergence: %d", divergence);// Task 3 Print the stats
fprintf(fout, "# of warps excuted conditional branch instrctions(no matter they have divergence or not): %d", branch);
```

```
//Task 3 Variables
int divergence = 0;
int branch = 0;
```

Output (SM6_TITANX config)

```
# of warps excuted conditional branch instrctions and have divergence: 0
# of warps excuted conditional branch instrctions(no matter they have divergence or not): 102600
```

TASK 4

Modified code shown below (line 666&2366 of shader.cc, line 1826 of shader.h).

```
//Task 4 Printing the local & global memory accesses
fprintf(fout, "# of global memory access: %d\n", global_memory);
fprintf(fout, "# of local memory access: %d\n", local_memory);
```

```
//Task 4 incrementing local & global memory accesses
if (inst.space.is_global())
  m_stats->global_memory++;
else if (inst.space.is_local())
  m_stats->local_memory++;
```

```
// memory access classification
int global_memory = 0;//Task 4 variables
int local_memory = 0;
```

Output (SM6_TITANX config)

```
# of global memory access: 448400
# of local memory access: 0
```