Module-2
CSEN 3104
Lecture 12

Dr. Debranjan Sarkar

Vector Processing

Problems of Vector Processing

- Vector lengths do not often correspond to the length of the vector registers - except by plan
- For shorter vectors, we can use a vector length register applied to each vector operation
- For longer vectors, we can split the long vector into multiple vectors (of equal, or of maximum plus smaller lengths)
- The process is called strip-mining

Strip Mining

- Strip mining is a technique to generate code in vector processors so that vector operation is possible for vector operands whose size is not equal to the size of vector registers
- Vector-length register (VLR) controls the length of any vector operation, including a vector load or store
- Length of any vector operation cannot be > the length of vector registers

```
      do 10 i=1,n                for (i=0;i<n;i++)
10    Y(i)=a*X(i)+Y(i)           y[i] = a * x[i] + y[i];
```

- As the value of n is known only during runtime, strip mining technique tackles the situation when n > Max. Vector Length (MVL)
- Strip mining technique generates code so that each vector operation is done for a size $\leq$ MVL

Strip Mining

- Suppose n = 135, MVL = 64
- 1st loop iterates for (n mod MVL) = 7 times
- 2nd and 3rd loop iterates for MVL = 64 times each
- This has a loop overhead

```
   low = 1
   VL = (n mod MVL)  /*find the odd size piece*/
   do 1 j = 0, (n / MVL)  /*outer loop*/
         do 10 i = low,low+VL-1  /*runs for length VL*/
         Y(i) = a*X(i) + Y(i)  /*main operation*/
10       continue
         low = low+VL  /*start of next vector*/
         VL = MVL  /*reset the length to max*/
1        continue
```

```
low = 0;
vl = n mod mvl;
for(j = 0; j <= n/mvl; j++)
{for(i = low; i < low+vl; i++)
{y[i] = a*x[i] + y[i];
}
low = low + vl;
vl = mvl;
}
```

Solution of Data Hazard in pipeline: data forwarding

- The data obtained at the Execution phase of the first instruction may be forwarded to the operand fetch unit of the 2nd instruction
- Instruction I: ADD R1, R2, R3
- Instruction J: SUB R4, R1, R6
- After data forwarding, the delay is reduced to 1 cycle

	1	2	3	4	5	6	7
IF	I	J					
ID		ADD	SUB				
OF			R_2, R_3	Delay	$R_{2+}R_3$, R_6		
EX				R_2+R_3		$R_{2+}R_3$-R_6	
WB					Res to R_1		

Delay Reduced to 1 cycle
By Data Forwarding

Addressing data hazards: Vector Chaining

- Vector Chaining allows the results of one vector operation to be directly used as input to another vector operation
- Vector Chaining is the equivalent to Data forwarding in pipelined processors
- Vector Chaining is used in case of data dependency among vector instructions.
- Consider the simple vector sequence

```
MULTV V1,V2,V3
ADDV   V4,V1,V5
```

- As the ADDV instruction is dependent on the MULTV instruction, these two instructions are to be put into two separate convoys

Vector Chaining

- A convoy is a set of vector instructions that can potentially execute together.
- If the vector register, V1 in this case, is treated not as a single entity but as a group of individual registers, then the ideas of forwarding can be conceptually extended to work on individual elements of a vector.
- This insight, which will allow the ADDV instruction to start earlier in this example, is called chaining
- Chaining allows a vector operation to start as soon as the individual elements of its vector source operand become available
- The results from the first functional unit in the chain are “forwarded” to the second functional unit
- Only structural hazards cause separate convoys as true dependences are handled via chaining in the same convoy

Vector Chaining

- Earlier, implementations of chaining worked just like forwarding
- But this restricted the timing of the source and destination instructions in the chain
- Recent implementations use flexible chaining
- Flexible chaining allows a vector instruction to chain to essentially any other active vector instruction, assuming that no structural hazard is generated
- Flexible chaining requires more read and write ports for the vector register file

Vector Stride

- Recapitulation of row-major and column-major order
- In a **row-major order**, the consecutive elements of a row reside next to each other
- In a **column-major order**, the consecutive elements of a column reside next to each other
- Matrix multiplication Example:

```
        do 10 i = 1,100
        do 10 j = 1,100
            A(i,j) = 0.0
            do 10 k = 1,100
10          A(i,j) = A(i,j) + B(i,k) * C(k,j)
```

Vector Stride

- This loop can be strip-mined as a vector multiplication
- Suppose adjacent elements are not sequential in memory
- Each row of B would be first operand and each column of C would be second operand
- For memory organization as column major order, B’s elements would be non-adjacent
- Stride is distance (uniform) between the non-adjacent elements in memory that are to be merged into a single vector i.e. will be adjacent in a vector register
- The unit stride is easiest to handle
- Caches deal with unit stride, and behave badly for non-unit stride

Vector Stride

- Recapitulation of interleaving within memory bank
- Memory banks are used to reduce load/store latency.

- Non-unit strides can cause major problems for the memory system, which is based on unit stride (i.e. all the elements are one after another in different interleaved memory banks)
- To account for non-unit stride, most systems have a stride register that the memory system can use for loading elements of a vector register
- To access non sequential memory elements, LVWS (load vector with stride) and SVWS (store vector with stride) instructions are used

Performance optimization in vector processors

- Memory banks are used to reduce load/store latency.
- Strip mining is used to generate code so that vector operation is possible for vector operands whose size is less than or greater than the size of vector registers.
- Vector chaining - the equivalent of forwarding in vector processors - is used in case of data dependency among vector instructions.
- Special scatter and gather instructions are provided to efficiently operate on sparse matrices.

Thank you