Comparative Architectures Supervision Answer (Set 1)

Wei Song 08/02/2017

This answer provides a summary of key points in each question. It is the understanding of the supervisor, which does not necessarily always correct and does not represent the view of lecturers. It is expected to expand the key points into detailed description when similar questions are asked in exams.

Lecture 1. Introduction

Q1: Why does power consumption now constrain processor design much more than it has historically?

1. Although dynamic power is reduced for each generation, supply voltage and interconnect do not scale at the same speed with transistors, slowing down the drop of dynamic power.
2. Leakage power actually increases for each generation due to the drop of supply voltage.
3. The increased transistor density and 3-D technology make heat dissipation more difficult than before.
4. Smaller transistors are more vulnerable to heat, which makes the heat dissipation issue even less tolerable.

Q2: Why is MIPS a very poor measure of processor performance?

1. The number of instructions is not a good measurement of workload. Work can be done by different ISAs with very different numbers of instructions. (CISC and RISC)
2. Speed performance is not the only metric important for processors.

Q3: How might recent advances in die stacking help to improve microprocessor performance and reduce costs?

1. Understand the difference between 3D IC and 3D package. 3D package still use traditional dies but use flipped chip or wire bounding to connect dies inside the same package. The term “die stacking” is not very clear.
2. Die stacking significantly reduces the length of interconnects, which improves speed performance.
3. It also allows dies to use different feature sizes for the best cost reduction and manufacture easiness.
4. The die size of each layer is significantly reduced, which improves manufacturing yield rate.

Lecture 2. Fundamentals of Computer Design

Q4: Discuss the pros and cons of architectures with a 64-bit word size versus those with a 32-bit size. What applications are likely to benefit most?

1. Supporting 64-bit word size allows easy access to memory > 4GB, which is beneficial for applications requiring large memory. (32-bit ISA is possible to access memory space > 4GB by using wider physical addresses, search “page address extension (PAE)” of x86)
2. It also reduces the number of instructions and the time to do long calculation, such as double precision floating point and 64-bit integer calculation. It would be helpful for scientific applications need such features.

Q5: 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?

Currently there is strong evidence that using shorter instructions for frequent operations can reduce the size of a program. So the common method is profiling the target applications to identify the most utilized operations. Then replace the most utilized operations with shorter instructions, such as 16-bit instructions in a 32-bit ISA.

Reducing the memory need for data is less obvious. Most time it will depend on programmers to write tight programs.

Lecture 3 & 4. Advanced Pipelining

Q6: What is the architecture requirement for an anti-dependence (WAR) to cause a data hazard?

Quote: “WAR hazards cannot occur in most static issue pipelines.” [Hennessy and Patterson] This is a hazard happens only in out-of-order issue and out-of-order commit processors (nearly non-existing) and can be resolved by renaming.

WAR in memory is also possible (think of a non-blocking data cache) and it is a real problem. It becomes even difficult to detect if using a virtually tagged and virtually indexed cache (two virtual addresses are actually mapped to the same physical address, normally called alias).

Q7: What parameters determine the optimal pipeline length for a microprocessor?

Optimal pipeline length is a trade-off between clock frequency and stall cost. Longer pipeline allows higher clock frequency but introduces longer stall penalty. As a result, deep pipelined processor must have highly accurate branch predictors, which may significantly increase area.

Q8: How does the use of multiple parallel pipelines make it more difficult for us to provide support for precise exceptions?

Precise exception requires all instructions before the excepted instruction to be committed while all instructions after the excepted one to be flushed. For processors using multiple pipelines, it is difficult to figure out the individual instruction on the non-excepted pipeline whether it should be committed or flushed. Extra synchronization or book-keeping logic is required.

Q9: Discuss the pros and cons of different address formats, from zero-address to three-address format.

From zero to three address formats, implicate register dependency is reduced; therefore, parallelization and speculation become easier. However, the number of read ports to the register file increased, which brings potential structural hazards.

Q10: Why are 2-bit saturating counters often used to predict a branch's direction in preference to single bit schemes?

It reduces the fluctuation of the miss predication on minor cases.

Q11: Why might a branch target buffer provide a poor prediction of procedure return addresses and what hardware solution may be employed to improve the accuracy of such predictions?

BTB is suitable for predicting the target address of a branch that frequently jumps to a single address. Procedure calls hardly return to a single address but the caller addresses. A return address stack can accurately predict the return addresses.

Q12: Why might a single conditional move instruction be supported rather than the conditional (or predicated) execution of every instruction in the instruction set?

This is a trade-off between pure predicated execution and branched execution. Predicted execution has the benefits of low instruction count and no jump but processors need to work out whether an instruction should be invalidated according to the condition. Such condition causes implicate dependence, which makes speculation hard. It is normally beneficial to use predicated execution for short code segments rather than long segments. To reach a balance between easier speculation (small forwarding network and simple pipeline control) and lower instruction count (less overhead for miss-prediction as well), some processor may choose to support only conditional move.