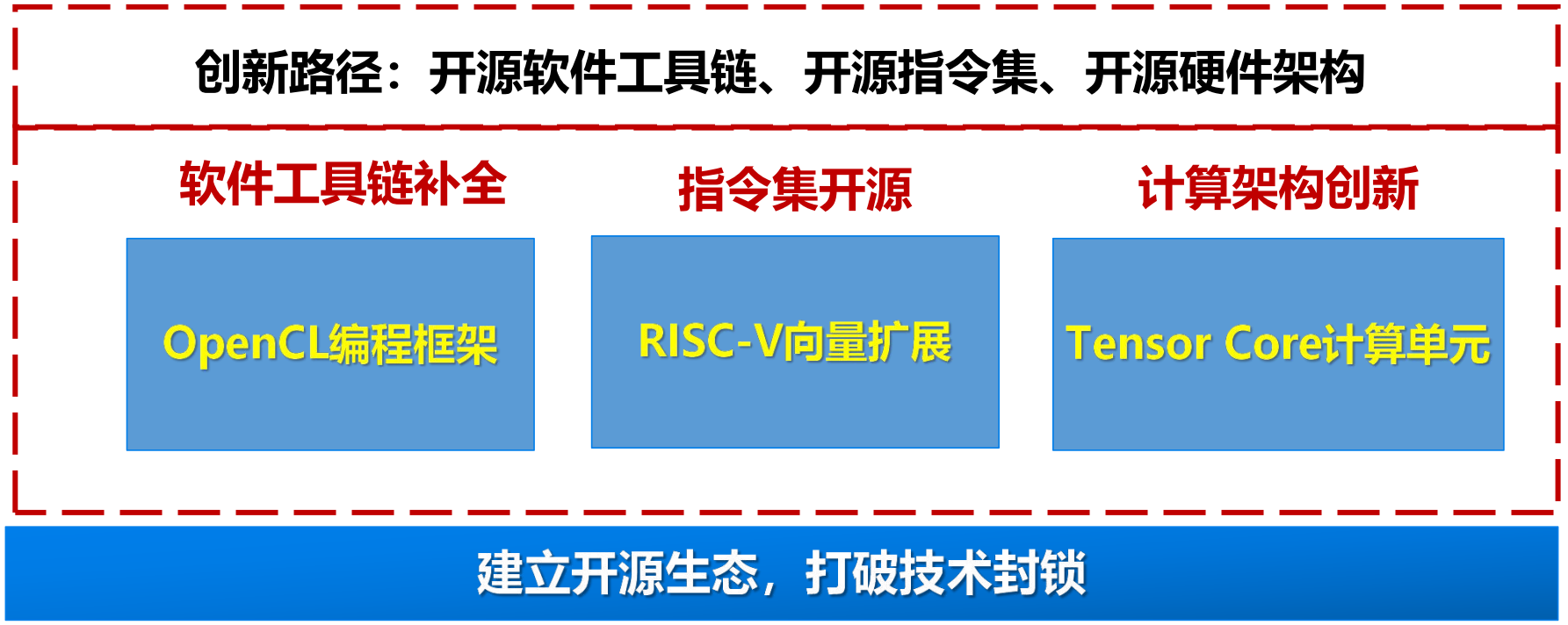
“乘影”：开源通用GPU指令集架构介绍

清华大学集成电路学院

于芳菲

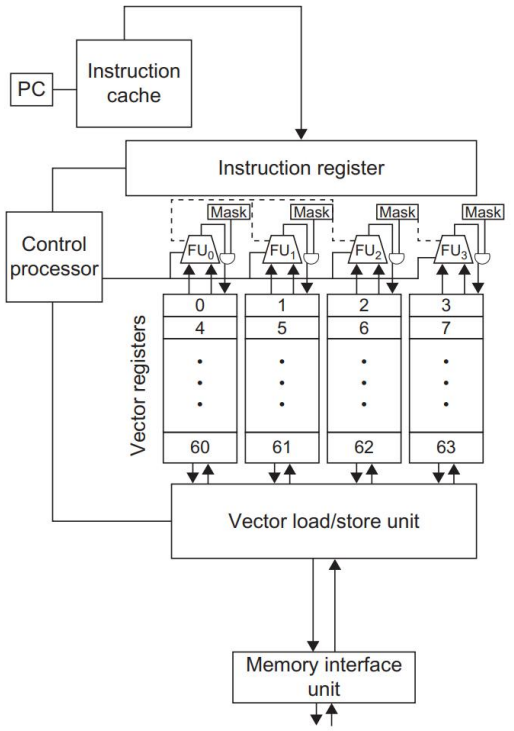
背景介绍

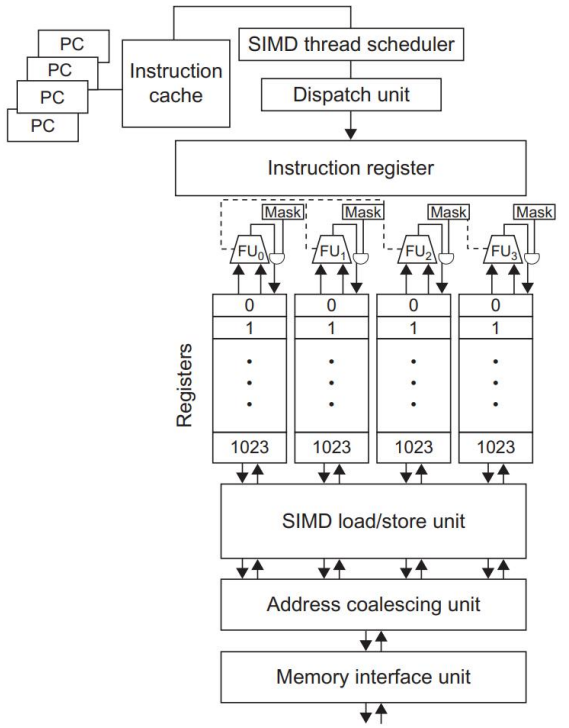
处理器芯片技术栈最重要的中间抽象层是指令集架构（ISA），高性能计算公司，尤其是并行计算芯片行业领导者英伟达（NVIDIA）和超微半导体（AMD）长期以来将其作为闭源技术规范。
开源指令集RISC-V有望突破这一现状，通过建立大众参与的开放上下游生态，建设积极旺盛的人才培养环境，使部分国内GPU设计公司核心技术摆脱海外IP授权，打破行业垄断玩家对先进技术的封锁。



向量处理器与GPGPU

