READ ME

Program functions:

1. **Void Instruction_decode(asmcode):**
   It reads each line of the program and stores it into the format of the object of Instruction_pattern class and save it into ArrayList of Instruction_pattern type.

2. **Void Data_dependency():**
   It finds out the data dependencies available in the assembly language code.

3. **Void Control_dependency():**
   It finds out the control dependencies available in the assembly language code.

4. **Void Structural_dependency():**
   It finds out the structural dependencies available in the assembly language code.

Input:

Input to the program is the input file "Testcase.asm" and the file should be in the same as of program.
Testcase.asm: Input file which contains the assembly language code.

Output:

- The first block of output shows the Data dependency in the assembly language code.
- The second block of output shows the Control dependency in the assembly language code.
- The third block of output shows the structural dependency in the assembly language code.

Dependencies:

Data Dependency:

Data dependence arises from two statements which access or modify the same resource that can be true dependency, anti-dependency, and output dependency.

Control Dependency:(Source Wikipedia)

Control dependency is a situation in which a program instruction executes if the previous instruction evaluates in a way that allows its execution.

A statement *S2* is *control dependent* on *S1* if and only if *S2*'s execution is conditionally guarded by *S1*. The following is an example of such a control dependence:

```
S1      if x > 2 goto L1
S2      y := 3
S3   L1: z := y + 1
```

Here, *S2* only runs if the predicate in *S1* is false.

Structural Dependence:

- We considering the instruction and data memory are the same.
- Instructions which results in **memory conflict** are displayed.
- Data dependency can result in structural dependency because of register conflicts that are not displayed in the output because we already show it in data dependency.

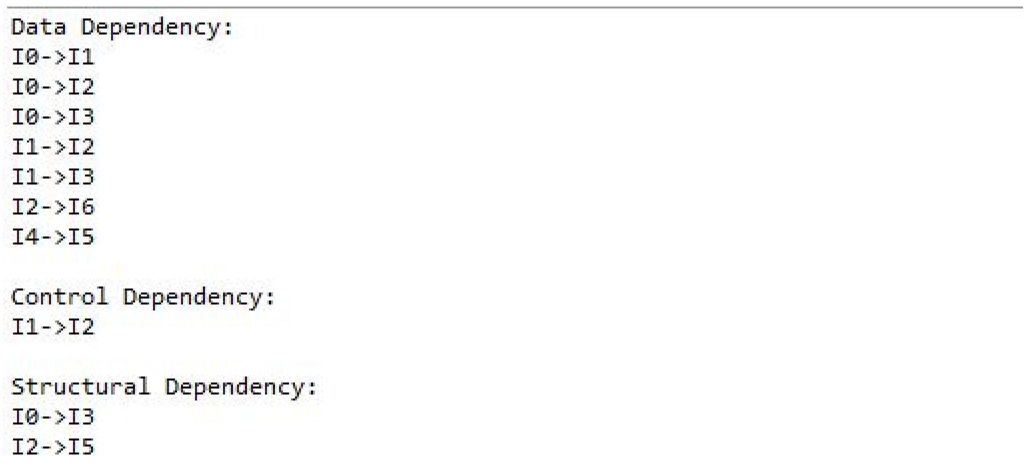
Assumptions:

- Consider the Base address of the assembly language code is 100H.
- Code can handle some standard instruction like add, sub, div,mul, data transfer, logical, conditional and unconditional branch.
- We considering the instruction and data memory are the same.
- To avoid errors input should be given in the same format as shown in Testcase.asm file.

Test Case1:

```
lw $1,100($2)
bne $1,$2,112
sw $2,200($1)
add $4,$1,$2
add $11,$2,$3
and $13,$11,$3
sw $11,100($2)
```

Output:



```
Data Dependency:
I0->I1
I0->I2
I0->I3
I1->I2
I1->I3
I2->I6
I4->I5

Control Dependency:
I1->I2

Structural Dependency:
I0->I3
I2->I5
```

Test case2:

```
sub $2,$1,$3
lw $1,100($2)
and $12,$2,$5
or $13,$6,$2
add $14,$2,$2
blt $12,$1,108
add $6,$19,$20
sw $15,100($2)
j 190
sw $15,100($2)
```

OUTPUT:

```
Data Dependency:
I0->I1
I0->I2
I0->I3
I0->I4
I0->I7
I0->I9
I1->I5
I2->I5
I3->I6
I7->I9

Control Dependency:
I5->I2
I5->I3
I5->I4
I5->I5
I5->I6
I5->I7
I5->I8
I5->I9

Structural Dependency:
I1->I4
```