Arquiteturas de Computadores

Computadores vetoriais

Fontes dos slides: Livro Patterson e Hennessy, Quantitative Approach

Introdução

- Arquiteturas Single Instruction Multiple Data podem explorar paralelismo de dados para:
  - Computação científica utilizando matrizes
  - Processamento de imagem e som
- SIMD é mais eficiente que MIMD
  - Só precisa buscar uma instrução para operação de dados
  - SIMD é atrativo para dispositivos móveis
- SIMD permite que programador continue a pensar sequencialmente

Paralelismo SIMD

- Arquiteturas vetoriais
- Extensões SIMD
- Graphics Processor Units (GPUs)

- Para processadores x86:
  - Esperam-se dois cores adicionais por chip por ano
  - Largura SIMD deve dobrar em 4 anos
  - Aceleração potencial de SIMD ser o dobro de MIMD!

Arquiteturas vetoriais

- Ideia básica:
  - Lê conjuntos de elementos de dados em “registradores vetoriais”
  - Opera nestes registradores
  - Armazena resultados na memória
- Registradores são controlados pelo compilador
  - Utilizado para esconder a latência de memória

VMIPS

- Exemplo de arquitetura: VMIPS
  - Baseada levemente no Cray-1
  - Registradores vetoriais
    - Cada registrador armazena um vetor de 64 elementos com 64 bits/elemento
    - O conjunto de registradores possui 16 portas de leitura e 8 portas de escrita
  - Unidades funcionais vetoriais
    - Totalmente pipelined
    - Conflito de dados e controle são detectados
  - Unidade de carga/armazenamento vetorial
    - Totalmente pipelined
    - Uma palavra por ciclo de relógio após a latência inicial
  - Registradores escalares
    - 32 registradores de propósito geral
    - 32 registradores de ponto flutuante

Instruções VMIPS

- ADDVV.D: soma dois vetores
- ADDVS.D: soma um vetor com escalar
- LV/SV: carrega e armazena vetor de um endereço
- Exemplo: DAXPY

```
L.D    F0,a                ; armazena escalar a
LV     V1,Rx               ; carrega vetor X
MULVS.D    V2,V1,F0        ; multiplica vetor por escalar
LV     V3,Ry               ; carrega vetor Y
ADDVV      V4,V2,V3        ; soma
SV     Ry,V4               ; armazena o resultado
```

- Requer 6 instruções vs. quase 600 para MIPS

Tempo de execução VMIPS

- Tempo de execução depende de 3 fatores:
  - Tamanho dos vetores
  - Conflitos estruturais
  - Dependências de dados
- As unidades funcionais consomem um elemento por ciclo de relógio
  - Tempo de execução é aproximadamente o tamanho do vetor
- *Convey*
  - Conjunto de instruções vetoriais que podem potencialmente serem executadas em conjunto

Chimes

- Sequências com dependência de dados leitura-depois-de-escrita podem estar no mesmo convey através de *chaining*
- *Chaining*
  - Permite que uma operação vetorial comece logo que os elementos individuais do operando vetorial fonte estejam disponíveis
- *Chime*
  - Unidade de temo para executar um convey
  - *m* conveys executam em *m* chimes
  - Para um vetor de tamanho *n*, necessitam-se *m* x *n* ciclos de relógio

Exemplo

```
LV        V1,Rx      ;carrega vetor X
MULVS.D   V2,V1,F0   ;multiplica vetor por escalar
LV        V3,Ry      ;carrega vetor Y
ADDVV.D   V4,V2,V3   ;soma dois vetores
SV        Ry,V4      ;armazena a soma
```

Convoys:

