

```
configuration_files/BOOMV3_PIM/processor_PIM.cfg

//=====RVV BIGGER LOADS=====
//0 -> Disabled
//N -> Number of bytes loaded by the first of a vector load
SIMULATED_VECTOR_LOAD_SIZE = x
//=====RVV BIGGER LOADS=====
```

Alteração/Solução: Ao invés de enviar um prefetch avulso, a primeira instrução que dá MISS vira um prefetch.

Alerta: Isso foi planejado para um processador in-order.
Problema: Se mais requisições forem feitas em paralelo, elas podem não aproveitar a expandida (já que a verificação do MSHR já passou quando ela é gerada).

processor_t

// Mininum number of bytes loaded by each first vector load;
// Default is 8
uint32_t SIMULATED_VECTOR_LOAD_SIZE;

INITANTIATE_GET_SET_ADD (uint32_t, SIMULATED_VECTOR_LOAD_SIZE)

bool is_vector_insn(opcode_package_t *);

processor_t::allocate()

...
// libconfig::Setting &cfg_root = orcs_engine.configuration->getConfig();
// libconfig::Setting &cfg_processor = cfg_root["PROCESSOR"][0];
if (cfg_processor.exists("SIMULATED_VECTOR_LOAD_SIZE"))
 set_SIMULATED_VECTOR_LOAD_SIZE(cfg_processor["SIMULATED_VECTOR_LOAD_SIZE"]);
else
 set_SIMULATED_VECTOR_LOAD_SIZE(8);

processor_t::rename()

// Adicionar
for (uint32_t r = 0; r < rob_line->uop.num_mem_operations; ++r)
{
 uint32_t pos = (pos_mob + r) % MOB_LIMIT;
 rob_line->mob_base[pos].is_first_of_vector_load = ((r == 0) && rob_line->uop.is_vector_insn);
}
...

cache_manager_t::process()

// Em:
} else if (irequest->sent_to_ram) {
 <QUIZ>
}
...
Adicionar:
// Check if its the first load request from RVV
if (((memory_order_buffer_line_t *)request->clients[0])->is_first_of_vector_load) {
 // Change the request memory_size
 request->memory_size = (request->memory_size < SIMULATED_VECTOR_LOAD_SIZE) ?
 SIMULATED_VECTOR_LOAD_SIZE : request->memory_size;
}

memory_order_buffer_line_t

...
bool is_first_of_vector_load;
...

memory_order_buffer_line_t::package_clean()

...
this->is_first_of_vector_load = false;

processor_t::is_vector_insn(opcode_package_t *insn)

// Deve verificar se o opcode_assembly começa com v
if (insn->opcode_assembly[0] == 'v') return true;
return false;

```
uop_package_t

...

bool is_vector_insn;
```

```
opcode_package_t

...

bool is_vector_insn;
```

```
uop_package_t::opcode_to_uop(...)

...

this->is_vector_insn = opcode.is_vector_insn;
```

```
processor_t::fetch(...)

//=====RVV BIGGER LOADS=====
//add control variables
//=====RVV BIGGER LOADS=====
...

//=====RVV BIGGER LOADS=====
//Mark vector insn
//=====RVV BIGGER LOADS=====
operation.is_vector_insn = is_vector_insn(&operation);
```