Solution notes

Comparative Architectures 2005 – Paper 7 Question 1 (IAP)

Most RISC architectures use a 32 bit fixed length instruction encoding. In contrast, Intel x86 and VAX use a variable length encoding, and the IA-64 uses 128 bit instruction 'bundles'. Compare and contrast these different instruction encodings, with particular reference to their ease (or otherwise) to decode in a super-scalar implementation, and the 'code density' they achieve. [10 marks]

- typical RISC (e.g. ppc, parisc, arm, alpha) use a 32 bit instruction with with 2 source and 1 destination operands which must all be registers. Special encodings are used for load/stores. Each register operand typically requires 4 or 5 bits. The format is regular, making the instruction decode logic simple. Multiple instructions are easily decided in parallel.

- x86 uses variable length instructions encodings, enabling 8,16 and 32bit immediate and displacement values. Instructions are two operand, of which one operand (either) can be a memory operand (the other must be a register). By examining the first couple of bytes of an x86 instruction it is possible to figure out the instruction length. This is important for building a super-scalar instruction decoder.

- VAX instructions can be either 2 or 3 operand, employing a variable length encoding that allows operands to be registers, immediate or memory operands (with displacements) in arbitrary combination. After decoding the instruction type, each operand must be decoded individually to discover their length, thus making parallel decoding of multiple instructions slower.

- IA-64 instruction bundles are 128 bits in length and encode 3 41 bit sub-instructions and a 5 bit template. This format was required because with 128 registers it was not possible to encode operands in the traditional 32 bit RISC operand. The next power of 2, 64 bits would have resulted in poor instruction density, so they decided to pack 3 sub instructions into a 128 bit bundle. The template bits assist instruction dispersement by identifying the type of each sub instruction; only certain combinations are possible.

- In general, variable length encodings result in higher instruction density than fixed length encodings: register-register instructions are able to have short encodings, and when large immediates are required they can be specified in a single instruction rather than requiring multiple instructions to synthesise.

Discuss the pros and cons of architectures with a 64 bit word size *vs.* those with a 32 bit word size. Which applications are likely to benefit most? [4 marks]

- 64 bit allows easy linear access to large data structures, memory mapped files etc.

- 64 bit means overflow rarely a concern when counting

- 64 bit good for implementing long integer arithmetic e.g. encryption. Also good in applications where you can use the wide register in a SIMD fashion e.g. string searching,

colour pixel processing etc. 64 bit usually implies fast memory copy too.

- having 64 bit C 'int' is likely to lead to program's data area increasing in size – and most words will have the msb's all 1's or 0's. This can lead to increased data bandwidth requirements

If you were a processor architect targeting embedded applications where memory is a scarce resource, how might you design a RISC-like instruction set that will achieve efficient use of memory? [6 marks]

- use a 32 bit word size

- move to a 16bit instruction encoding, by using the following techniques to compress the encoding:

- use a two operand read-modify-write instruction set rather than the usual RISC 3 operand format

- even if the machine only has 16 GPRs, specifying two register operands would consume 4+4=8 of the 16 bits. We can potentially save a couple of bits here by using an instruction specific encoding to map a 3 bit operand to a subset (8) of the 16 registers. A different mapping would probably be used for the src and src/dst operands. The non-orthogonality this scheme creates is going to cause the compiler to have to work hard to most effectively allocate registers.

- 16 bit operands doesn't leave much room for displacements and immediates. It's likely that multiple instructions may need to be used to synthesise constants.