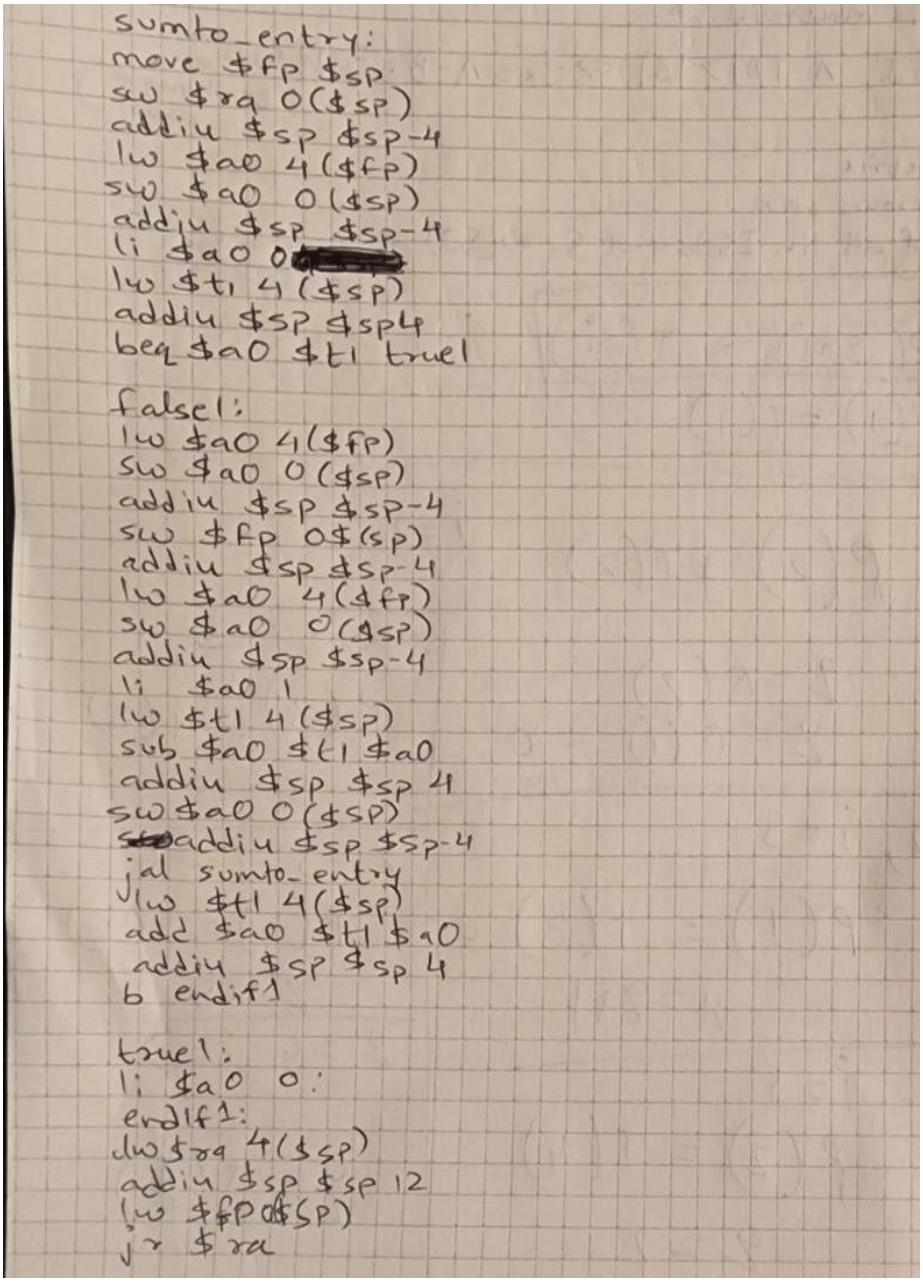
CS 6353.001, Compiler Construction, Spring 2022
Homework Assignment #3

I Code Generation

Generate MIPS code for the following program using the code generation functions discussed in the code generation lectures. This is the same code provided in slide 111 of lecture 11. You can check the result AFTER you work out the problem.

```
def sumto(x) = if x = 0 then 0 else x + sumto(x-1)
```



