线程结构和内存结构说明	共享数组	sharedata[blockdim.x/32]	数组的大小为单个block内的warp的个数															
	block的大小	blockdim.x	64								64							
	block的索引	blockidx.x	block0								block1							
	warp的索引	warpid	warp0				warp1				warp0				warp1			
	总的线程索引	index	0	1	…	31	32	…	…	63	64	…	…	95	96	…	…	127
	warp内的线程索引	laneid	0	1	…	31	0	1	…	31	0	1	…	31	0	1	…	31
	block内的线程索引	threadidx.x	0	1	…	31	32	…	…	63	0	1	…	31	32	…	…	63
	输入的数组的值	input	93	95	86	84	81	77	74	70	67	63	60	56	53	49	46	42
第一步	获取总的线程索引：	index=blockdim.x*blockidx.x+threadidx.x																
第二步	数组的值赋值给每个线程的寄存器	每个输入数组的值赋值给每个线程的寄存器data=input[index]																
第三步	warp内循环找最大值使用shuffle操作，所有warp并行进行。Shuffle操作会是warp内的32个线程的寄存器等于同一个值，因此取warp内的第一个线程的寄存器值就行。	warp内规约																
			0				1				0				1			
			0	1	…	31	0	1	…	31	0	1	…	31	0	1	…	31
			↓				↓				↓				↓			
第四步	并行赋值sharedata[warpid]=x	x=laneid==0的值					x=laneid==0的值				x=laneid==0的值				x=laneid==0的值			
		sharedata[warpid]=x									sharedata[warpid]=x							
第五步	数组的值赋值给每个线程的寄存器	temp=sharedata[threadidx.x]																
第六步	第二次warp内规约	blcok内的warp内的规约									blcok内的warp内的规约							
			threadidx.x=0				threadidx.x=1				threadidx.x=0				threadidx.x=1			
				↓								↓						
第七步	输出赋值out[griddim]	y=threadidx.x==0的值									y=threadidx.x==0的值							
		out[blockidx.x]=y									out[blockidx.x]=y							