CS 451 / 551 / ECE 541

ADVANCED COMPUTER ARCHITECTURE

SESSION no. 21

University of Idaho

IN-CLASS EXAM: NOV 21
TAKE HOME FINAL
- PAPERS - DUE MON. OF FINAL EXAM WEEK
  - WILL REVIEW DRAFTS ANY TIME!

FLEXIBILITY FEATURES FOR VECTOR PROCESSORS

- LOOPS (DATASET) ≠ VECTOR LENGTH
  - LENGTH REGISTER
- IF STATEMENTS IN LOOP
  - MASK REGISTER 1 ⇒ DO OP.
    0 ⇒ DON'T
- 2D ADDRESSING
  LD & ST WITH "STRIDE"

University of Idaho

SPARSE MATRICES

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & \end{bmatrix} \cdots \text{GRAPH}$$

$$A = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 9 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 \\ 1 & 6 & 0 & 0 \end{bmatrix} \end{array} \cdot \begin{array}{c} b \\ \begin{bmatrix} 0 \\ 2 \\ 1 \\ 0 \end{bmatrix} \end{array} = \begin{bmatrix} 2\times 9 \\ 0 \\ 0 \\ 2\times 6 \end{bmatrix} = \begin{bmatrix} 18 \\ 0 \\ 0 \\ 12 \end{bmatrix}$$

- MANY ZEROES
- PUTTING ZEROES THRU VECTOR PROCESSOR WASTES TIME & ENERGY

University of Idaho

COMPACT REPRESENTATION

- LIST OF LISTS (PYTHON)
- VALUE - ROW - COLUMN (VRC)

VAL ROW COL

[9 1 0 | 2 0 1 | 3 3 2 | 1 0 3 | 6 1 3]

0 1 | 1 0 | 2 3 | 3 0 | 3 1

for (i=0; i<n; i++)

~~A[k][i]~~

A[K[i]] = A[K[i]] + C[M[i]]

K: VECTOR OF INDICES

M: VECTOR OF INDICES

~~INDEXES~~

(INDEXES)

INDEX VECTOR

K = [[1,0], [0,1], [3,2], [0,3], [1,3]]

K[0] = [1,0] ...

"GATHER" - SCAN K, LOAD CORRESPONDING DATA INTO VECTOR REGISTER

LV Vn, Rk Load Vector @ R

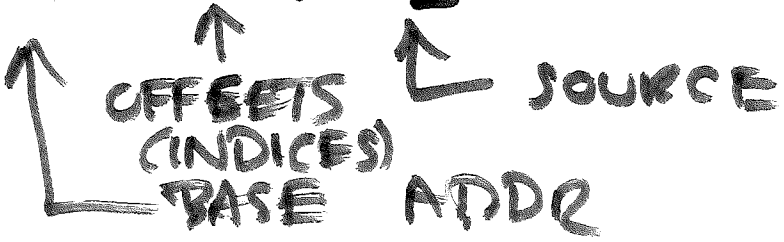
LVI Va (Ra + Vk) - Load Vector register, Ra = BASE, Vk = index

DO VECTOR OPERATION

"SCATTER"

SVI Store vector in memory

SVI (R1 + V2) V_1



REVIEW VMIPS INSTRUCTIONS

SIMD INSTRUCTION SET EXTENSIONS FOR MULTIMEDIA

- ENHANCE INSTRUCTION SET WITH "MINI VECTOR" INSTRUCTIONS.
- NOT AS COMPLEX AS VECTOR PROCESSOR

EX INTEL: ADVANCED VECTOR EXTENSIONS (AVX)

- 256-BIT REGISTERS
- PACK 4 64-BIT OPERANDS INTO REG.

VADDPD - VECTOR ADD ON PACKED DOUBLE PRECISION F.P.

NOT A LOT OF "VECTOR ADVANTAGE"

BUT -

- EASY TO BUILD INTO CHIP
- DON'T NEED TO TREAT AS
  - I/O DEVICE
  - CO-PROCESSOR. EX GPU

- ADD TO UNUSED ADDRESS SPACE OF PROCESSOR
- TRAP INSTRUCTION TO INVOKE

INTERRUPT

- SUPERVISOR MODE. SAVE STATE
- GO TO IVEC.

