VLIW

VLIW processors are another way to improve performance. These processors work best with regular tasks- such as loops and array manipulations.

More than 1 IPC

Processors that can issue more than 1 instruction per cycle:

- -Out of Order Superscalar
  - -It can issue multiple instructions per cycle
  - -It can look at a lot of instructions at a time for scheduling
  - -Very expensive - with many reservation stations, etc.
  - - A compiler can help with improving IPC
- -In Order Superscalar
  - -It can issue multiple instructions per cycle
  - -It can look at fewer instructions at a time for scheduling than OOO processor
  - -It is less expensive that OOO processor
  - -It needs help from a compiler to improve IPC
- -Very Long Instruction Word (VLIW)
  - -It executes 1 big instruction per cycle
  - -It does not do instruction scheduling, it just executes the next large instruction
  - -It is the least expensive of the three listed
  - -It really requires a good compiler

Superscalar Vs. VLIW

A superscalar processor:

1. Gets multiple instructions
2. Checks for dependencies
3. Then sends instructions to the execution units for parallel execution when it can.

A VLIW Processor:

1. The compiler looks for dependencies
2. If there are dependencies it loads them into separate instruction words. This can lead to much larger number of bytes for a program in VLIW.

VLIW: The Good and the Bad

Good:

-The compiler does the work and this program is run over and over. Thus, the compiler can take the time to find good instruction scheduling.
-The hardware is simpler than for Superscalar
-It can be energy efficient
-It works well on “regular code” such as loops and arrays.

Bad:

-Latencies of instructions are not always the same
-Many applications are irregular
-Code bloat

VLIW Instructions

-VLIW instructions have all the usual ISA opcodes
-Fully support predication
-Require many registers because of the scheduling optimizations
-Branch hints because the compiler needs to tell the hardware its predictions
-VLIW instruction compaction - instead of using NOPs for empty instruction slots there are stops. This reduces the number of instructions required, thus reducing code bloat.

VLIW Examples

Examples of VLIW processors:

Itanium Processor - too complicated, not good with irregular code

DSP Processors - usually have excellent performance and energy efficient