Welcome

To Advance through Presentation
Use Page Up and Page Down Keys

AltiVec Overview

A. Sazegari

AltiVec Technical Lead
Core OS

Introduction

- AltiVec™ is an extension to the PowerPC Instruction Set Architecture
- Designed to extend Apple's leadership position in multimedia processing

AltiVec is a trademark of Motorola, Inc.

What You'll Learn

- AltiVec architecture
- AltiVec performance potential
- AltiVec tools and libraries

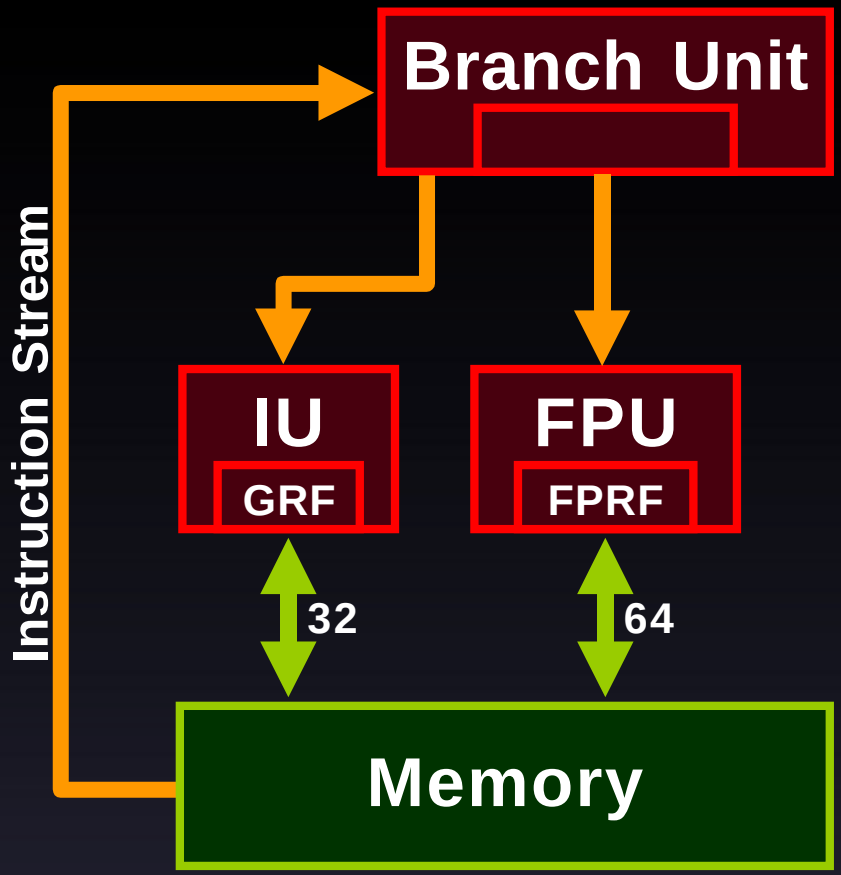
AltiVec Technology

- Vector/SIMD technology
  - Fixed-length vector operands (packed data)
  - Single Instruction Multiple Data
  - RISC-style instruction set
  - Optimized for digital signal processing
- Elevates multimedia to first-class data type
- Useful wherever data-parallelism exists