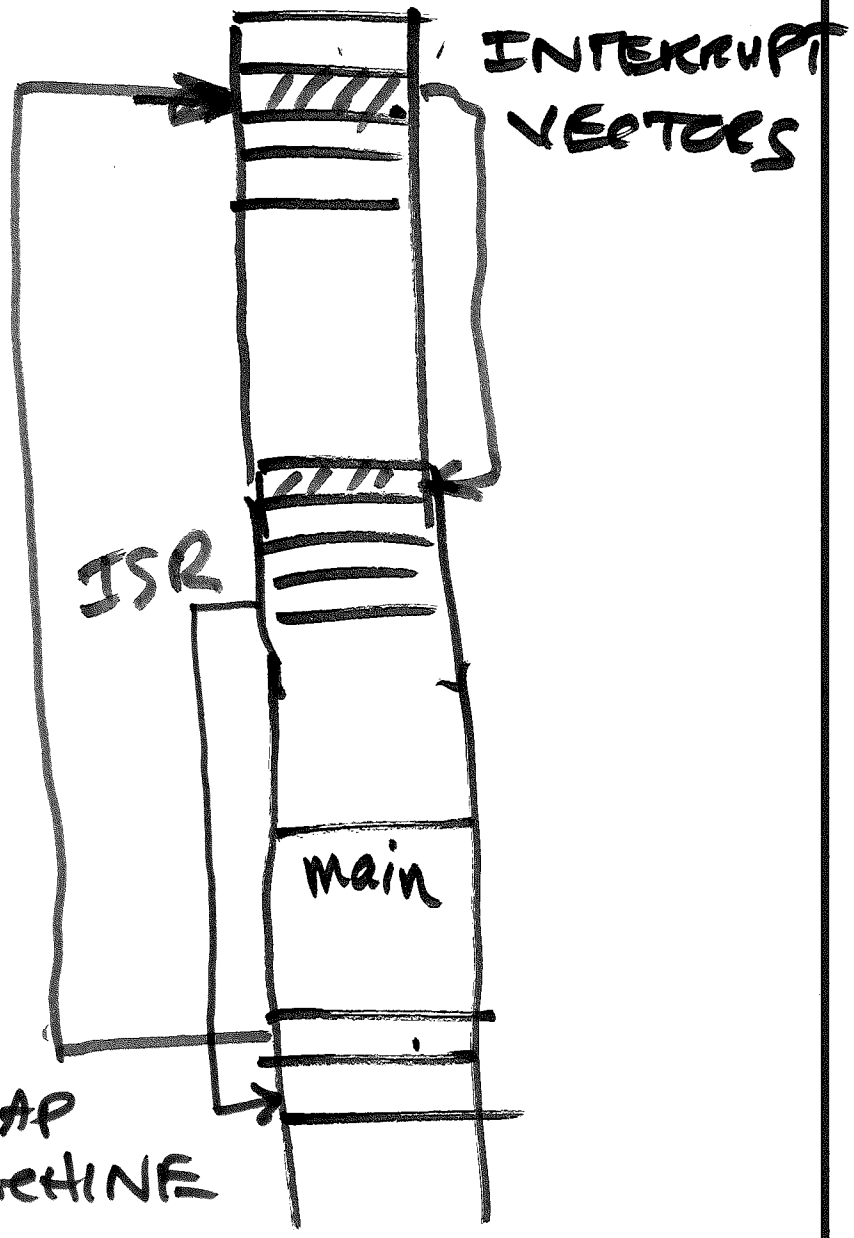
TRAP

- SAME IDEA
- SOFTWARE TRIGGERED

ex

FADD - TRIGGER A TRAP.

VADD - TRIGGER A TRAP TO VECTOR MACHINE



PROGRAMMING!

HOW DO I PROGRAM MY (MEDIA INSTRUCTION, GPU, VECTOR PROCESSOR) TO GET THE SPEED ADVANTAGE?

1) COMPILER THAT "KNOWS" HOW TO USE NEW FEATURES.