1	LV	MULVS.D
2	LV	ADDVV.D
3	SV	

3 chimes, 2 ops FP ops por resultado, ciclos por FLOP = 1.5

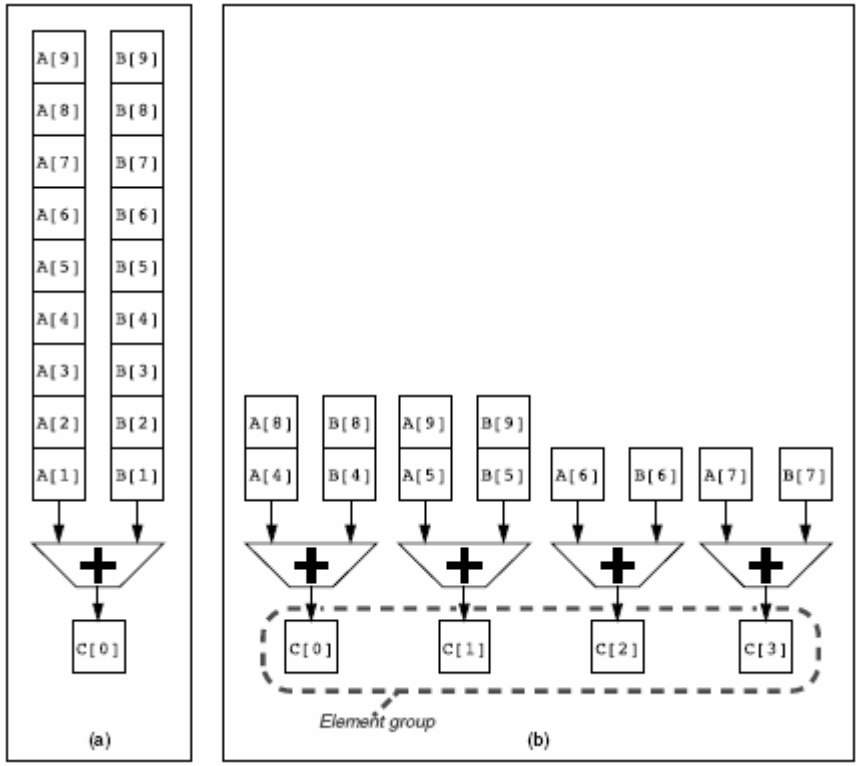
Para vetores com 64 elementos, necessita-se de 64 x 3 = 192 ciclos de relógio