- 典型向量处理器与典型SIMT架构在计算、访存、分支、任务控制、寄存器堆较为相似
- 动机：将向量lane与GPGPU thread一一对应



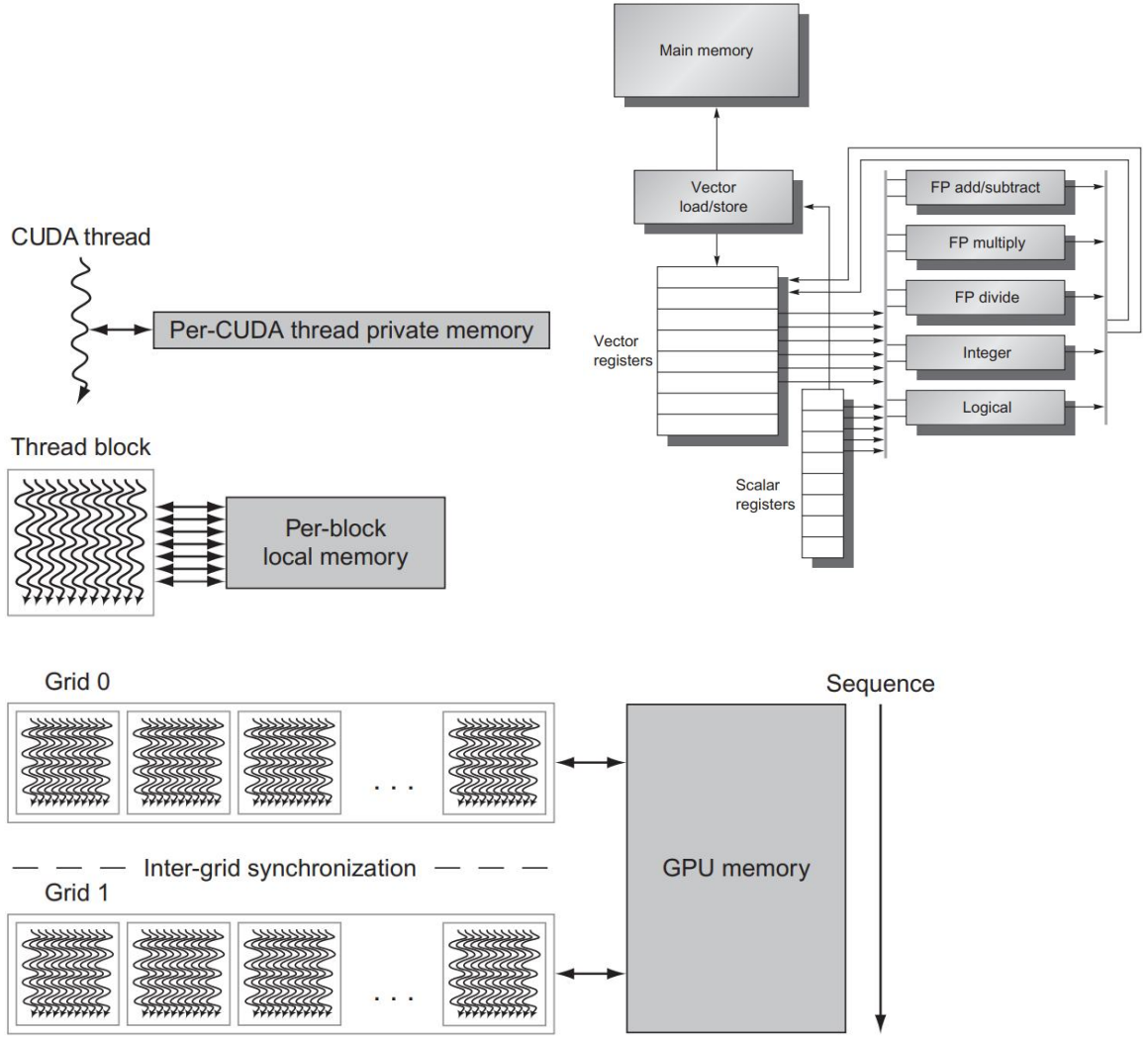


	计算单元	访存单元	分支控制	任务控制	寄存器堆	整体结构
Vector	Vector Lane	Gather/Scatter	Mask Registers	Control Processor	Vector Registers	Vector Processor
GPGPU	SIMD Lane	Global load/store	Predicate Registers	Thread Block Scheduler	SIMD Lane Registers	Multithreaded SIMD Processor

