efficient control.

Issue as many instructions per cycle as you can

Terminology:

Efficient processing translates into instruction issuing

Vector processors are motivated by the first rule of thumb of

the type of inquiries we will make in the rest of the semester.

We will examine each of the 4 main areas of concern to show

. We will consider a very standard approach to the problem of

Vector processors
The EX stage could also be pipelined.

- Write result (WR)
- Execution of operation (EX)
- Fetch operands (OF)
- Compute addresses of operands (AD)
- Decode instruction (DC)
- Fetch instruction (IF)

but the concept is identical.

Consider the stages needed to execute an instruction.

Pipelining
Institution

output of one is not needed as input to a later
Suppose they do not depend on each other, i.e., the

INSTM

Instructions INSTM, INSTM2, INSTM3, INSTM4, INSTM5
Suppose a code fragment consists of a series of

•
at the same time

Then each instruction can be active in a different stage.
Computational rate increases 6-fold
waiting for 1 to ripple through all 6 stages.
So 6 instructions execute simultaneously rather than
In instructions that are dependent cause problems.

Conditional branches can cause problems.

Requiring general address computation on each cycle.

Succeeding instructions can differ.

Requiring decoding an instruction on each cycle (since

Numerical computing can be a very powerful technique when applied to

Can be applied to memory system requests

It is used in many different forms and places

Standard technique to increase performance of machines

Pipelining
The EX stage is pipelined

(hardware)

Therefore can be computed simply (possibly by dedicated hardware)

The addresses of the required data are related and

ea count known

Point address, so only one instruction must be decoded and

A series of instructions are identical, e.g., all floating

Vector processing simplifies things by assuming:
1. compare exponents (c-stage)

2. shift mantissa (s-stage)

3. add mantissas (a-stage)

4. normalize (n-stage)

\[ h + x \text{ implies } h + x \]

Suppose \( x \) and \( z \) * \( t \) * * \( d \) = \( x \) and \( y \) * * \( * \) then, 

\text{Floating point addition}
\[
(642.0)_{10} = (642.0)_{10} + \text{(shift needed)} -
\]
\[
(642.0)_{10} = (642.0)_{10} + (4.6314)_{10} = (10^2)_{10} = x -
\]
\[
\text{Example 1} -
\]
\[
(8138)_{10} = (8138)_{10} + (4.6314)_{10} = (10^2)_{10} = x -
\]
\[
\text{Example 1} -
\]

4-digit floating point decimal arithmetic
Cycle by cycle profile of pipelined add. (O indicates subcomputations for the i-th component operation.

\[ u \cdot \ldots \cdot I = \gamma \quad \gamma i + \gamma x = \gamma z \]

\[ \text{Input} \leftarrow \text{N} \leftarrow \text{A} \leftarrow \text{S} \leftarrow \text{C} \leftarrow \gamma i \quad \text{Output} \]
<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>6</td>
<td>7</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>6</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>6</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>1</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### In-Out

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>A</td>
<td>S</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Cycle**
an array to define an aggregate operand.

More generally, it can be applied in any or all dimensions of

\[ \text{HPF / Fortran90 Notation: } A[:N:]R \text{ or } A\text{[low:high:stride]} \]

- \( N \) : Length of the vector
- \( R \) : Stride of the vector
- \( A \) : Base address of the vector

(contents of the addresses)

The form \( A + (y - 1)R \) for \( y = 1, \ldots, N \). (also refers to the

Definition: a vector is defined to be a series of addresses of

Vectors
vectorization.

Vector instructions still benefit from exposing

Many highly pipelined machines that do not have actual
computations.

Pipelined functional units perform both scalar and vector

Cray is a notable exception to this rule where the same
execute instructions with one or more vector operands. A
execute instructions with one or more vector functional
units and other special vector functional units which
Such machines typically have normal scalar functional
well as scalars.

Definition: A vector machine is characterized by the fact
that it has instructions that can operate on vectors as
which may be integer, floating point or logical.

where \( \mathbf{v} \) is a vector, \( s \) is a scalar, and \( \circ \) is an operation

\[
(\text{reduction}) \quad s = \mathbf{v} \cdot s \\
(\text{reduction}) \quad s = \mathbf{v} \cdot s \\
(\text{nonreduction}) \quad \mathbf{v} = (\mathbf{v} \circ s) \circ \mathbf{v} \\
(\text{nonreduction}) \quad \mathbf{v} = s \circ \mathbf{v} \\
(\text{nonreduction}) \quad \mathbf{v} = \mathbf{v} \circ \mathbf{v}
\]

\text{Vector operations}
Systolic arrays
machines such as the CM-5 (Connection machine) or not restricted to vector machines, e.g., data parallel

- flexible (associative) dependence for reductions

- nonreductions

- the same operation on independent data

Note this is highly structured parallelism

\[
\begin{align*}
\langle \text{dot product, BLAS1} \rangle & \quad h' \cdot x + g' = a \\
\langle \text{trim, SAXPY, DAXPY, BLAS1} \rangle & \quad z = h' a + x
\end{align*}
\]
end do

A(2*I - 1) = B(2*I) + a * c(3*I + 2)

do I=1,N
equivalent to


Example
Note: \( B[2^N] \) is not accessed, i.e., it is used as upper bound.

\[
\begin{align*}
\text{end do}\\
A(\tau) = B(2^\tau-1) + p\\
do \tau = 1, N\\
\end{align*}
\]

Example
end do

\[ c = \forall (2^i - 1) + c \]

\[ do \quad i = 1, N \]

\[ 0 = c \]

Example

\[ (\exists z : \forall x : I) \land S = c \]
The other form of vector syntax can be particularly problematic since it always requires symbolic analysis. Things are not always so easy for the compiler. Length was shown explicitly and is symbolically combined. Note vectors must be of the same length to be equal in $\$\text{ form.}$
data parallel processors.

- HPF and Fortran 90 used for vector processors and software.
- Recall the portability compromise of architecture (consider the particular mappings of the operations to the dimensioned array structures but they do not require multiple Fortran (HPF) have explicit vector and multiple languages such as Fortran 90 and High-Performance
- Typical for true vector machines.
- Software support at the assembler level is quite
Exceptions to the structure.

Universally apply the basic idea, i.e., to handle and hardware support addresses the need to not.

After defining the basic idea, much of the software can be done reasonably well (but there is still the need for explicit programming and library routine). Automatic vectorization via restructuring compilers.
Example: CDC Cypher 205

- Requires very rich instruction set or many passes to/from memory
- Instruction computations, termination tests
- All loop control is done in hardware when using vector memory during vector operations
- All operands come from memory and results return to

MEMORY-TO-MEMORY ARCHITECTURE

The most natural way to implement this type of processor is...
are used.

processor arrays. Register-based vector processors

Not used in practice, except for special purpose processor pipelines.

through a memory system pipeline much like the

Note multiple outstanding memory requests circulate

one)

Requires multiple ports to memory (or one very fast

cycle.

At least two reads and one write are required per

demand on the memory system.

A memory-to-memory machine places a very high