- OFTEN - VENDOR - SPECIFIC
- SOMETIMES: "PRAGMAS"
  COMPILER DIRECTIVES (HINTS)
  "YOU CAN VECTORIZE THIS LOOP"

"VECTORIZING COMPILER"

Benchmark name	Operations executed in vector mode, compiler-optimized	Operations executed in vector mode, with programmer aid	Speedup from hint optimization
BDNA	96.1%	97.2%	1.52
MG3D	95.1%	94.5%	1.00
FLO52	91.5%	88.7%	N/A
ARC3D	91.1%	92.0%	1.01
SPEC77	90.3%	90.4%	1.07
MDG	87.7%	94.2%	1.49
TRFD	69.8%	73.7%	1.67
DYFESM	68.8%	65.6%	N/A
ADM	42.9%	59.6%	3.60
OCEAN	42.8%	91.2%	3.92
TRACK	14.4%	54.6%	2.52
SPICE	11.5%	79.9%	4.06
QCD	4.2%	75.1%	2.15

Figure 4.7 Level of vectorization among the Perfect Club benchmarks when executed on the Cray Y-MP [Vajapeyam 1991]. The first column shows the vectorization level obtained with the compiler without hints, while the second column shows the results after the codes have been improved with hints from a team of Cray Research programmers.

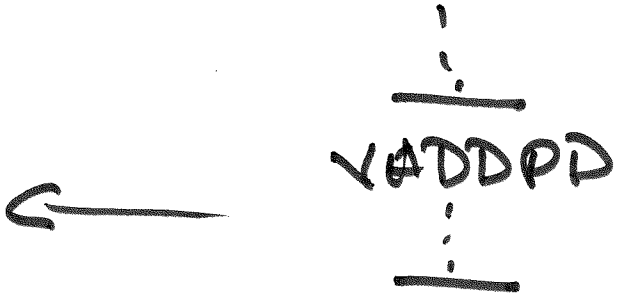
AVX Instruction	Description
VADDPD	Add four packed double-precision operands
VSUBPD	Subtract four packed double-precision operands
VMULPD	Multiply four packed double-precision operands
VDIVPD	Divide four packed double-precision operands
VFMADDPD	Multiply and add four packed double-precision operands
VFMSUBPD	Multiply and subtract four packed double-precision operands
VCMPxx	Compare four packed double-precision operands for EQ, NEQ, LT, LE, GT, GE, ...
VMOVAPD	Move aligned four packed double-precision operands
VBROADCASTSD	Broadcast one double-precision operand to four locations in a 256-bit register

Figure 4.9 AVX instructions for x86 architecture useful in double-precision floating-point programs. Packed-double for 256-bit AVX means four 64-bit operands executed in SIMD mode. As the width increases with AVX, it is increasingly important to add data permutation instructions that allow combinations of narrow operands from different parts of the wide registers. AVX includes instructions that shuffle 32-bit, 64-bit, or 128-bit operands within a 256-bit register. For example, BROADCAST replicates a 64-bit operand 4 times in an AVX register. AVX also includes a large variety of fused multiply-add/subtract instructions; we show just two here.

University of Idaho

2) DROP BACK TO ASSEMBLY CODE

```
float vadd (*v1,*v2)
{
    float *v3;
    :
    :
    return v3
}
```



CAREER TIP: PROJECT STOPPERS

- ASSEMBLY LANGUAGE
- HARDWARE

University of Idaho

3) LIBRARIES — PRE-BUILT

PROGRAMMING GPU's

- OFTEN CO-PROCESSORS
- HYBRID - MIMD & SIMD
- THINKING IN PARALLEL
  - C/C++ DON'T HANDLE WELL
  - CUDA - Compute Unified Device Architecture NVIDIA (Wikipedia)
    - Extensions to C++
  - Open GL - Graphics
  - Open CL - Heterogenious

University of Idaho

PARALLEL PROCESSORS - MIMD

THREAD PARALLELISM

- REQUEST - RESPONSE
- INDUSTRIAL CONTROL
- ROUTERS, SERVERS (RESTFUL)
  "DEMAND - RESPONSE"

THREADS VS PROCESSES

PROCESS - O/S LEVEL

- CONTEXT, PRIVATE
- MEMORY, PRIVATE

SWITCH - 100's - 1,000,000's OF INSTRUCTIONS

THREADS

- PROGRAM LEVEL
- SHARED MEMORY
  - SOME CONTEXT SHARED
- SWTICH - 1 - 100 INSTRUCTIONS

PERFORMANCE ENHANCEMENT

- NO HDWE SUPPORT — ALL BUSY
- HDWE SUPPORT - BUSIER!

ENTRY INTO MULTICORE