High-Level Tensor Operation	Micro-Op Semantics	CISC Instruction Semantics

R = LOAD(addr)	load/store operations don't rely on micro-ops	**LOAD_OP(src, dst, x_size, y_size, x_stride, x_pad, y_pad)**
		for i in range(0, y_size+2*y_pad):
		  for j in range(0, x_size+2*x_pad):
		  if (i < y_pad || i >= y_size + y_pad ||
		  j < x_pad || j >= x_size + x_pad):
		  R[dst + i * (x_size+2*x_pad) + j] = 0
		  else:
		  R[dst + i * (x_size+2*x_pad) + j] =
		  DRAM[src + (i-x_pad) * x_stride + (j-y_pad)

STORE(R, addr)		**STORE_OP(src, dst, x_size, y_size, x_stride)**

R = GEMM(R, A, K)	**r[x]** : **r[x]** + MMUL(**a[y]**, **k[z]**)	**GEMM_OP(MICRO_OP*, end0, end1, x0, x1, y0, y1, z0, z1)**
		for i0 in range(0, end0):
		  for i1 in range(0, end1):
		  r[i0*x0 + i1*x1 + x] +=
		  GEMM(**a**[i0*y0 + i1*y1 + y],
		  k[i0*z0 + i1*z1 + z])

R0 = VMIN(R0, R1)	**r[x]** : if (**r[x]**<**r[y]**) then **r[x]** else **r[y]**	**ALU_OP(MICRO_OP*, OPCODE, USE_IMM, IMM_VAL, end0, end1, x0, x1, y0, y1)**
		
		for i0 in range(0, end0):
		  for i1 in range(0, end1):
		  r[i0*x0 + i1*x1 + x] =
		  OP(OPCODE, **r**[i0*x0 + i1*x1 + x],
		  r[i0*y0 + i1*y1 + y]) if USE_IMM else
		  OP(OPCODE, **r**[i0*x0 + i1*x1 + x],
		  IMM_VAL)

R0 = VMAX(R0, R1)	**r[x]** : if (**r[x]**>**r[y]**) then **r[x]** else **r[y]**	
R = VADDI(R, C)	**r[x]** : **r[x]** + **c**	
R0 = VADD(R0, R1)	**r[x]** : **r[x]** + **r[y]**	
R = VMULI(R, C)	**r[x]** : **r[x]**.low * **c**.low	
R0 = VMUL(R0, R1)	**r[x]** : **r[x]**.low * **r[y]**.low	
R = VSHLI(R, C)	**r[x]** : **r[x]** << **c**[log(bits):0]	
R = VSHRI(R, C)	**r[x]** : **r[x]** >> **c**[log(bits):0]	