ECE 468 & 573
Problem Set 5: Instruction scheduling & loop transformations

Instruction scheduling

1. Assume that your machine has two ALUs, which can execute ADDs in a single cycle, SUBs in two cycles, and *one* of the two ALUs can execute MULs in two cycles. There is also a single LD/ST unit. LOADs take up the LD/ST for two cycles, while STs take up the LD/ST for one cycle. The LD/ST unit can also perform ADD instructions in a single cycle. Draw the reservation tables for this machine.

   Answer:

   ADD (ver. 1):

ALU0	ALU1	LD/ST
X		

   ADD (ver. 2):

ALU0	ALU1	LD/ST
	X	

   ADD (ver. 3):

ALU0	ALU1	LD/ST
		X

   SUB (ver. 1):

ALU0	ALU1	LD/ST
X		
X		

   SUB (ver. 2):

ALU0	ALU1	LD/ST
	X	
	X	

   MUL:

ALU0	ALU1	LD/ST
X		
X		

   LOAD:

ALU0	ALU1	LD/ST
		X
		X

STORE:

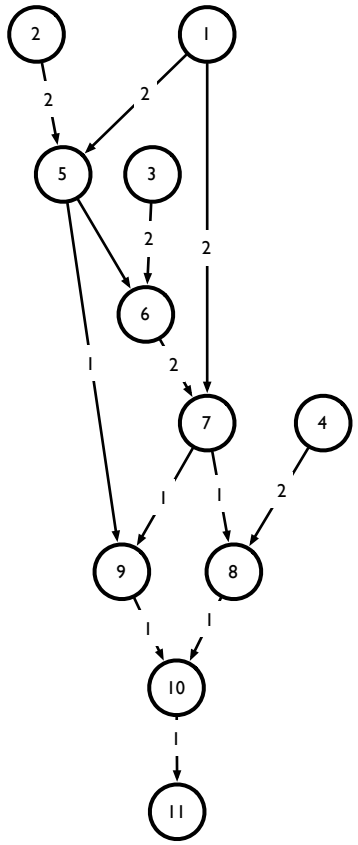
ALU0	ALU1	LD/ST
		X

2. Draw the scheduling DAG for the following program:

```
1.  LOAD(A) R1
2.  LOAD(B) R2
3.  LOAD(C) R3
4.  LOAD(D) R4
5.  R5 = R1 + R2
6.  R6 = R5 * R3
7.  R7 = R1 + R6
8.  R8 = R6 + R4
9.  R9 = R5 + R7
10. R10 = R9 + R8
11. ST(R10) E;
```

Answer:

Instructions, with heights in parentheses:



3. Give the schedule you obtain after performing height-based list scheduling on the above code.

Answer:

Each instruction is marked in the cycle(s) it is executing, in the appropriate functional units.

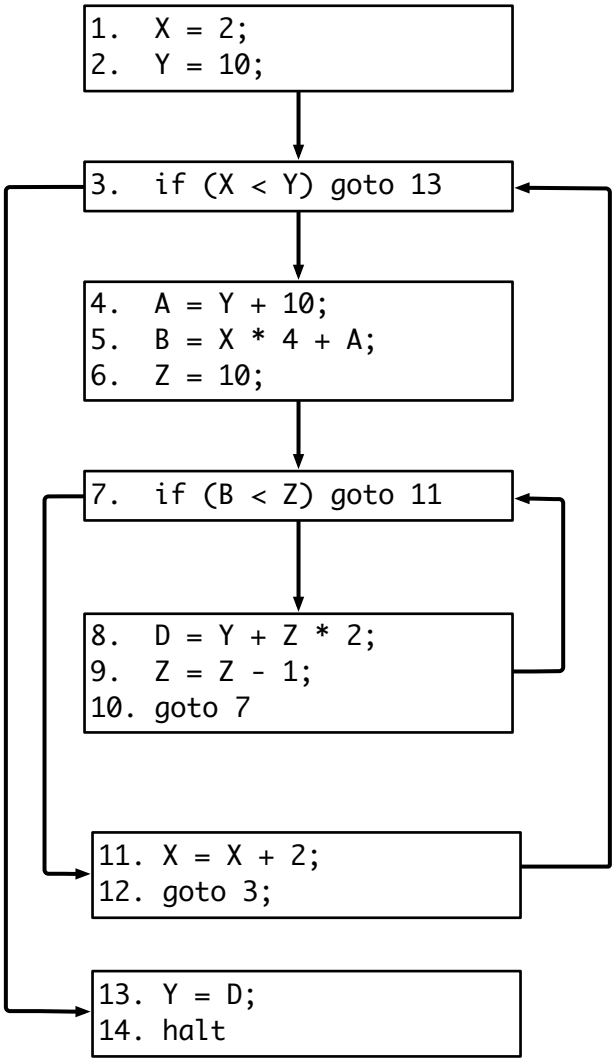
Cycle	ALU0	ALU1	LD/ST
1			1
2			1
3			2
4			2
5	5		3
6			3
7	6		4
8	6		4
9	7		
10	8	9	
11	10		
12	11		

