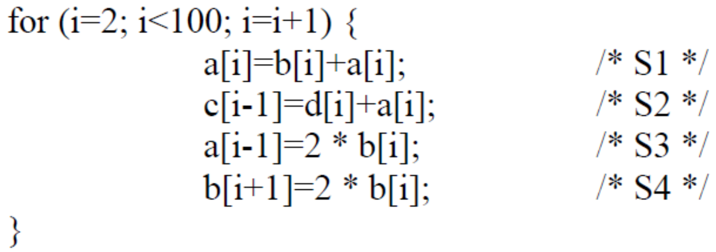
CS572 Homework3

1. [20 points] List all the dependencies (output, anti and true) in the following code fragment. Indicate whether the true dependences are loop-carried or not. Show why the loop is not parallel (hints: For a loop to be parallel, each iteration must be independent of all others; Loop-carried: data access in later iterations is dependent on data values produced in earlier iterations.)



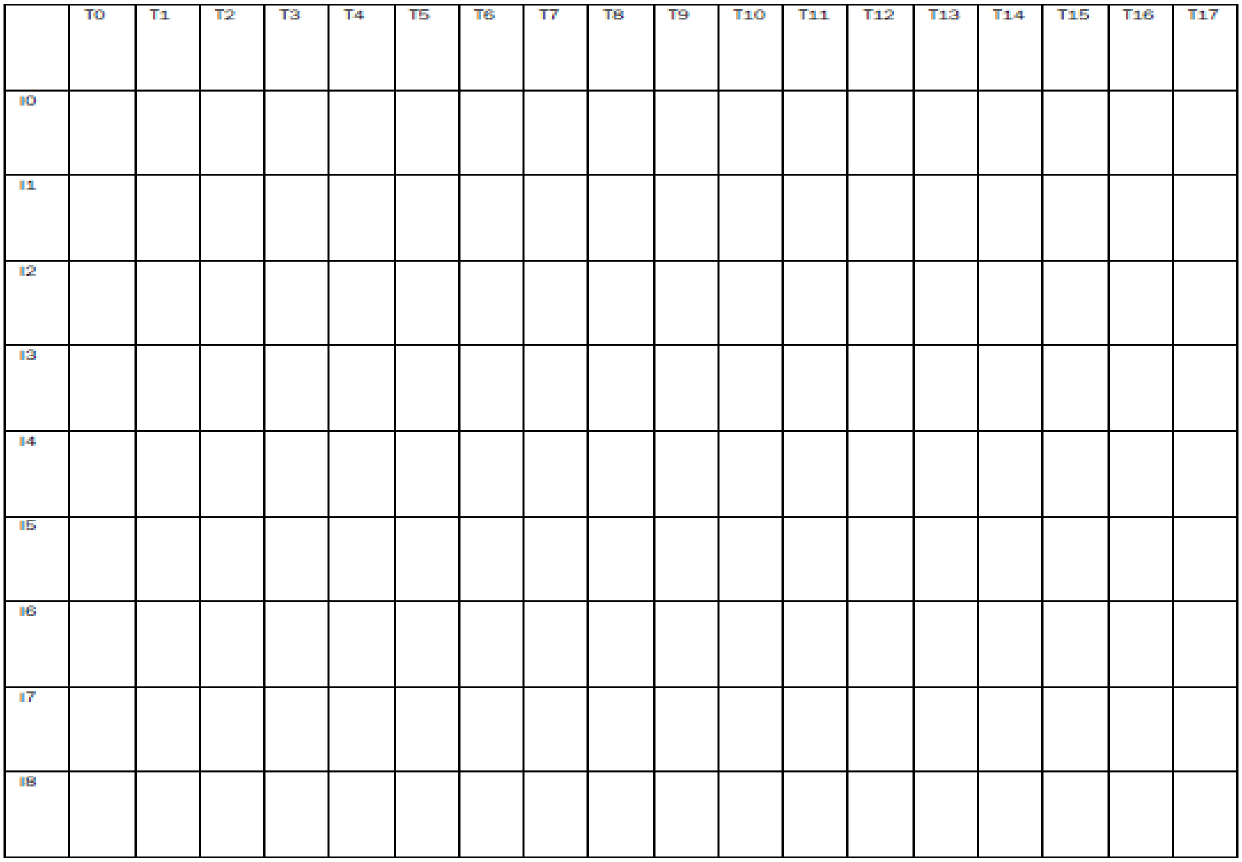
```
for (i=2; i<100; i=i+1) {
                a[i]=b[i]+a[i];                 /* S1 */
                c[i-1]=d[i]+a[i];               /* S2 */
                a[i-1]=2 * b[i];                /* S3 */
                b[i+1]=2 * b[i];                /* S4 */
}
```