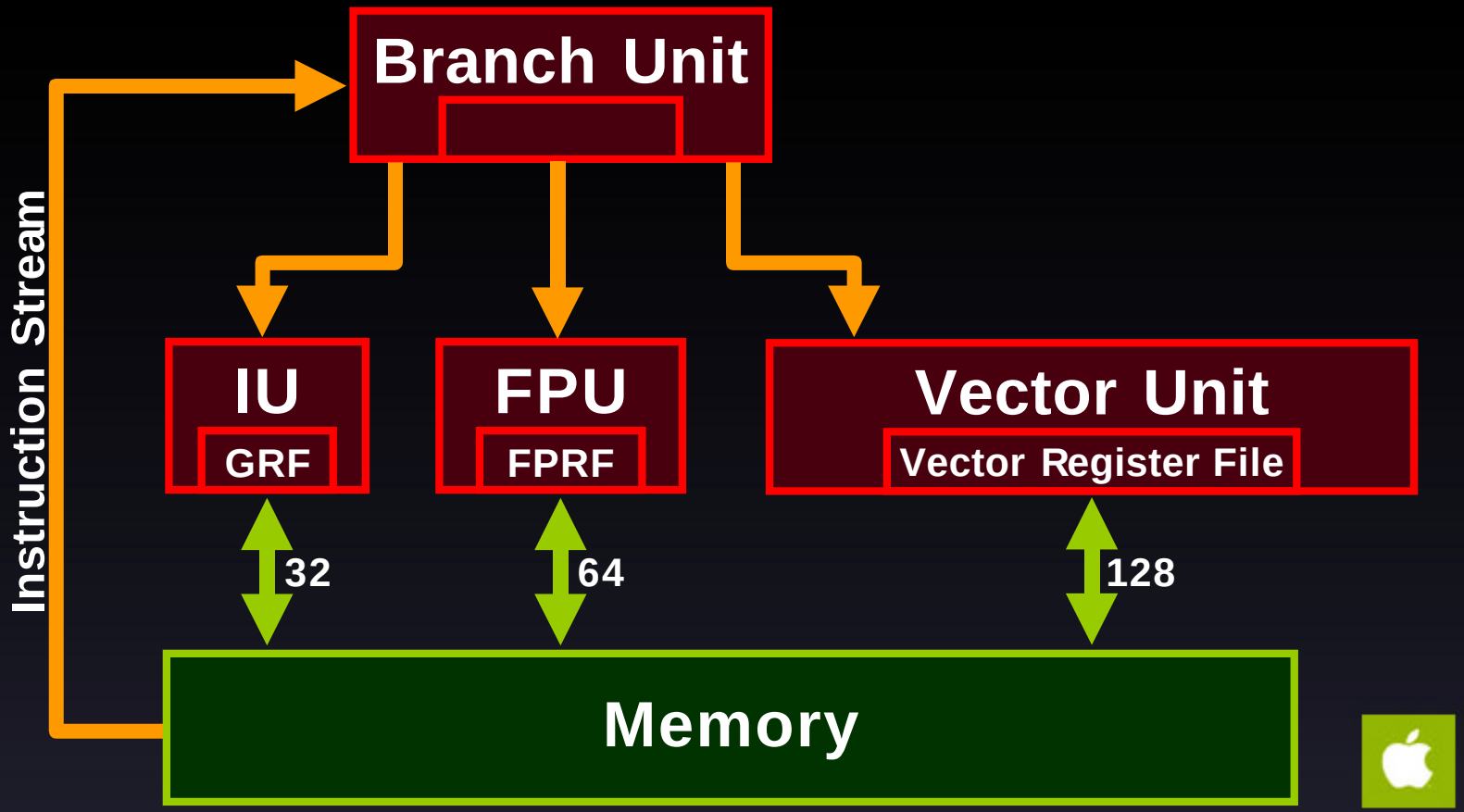
AltiVec Architecture

- New Vector Register File:
  - 32 new 128-bit wide registers
- New data-types:
  - Packed byte, halfword, and word integers
  - Packed IEEE single-precision floats
- Saturation Arithmetic capability
- 160 new PowerPC instructions

