Parallel Processing SIMD, Vector and GPU's

EECS4201 Fall 2016 York University

1

Introduction

- Vector and array processors
- Chaining
- GPU

2

Flynn's taxonomy

- **SISD**: **S**ingle instruction operating on **S**ingle Data
- **SIMD**: **S**ingle instruction operating on **M**ultiple Data
- **MISD**: **M**ultiple instruction operating on **S**ingle Data
- **MIMD**: **M**ultiple instructions operating on **M**ultiple Data

3

SIMD

- SIMD architectures can exploit significant data-level parallelism for:
  - matrix-oriented scientific computing
  - media-oriented image and sound processors
- SIMD is more energy efficient than MIMD
  - Only needs to fetch one instruction per data operation
  - Makes SIMD attractive for personal mobile devices
- SIMD allows programmer to continue to think sequentially

4

Vector vs. Array Processors

- **Array processors** same instruction operating on many data elements at the same time (space)
- **Vector processors** Same instruction operating on many data in a pipeline fashion (what is the difference between this and regular pipelined processors?)

5

Vector vs. Array processors

		Array Processor				Vector Processor				
		PE0	PE1	PE2	PE3	LD	ADD	MUL	ST	SISD
LD	V1←A[]	LD0	LD1	LD2	LD3	LD0				LD0
ADD	V2 ←V1+2	AD0	AD1	AD2	AD3	LD1	AD0			AD0
MUL	V3 ←V2 ×2	ML0	ML1	ML2	ML3	LD2	AD1	ML0		ML0
St	A[] ←V3	ST0	ST1	ST2	ST3	LD3	AD2	ML1	ST0	ST0
							AD3	ML2	ST1	LD1
								ML3	ST2	AD2
									ST3	ML1
										ST1
										LD2
										AD2
										ML2
										ST2
										LD3
										AD3
										ML3

6

Vector processors

- Energy efficient: we fetch only one instruction to perform many operations.
- Can have much deeper pipelines, no interlocks, no dependence between the vector elements
- Stride may not be 1
- Performance depends on what kind of parallelism in your program.

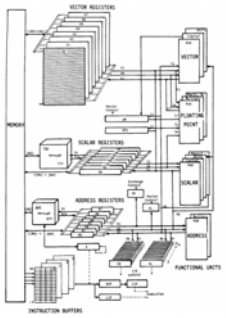
7

Vector Processors

- Cray-1 was the first commercially successful vector processor

8

Cray-1



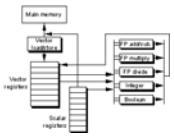
9

VMIPS

- Example architecture: VMIPS
  - Loosely based on Cray-1
  - Vector registers
    - Each register holds a 64-element, 64 bits/element vector
    - Register file has 16 read ports and 8 write ports
  - Vector functional units
    - Fully pipelined
    - Data and control hazards are detected
  - Vector load-store unit
    - Fully pipelined
    - One word per clock cycle after initial latency
  - Scalar registers
    - 32 general-purpose registers
    - 32 floating-point registers

10

VMIPS



11

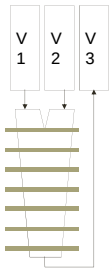
VMIPS Instructions

ADDVV.D	V1,V2,V3	add two vectors
ADDVS.D	V1,V2,F0	add vector to a scalar
LV	V1,R1	vector load from address
SV	R1,V1	Vector store at R1
MULVV.D	V1,V2,V3	vector multiply
DIVVV.D	V1,V2,V3	Vector div (element by element)
LVWS	V1,(R1,R2)	Load vector from R1, stride=R2
LVI	V1,(R1+V2)	Load V1 with elements at R1+V2(i)
CVI	V1,R1	create an index vector in V1 (0, R1, 2R1,3R1,…
SEQVV.D	V1,V2	Compare elements V1,V2 0 or 1in VM EQ, NE, GT, …
MVTM	VM,F0	Move contents of F0 to vec. mask reg.
MTCI	VLR,R1	Move r1 to vector length register

12

Vector Processing

- ADDV V3, V1, V2
- After an initial latency (depth of pipeline) we get one result per cycle.
- We can do this with a simple loop, what is the difference?



13

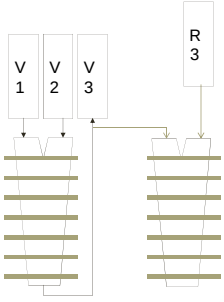
Vector Code

- Chaining: No need to wait until the vector register is loaded, you can start after the first element is ready.
-

14

Chaining

- Chaining
- ADD V3,V2,V1
- MUL V4, V3, R
- Chaining: No need to wait until the vector register is loaded, you can start after the first element is ready.



15

Vector ADD scalar A=B+C

•	R1 ← A	1
•	R2 ← B	1
•	R3 ← C	1
• X:	LD R4, 0(R2)	11
•	LD R5, 0(R3)	11
•	R2++; R3++;R1++	3
•	ADD R6, R4, R5	4
•	ST R6, -8(R1)	11
•	Branch X	1

16

Vector Execution Time

- Execution time depends on three factors:
  - Length of operand vectors
  - Structural hazards
  - Data dependencies
- VMIPS functional units consume one element per clock cycle
  - Execution time is approximately the vector length
- *Convey*
  - Set of vector instructions that could potentially execute together (could be more than one instruction)

17

Example

- Consider the following example:
- For (i=0;i<50.i++)
- c[i] = (a[i] + b[i])/2
- Sequence of improvements from in order execution with one bank to chained vector processor with multiple banks

18

Assembly Code

```
Initialize registers R0, R1, R2, R3
LOOP    LD      R4, 0(R1)    11
        LD      R5, 0(R2)    11
        ADD     R6,R4,R5     4
        SR      R6, R6, 1    1
        ST      R6, 0(R3)    11
        ADDI    R1,R1,4      1
        ADDI    R2, R2, 4    1
        ADDI    R3, R3, 4    1
        ADDI    R0, R0, -1   1
        BEQZ    R0, LOOP     2   = 44*50
```

19