

VLIW Processors

1 very long instruction contains suboperations

- change in the instruction set architecture

Machines: Multiflow & Cydra 5 (8 to 16 suboperations)
IA-64, Crusoe today (3 & 4 suboperations)

VLIW Processors

Goal of the hardware design:

reduce hardware complexity & therefore reduce:

cycle time

power

- only 1 compiler-generated instruction issued
 - 1 program counter
- operations in a bundle issue in parallel
 - fixed format so could decode operations in parallel
 - enough FUs for types of operations that can issue in parallel
 - pipelined FUs
- ramifications for hardware complexity
 - no multiple issue hardware (no instruction grouping)
 - fewer paths between instruction issue slots & FUs
 - ideally no structural hazard checking logic
 - no dependence checking for instructions in a bundle
 - no hardware for out-of-order instruction issue

VLIW Processors

Compiler support to increase ILP

- compiler creates each VLIW word
- need for good scheduling greater than with in-order issue superscalars
 - instruction doesn't issue if 1 operation can't
- detects hazards & hides latencies
 - structural hazards
 - no 2 operations to the same functional unit
 - no 2 operations to the same memory bank
 - hiding latencies
 - data prefetching
 - hoisting loads above stores
 - data hazards
 - no data hazards among instructions in a bundle
 - control hazards
 - predicated execution
 - static branch prediction
- techniques for increasing ILP
 - loop unrolling
 - global code scheduling
 - software pipelining
 - trace scheduling
 - aggressive inlining

IA-64 EPIC

Explicitly Parallel Instruction Computing, aka VLIW

Itanium IA-64

Bundle of instructions

- 3 instructions/bundle
- 128 bit bundles
- 2 bundles can be issued; issue one, get another
 - implications for performance:
 - less delay in bundle issue than 21164-style slotting

IA-64 EPIC

128 integer & FP registers

- 128 additional registers for loop unrolling & similar opts
- implications for architecture:
 - 7-bit register specifiers in the instruction format
- implications for hardware:
 - no register renaming to eliminate WAW & WAR hazards
- implications for performance:
 - less spill code
 - longer access time
 - more time to save/restore on a context switch
 - fewer stalls due to WAR & WAW hazards

IA-64 EPIC 2

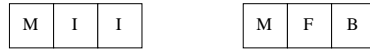
Full predicated execution

- supported by 64 predicate registers
- implications for architecture:
 - extra register specifier in the instruction format
- implications for the hardware:
 - additional functional units to support both branch paths
- implications for exploiting ILP:
 - eliminate branch delays & mispredictions
 - more independent instructions to execute since can schedule both paths & merge code together

IA-64 EPIC

Template in a bundle that indicates:

- type of operation for each instruction
- instruction order
- restrictions on which instructions can be in which slots



- M: load & increment index
- I: integer ALU op
- F: FP op
- B: branch
- if can't issue all instructions in a bundle, insert a stop bit
- implications for hardware:
 - no instruction grouping
 - fewer paths between issue slots & functional units
 - potentially no structural hazard checks
 - hardware not have to determine intra-bundle data dependences

IA-64 EPIC

Branch prediction

- static branch prediction
- full predicated execution
- hierarchy of BTBs
 - 4 target BTB for repeatedly executed code
 - instruction to put a specific target in it
 - larger BTB for hard-to-predict branches
 - instruction hint a target could not be placed in it
 - 2-level branch prediction
 - private history registers
 - 4 history bits
 - shared PHTs
- separate 2-level structure for multiway branches

IA-64 EPIC

Still seems complicated

- compatibility
 - IA-32
 - x86 compatible at the subroutine level
 - PA-RISC compatible memory model (segments)
- the sense of little niggling hardware

Superscalars vs. VLIW

Superscalar has more complex hardware for instruction scheduling

- instruction slotting or out-of-order hardware
- more paths between instruction issue structure & functional units
- possible consequences:
 - slower cycle times
 - more chip real estate

VLIW has larger code size

- estimates of IA-64 code of up to 2X - 5X over x86
 - 128b holds 4 (not 3) instructions on a RISC superscalar
 - nops if don't have an instruction of the correct type
 - branch targets must be in the beginning of a bundle
 - predicated execution to avoid branches
 - extra, special instructions
- consequences:
 - increase in instruction bandwidth requirements
 - decrease in instruction cache effectiveness

Superscalars vs. VLIW

VLIW require more complex compiler

Superscalars can more efficiently execute pipeline-independent code

- consequence: don't have to recompile if change the implementation