CS 451 / 551 / ECE 541

ADVANCED COMPUTER ARCHITECTURE

SESSION no. 18

AUTHORS': MORE PERFORMANCE GROWTH WITH DATA PARALLELISM (SIMD) THAN MULTICORE (MIMD)

- CORES: ADD 2/YEAR (linear)
- VECTORS: DOUBLE WIDTH (LENGTH) EVERY 4 YEARS (EXPONENTIAL)
- BEST TO USE BOTH

BASIC IDEA

- READ SETS OF DATA ELEMENTS INTO VECTOR REGISTERS
- PROCESS VECTOR, UPDATE REGISTERS
- "SCATTER" RESULTS BACK TO MEMORY

University of Idaho

VMIPS · ~~Inspired by~~ INSPIRED BY [redacted]

CRAY 1.

SCIENTIFIC COMPUTATIONS

MATRIX - VECTOR

LINPACK } MATRIX LIBRARIES
EISPACK }

$\underline{Y} = a \cdot \underline{X} + \underline{Y}$ X,Y VECTORS, 64 ELEM,

BENCHMARK: SAXPY, DAXPY

EXECUTION TIME

~1 DATA ITEM / CLOCK CYCLE

· DEPENDS ON: LENGTH OF VECTOR,
STRUCTURAL HAZARDS, DATA DEPENDENCIES

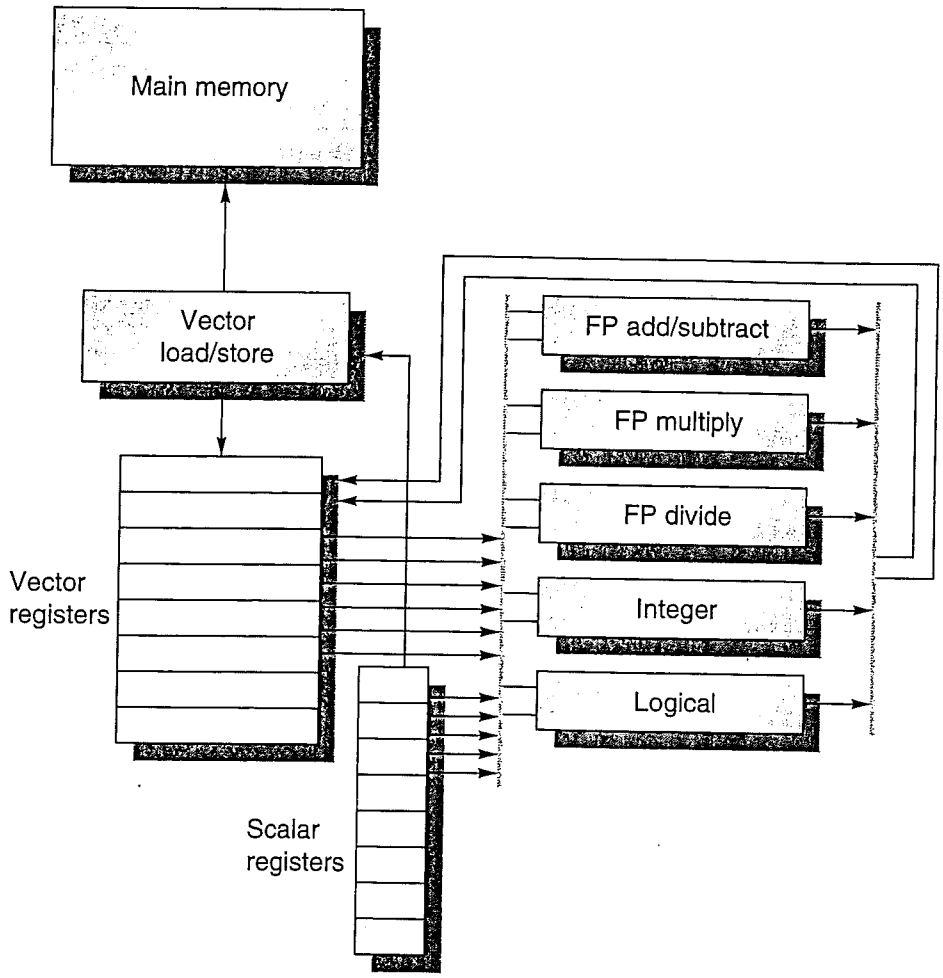


Figure 4.2 The basic structure of a vector architecture, VMIPS. This processor has a scalar architecture just like MIPS. There are also eight 64-element vector registers, and all the functional units are vector functional units. This chapter defines special vector instructions for both arithmetic and memory accesses. The figure shows vector units for logical and integer operations so that VMIPS looks like a standard vector processor that usually includes these units; however, we will not be discussing these units. The vector and scalar registers have a significant number of read and write ports to allow multiple simultaneous vector operations. A set of crossbar switches (thick gray lines) connects these ports to the inputs and outputs of the vector functional units.