*图片来自《计算机体系结构 量化研究方法》

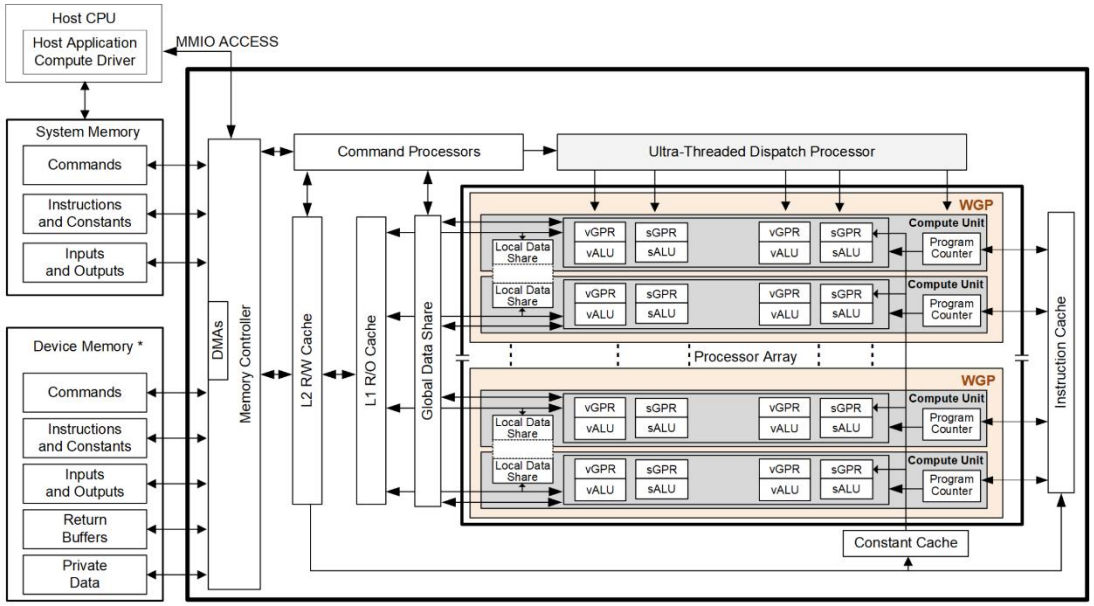
向量处理器与GPGPU

- RISC-V向量扩展优点：
  - 具备scalarization：用scalar寄存器和scalarALU计算公共数据地址
  - 访存特性表征：指令中指示了访存类型：连续、步进和索引
- SIMT架构优点：
  - warp级别并行调度：硬件将thread按32个分为一组整体执行，通过切换warp掩盖延时
  - 编程灵活性高，对程序员友好： CUDA编程仅需针对单个thread行为进行描述；OpenCL编程针对workitem行为进行描述



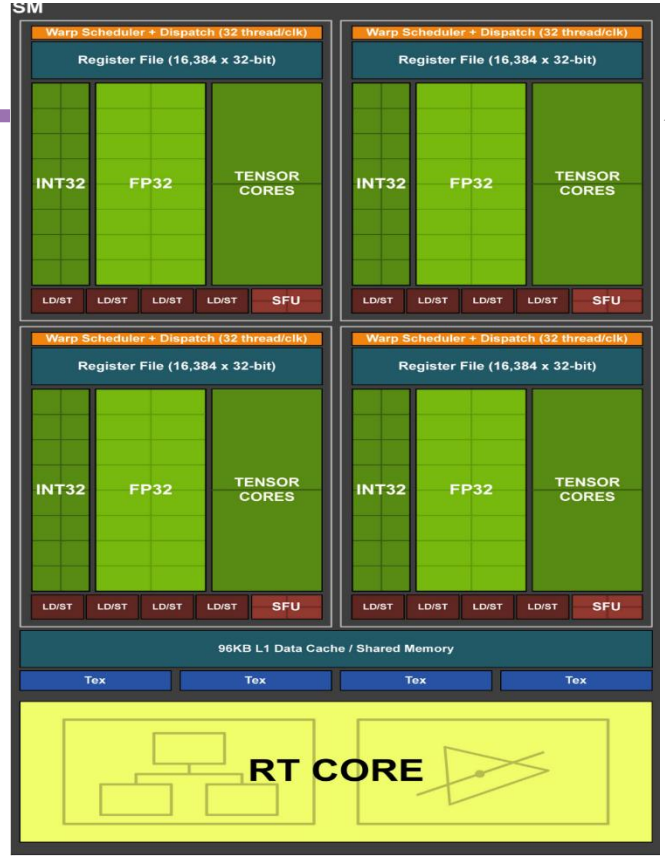
*图片来自《计算机体系结构 量化研究方法》

AMD与NVIDIA指令集



*Discrete GPU – Physical Device Memory; APU – Region of system for GPU direct access

