classmate
Date 5. 3. 2020
Page 1

SIMD

NVIDIA tech reports fermi, kepler arch.

VECTOR ARCHITECTURE

vector achitecture grab sets of data elements scattered about memory, place them into large sequential register files, operate on data, in those register files and then disperse the results back to memory

single instruction works on vector of data, which results in dozens of operations on individual elements. these large registers act as compiler controlled buffers, both to hide memory latency and to leverage memory bandwidth.

vector loads and stores are deeply pipelined.

RVV -> vector extensions for RISC-V. loosely based on 40 years-old-cray. (RV64V)

the primary components of ISA of RV64V

vector registers -> each vector register holds a single vector, and RV64V has 32 of them, each 64-bits wide.

atleast 16 read ports and 8 write ports.

Vector functional units:

each unit is fully pipelined and it can start a new operation of every clock cycle. a control unit is needed to detect hazards (structural). for

functional units and data hazards on register accesses.. we have a third unit, load & store unit, the vector memory unit loads or stores a vector to or from memory

the vector loads and stores are fully pipelined. this unit also normally handles scalar loads and stores.

scalar registers likewise provide data as input to vector functional units as well as computed addresses to pass to the vector load and store unit.

there are normal 31 general purpose registers, and 32 floating point registers for the RV64V.

$$Y = a \times X + Y$$

```
    fld    f0, a        # load scalar a
    addi   x28, x5, #256   # local address to load
loop: fld  f1, 0(x5)    # load X[i]
    fmuld  f1, f1, f0    # a x X[i]
    fld    f2, 0(x6)     # load Y[i]
    faddd  f2, f2, f1    # a * X[i] + Y[i]
    fsd    f2, 0(x6)     # store into Y[i]
    addi   x5, x5, #8    # increment index to X
    addi   x6, x6, #8    # increment index to Y
    bne    x28, x5, loop # check if done
```

```
vsetdcfg    4*FP64      # enable 4 DP FP regs.
fld         f0, a       # load scalar.
vld         v0, x5      # load vector x.
vmul        v1, v0, f0  # vector scalar mult.
vld         v2, x6      # load vector y
vadd        v3, v1, v2  # vector-vector add.
vst         v3, x6.     # store the sum.
vdisable                # disable vector registers.
```

8 instr. vs 258 for RV64V.

Vector execution time depends upon

① length of operand vectors.

② structural hazards among the operations.

③ data dependencies

Load and add could happen concurrently.

convoy: set of vector instructions that could potentially execute together, the instruction in a convoy must not contain any structural hazards. chaining allows a vector operation to start as soon as the individual elements of its vector source operands becomes available.

flexible chaining allows a vector instruction to chain to essentially any other active vector instruction.

Chime:- simply a unit of time taken to execute one convoy.

DMIPS

chime approximation ignores some processor specific overheads.

1 convoy {
```
vld  v0, x5      # load vector X
vmul v1, v0, f0  # vector scalar multiply
vld  v2, x6      # load vector Y
vadd v3, v1, v2  # vector-vector add
vst  v3, x6      # store the sum
```

```
vld, vmul
vld, vadd
vst
```

fp-add	:	6 cycles
fp-multiply	:	7 cycles
fp-divide	:	20 cycles
vector-load	:	12 cycles

optimization that either improve the performance or increase the the type of programs that can run well on vector arch.

how can a vector processor execute a single vector faster than one element per clock cycle, multiple elements per clock cycle to improve the perf.

how does a vector processor handle programs where the vector lengths are not the same as maximum vector length. we need an efficient soln. in this common case.

Strip mining -

what happens when there is an if statement inside the coded to be vectorized?

what does a vector processor need from memory system?

how does a vector processor handle multiple dimensional matrices (stride).

if there are sparse matrices, how does a vector processor handle sparse matrices.

(base vector, index vector)

how do you program a vector computer.