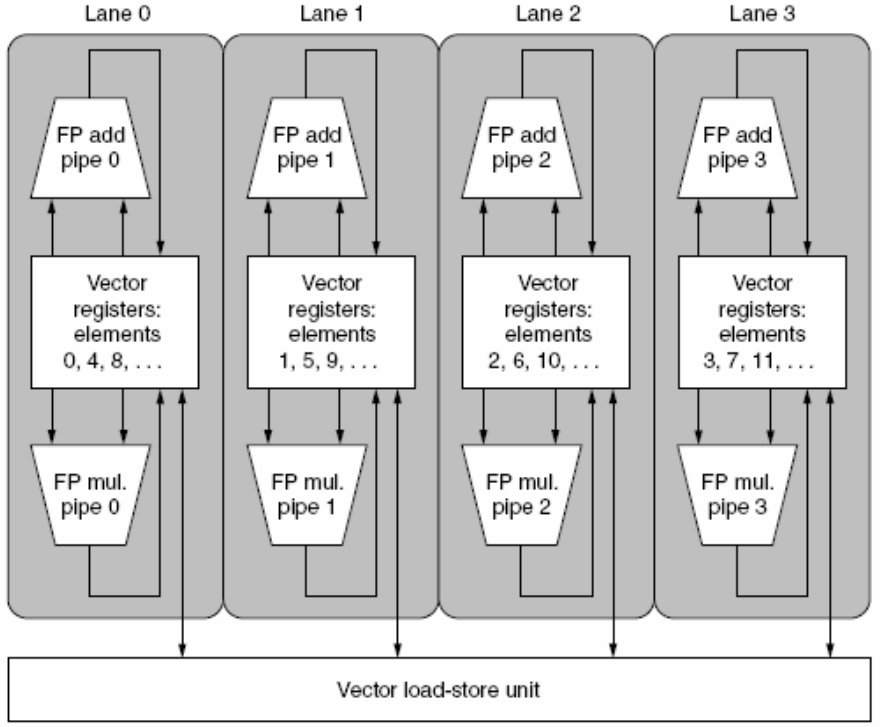
Desafios

- Tempo de início
  - Latência de unidade funcional vetorial
  - Assumindo o mesmo que Cray-1
    - Floating-point add => 6 ciclos de relógio
    - Floating-point multiply => 7 ciclos de relógio
    - Floating-point divide => 20 ciclos de relógio
    - Vector load => 12 ciclos de relógio
- Melhorias:
  - > 1 elemento por ciclo de relógio
  - Vetores com tamanho diferente de 64
  - Instruções IF no código vetorial
  - Otimizações na memória para apoiar processadores vetoriais
  - Matrizes multidimensionais
  - Matrizes esparsas
  - Programação de um computador vetorial

Múltiplas faixas

- Elemento *n de um registrador vetorial A é* “hardwired” ao elemento *n de um registrador vetorial B*
  - Permite múltiplas faixas de hardware

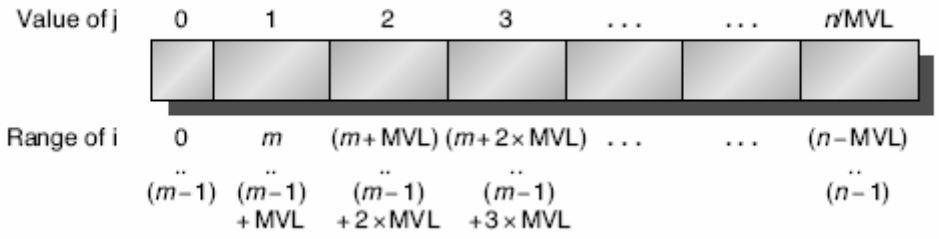




Tamanho do registrador vetorial

- Tamanho do vetor não é conhecido em tempo de compilação?
- Use Vector Length Register (VLR)
- Use extração de faixas dos vetores que tem comprimento maior:

```
low = 0;
VL = (n % MVL); /*find odd-size piece using modulo op % */
for (j = 0; j <= (n/MVL); j=j+1) { /*outer loop*/
      for (i = low; i < (low+VL); i=i+1) /*runs for length VL*/
           Y[i] = a * X[i] + Y[i] ; /*main operation*/
      low = low + VL; /*start of next vector*/
      VL = MVL; /*reset the length to maximum vector length*/
}
```



Registradores vetoriais de máscara

- Considere:

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

- Use registradores vetorial de máscara para “desabilitar” elementos:

```
LV          V1,Rx       ;load vector X into V1
LV          V2,Ry       ;load vector Y
L.D         F0,#0       ;load FP zero into F0
SNEVS.D     V1,F0       ;sets VM(i) to 1 if V1(i)!=F0
SUBVV.D     V1,V1,V2    ;subtract under vector mask
SV          Rx,V1       ;store the result in X
```

- Taxa GFLOPS diminui !

Bancos de memória

- O sistema de memória deve ser projetado para suportar a grande largura de banda para carregamento e armazenamento de vetores
- Acesso espalhado em vários bancos:
  - Controla endereços de banco independentemente
  - Carrega ou armazena palavras não sequenciais
  - Suporta múltiplos processadores vetoriais compartilhando a mesma memória
- Exemplo:
  - 32 processadores, cada um gerando 4 loads e 2 stores/ciclo
  - Tempo de ciclo do processador é 2.167 ns, e da SRA é 15 ns
  - Quantos bancos são necessários?

Passo

- Considere:

```
for (i = 0; i < 100; i=i+1)
        for (j = 0; j < 100; j=j+1) {
                A[i][j] = 0.0;
                for (k = 0; k < 100; k=k+1)
                A[i][j] = A[i][j] + B[i][k] * D[k][j];
        }
```

- Precisa vetorizar a multiplicação de linhas de B com colunas de D
- Use *non-unit stride*
- Ocorre conflito no banco quando o mesmo banco é acessado mais rapidamente que o tempo de ocupação:
  - #banks / LCM(stride,#banks) < bank busy time

Programando Arquiteturas Vetoriais

- Compiladores podem prover retorno aos programadores
- Programadores podem prover dicas para o compilador

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