2. [20 Points] Consider the following assembly language code:

I0: ADD R4 = R1 + R0;
I1: SUB R9 = R3 - R4;
I2: ADD R4 = R5 + R6;
I3: LDW R2 = MEM[R3 + 100];
I4: LDW R2 = MEM[R2 + 0];
I5: STW MEM[R4 + 100] = R2;
I6: AND R2 = R2 & R1;
I7: BEQ R9 == R1, Target;
I8: AND R9 = R9 & R1;

Consider a pipeline with forwarding, hazard detection, and 1 delay slot for branches. The pipeline is the typical 5-stage **IF, ID, EX, MEM, WB** MIPS design. For the above code, complete the pipeline diagram below (instructions on the left, cycles on top) for the code. Insert the characters **IF, ID, EX, MEM, WB** for each instruction in the boxes. Assume that there two levels of bypassing, that the second half of the decode stage performs a read of source registers, and that the first half of the write-back stage writes to the register file.

Label all data stalls (Draw an X in the box). Label all data forwards that the forwarding unit detects (arrow between the stages handing off the data and the stages receiving the data). What is the final execution time of the code?



	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17
I0																		
I1																		
I2																		
I3																		
I4																		
I5																		
I6																		
I7																		
I8																		

3. [20 points] This question covers your understanding of dependences between instructions. Using the code below, list all of the dependence types (RAW, WAR, WAW). List the dependences in the respective table (example INST-X to INST-Y) by writing in the instruction numbers involved with the dependence.

```
I0:  A = B + C;
I1:  C = A - B;
I2:  D = A + C;
I3:  A = B * C * D;
I4:  C = F / D;
I5:  F = A ˆ G;
I6:  G = F + D;
```

RAW Dependence		WAR Dependence		WAW Dependence	
From Instr	To Instr	From Instr	To Instr	From Instr	To Instr

4. [20 points] Consider the following MIPS assembly code.

```
LD       R1, 45(R2)
DADD     R7, R1, R5
DSUB     R8, R1, R6
OR       R9, R5, R1
BNEZ     R7, target
DADD     R10, R8, R5
XOR      R2, R3, R4
```

a. [10 points] Identify each dependence by type; list the two instructions involved; identify which instruction is dependent; and, if there is one, name the storage location involved.

b. [10 points] Assume the 5-stage MIPS pipeline (IF, ID, EX, MEM, WB), and a register file that writes in the first half of a clock cycle and reads in the second half. Which of the dependencies that you listed in part (a) become hazards, and which do not? Why?

5. [20 points] This exercise is intended to help you understand the relationship between delay slots, control hazards, and branch execution in a pipelined processor. In this exercise, we assume that the following MIPS code is executed on a pipelined processor with a 5-stage pipeline, full forwarding, and a predict-taken branch predictor:

```
Label1:  LW $1, 40 ($6)
         BEQ $2, $3, Labe2; Taken
         ADD $1, $6, $4
Label2:  BEQ $1, $2, Label1; Not Taken
         SW $2, 20($4)
         AND $1, $1, $4
```

a. [10 points] Draw the pipeline execution diagram for this code, assuming there are no delay slots used and that branches execute in the EX stage.

b. [10 points] Repeat 5.a, but assume that delay slots are used. In the given code, the instruction that follows the branch is now the delay slot instruction for that branch.