PowerPC Architecture



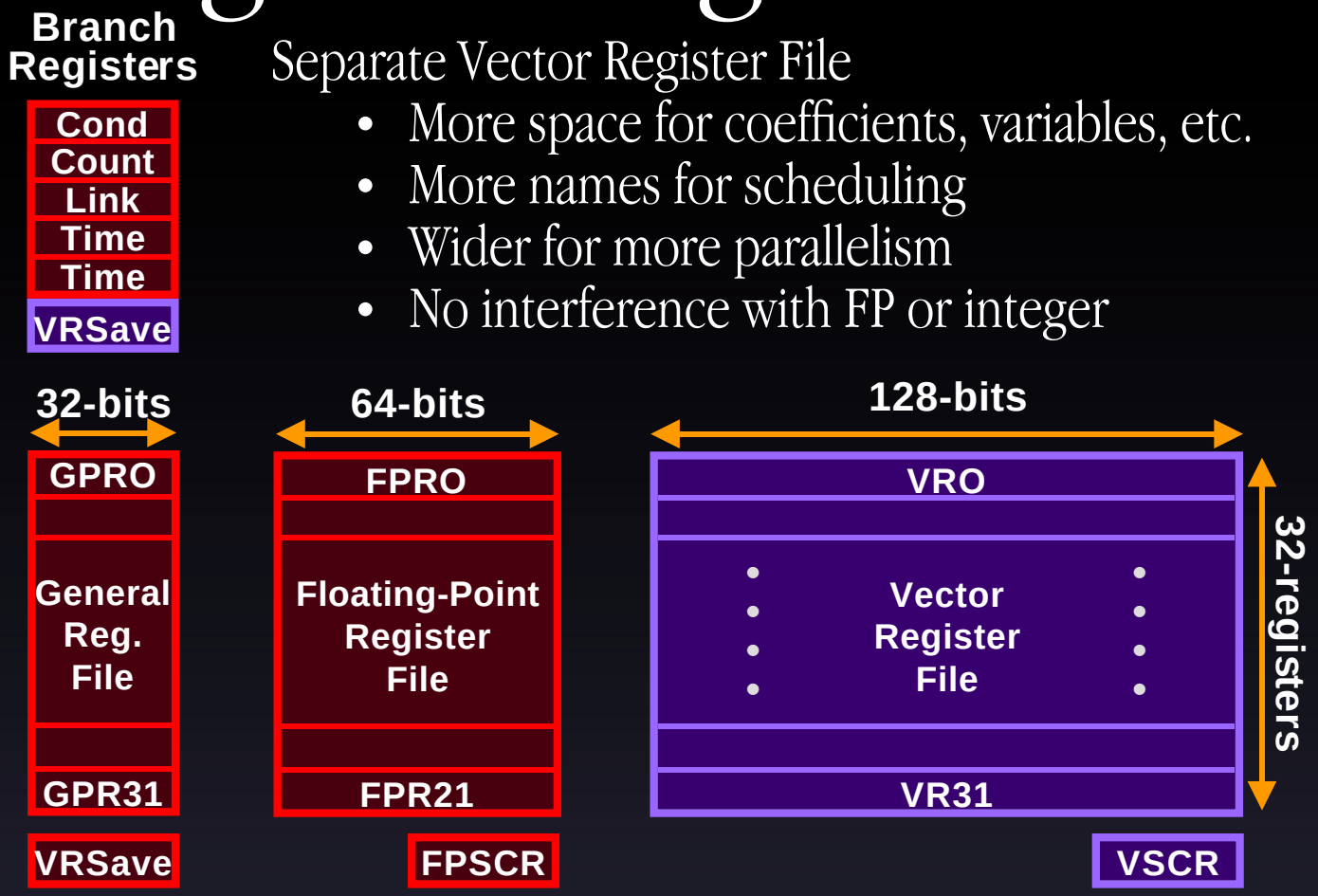
AltiVec Architecture



Programming Model

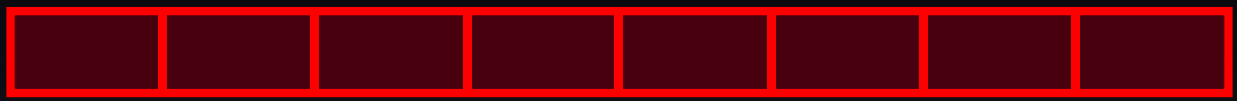
Separate Vector Register File

- More space for coefficients, variables, etc.
- More names for scheduling
- Wider for more parallelism
- No interference with FP or integer