Figure 1. AMD RDNA3 Generation Series Block Diagram



AMD RDNA3系列的整体硬件架构
被分为scalar pipeline、vector pipeline
Vector pipeline用于提供计算带宽
Scalar pipeline用于控制流以及公共地址计算

Nvidia 自18年Turing架构推出后加入Uniform Datapath
用于加速整数操作如更新数组索引、循环索引或者循环边界检查等（ typically updating array indices, loop indices or pointers; or performing array or loop boundary checks.）

*图片来自RDNA3、Turing白皮书

“乘影”GPGPU指令集 - 设计概览

- 指令集：RISC-V + 向量扩展
  - GPGPU与向量处理器的结构有许多相似之处
  - 大部分GPGPU所需指令可以由RISC-V向量扩展集提供
- GPGPU额外的特性和功能需求：
  - 线程束的分支和同步、多线程束执行
  - 按OpenCL模型要求，线程在运行时能够获取自身索引号
  - 更高的指令表达能力，以提升执行效率

扩展	指令类型	乘影GPGPU
V(zve32f)	Configuration-Setting	部分支持
	Loads and Stores	支持
	Integer Arithmetic	支持
	Floating-Point	支持fp32
	Fix-Point	参照需求添加
	Reduction、Mask、Permutation	参照需求添加
I		支持
M		支持
A		支持
zicsr		支持
F(zfinx)		支持

“乘影”GPGPU指令集 – 标量指令

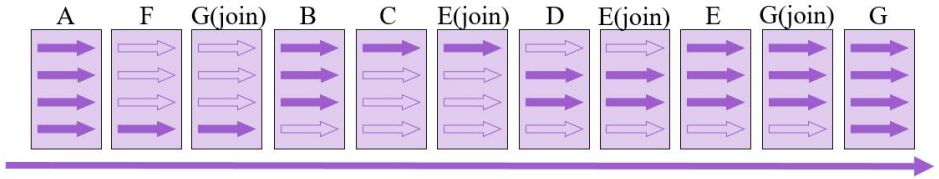
- 覆盖范围：RV32IMA_zicsr_zfinx
- I：基本指令
  - 整数基本运算、访存、线程束整体分支、函数调用
  - *未支持ecall和ebreak指令
- M：乘法指令
- A：原子指令
  - 约束多线程对同一地址的读写操作，描述的是单个线程行为。
- zicsr：CSR寄存器操作
  - 线程和线程块索引在启动线程束前存入特定CSR以供读取
- zfinx：单精度浮点指令
  - 不设标量浮点寄存器，标量浮点数同样存入整数寄存器

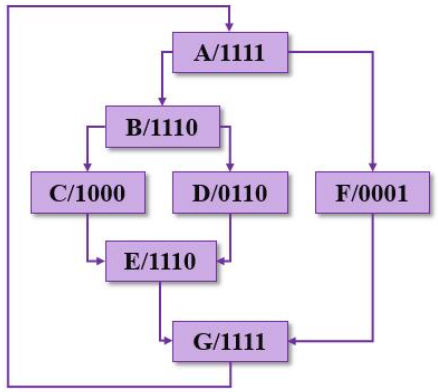
“乘影”GPGPU指令集 – 向量指令

- 在V扩展的zve32f子集上进行裁剪与修改，使之符合SIMT编程模型：
- 支持基本运算和访存指令，保证线程标量操作均存在向量版本指令
- GPGPU计算方式特性考量：
  - 未支持元素数目和宽度的变长特性，固定元素数目32+宽度32位
  - 未支持向量元素间数据交换（gather, scatter, shuffle等）
  - 写回数据位宽小于32时（如掩码计算、读字节），仍采用32位对齐方式写回

自定义指令：线程束分支与汇合

- 使用自定义指令和硬件分支同步栈
- 重汇聚pc设置指令：setrpc
  - 将即将出现的分支的重汇聚PC写入rpc CSR寄存器
- 线程束分支指令： vbeq、vbge等
  - 从CSR中取出上一次写入的rpc值
  - 当发现各个线程判断结果不一致、出现分歧（THEN和ELSE）时，将取出的rpc和当前掩码压栈、将线程数多的分支的PC值和掩码压栈，按线程数少的分支掩码执行线程数少的分支 （线程数量相同时先执行ELSE分支）
- 线程束汇合指令：join
  - 分支执行到此处，若栈顶rpc与当前执行PC相同时，弹出栈顶信息，将PC置为PC，掩码置为栈顶掩码；若栈顶rpc与当前执行PC不同，则不做任何操作
- 在未分歧情形下，简化压栈和汇合操作，只执行一个分支





