CS352 Homework 4 (Due March 31, 2023, 11:59pm, on Gradescope)

IMPORTANT NOTE: Homework solutions must be submitted on Gradescope and must satisfy the requirements listed in Homework 1 handout.

NOTES Common to All Problems in HW2 We assume all IDs are either symbolic registers or labels. There is only one special hardware register %sp, which is the stack pointer used to access the current stack frame. A function call is made by instruction "bl <function_label>". There is a return instruction "*return*" or "*return a*", where a is a symbolic register. In this course, the "bl" instruction is not treated as a branch when constructing the basic block, because, upon return of the call, the program continues to execute the next instruction.

The remaining instruction formats are listed below. The assignment operator in a 3AC instruction may be written as "←" or "=".

- Each line contains exactly one instruction in its entirety, which may be followed by a comment in the same line or separate lines. A comment must follow a ";" Each comment line must have its own ";"
- ALU operations: Rx = Ry op *op2*, where Rx and Ry are symbolic registers and op2 is either a symbolic register or a constant.
- In the above, if op is a logical or comparison operation, then Rx becomes 0 if the result is false and becomes 1 if true.
- Register copying: Rx = op2, where Rx is a symbolic register and op2 is either a symbolic register or a constant. (We do not worry about the size of the constant in operations in the above.)
- A symbolic label is an ID followed immediately by a ":"
- Whether the instruction following the label appears in the same line or a new line does not matter.
- Conditional branch operations: if cond goto Label, where cond is either a symbolic register or a single comparison such as *a* > *b.* Label is a symbolic label mentioned above.
- Unconditional operations: goto Label
- **Load operations***: Rx = [r, #offset],* where #offset is a symbolic constant indicating a byte offset in the current stack frame. Rx is a symbolic register, and r may be either a symbolic register or %sp.
- **Store operation**: *[r, #offset] = Rx. The format is similar to that of load operations.*

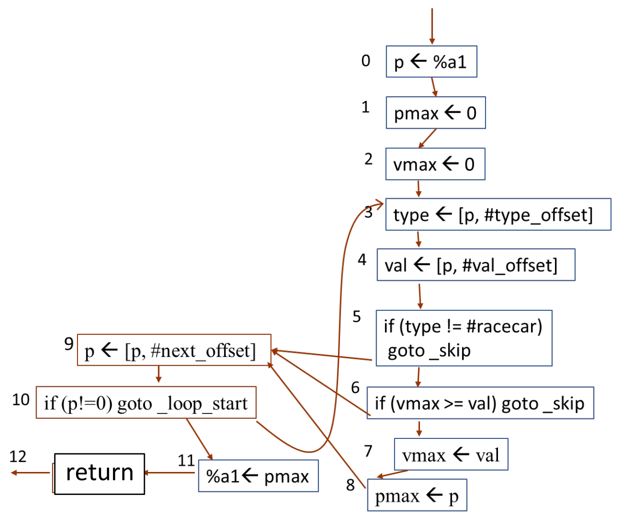
Problem 1 (30 pts) You are given the following 3AC:

```
(1) i ← 0
(2) j ← 0
(3) k ← 0
(4) L4: if k > 30 goto L11
(5)   if j > k goto (9)
(6)     i ← k
(7)     j ← j+1
(8)    goto L4
(9)    k ← k +1
(10)   goto L4
(11) L11: return i
```

(1.1) (10 pts)
Suppose we let *each block contain only one instruction*. Draw the control flow graph such that each node is marked by the number marked on the corresponding instruction in the given 3AC.

(1.2) (20 pts) Partition the given code segment into basic blocks such that each block contains as many instructions as possible. Draw a control flow graph such that each node represents a basic block. For each basic block, list the range of instructions it contains.

Problem 2 (30 pts) For the control flow graph below (in the left), please draw a table to list USE(B) and DEF(B) for each block B. Next, ***use a worklist*** to compute LIVEin(B) and LIVEout(B) for each block B. (Note: # followed by an ID denotes a symbolic constant, not a symbolic register. %a1 is an argument register and should be included in the computation of the USE, DEF, LIVEin and LIVEout sets. We assume %a1 is in LIVEout of block 12.)



f1:

```
1.  a←%a1
2.  b←%a2
3.  Loop: c1 ←a+b
4.  d← c1 * 2
5.  c2 ←a-c1
6.  b ←c2 * 2
7.  f ←d / b
8.  If (a > b) goto Loop
9.  %a1←f
10. return
```

Problem 3 (40 Pts) You are given the 3AC code for function **f1** in the text box above. All variables are treated as symbolic registers**. %a1 and %a2 are argument registers.**

3.1 (20 pts) Assuming *minimum-block* partition, i.e., each instruction in the 3AC forming a basic block, please use a worklist to compute LIVEin(B) and LIVEout(B) for each block B. The liveness computation must cover %a1 and %a2, in addition to all symbolic registers. %a1 is assumed to be in LIVEout of block 10. (NOTE: Listing of USE/DEF sets is optional.)

3.2 (20 pts) Draw the interference graph, which should include %a1, %a2 and all symbolic registers.