Vector Data Types

16 signed or unsigned integer bytes

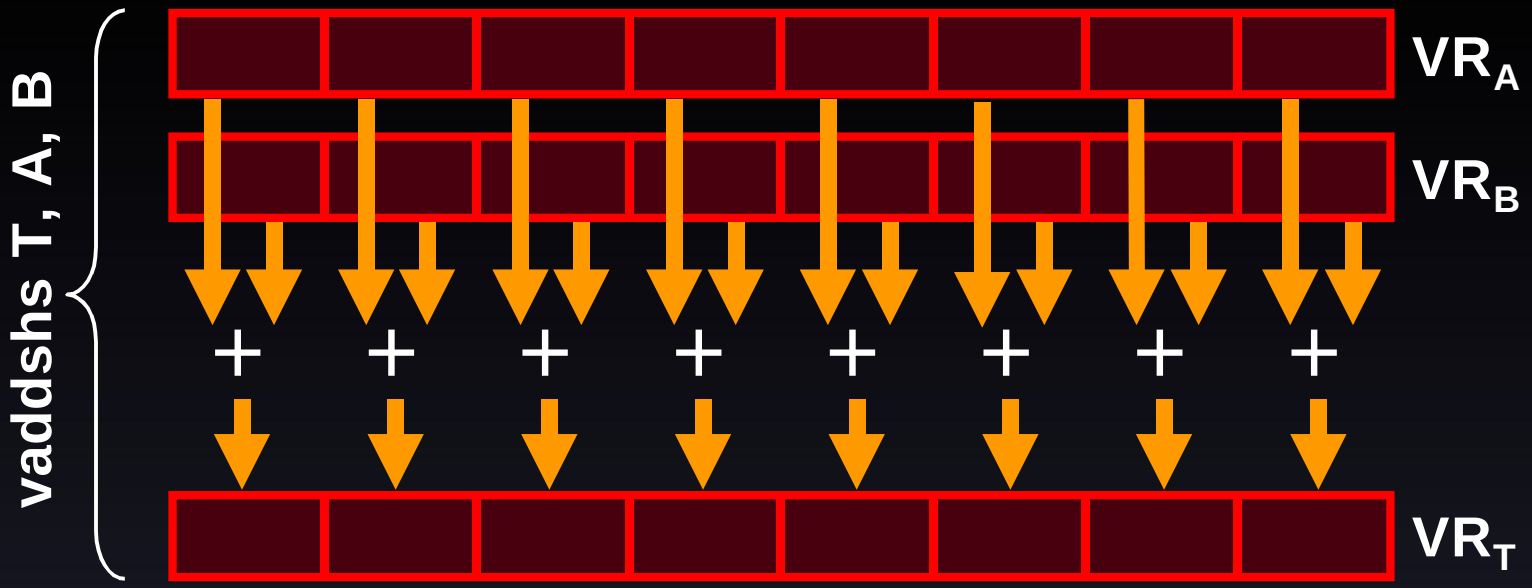


8 signed or unsigned integer halfwords

4 signed or unsigned integer words
or
4 IEEE single-precision floating-point numbers

