Problem 1. Software Pipelining
Consider the following loop

```c
for(i = 0; i < n; i++) {
    X[i+1] = X[i] * 2
    Y[i+1] = X[i+1] + Y[i]
}
```

For this problem, assume that ADD/MUL takes 2 clocks, while LD/ST takes 1 clock. The machine has two MEM units that can execute a LD and ST, and two ALU units that can execute ADD and MUL in one clock. The machine can auto-increment address registers.

1. Draw the data dependence graph by showing four types of nodes, LD, ADD, MUL, and ST.

2. Label the edges of the dependency graph according to the type of dependency (true dependency, anti-dependency or output dependency)

3. What is the lower bound on the initiation interval?
Problem 2. Dependency analysis and parallelization
Consider the following program:

```c
for(i = 1; i < n; i++) {
    for(j = i; j < 2 * i; j++) {
    }
}
```

1. Draw the iteration space for this loop.

2. What are the possible data dependencies in this loop? A data dependency is something of the form “read/write A[i], read/write B[j]”.

3. Is this loop nest 1 or 2-d parallelizable without loop transformations? You do not need to show the exact solution to the equations, but justify your answer.
Problem 3. Global Instruction Scheduling

Assume you have a statically scheduled machine that can only issue one operation every clock. All operations have a latency of one clock cycle, with the exception of its memory load operation, which has a latency of three clock cycles. Consider the following locally scheduled program:

Assume that only $r$ is live at the end of the program, and that all variables have been defined at the beginning of the program. Each branch in the flow graph is labeled with the probability that it is taken dynamically. To answer the following, you may apply any of the code motions discussed in class, but no other optimizations.

1. Is this the best globally scheduled code that can be generated given that $p = 0.1$? If not, provide the improved code along with its expected execution time.

2. Repeat part 1 given that $p = 0.9$.

3. What is the percentage improvement in execution time from the original program for your program(s) in part 1 and 2, if one was provided?
Problem 4. More Software Pipelining
Consider the following dependence graph for a single iteration of a loop, with resource constraints:

All instructions take 1 cycle each.

1. What is the bound on the initiation interval $T$ according to the precedence and resource constraints for this program?

2. What is the minimum initiation interval? Show a modulo reservation table for an optimal software pipelined schedule. Also show the code and schedule for an iteration in the source loop.

3. Can the scheduling algorithm described in class produce the optimal schedule for this loop?
Problem 5. Affine Transforms. Apply affine transform to find the largest degree of outermost loop parallelism. Show the transformed code, marking the loops that are parallelizable.

```c
for (i = 1; i <= N; i++) {
    for (j = 1; j <= N; j++) {
        A[i,j] = X[i,j] + X[i,j];
    }
}
for (i = 1; i <= N; i++) {
    for (j = 1; j <= N; j++) {
        for (k = 1; k <= N; k++) {
            B[i,k] = B[i,k] + Y[j];
        }
    }
}
for (i = 2; i <= N; i++) {
    for (j = 1; j <= N; j++) {
        C[i,j] = A[i-1,j] * A[i-1,j];
    }
}
for (i = 1; i <= N; i++) {
    D[i] = C[2,i] + B[2,i];
}
```