(a) 程序样例

rPC	PC	掩码(mask)
-	-	-

(b) 栈初始状态

rPC	PC	掩码(mask)
G	G	1111
G	B	1110

(c) 分支1发生后

rPC	PC	掩码(mask)
G	G	1111

(d) F段程序执行完毕后

rPC	PC	掩码(mask)
G	G	1111
E	E	1110
E	D	0110

(e) 分支2发生后

rPC	PC	掩码(mask)
G	G	1111
E	E	1110

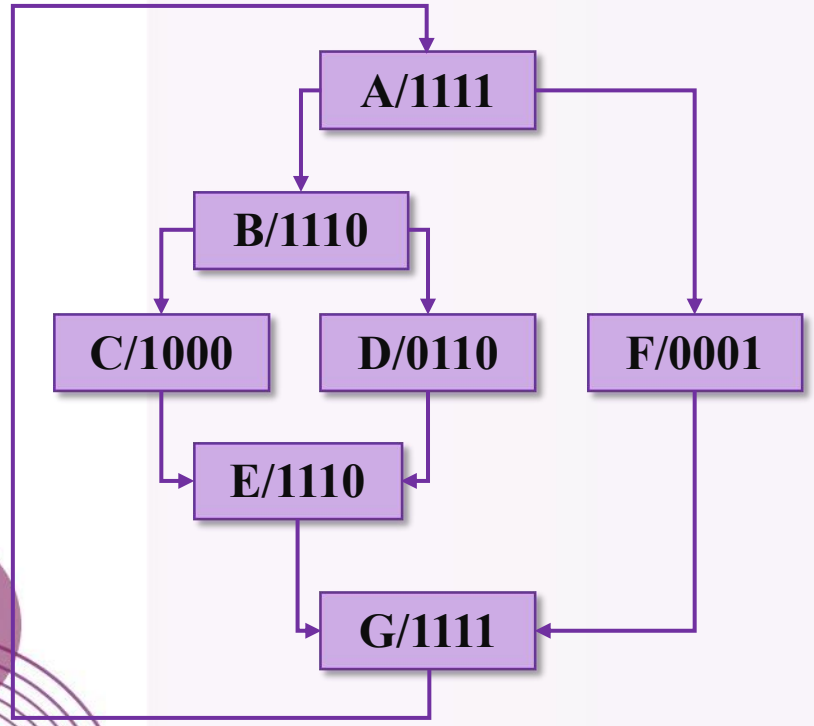
(f) C段程序执行完毕

rPC	PC	掩码(mask)
G	G	1111

(g) 分支2执行完毕

rPC	PC	掩码(mask)
-	-	-

(h) 分支1执行完毕



(a) 程序样例

rPC	PC	掩码(mask)
-	-	-

(b) 栈初始状态

rPC	PC	掩码(mask)
G	G	1111
G	B	1110

(c) 分支1发生后

rPC	PC	掩码(mask)
G	G	1111

(d) F段程序执行完毕后

rPC	PC	掩码(mask)
G	G	1111
E	E	1110
E	D	0110

(e) 分支2发生后

rPC	PC	掩码(mask)
G	G	1111
E	E	1110

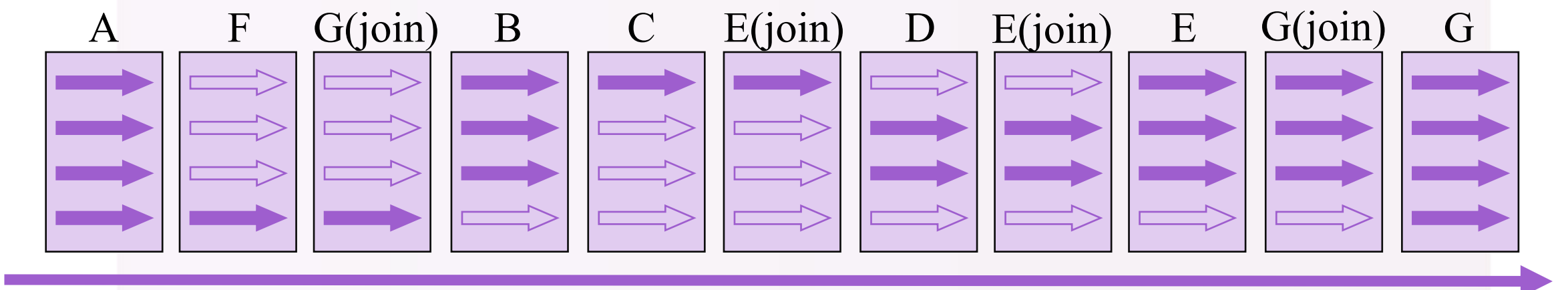
(f) C段程序执行完毕

rPC	PC	掩码(mask)
G	G	1111

(g) 分支2执行完毕

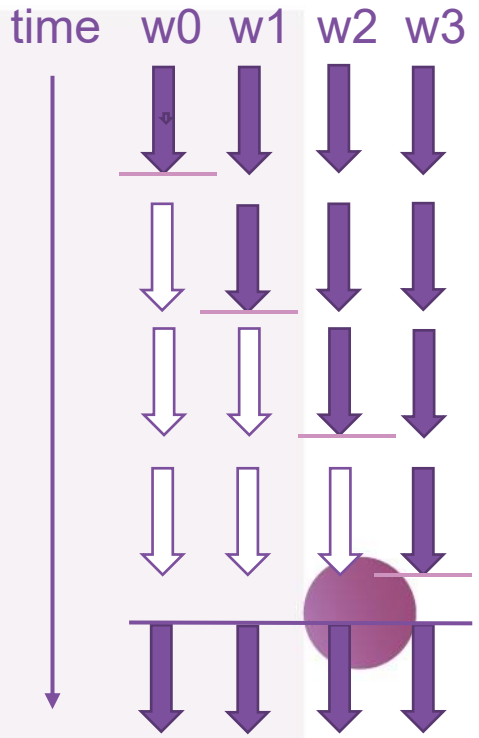
rPC	PC	掩码(mask)
-	-	-

(h) 分支1执行完毕


A
F
G(join)
B
C
E(join)
D
E(join)
E
G(join)
G

自定义指令：线程同步和退出

- SIMT编程模型需求线程间的数据交换
  - 显式插入栅栏（barrier）
  - 线程到达栅栏后需等待线程块内所有线程均到达，方可继续执行
- 栅栏指令：barrier、barriersub
  - 可阻拦特定内存或线程范围的访存操作
  - 可指定解除栅栏所需的到达线程数目
- 线程退出指令：endprg
  - warp的核函数退出，回收PC控制器、指令缓冲、寄存器堆、共享内存等资源



