Problem No. 1 (11 points)

(a) (6 points) Consider the two-level GAg predictor proposed in the Yeh & Patt paper. Assume that you have a budget of 16 Kbytes to implement the storage for the branch predictor. It is required that the pattern history table (PHT) must have 4-bit entries. How many bits of global branch history can you keep in the branch history register (BHR)?

(b) (2 points) What is the main advantage of the GSHARE branch predictor compared to the GAg branch predictor?

(c) (3 points) Consider a Return Address Stack (RAS) with 4 entries. Describe a scenario in which this RAS will be unable to predict return addresses correctly.

Problem No. 2 (17 points)

(a) (5 points) State one potential benefit of allowing partial matches in trace cache design. What change(s) needs to be made to the baseline trace cache design in order to support partial matches?

(b) (3 points) Can the same instruction appear multiple times in a single trace cache block? Why or why not?

(c) (6 points) Discuss the impact of each of the following factors on the effectiveness of a trace cache: (i) Temporal locality, (ii) Percentage of indirect branches, (iii) Average basic block size.

(d) (3 points) Consider a design upgrade in which the simple 1-bit branch predictor used in a processor is being replaced by a more sophisticated PAp predictor with 10 bits of branch history. How does that change impact the effectiveness of the trace cache being used in the processor?
Problem No. 3 (14 points)

(a) **(2 points)** State one advantage of the “future file” approach as compared to the “history buffer” approach.

(b) **(4 points)** State one advantage and one disadvantage of using the “future file” approach as compared to using the “reorder buffer without bypassing” approach.

(c) **(4 points)** Consider a scenario where an instruction just caused an exception and the history buffer is being used to recover the precise program state. In which order should the processor read the contents of the history buffer to ensure precise recovery? LIFO (Last-in-first-out) order or FIFO (First-in-first-out) order? Provide reasoning for your answer.

(d) **(4 points)** A processor uses a re-order buffer (ROB) with 64 entries and a store buffer with 32 entries. In the next generation of the processor, the ROB size is increased to 128 entries. There are three options regarding the store buffer size: (i) retain the old size (32 entries), (ii) quadruple the size to 128 entries, (iii) double the size to 64 entries. Which of the three options are the processor architects most likely to choose and why? What are the reasons for not choosing the other two options?

Problem No. 4 (8 points)

(a) **(5 points)** Assume that the issue window is quadrupled in size from 16 entries to 64 entries. What impact does it have on: (i) number of comparators in the wakeup logic, (ii) delay of selection logic?

(b) **(3 points)** Can the RUU entry occupied by an instruction be de-allocated immediately after the instruction is sent to the functional unit for execution? Why or why not?