5

```
        L.D     F0,a            ;load scalar a
        DADDIU  R4,Rx,#512      ;last address to load
Loop:   L.D     F2,0(Rx)        ;load X[i]
        MUL.D   F2,F2,F0        ;a × X[i]
        L.D     F4,0(Ry)        ;load Y[i]
        ADD.D   F4,F4,F2        ;a × X[i] + Y[i]
        S.D     F4,9(Ry)        ;store into Y[i]
        DADDIU  Rx,Rx,#8        ;increment index to X
        DADDIU  Ry,Ry,#8        ;increment index to Y
        DSUBU   R20,R4,Rx       ;compute bound
        BNEZ    R20,Loop        ;check if done
```

600 instr

Here is the VMIPS code for DAXPY.

```
        L.D     F0,a            ;load scalar a
        LV      V1,Rx           ;load vector X
        MULVS.D V2,V1,F0        ;vector-scalar multiply
        LV      V3,Ry           ;load vector Y
        ADDVV.D V4,V2,V3        ;add
        SV      V4,Ry           ;store the result
```

6 instr

Convoy
Convoy
convoy

P. 268

$$Y = a \cdot X + Y$$

CONVOY

- SET OF VECTOR INSTRUCTIONS THAT CAN EXECUTE TOGETHER
  - GROUP INTO CONVOY, DISPATCH TO PROCESSOR
- NO STRUCTURAL HAZARDS
- CONVOY FINISHES BEFORE NEXT ONE STARTS

CHAINING

- AVOID READ-AFTER-WRITE DEPENDENCIES
  FORWARD RESULTS
- CHIME - TIME TO EXECUTE A CONVOY

3 CONVOYS => ≤ 3 chimes

2 FP OPERATIONS / RESULT

3 CYCLES FOR FLOPS => 1.5 CYCLES/FLOP

. IGNORED: STARTUP TIME OF 2 CYCLES

NOT MUCH OF A BIAS:

$$64 \frac{\text{ELEMENTS}}{\text{VECTOR}} \times 3 \frac{\text{FLOPS}}{\text{ELEMENT}} \times \frac{1 \text{ CLOCK}}{\text{FLOP}}$$

= 192 CLOCKS/VECTOR

2 MORE INSIGNIFICANT

STARTUP LATENCY

FP ADD	6 CYCLES
FP MULT	7 "
FP DIV	20 "
VECTOR LOAD	12 "

BIG QUESTIONS

1. HOW TO EXCEED 1 CLOCK CYCLE/ELEMENT?
2. VECTORIZE A LOOP
   - WHAT IF HAS IF STATEMENTS?
     CONDITIONAL EXECUTION
2. WHAT IF VECTORS NOT 64 ELEMENTS LONG?

4. HOW TO GET MEMORY BW TO MATCH PROCESSOR SPEED?

5. HOW TO HANDLE MULTIDIMENSIONAL MATRICES?

6. HOW TO PROGRAM THIS BEAST?

---

1. > 1 ELEMENT / CYCLE

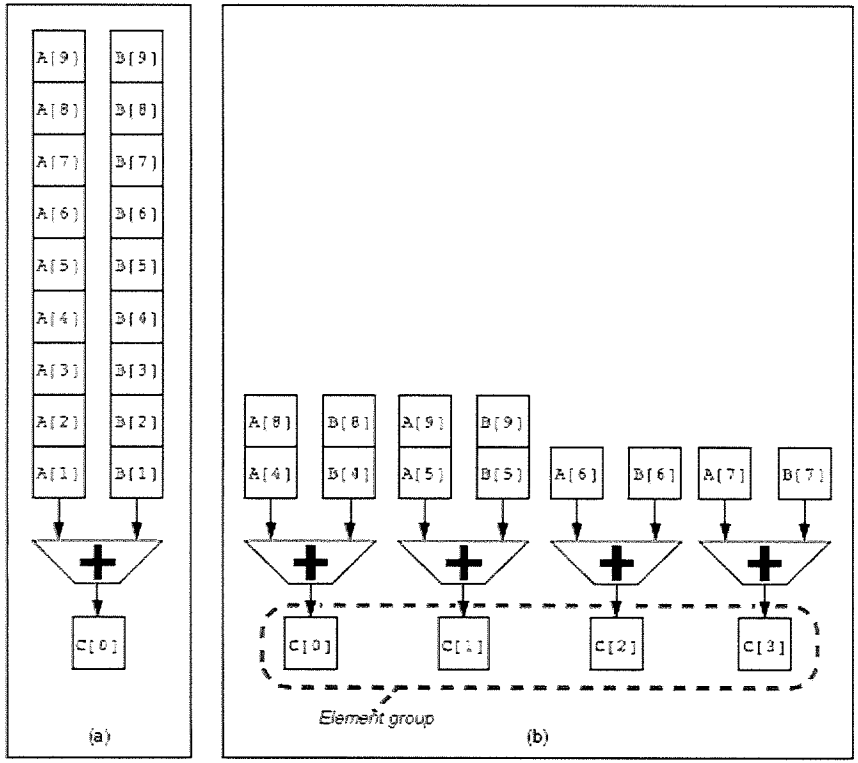
   . MULTIPLE "LANES" - REPLICATED EXECUTION UNITS

