The following problems are mostly descriptive. Diagrams or charts are not needed in the answers. Therefore it is highly recommended to type the answers. Please keep your answers organised and tidy. If you are not sure about the answers, you could look up from the Hennessy and Patterson book or directly from the Internet. If you could not find the answer, please make it clear that you are not sure.

1 Superscalar techniques

Read the paper about the ALPHA 21264 superscalar processor and answer the following question.

Link: https://www.cis.upenn.edu/ milom/cis501-Fall09/papers/Alpha21264.pdf

Figure 1: Stages of the Alpha 21264 instruction pipeline

(a) How many instructions are discarded if the processor finds a single branch is mis-predicted? In Figure 1 why “Slot 1” were added as stage 1 and why there is no decode stage?

(b) For the Alpha 21264 processor, two architectural techniques boosted the instruction fetch efficiency respectively, one is line and way prediction, and the other is branch prediction.
(i) The line and way predictor predicts the line-way of the instruction cache that will be accessed in the next cycle. Please read the paper and explain how it could hide the long delay of the branch predictor and why the line and way predictor could be trained by the results of the branch predictor?

(ii) What kind of patterns in branches are better predicted by global branch predictors and what patterns are preferred by local branch predictors?

(c) Register renaming exposes application instruction-level parallelisms since it eliminates unnecessary dependencies. Out-of-order execution then exploits the benefits and deploys them on multiple execution units.

(i) Which kind of dependences are removed and how the renaming could remove the hazards caused by the false dependences?

(ii) Instructions could be issued out of order, why do we need to fetch and commit (retire) instructions in order? What problems occur when you commit instructions
1 SUPERSCALAR TECHNIQUES

out of order?

![Register renaming state transition](image)

Figure 4: Register renaming state transition. AR: architectural register, RB rename buffer (or scoreboard), source: [The design space of register renaming techniques, IEEE](#)

(iii) Read the paper and Figure 3 and 4 and try to fill in the following blanks:

i. On decoding an instruction: search the map content addressable memories for the instruction’s _______ register(s) and allocate one of the _______ for the instruction’s _______ register.

ii. On issuing an instruction: the instruction is assigned to one of the two _______. When it is assigned with an instruction which _______, it would assert a request to the _______. When the request is granted, the instruction is issued to _______.

iii. On finishing an instruction: write the result back to _______ register and mark it as _______ state. Broadcast the results to _______ so that the depending instruction can ________________.

iv. On committing/retiring an instruction: mark the scoreboard as _______ state and free the _______ in the _______. The state transits to _______.

(iv) When a mis-predicted branch or an exception is detected, how would the register map be reverted to the latest valid state? What additional hardware is needed?

(d) Out-of-order memory references are most significant factor that limits the superscalar performance.

(i) The out-of-order execution of ALU instructions in a superscalar processor is only constrained by the availability of functional units and true data dependencies. Why must the out-of-order execution of memory instructions be constrained further? (from 2009 Paper 7 Question 7)

(ii) Read the section: Internal memory system and Dynamic execution examples in the paper and fill the following blanks.

i. The Alpha 21264 maintains a load queue (LDQ) and store queue (STQ). Both queues follow strict memory order as the fetch order. A load in the LDQ is flushed when _______ and a store exits the STQ after _______.

ii. When a store is issued, the store address is compared against the addresses of younger loads in the LDQ using CAM search. If the older store issues to
the same memory address as a younger load, then ________________. This memory data dependence is called ________.

iii. When a load is issued, the load address is compared against the addresses of younger stores. If a match is found and the store data is available, then ________________. If a match is found but the store data is unavailable, then ________________. This memory data dependence is called ________.

(e) The Alpha 21264 also supports speculative executions to break RAW dependences on available execution units.

(i) In what circumstances does it need speculative execution?

(ii) Reorder buffer and unified registers are the two major techniques to support speculative execution. Please describe the difference of the two techniques.

2 VLIW Processor

(a) What components in typical superscalar processors are no longer needed in VLIW processors?

(b) Why might dynamic binary translation be particularly useful in the case of a VLIW machine?

(c) (Optional) Some VLIW processors contain additional hardware to permit memory reference speculation.

(a) What optimisations does memory reference speculation permit?

(b) Briefly describe the additional hardware required to support this type of speculation.

3 Multi-threaded processors

(a) What are the advantages of exploiting Thread-Level Parallelism (TLP) in addition to Instruction-Level Parallelism (ILP)?

(b) Briefly describe the differences between coarse-grained, fine-grained and simultaneous multithreading.

(c) Why couldn’t we combine all the stages from all the N cores from a chip-multicore processor into a massive core to run N simultaneous threads with N times wider instruction fetch, N times number of ALUs, N times of other execution units?

(d) (Optional) Describe one technique for reducing the thread switch penalty in a coarse-grained multithreaded processor.

(e) ILP and coarse-grained TLP (parallel execution) are almost orthogonal ways of exposing parallelism on multicore processors. The following table shows the major software/hardware techniques to expose different factors of ILP. Please fill in the third column for TLP if you know the technique or discuss a way if this technique hasn’t been developed.
## MULTI-THREADED PROCESSORS

<table>
<thead>
<tr>
<th>Feature</th>
<th>Instruction-level parallelism</th>
<th>Thread-level parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unit</strong></td>
<td>Instruction</td>
<td>Task</td>
</tr>
<tr>
<td>Source/Dest</td>
<td>Register/Memory</td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td>ALU/EU</td>
<td></td>
</tr>
<tr>
<td>Pipeline</td>
<td>Pipeline stages</td>
<td></td>
</tr>
<tr>
<td>Relaxation</td>
<td>Out-of-order execution</td>
<td></td>
</tr>
<tr>
<td>WAR/WAW</td>
<td>Register renaming</td>
<td></td>
</tr>
<tr>
<td>RAW</td>
<td>Reorder buffer/Scoreboard</td>
<td></td>
</tr>
<tr>
<td>Speculation</td>
<td>Reorder buffer</td>
<td></td>
</tr>
<tr>
<td>Memory order</td>
<td>Load/Store buffer</td>
<td></td>
</tr>
<tr>
<td>Prediction</td>
<td>Branch prediction</td>
<td></td>
</tr>
<tr>
<td>Broadcasting</td>
<td>Internal bus</td>
<td></td>
</tr>
<tr>
<td>Schedule</td>
<td>Issue stage control logic</td>
<td></td>
</tr>
<tr>
<td>Compiler assistance</td>
<td>VLIW</td>
<td></td>
</tr>
</tbody>
</table>

End of Problems