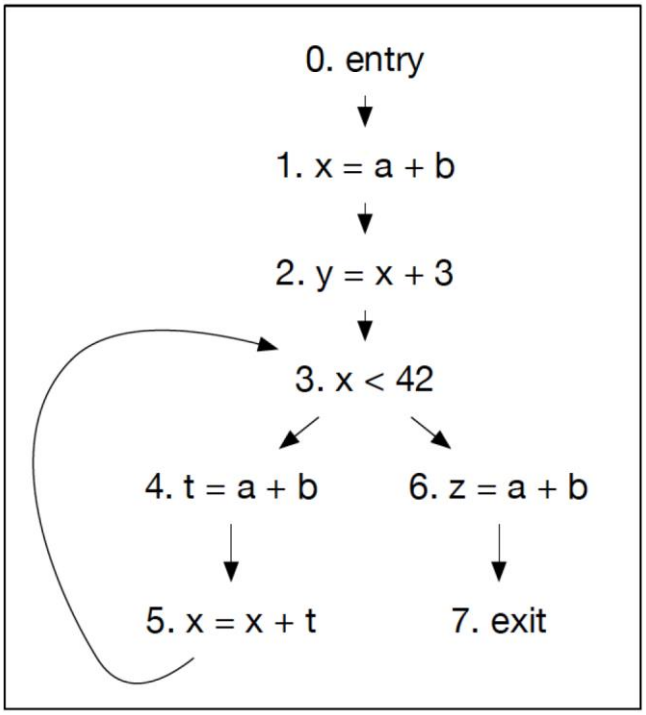
```
sumto_entry:
move $fp $sp
sw $ra 0($sp)
addiu $sp $sp-4
lw $a0 4($fp)
sw $a0 0($sp)
addiu $sp $sp-4
li $a0 0
lw $t1 4($sp)
addiu $sp $sp4
beq $a0 $t1 true1

false1:
lw $a0 4($fp)
sw $a0 0($sp)
addiu $sp $sp-4
sw $fp 0$(sp)
addiu $sp $sp-4
lw $a0 4($fp)
sw $a0 0($sp)
addiu $sp $sp-4
li $a0 1
lw $t1 4($sp)
sub $a0 $t1 $a0
addiu $sp $sp 4
sw$a0 0($sp)
saddiu $sp $sp-4
jal sumto_entry
lw $t1 4($sp)
add $a0 $t1 $a0
addiu $sp $sp 4
b endif1

true1:
li $a0 0.
endif1:
dw$ra 4($sp)
addiu $sp $sp 12
lw $fp0($sp)
jr $ra
```

PART II Dataflow Analysis

In the following table, show each iteration of *very busy expressions* for the control-flow graph below. For each iteration, list the statement taken from the worklist (*wl*) in that step, the value of *in* computed for that statement, and the new worklist at the end of the iteration. You may or may not need all the iterations; you may also add more iterations if needed. Do not add the exit node to the worklist. Assume that x<42 is not a possible very busy expression. Use one of the algorithms in slides 57-58 of lecture 14 to work out this problem.



Use ϕ for the set of no expressions, and T for the set of all expressions. What is T?
T = (a+b, x+3, x+t)

What are the initial *in's* for each statement?

