In class we covered the “LLPex” example, and did constraint graphs for the normal code, and for a version that was 2x unrolled (i.e. the loop body handles both \([X0, #0]\) and \([X0, #1]\) and compiler register renamed. For this homework, do the following. NOTE: We are ONLY Scheduling the body of the loop – you can ignore the ADDI at the beginning to initialize the loop variable. Also, when scheduling do NOT change the code, just schedule what you produced in the previous steps, or got from lecture:

1.) Create a 4x unrolled version of the code (let’s call it LLPex4) with compiler register renaming (i.e. use different registers in the code to increase parallelism, but do NOT assume hardware register renaming, the a0’ stuff).

2.) Draw the constraint graph for LLPex4. Each edge of the constraint graph should indicate the cause (i.e. “RAW X0”, “Control”, etc.).

3.) Schedule both LLPex (the original code) and LLPex4 onto the lab #4 processor.

4.) Schedule both LLPex and LLPex4 onto a 2-way VLIW. Only one of the issue slots is allowed to do loads and stores, there are load delay slots (cannot use the value loaded in the subsequent cycle), and branch delay slots (the other instruction in the same cycle, and both instructions in the next cycle, ALWAYS execute regardless of whether the branch is taken or not). Branches can be done in either issue slot.

For #3 and #4 please make sure the code schedule is as short as possible.