自定义指令：寄存器与立即数扩展

- 问题
  - RISC-V原始寄存器编码仅有5位，32个可用寄存器
  - OpenCL内建向量数据类型在“乘影”架构中会占用多个向量寄存器，容易发生寄存器溢出
  - 向量指令操作数较多，立即数仅有5位，只能表示-16~15的整数
- 解决
  - 定义寄存器/立即数扩展指令，携带下一条指令的寄存器/立即数高位编码
  - 寄存器扩展指令：四个操作数寄存器扩展高3位
  - 立即数扩展指令：两个操作数寄存器扩展高3位、立即数扩展高6位
  - 支持8位寄存器编码（256寄存器）、11位立即数，并保持了不扩展时的兼容性

imm[11:0](x3,x2,x1,xd)	rs1	0 1 0	rd	0 0 0 1 0 1 1	regext x0,x0,imm
imm[11:0](imm[10:5],x2,xd)		0 1 1			regexti x0,x0,imm

regext x0,x0,imm //{x3,x2,x1,xd}, reg id += xi<<5 respectively in next inst
regexti x0,x0,imm //{imm_n,x1,xd}, reg id += xi<<5 and imm += imm_n << 5 in next inst

示例：
regext x0, x0, 0b000_001_000_010
vadd.vx v16, v20, x8
等价于vadd.vx v80, v20, x40

自定义指令：寄存器对拼接指令

- 问题
  - 32位寄存器的架构只能访问4GB的内存空间
  - 将寄存器全部扩展为64位带来额外的硬件开销和资源浪费
  - 混合32位64位数据类型在编译器支持上存在问题
- 解决
  - 定义寄存器对拼接指令，同时兼容寄存器扩展语义
  - 将相邻32位寄存器对拼接为64位数据，偶对齐
  - 按需支持64位操作

imm[11:0](x3,x2,x1,xd) imm[11:0](imm[10:5],x2,xd)	rs1	1 0 1 1 1 1	rd	0 0 0 1 0 1 1	regpair x0,x0,imm regpairi x0,x0,imm

示例：
regpair x0, x0, 0b000_000_000_001
lw x9, 0(x6)
等价于lw x41, 0([x7,x6])