Loop transformations
For the following problems, consider the code below:

```
1.   X = 2;
2.   Y = 10;
3.   if (X < Y) goto 13
4.     A = Y + 10;
5.     B = X * 4 + A;
6.     Z = 10;
7.     if (B < Z) goto 11
8.        D = Y + Z * 2;
9.        Z = Z - 1;
10.       goto 7;
11.    X = X + 2;
12.    goto 3;
13.  Y = D;
14.  halt;
```

1. Draw the CFG for the code above. Identify the loops in the code.

   Answer:



The loops are the following sets of instructions (with the first instruction as the loop header): {3, 4, 5, 6, 7, 8, 9, 10, 11, 12} and {7, 8, 9, 10}.

2. Which statements are loop invariant? Can they be moved outside their enclosing loop? Show the code that results after hoisting any loop invariant code outside the loop.

   The following statements are loop invariant:

   - Statement 4 (Y is only defined outside the loop)
   - Statement 6 (The rhs is constant)

Statement 4 can be hoisted (A is not live at any loop exit, it is only defined once inside the loop, and it is not live at the beginning of the loop). Statement 6 cannot (Z is defined more than once inside the loop).

The rewritten code looks like:

```
1.   X = 2;
2.   Y = 10;
4.   A = Y + 10;
3.   if (X < Y) goto 13
5.     B = X * 4 + A;
6.     Z = 10;
7.     if (B < Z) goto 11
8.        D = Y + Z * 2;
9.        Z = Z - 1;
10.       goto 7;
11.    X = X + 2;
12.    goto 3;
13.  Y = D;
14.  halt;
```

3. Identify the induction variables in this code. Show the code that results after performing any possible strength reduction.

   In the outer loop, X is an induction variable and B is a mutual induction variable (because A is loop invariant). In the inner loop, Z is an induction variable and D is a mutual induction variable (Y is loop invariant). After strength reduction, the code looks like:

```
1.   X = 2;
2.   Y = 10;
4.   A = Y + 10;
4'.  B' = X * 4 + A;
3.   if (X < Y) goto 13
5.     B = B'
6.     Z = 10;
6'.    D' = Y + Z * 2;
7.     if (B < Z) goto 11
8.        D = D'
9.        Z = Z - 1;
9'.       D' = D' - 2;
```

```
10.       goto 7;
11.    X = X + 2;
11'.   B' = B' + 8;
12.    goto 3;
13.  Y = D;
14.  halt;
```

4. Show the code after performing any possible linear test replacement.

```
1.   X = 2;
2.   Y = 10;
4.   A = Y + 10;
4'.  B' = X * 4 + A;
3.   if (B' < Y * 4 + A) goto 13
5.     B = B'
6.     Z = 10;
6'.    D' = Y + Z * 2;
7.     if (2 * B + Y < D') goto 11
8.        D = D'
9'.       D' = D' - 2;
10.       goto 7;
11'.   B' = B' + 8;
12.    goto 3;
13.  Y = D;
14.  halt;
```

Note that at this point, statement 6' is loop invariant (though it cannot be hoisted), and statement 6 is still loop invariant and *can* now be hoisted (since Z is no longer defined a second time in the loop).