2000 Paper 7 q3
IAP

## *Comparative Architectures / IAP Q1 Extended Answer*

*Describe the characteristic features of a VLIW (Very Long Instruction Word) processor architecture.* [3 marks]

VLIW processors use a long instruction word (e.g. 128 bits) that encodes several operations that are to be performed concurrently on different function units of the processor. The compiler is effectively responsible for scheduling these sub-operations to function units, and ensuring that there are no data dependencies between them.

*VLIW processors aim to achieve high instruction throughput using a different strategy to that of out-of-order dynamic execution RISC implementations. Compare and contrast these different approaches, pointing out the pros and cons of each strategy in achieving this goal.* [9 marks]

The instruction issue logic of a VLIW processor is much simpler since the compiler has already done the work of selecting the function unit that instructions are to be dispatched to, and ensuring there are no dependencies between instructions in the same instruction word.

The processor typically needs only relatively simple score-boarding to stall the pipeline if an instruction depends on the result of a load that has yet to be satisfied. Instructions execute in order.

Because of this simplicity, it is likely that a VLIW processor will have a higher clock rate than a comparable dynamic execution processor, since the later will require a complex instruction reordering buffer to identify instructions that are ready to execute (both operands are ready and an appropriate function unit is free).

However, VLIW is not without its drawbacks. If the compiler is unable to identify an instruction to fill one of the instruction word slots then it must insert an appropriate no-op. This results in a wasted issue slot, and reduces the density of the I-stream, requiring a larger I-cache.

Furthermore, the static scheduling by the compiler is unable to take advantage of information discovered at execution-time. This makes the processor more vulnerable to stalls due to L1 cache misses, or long-latency function units that are occupied etc. A dynamic execution processor is more likely to be able to work around such problems without stalling, due to its ability to search ahead into the instruction stream to find other work. Thus, VLIW machines tend to be very sensitive to the quality of their compilers. Without a good compiler, many issue slots will be wasted, thus the function unit utilization of a VLIW processor is likely to be lower than that of a dynamic execution processor. However, the reduced issue logic complexity may enable a greater number of functional units to be incorporated.

*Why have VLIW architectures traditionally only been used for special purpose applications, such as Digital Signal Processors?* [3 marks]

The instruction word of VLIW machines typically exposes many of the internals of the processor, such as number of functional units etc. Different generations of a VLIW processor family are likely to have different instruction words formats. Thus, backwards binary compatibility is not easily achieved. Such a facility is essential for a general purpose processor in order to protect customer's software investment.

VLIW processors require good compilers. Until recently, it is arguable that they were not really up to the job of scheduling code for VLIW processors.

VLIW processors have frequently been used in DSP applications as backwards compatibility is less of an issue – applications are typically re-written for every new hardware design. Furthermore,

the developers of such applications are frequently interested in extracting maximum performance from their processors by programming in hand-scheduled assembler to ensure good function unit utilization.

*What techniques have been developed to try and make VLIW-like architectures more suitable for general purpose use?* [5 marks]

Intel's IA-64 architecture uses Explicitly Parallel Instruction Computing (EPIC) a development from VLIW. It attempts to keep the benefits of simple issue logic while trying to avoid explicitly exposing details about an implementation's function units, hence enabling backwards code compatibility.

Instructions are 41 bits wide, but are arranged into a 128 bit 'bundle' containing 3 instructions. Template bits in the bundle identify the kind of function unit needed by each instruction.

The compiler identifies groups of non-dependent instructions and inserts 'stop' bits in the bundle's template to identify the boundaries between groups (groups may span multiple bundles). Since the processor knows that instructions in a group contain no data-dependencies between them, it can execute them in parallel (if there are no structural hazards and all of the source operands are available).

Although EPIC should enable backwards code compatibility, there may be significant benefit in recompiling for different implementations. Like VLIW, EPIC relies on a high-quality compiler.

An alternative way of making VLIW 'mainstream' has been demonstrated by Transmeta's Crusoe chip. This chip has a VLIW core, but it is hidden behind a builtin software translator that takes x86 instructions and 'morphs' them into VLIW code. Crusoe's instruction set contains a number of features that aid the efficiency of this interpretation/translation process. Extensive use is made of run-time code re-writing to improve the performance of the translated code based on observed behaviour.

Transmeta's code morphing approach means they get the benefits of a simpler, lower power (and potentially faster) VLIW core, but avoid exposing it to the user-visible instruction set.