Vector and Parallel Processors.

- Vector processors are processors which have special hardware for performing operations on vectors:
  - generally, this takes the form of a deep pipeline specialized for this task.
  - Some modern general-purpose CPUs also have vector facilities
    - Pentium P4
    - Power PC7451
- Parallel processors are processors with many near-independent CPUs which collaborate on a single task.
- Both forms of processor can supply a major speedup in processing:
  - but before we get carried away with this, let us examine speedup in a little more detail

Amdahl's Law

- Amdahl's law provides the speedup we obtain by providing some enhancement to performance that applies only to some fraction of the task:

$$\text{Overall Speedup} = \frac{\text{Execution time without enhancement}}{\text{Execution time with enhancement}}$$

If we write

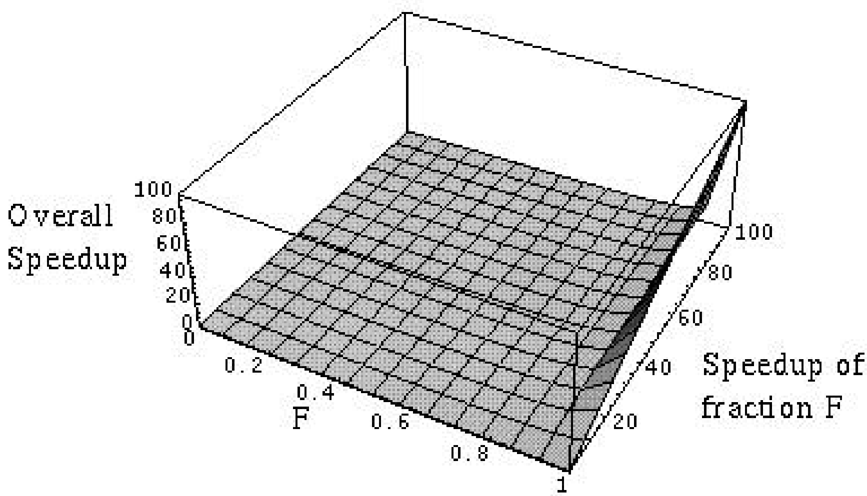
F for the fraction of the process speeded up

S for the speedup of this fraction

- Then the new execution time $E_{new} = E_{old}(1-F) + E_{old}(F/S)$ so that
- so that the overall speedup, $S_{overall}$ is

$$S_{overall} = 1/((1-F) + (F/S))$$

Amdahl's Law : graphically…



Vector Processors: why?

- Many computations involve vector operations:
  - these may be operations directly on vectors, or matrix operations.
- They arise commonly in scientific and engineering calculations.
- Such operations commonly require hundreds of floating point operations - mostly FP multiplies and adds.
- They also arise in graphics: there the operations are sometimes integer, rather than FP.
- By optimizing the machine for these operations, they can run considerably faster.
- On a normal CPU, there will be one instruction per arithmetic operation.
- On a vector CPU, there will be one instruction per vector operation - and this may involve hundreds of floating point operations.
- This also means that the instruction bandwidth is much reduced.

Vector Processors: how?

- The usual technique is to add

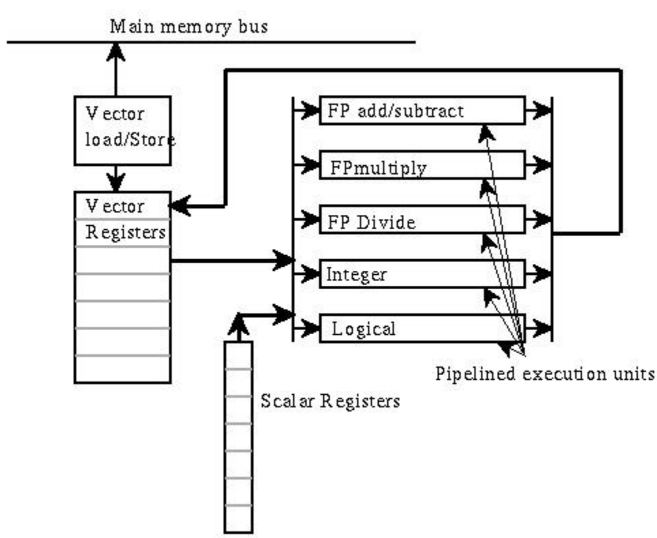
  * a number of *vector registers*, each able to store a vector (of some fixed size)

  * a number of *vector functional units* which can operate on these vector registers.

  * a *vector load/store* unit to interface the vector registers to the memory

  * some scalar registers for operations which involve a scalar and a vector

Example Vector Processor Architecture



Pipelining and performance

- Since one is aiming for performance, the arithmetic operations are all highly pipelined.
- Long pipelines are particularly advantageous in a vector processor:
  - if one is (e.g.) multiplying two vectors together, one knows in advance that one needs to perform a large number of FP multiplies,
  - there will not be any branches or any data hazards in the middle of the operation.
- Thus a deep pipeline may be used effectively without fear of stalling
  - it is worthwhile providing the hardware required to support this.

Compiler implications

- There are language implications:
- the compiler needs to know that certain operations are vector operations, so that it can generate appropriate code

```
vector of float  x[100], y[100], z[100];
float a,b,c ;
.
.
z = a * x + b * y ;
```

- The word `vector` tells the compiler something about the data type
- It can then ensure that the vector hardware is used.

- This can present a mismatch between general-purpose programming languages and vector processors.

Measuring performance: Mflops and Mips.

- Performance of machines is frequently measured in terms of the number of instructions which they can execute per second.
- For integer (and branch) instructions, the standard measure is Mips: mega-instructions per second.
- For floating point instructions, the measure is Mflops: mega-floating point instructions per second.
- It is clear that using a pipelined/superscalar integer pipeline will improve the Mips rating of the processor;
- additionally, using a vector processing approach will increase the Mflops rating. (Mips if it's an integer vector processor)

Mflops and Mips: a word of warning

- These measurements need to be considered with a pinch of salt: which instructions are being measured?
- can the processor sustain this speed on a **real** instruction mix, or is this number the maximum ever possible?
- can the Mflops rating really be achieved, or is the integer performance such that there will be delays between vector operations?
- With a vector processor, one needs to consider exactly how much of a speedup one will actually achieve.
- Amdahl's law is relevant here:
  - if one speeds up the vector processing section by a factor of 100, and the vector processing is 3/4 of the total processing,
  - then the actual speedup will be $1/((1-3/4) + (3/4)/100) = 400/103$: i.e. nearly 4
  - … not quite so impressive!