Intro. The main goals of this homework are the following:
(a) To introduce quantitative approach in computer architecture,
(b) To demonstrate quantitative approach in a “real-world” example, and
(c) To introduce tools students are likely to use for their projects; for this purpose your task is to write simple C
programs as explained below, and to measure performance on the sr4 machine using PAPI (Performance
Application Programming Interface).

Question #1 (50 points)
Extract as much as possible information about the memory hierarchy of the sr4 machine by utilizing simple C
microbenchmarks and PAPI. You would like to determine the size and organization of caches, in particular, if
possible to answer the following questions:
a. Are L1D and L1I caches split or there is a L1U (unified) cache for both instructions and data?
b. Is there L2 cache(s)?
b. What is the size of LXY (in KB)?
c. What is the size of a cache line for LXY?
d. What is cache associativity for LXY?
e. What is cache replacement policy of LXY?

Note: You can get all these answers for the corresponding reference manuals; however, your goal is to try to get all
these information using microbenchmarks and PAPI. Developing some of the microbenchmarks could be a tricky
task, and failing to get some of these parameters should not disturb you.

In your report, please clearly indicate the flow of experiments you have employed and what are findings you have
gotten.

The following documents can give you useful information:
1. M. Milenkovic, A. Milenkovic, J. Kulick,
"Microbenchmarks for determining branch predictor organization,"
== Describes how to stage your effort, though it addresses branch predictors.
== How to deal with caches.

Extraordinary efforts will qualify for bonus marks (up to 25% of the original score).

Question #2 (20 points). Exercise 1.2 (Textbook, pp. 74)

Question #3 (10 points).
A. Four enhancements with the following speedups are proposed for a new architecture:
E1: Speedup1 = 10
E2: Speedup2 = 20
E3: Speedup3 = 5
E4: Speedup4 = 50.
Only one enhancement is usable at a time (enhancements are non-overlapping).
Assume we are considering two approaches in maximizing performance (A1 and A2) where A2 requires slightly
more complex hardware support.
A1: Enhancements E1 and E2 are in use; the distribution of enhancement usage is 30% (E1) and 40% (E2).
A2: Enhancements E3 and E4 are in use; the distribution of enhancement usage is 20% (E3) and 50% (E4).
Which of these 2 approaches (A1 or A2) is better?

Question #4 (20 points). Exercise 1.17 (Textbook, pp. 82)