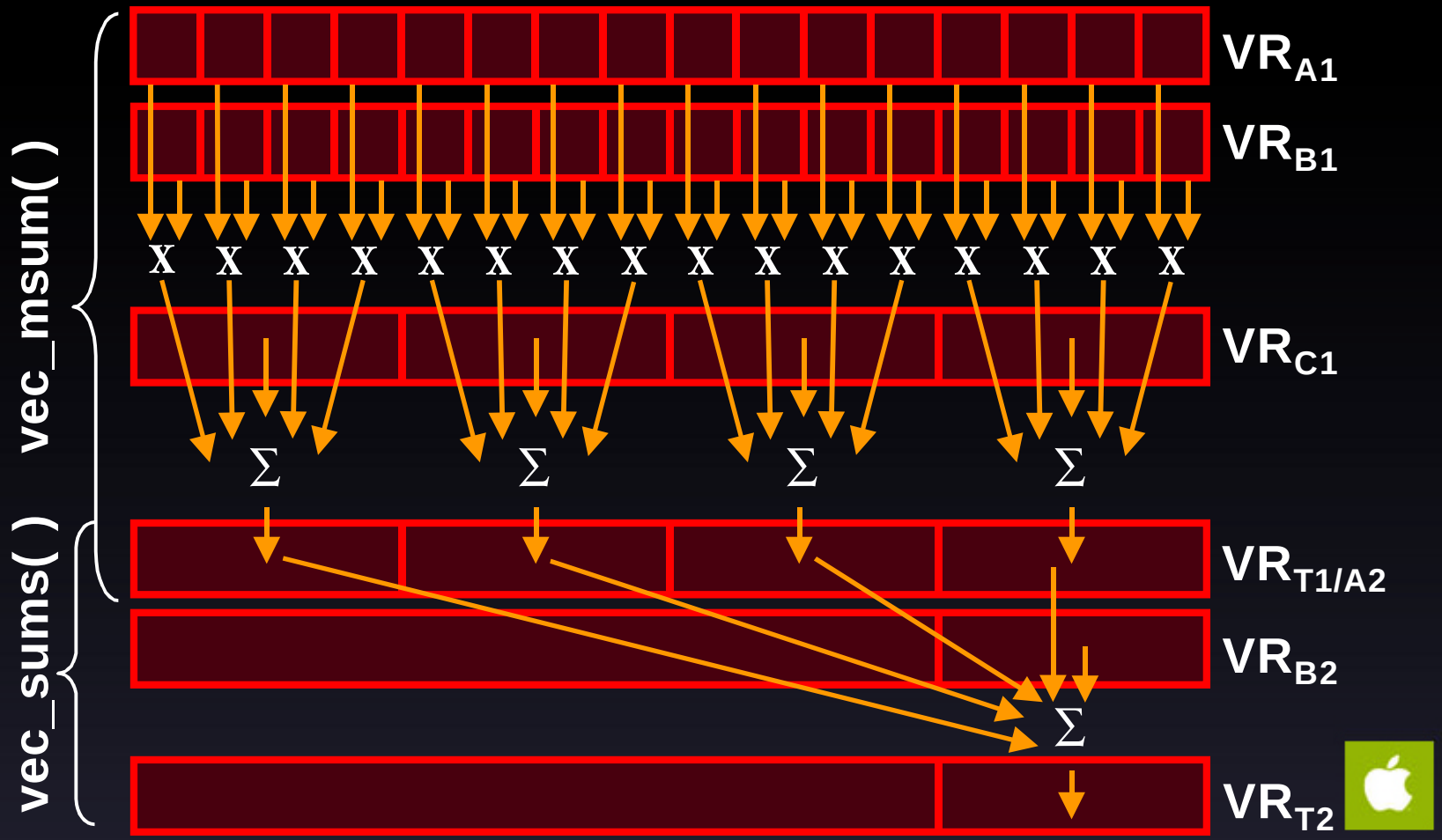
Simple SIMD Example

```
T = vec_adds (A, B);     // vector signed short T, A, B
```



- 8 halfword additions in one instruction
- Saturation arithmetic (clamp to max or min on overflow)