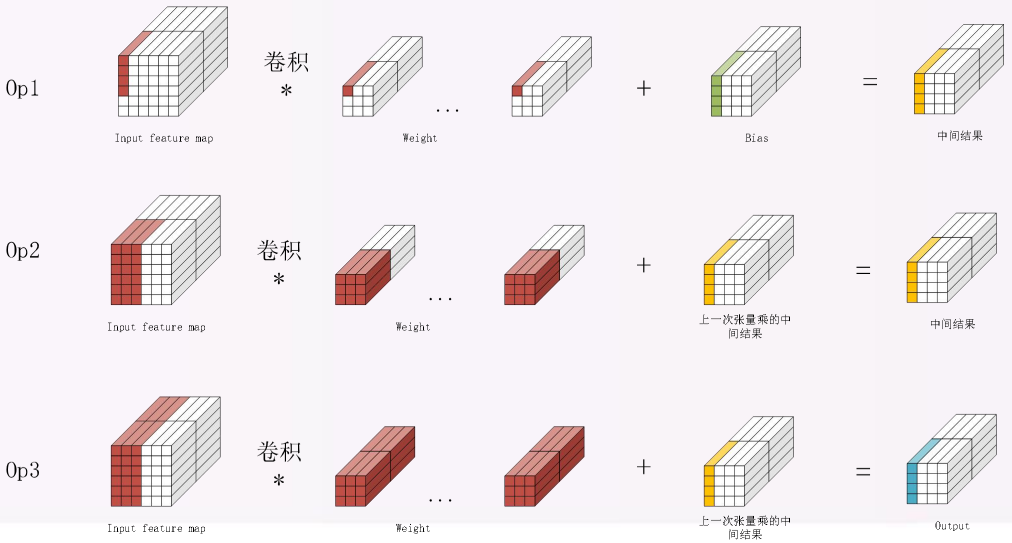
自定义指令：额外访存指令

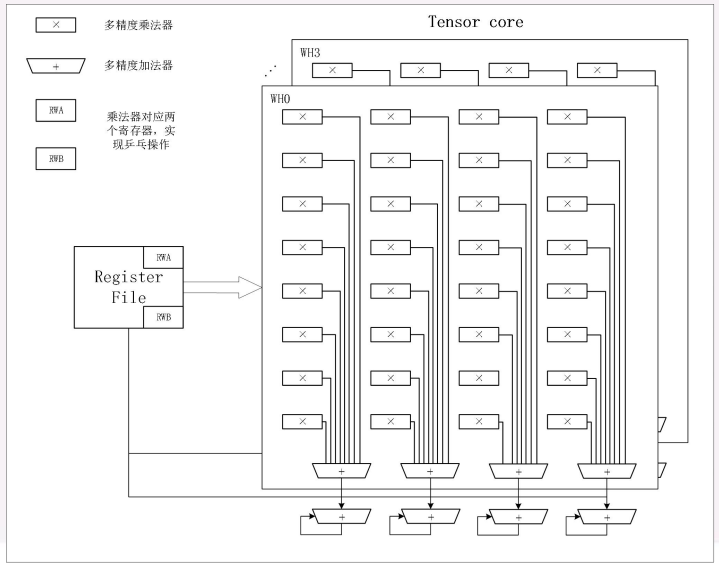
- RISC-V向量扩展的访存指令设计与GPGPU需求并不一致
- RISC-V向量扩展的长立即数版本向量访存指令缺失
  - 添加vls12系列指令，支持向量寄存器基址+12位立即数偏移量访存
- OpenCL需求访问多类地址空间，RISC-V向量扩展并无相关支持
  - 添加vls系列指令，支持标量寄存器基址+11位立即数偏移访存
  - vls指令只用于每个线程访问仅对自己可见的私有内存，硬件按驱动提供的信息映射为真实地址

自定义指令：额外计算指令

- 添加vadd12指令，以12位立即数为源操作数
  - 降低寻址、立即数装载/加减、跳转时的立即数扩展频率

- 添加张量计算指令和特殊函数（指数）指令
  - 支持4*8*4规模的矩阵/张量运算



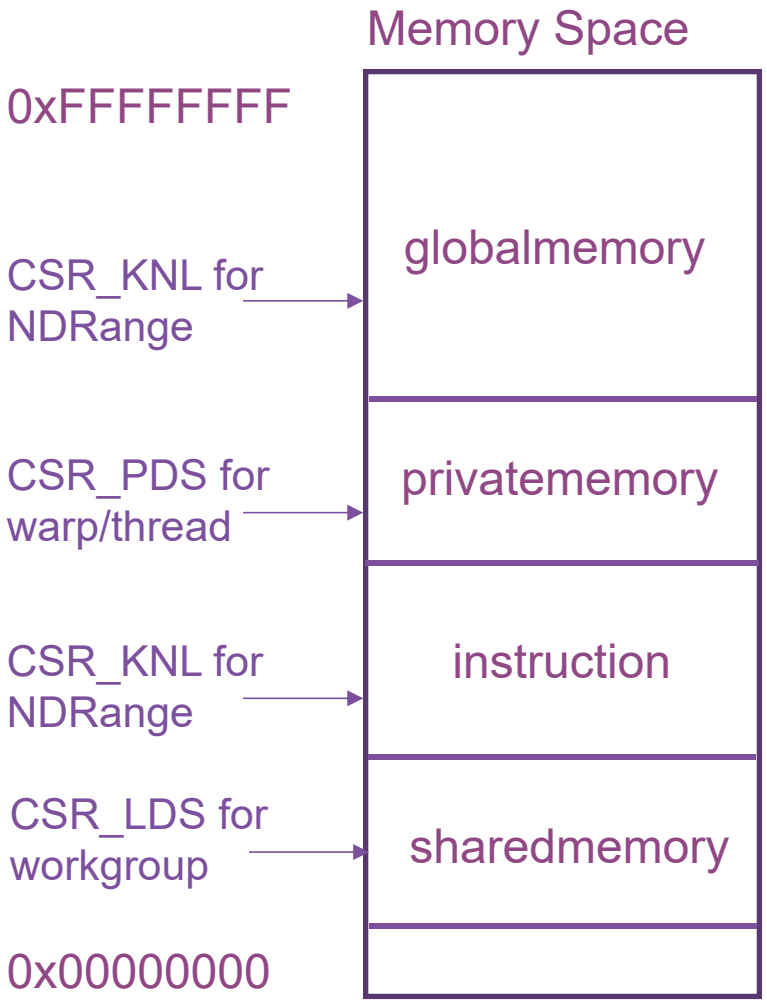


“乘影” GPGPU指令集 – 寄存器堆

- 每个warp有64个32bit sGPR，256个1024bit vGPR(num_thread=32时)。标量寄存器由同一warp内的线程共享。
- 特殊寄存器：x0为0寄存器，x1为返回pc寄存器，x2为栈指针寄存器、sharedmemory基址，x4为privatememory基址
- CSR：添加warp id、workgroup id、kernel metadata baseaddr等
- ABI：约定8个sGPR和32个vGPR为函数参数，16个vGPR为返回值