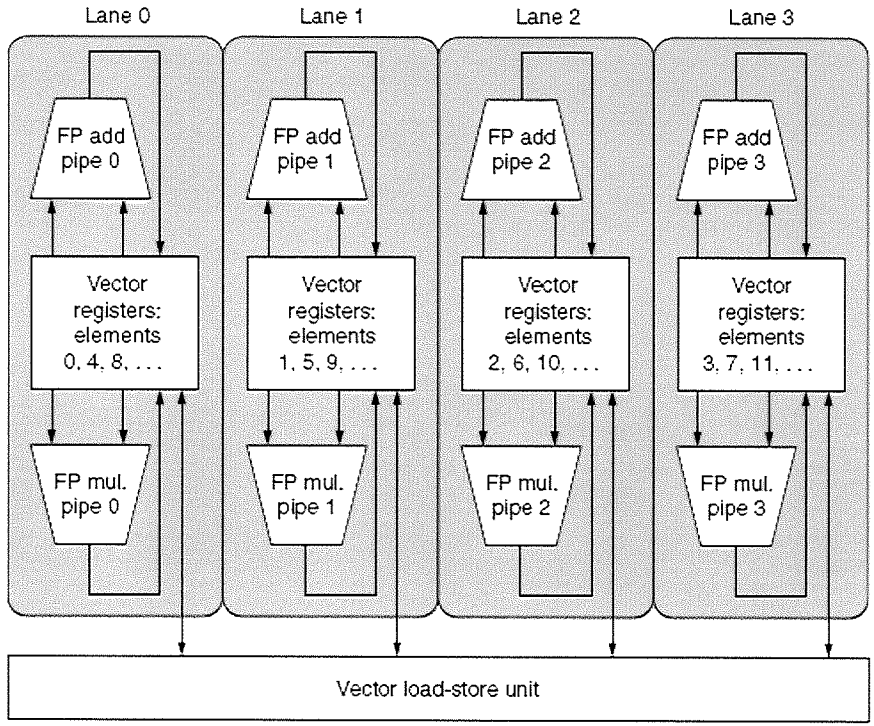
   RESTRICTION: OPERANDS MUST "LINE UP" BETWEEN VECTORS.

   NO COMMUNICATION/SWITCHING BETWEEN LANES.

Multiple Lanes

- Element *n* of vector register *A* is "hardwired" to element *n* of vector register *B*
  - Allows for multiple hardware lanes





2. LOOPS (VECTORS) ≠ 64 IN LENGTH

- ~~RARE~~
- VARIES BETWEEN APPLICATIONS
- MAYBE NOT KNOWN @ COMPILE TIME

```
for(i=0; i<n; i=i+1)
    Y[i] = a * X[i] + Y[i]
```

SOLN: ADD A VECTOR LENGTH REGISTER

P.274 EXAMPLE - UNDERSTAND FOR NEXT TIME!

"STRIP MINING"

DEALING WITH IF STATEMENTS IN LOOP

- CONDITIONAL EXECUTION

```
for(i=0; i<64; i=i+1)
    if(X[i] != 0)
        X[i] = X[i] - Y[i]
```

MASK REGISTER - A BIT FOR EVERY ITERATION THRU LOOP: 1 ⇒ EXECUTE
0 ⇒ DON'T EXECUTE

```
LV    V1,RX      ; V1 ← VECTOR @ [RX]
LV    V2,RY      ; V2 ← VECTOR @ [RY]
LD    F0,#0      ; CLEAR F0
SNEVS.D V1,F0    ; IF V1(i) != F0
                 ; SET VM(i) TO 1
```

Ex

X =	1	5	0	4	0	6	9 ...
Y =	3	6	2	7	4	3	4 ...
VM =	1	1	0	1	0	1	1 ... (BITS)
RES =	-2	-1	0	-3	0	3	5 ...

PRECODE: THE RESULTS OF IF INTO MASK. "IF CONVERSION"

- RESULT OF "IF" ALWAYS BOOLEAN E(0,1)
- MASK = AUXILLIARY DATA STRUCTURE