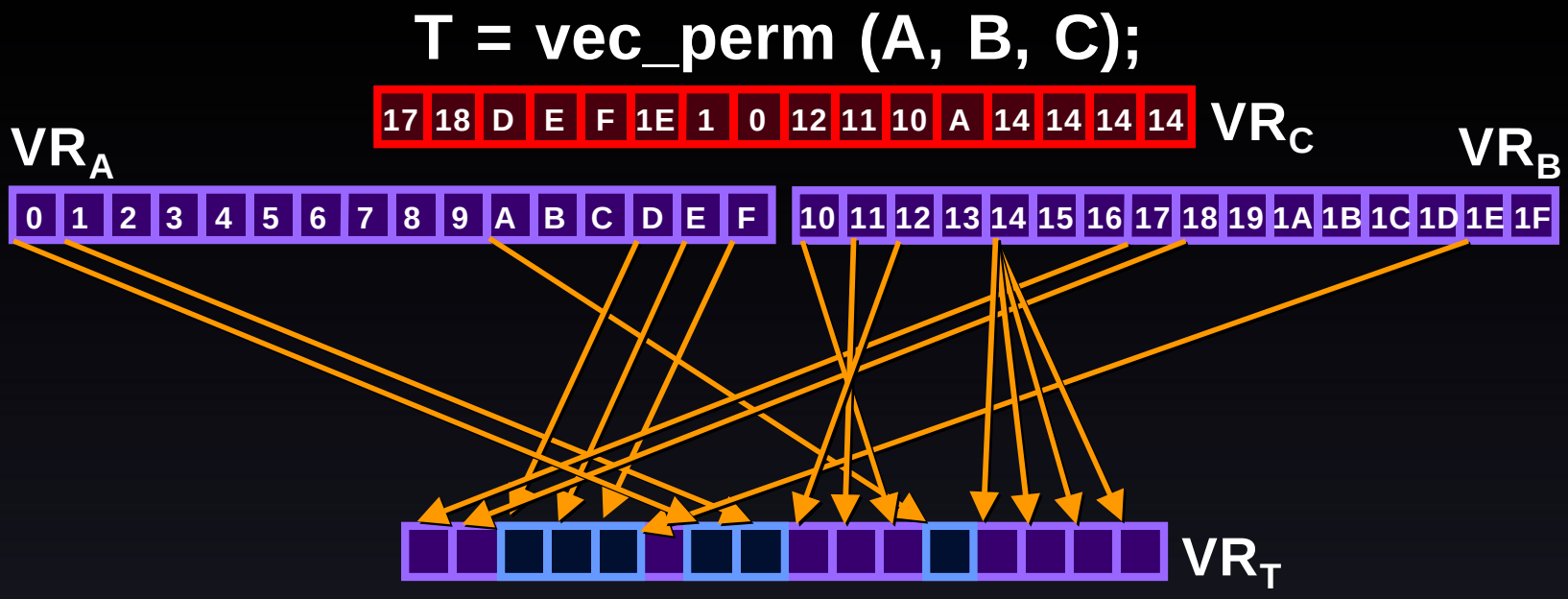
Vector Dot Product



Arithmetic Operations

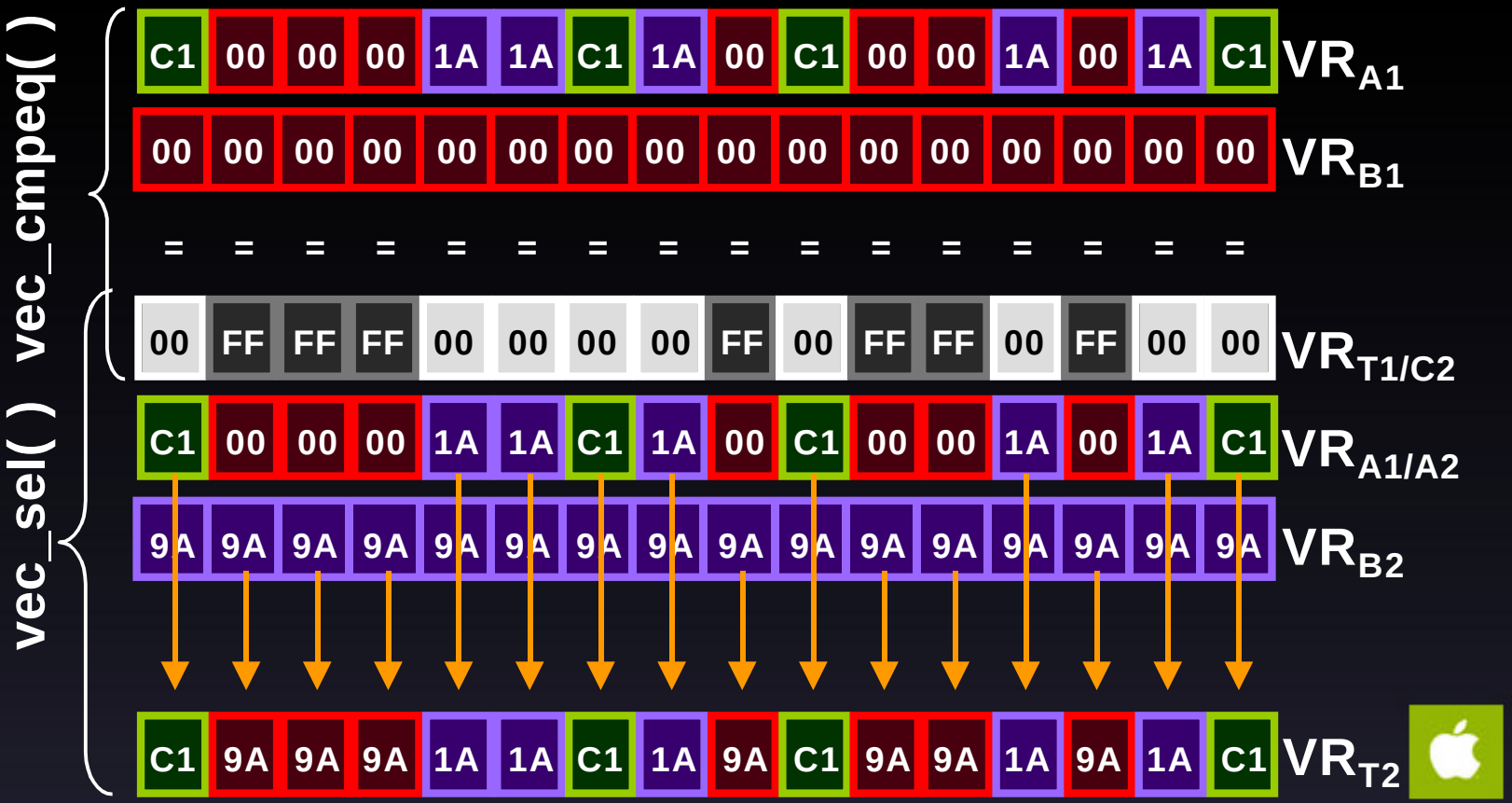
- Add, Subtract, Average
- Multiply, Multiply-add, Multiply-sum
- Logicals (and, andc, or, nor, xor)
- Rotates and shifts
- Compares
- Convert float  fixed (scaled)
- ÷ and √ via Newton-Raphson refinement of reciprocal estimate

