processor_t::decode()

```
# Alterar
if (instr->num_reads <= 2) num_uops += instr->num_reads;
else {
    num_uops += 1; // Gather
}

if (instr->num_writes <= 1) num_uops += instr->num_writes;
else {
    num_uops += 1; // Scatter
}

# Para
if (instr->num_reads <= 2) {
    num_uops += instr->num_reads;

} else if (instr->num_reads > MOB_READ) {
    num_uops += ceil((instr->num_reads + 0.0f) / MOB_READ);

}else {
    num_uops += 1; // Small Gather
}

if (instr->num_writes <= 1) {
    num_uops += instr->num_writes;

} else if (instr->num_writes > MOB_WRITE) {
    num_uops += ceil((instr->num_writes + 0.0f) / MOB_WRITE);

} else {
    num_uops += 1; // Small Scatter
}

# ================================================================

# Alterar
//// Read 1 and Read 2
if (instr->num_reads <= 2) {
 ...
}

# Para
//// Read 1 and Read 2
if ((instr->num_reads <= 2){
 ...
}
// big Gather
else if (instr->num_reads > MOB_READ) {
    for (uint32_t r = 0;  r < instr->num_reads; r += MOB_READ)
    {
        ...
        for (uint32_t req = 0; req < MOB_READ; ++req) {
             new_uop.add_memory_operation(instr->reads_addr[r + req], instr->reads_size[r + req]);
         }
        ...
    }

} else {
 ...
}


# ================================================================

# Alterar
//// Last write uops from an instruction
if (instr->num_writes > 0) {
 ...
}

# Para
//// Big scatter
if (instr->num_writes > MOB_WRITE) {

  // One uop per store
  for (uint32_t w = 0; w < instr->num_writes; w += MOB_WRITE) {
     new_uop.package_clean();
     new_uop.opcode_to_uop(this->uopCounter++,
                           INSTRUCTION_OPERATION_MEM_STORE,
                           this->LATENCY_MEM_STORE,
                           this->WAIT_NEXT_MEM_STORE,
                           &(this->fu_mem_store),
                           *instr, uops_created, is_masked);

    for (uint32_t req = 0; req < MOB_WRITE; ++req) {
        new_uop.add_memory_operation(instr->writes_addr[w + req], instr->writes_size[w + req]);
    }

     ++uops_created;

     // It there are other uops from the same instruction that this uop depends on
     if ((uops_created-1) > w) {
        // Create a dependency with the previous uop from the same instruction
         ... # Mantém o mesmo do código anterior
     }
     ... # Mantém o mesmo do código anterior
  }



}
//// Store and small scatter
else if (instr->num_writes > 0) {
# Mantém o código anterior
...

}

# ================================================================


# Adicionar
assert(uops_created == num_uops);
```