```
const __m512i offsets = _mm512_set4_epi32(3,2,1,0);
const __m512i four = _mm512_set4_epi32(4,4,4,4);
int int_mask = 0x1111;
__mmask16 mask = _mm512_int2mask(int_mask);

#pragma omp parallel for
for (int r=0; r < nb_rows; r++) {
 __m512 accu = _mm512_setzero_ps();

 for (int n=0; n < nz; n+=4) {
   float* a = &dom.col_id_t[0]+n+nz*r;
   __m512i vect_i = _mm512_setzero_ps();
   vect_i = _mm512_mask_loadunpacklo_ps(vect_i, mask, a);
   vect_i = _mm512_mask_loadunpackhi_ps(vect_i, mask, a);
   vect_i = _mm512_swizzle_ps(vect_i, _MM_SWIZ_REG_AAAA);
   vect_i = _mm512_fmadd_epi32(vect_i, four, offsets);

   all_vect = _mm512_i32gather_ps(vect_i, dom.vec_vt, scale);

   mat_ent = _mm512_load_ps(dom.data_t + nb_mat*(n + r*nz));

   //perform first tensor multiplication with AAAA pattern
   vect = permute(all_vect, _MM_PERM_AAAA);
   mat = _mm512_swizzle_ps(mat_ent, _MM_SWIZ_REG_AAAA);
   accu = _mm512_fmadd_ps(vect, mat, accu);

   //three more times with BBBB, CCCC, and DDDD patterns
 }
 _mm512_storenrngo_ps(dom.result_vt+nb_mat*nb_vec*r, accu);
```