Vector Permute



- Arbitrary bytewise data reorganization
- Small table-lookup

Compare and Select

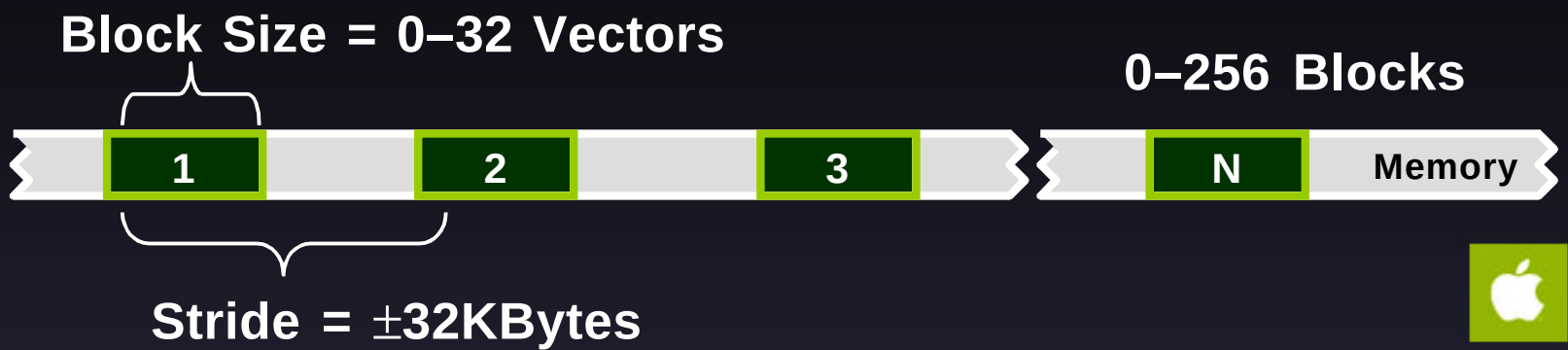


Other AltiVec Instructions

- Load and Store (vector or scalar element)
- Pack, Unpack, and Merge elements
- Splat (element or literal replication)
- Bitwise vector shifts
- Double-vector bytewise shifts