Stmt	0	1	2	3	4	5	6
Init *in*	**T**	**T**	**T**	**T**	**T**	**T**	**T**

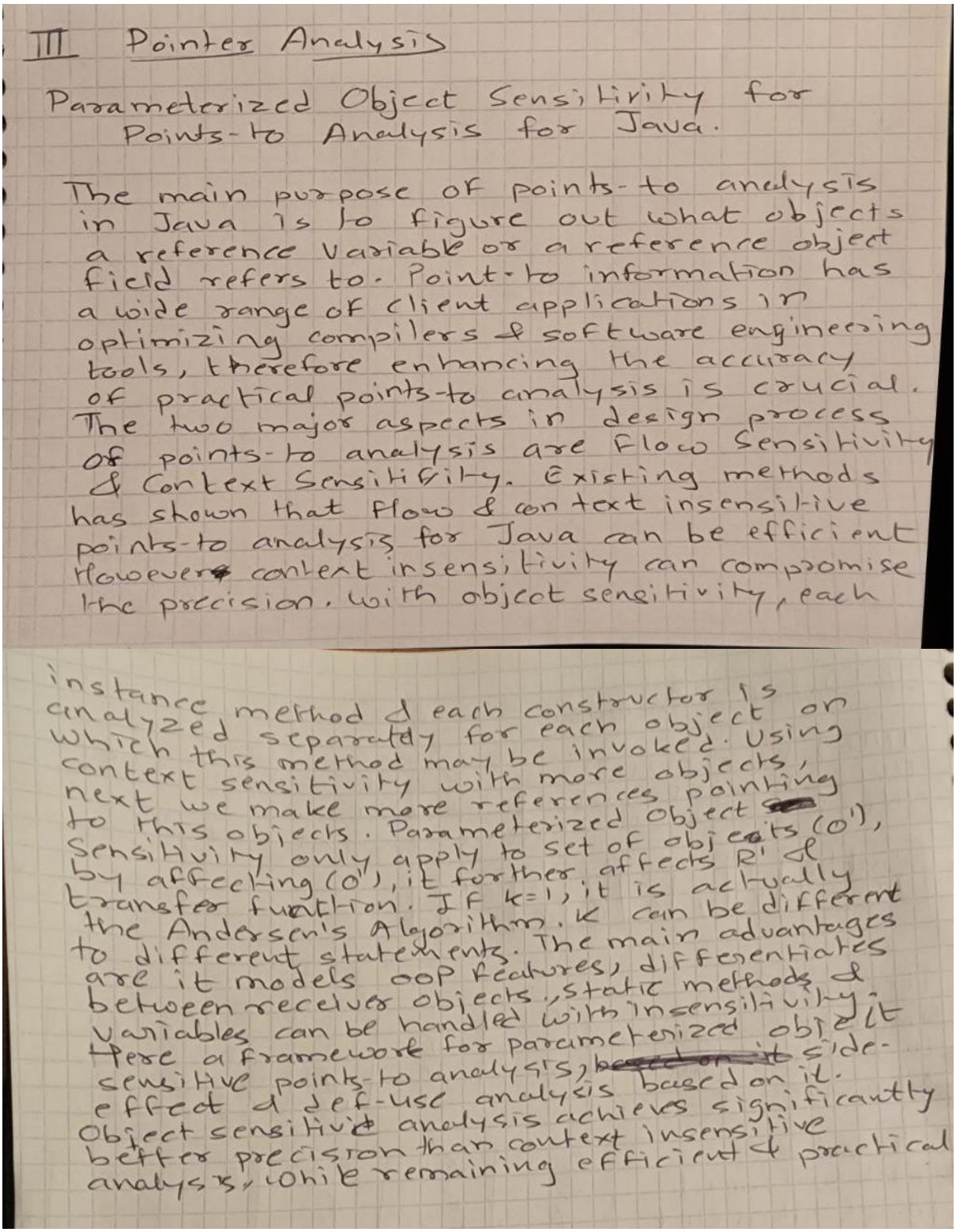
Iteration	0	1	2	3	4
Stmt taken from *wl*	6	5	4	3	2
in of taken stmt	a+b	x+t, a+b	a+b	a+b	a+b, x+3
New worklist	5,4,3,2,1,0	4,3,2,1,0	3,2,1,0	2,1,0,5	1,0,5

Iteration	5	6	7	8	9
Stmt taken from *wl*	1	0	5		
in of taken stmt	a+b	a+b	x+t, a+b		
New worklist	0,5	5			

PART III Pointer Analysis

Read this research paper that describes a technique called object-sensitive analysis:
http://web.cse.ohio-state.edu/presto/pubs/issta02.pdf

Write a one-paragraph summary of this paper (<350 words). You will receive full credit if the summary accurately captures the <problem, solution, result> and is not directly copied from the text in the paper. One problem in Exam II will rely on your understanding of this analysis.



III Pointer Analysis

Parameterized Object Sensitivity for Points-to Analysis for Java.

The main purpose of points-to analysis in Java is to figure out what objects a reference variable or a reference object field refers to. Point-to information has a wide range of client applications in optimizing compilers & software engineering tools, therefore enhancing the accuracy of practical points-to analysis is crucial. The two major aspects in design process of points-to analysis are flow sensitivity & Context Sensitivity. Existing methods has shown that flow & context insensitive points-to analysis for Java can be efficient However context insensitivity can compromise the precision. With object sensitivity, each instance method & each constructor is analyzed separately for each object on which this method may be invoked. Using context sensitivity with more objects, next we make more references pointing to this objects. Parameterized object sensitivity only apply to set of objects (o'), by affecting (o'), it further affects R' & transfer function. If k=1, it is actually the Andersen's Algorithm. K can be different to different statements. The main advantages are it models OOP features, differentiates between receiver objects, static methods & variables can be handled with insensitivity. Here a framework for parameterized object sensitive points-to analysis, based on it. effect & def-use analysis based on it. Object sensitivity analysis achieves significantly better precision than context insensitive analysis, while remaining efficient & practical