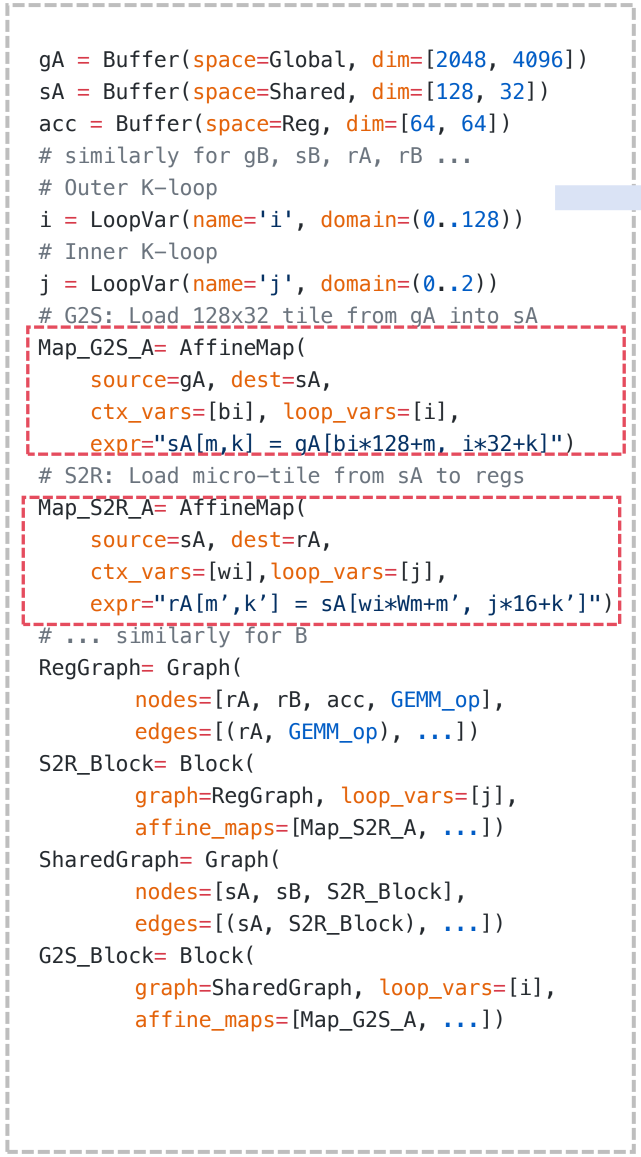
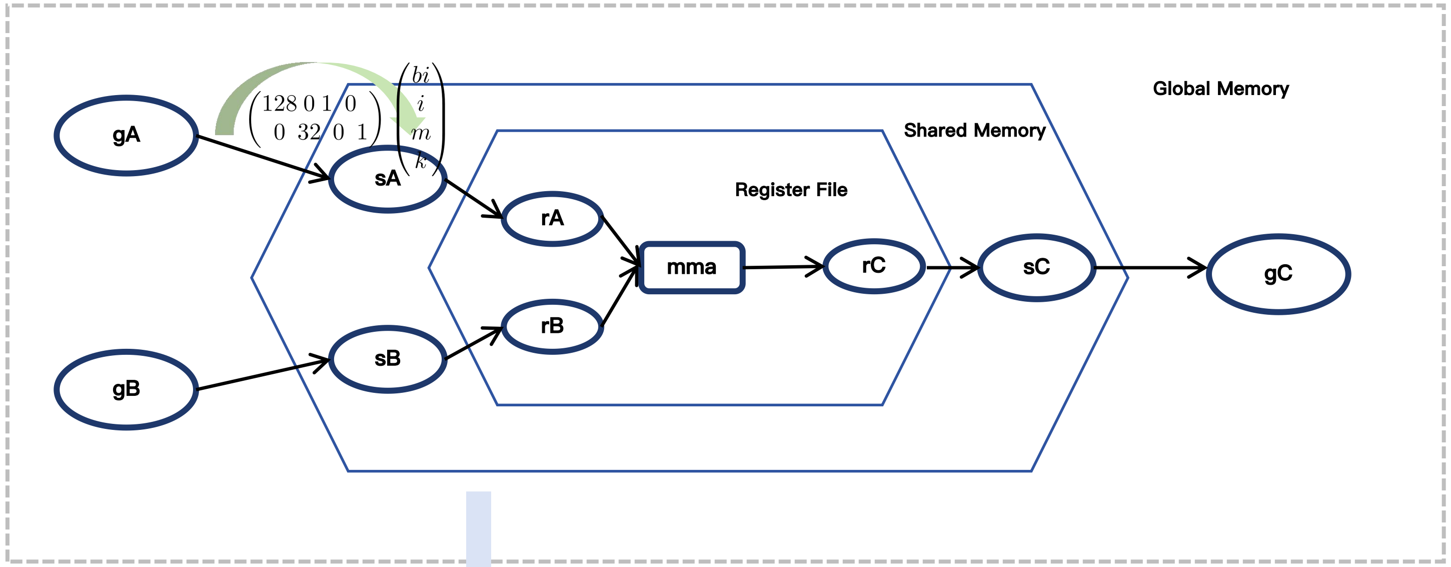
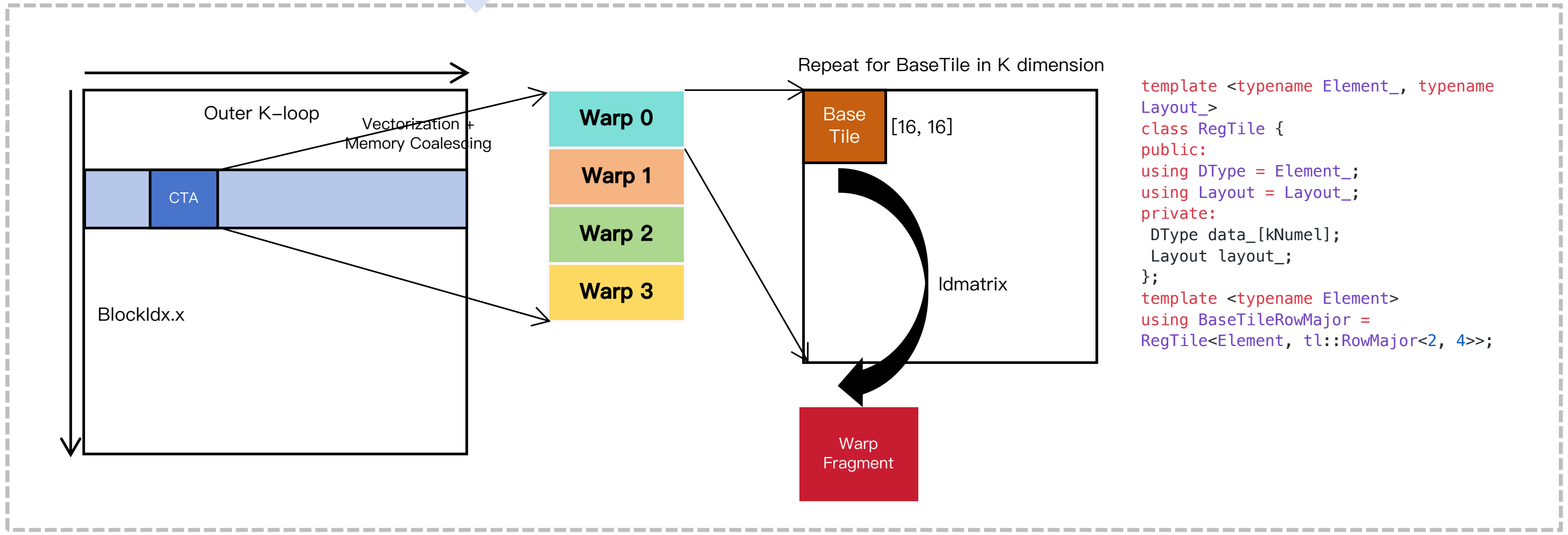


```
gA = Buffer(space=Global, dim=[2048, 4096])
sA = Buffer(space=Shared, dim=[128, 32])
acc = Buffer(space=Reg, dim=[64, 64])
# similarly for gB, sB, rA, rB ...
# Outer K–loop
i = LoopVar(name='i', domain=(0..128))
# Inner K–loop
j = LoopVar(name='j', domain=(0..2))
# G2S: Load 128x32 tile from gA into sA
Map_G2S_A= AffineMap(
    source=gA, dest=sA,
    ctx_vars=[bi], loop_vars=[i],
    expr="sA[m,k] = gA[bi*128+m, i*32+k]")
# S2R: Load micro–tile from sA to regs
Map_S2R_A= AffineMap(
    source=sA, dest=rA,
    ctx_vars=[wi],loop_vars=[j],
    expr="rA[m',k'] = sA[wi*Wm+m', j*16+k']")
# ... similarly for B
RegGraph= Graph(
        nodes=[rA, rB, acc, GEMM_op],
        edges=[(rA, GEMM_op), ...])
S2R_Block= Block(
        graph=RegGraph, loop_vars=[j],
        affine_maps=[Map_S2R_A, ...])
SharedGraph= Graph(
        nodes=[sA, sB, S2R_Block],
        edges=[(sA, S2R_Block), ...])
G2S_Block= Block(
        graph=SharedGraph, loop_vars=[i],
        affine_maps=[Map_G2S_A, ...])
```

(a) Multi–memory level GEMM described using AffineGraph IR.


$\begin{pmatrix} 128 & 0 & 1 & 0 \\ 0 & 32 & 0 & 1 \end{pmatrix} \begin{pmatrix} bi \\ i \\ m \\ k \end{pmatrix}$
Global Memory
Shared Memory
Register File
gA
sA
gB
sB
rA
rB
mma
rC
sC
gC

(b) Transform based on AffineGraph IR and generate a complete data flow graph



```
template <typename Element_, typename
Layout_>
class RegTile {
public:
using DType = Element_;
using Layout = Layout_;
private:
 DType data_[kNumel];
 Layout layout_;
};
template <typename Element>
using BaseTileRowMajor =
RegTile<Element, tl::RowMajor<2, 4>>;
```

(c) The complete copy process of the A matrix in all memory hierarchies based on the Affine Graph mapping.