Data Stream Prefetch

- Software directed prefetch into cache
- 4 simultaneous streams
  - Independent and asynchronous
  - Can be non-contiguous



Typical Implementation

- ALL instructions fully-pipelined with single-cycle throughput
  - Simple ops: 1 cycle latency
  - Compound ops: 3–4 cycle latency
- Dual AltiVec instruction issue
  - One arithmetic, one “permute”
- No restriction on issue with scalar instructions

Peak Performance

- Vector operations at 400MHz:
  - Integer
    - 12.8 billion arithmetic ops/sec
    - + 6.4 billion byte crossbar ops/sec
  - Floating-point
    - 3.2 gigaflops
    - + 1.6 billion FP crossbar ops/sec

AltiVec vs. MMX and SSE

- Both SIMD, but AltiVec:
  - Twice the SIMD parallelism
  - 32 128-bit AltiVec registers
    vs.
    8 64-bit MMX and 8 128-bit SSE registers
  - No mode switch or use overhead
  - Generalized Permute instruction
  - Richer set of DSP instructions

AltiVec vs. MMX and SSE

	AltiVec	MMX & SSE
1024 dim. square (bsqr1)	0.5	1.3
1024 dim. multiply (bMpy2)	0.75	1.33
1024 dim. dot product (DotProd)	0.5	2.21
256 point complex FFT (FFT)	<4	6.94
32 tap x 1024 dim. FIR filer (bfir)	0.33	0.95
32 tap x 1024 convolution (conv)	11	85

Clock Cycles per Element

Application Performance

- Maxon perspective shading engine 3x
- Photoshop
  - Rotate CW90 1.6x
  - Gaussian Blur 1 2x
  - Unsharp Mask 1,50,0 1.6x
  - Maximum 5 6x
- Media 100 audio processing code 4x
- Mac OS block move 1.5x

Demos

AltiVec Tools

- Programming Model and API
- Compilers and assemblers
  - Apple's MrC and PPCASM
  - Metrowerks
- Emulator/Trace generator
- Vector Libraries
- MacsBug
- Cycle-accurate simulator (SIM_G4)

Programming in C

- 11 new fundamental packed data types
- AltiVec operators
  - Parse like function calls
  - Specific operators → assembly instructions
  - Generic operators type sensitive
  - sizeof(), a=b, &a, *p, etc.
- Compiler does register allocation, inlining, code scheduling, etc.

AltiVec at Apple

- OS integration—transparent app support
  - OS 8.6 and OS X will support AltiVec
  - Core graphics routines: QD, QT, OpenGL
  - Other OS functions
- Core libraries for developer use
- altivec@apple.com
- http://developer.apple.com/hardware/altivec/

AltiVec Libraries

- Libraries for developer use
  - vBasicOps
  - vectorOps
  - vMathLib
  - vBigNum
  - DSPLib
  - imageLib

AltiVec Summary

- Major architectural extension will make future PowerPCs great media processors
- Programming tools available now
- G3 based development systems available now
- AltiVec based systems early ’00

Call to Action

- Attend other AltiVec sessions
- Download the SDK:
  <developer.apple.com/hardware/altivec>
- Identify data parallelism in your programs
- Vectorize critical sections first without DST
- Add DST instructions last
- Use SIM_G4 to tune performance
- Sign up for the next AltiVec kitchen

More AltiVec Sessions

211: Writing AltiVec Code Performance and data streaming	Hall A1 **Fri., 9:00am**
911: AltiVec Feedback Forum Feedback on tools, support, program	Hall J2 **Fri., 10:15am**
AltiVec Workshops 2+ hours of hands-on with AltiVec	Hall L **Sign up now!**

Q&A

Think different.™

Welcome

To Advance through Presentation
Use Page Up and Page Down Keys