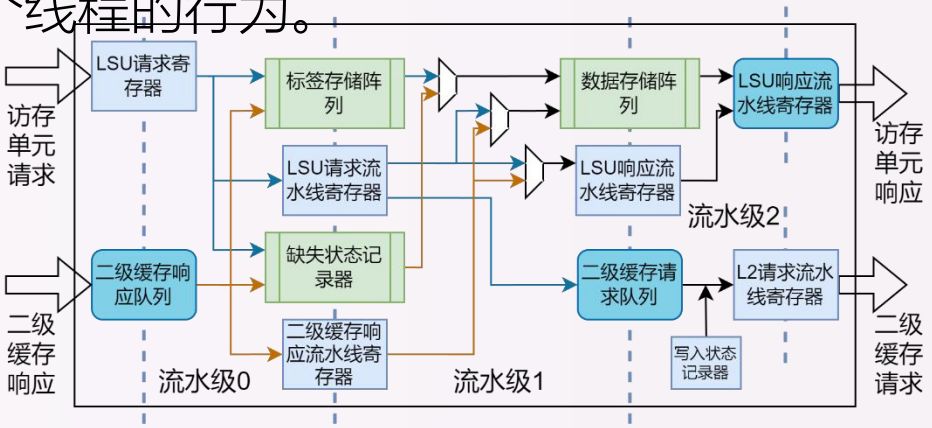
“乘影” GPGPU指令集 – 存储模型

- 不同线程层次对应不同存储层次：
  - warp地址空间：PrivateMemory（栈）
  - workgroup地址空间： SharedMemory(LDS)
  - kernel地址空间：GlobalMemory（数据）、ConstantMemory（数据、指令）
- 驱动和线程块调度器完成地址分配
- 针对不同空间数据采用不同指令/地址范围进行访问
  - 分配的内存基址初始化到CSR里
  - vls指令专用于访问私有内存
  - 其余所有访存指令（RVV访存和自定义访存指令）均可访问任意内存，以地址范围区分



“乘影” GPGPU指令集 – 缓存一致性

- 在RISC-V弱存储模型的基础上，“乘影”进一步部署了释放连贯性指导的缓存一致性（RCC）。让SM私有缓存间具备一致性功能的同时，避免了硬件一致性协议（比如基于directory的MESI协议）带来的高昂硬件成本和运行时带宽开销。原子指令也处理成对单个线程的行为。



微架构操作	④全局无效化	③全局无效化	④全局无效化	①清空MSHR	④全局无效化	③全局冲刷
RVWMO连贯性语义	.aq标识符	.rl标识符	FENCE R,R	FENCE R,W	FENCE W,R	FENCE W,W
附带的RCC一致性操作	“获取”和“释放”	“释放”	“获取”和“释放”	无	“获取”和“释放”	“释放”

“乘影”GPGPU与其它ISA对比

ISA	AMD RDNA	AMD GCN	NVIDIA PTX	Intel GEM	Imagination PowerVR	Vortex RISC-V	乘影 RVV
内存模型	GDS, LDS Constants Global	GDS, LDS Constants Global	Shared, Texture Constants Global	由软件管理	Global Common Store Unified Store	Shared Global	Private, LDS Constants Global
线程模型	Workgroup Wavefront 32/64线程	Compute Unit Wavefront 64线程	Grid/CTA Warp	Root Thread Child Thread	USC 32线程	Compute Unit Wavefront	Workgroup Warp(Wavefront) 32线程
指令字长	32/64位	32/64位	128位 (SASS, Ampere)	128位		32位	32/64位
寄存器堆	向量和标量 256 VGPRs 106 SGPRs	向量和标量 256 VGPRs 102 SGPRs	标量 (虚拟指令集)	向量 128 GRFs	128-bit 向量	标量	向量和标量 256 VGPRs 64 SGPRs
同步	barrier wait_cnt	barrier wait_cnt	barrier membar	wait fence	fence	barrier flush	barrier (membar) fence

“乘影”GPGPU与其它ISA对比（续）

ISA	AMD RDNA	AMD GCN	NVIDIA PTX	Intel GEM	Imagination PowerVR	Vortex RISC-V	**乘影 RVV**
线程控制	endpgm指令 线程mask	endpgm指令 线程mask	谓词（predicate）	消息（Message）	谓词（predicate）	线程mask	**endprg指令 线程mask**
指令流和分支控制	branch 线程mask	branch 线程mask split/join	branch predicate	branch SPF Regs split/join	branch predicate	split/join	**branch vbranch/join**
ALU操作	算术 条件判断 位运算	算术 条件判断 位运算	算术 条件判断 位运算	算术 条件判断 位运算	算术 条件判断 位运算	算术 条件判断 位运算	**算术 条件判断 位运算**
访存操作	load store prefetch	load store prefetch	load store prefetch	load store	load store	load store	**load store**
其他	图形 Matrix Core 光线追踪	图形	图形 Tensor Core 光线追踪	图形 Matrix Engine 光线追踪	图形	仅实现Texture Sampling	**Tensor Core**

“乘影”GPGPU指令集-演进方向

- 有限的编码空间
  - 32位/64位混编形式
  - 利用RISCV指令集中GPU用不到的指令拓展编码空间
- 更多数据类型
  - int8, int16, fp16, bp16, fp8, ...
- 更多计算方式
  - packaged math
  - 矩阵运算
- 存储模型
  - 添加域标签，采用不同缓存策略
  - 添加数据流特性标签，采用不同缓存策略

THANK YOU

上海清华国际创新中心
集成电路研究平台
International Innovation Center of Tsinghua University Shanghai
Integrated Circuit Research Platform

OpenGPGPU
乘影