University of Cape Town

Department of Computer Science

CSC3005h Class Test

2006

Marks **: 35**

Time **: 45 minutes**

Instructions:

- Answer all questions.
- Show all calculations where applicable.

Question 1: Symbol Tables and Activation Records [10]

a) What is a symbol table? [2]

a list of names and associated attributes

b) Briefly describe 3 context-sensitive tests that can be conducted with the aid of a symbol table. [3]

declaration before use

duplicate declarations

unused variables

array within bounds

etc.

c) What is an activation record? [2]

the layout of data necessary to support invocation of a subprogram

d) In many instances activation record fields are stored in registers for faster execution. Briefly describe one instance where stack memory may be necessary. [1]

nested subprograms, too many variables, large data structures, etc.

e) What is the purpose of the static link in a frame for a statically-scoped language? [2]

points to static parent to resolve non-local references

Question 2: Intermediate Code [15]

a) Discuss 2 advantages of using intermediate representations. [2]

separation of front/back ends, easier to apply optimisations to

b) Using the attached IR language, convert the following C-like expression to an unoptimised IR tree. Assume **b** and **c** are stack variables at offsets *k_b* and *k_c* respectively from the frame pointer TEMP(FP). Assume **a** and **x** are as-yet-undefined constants. Provide the final tree and do not use the Nx/Cx/Ex expression types/objects. [3]

```
b = a + 2 + 5; c = 1 + x + 3
```

SEQ(MOVE(MEM(+(TEMP(FP),CONST(k_a))), +(+(CONST(a),CONST(2)),CONST(5))),

MOVE(MEM(+(TEMP(FP),CONST(k_c))), +(+(CONST(1),CONST(x)),CONST(5))))

Minus one mark for each major error.

c) Generate a new tree, applying constant folding as an optimisation. [2]

SEQ(MOVE(MEM(+(TEMP(FP),CONST(k_a))), +(CONST(a),CONST(7))),

MOVE(MEM(+(TEMP(FP),CONST(k_c))), +(CONST(4),CONST(x))))

Minus one mark for each major error.

d) Discuss 2 other optimisations that may be applied to IR trees. [2]

inlining – replace subprogram calls with body of subprogram

unreachable code elimination – remove code that nevers gets executed

constant propagation – convert uses of a constant name to its value

etc.

e) What is a basic block? [2]

a linear sequence of statements starting with a label and ending with a jump

f) What benefit is there in rearranging basic blocks into traces? [2]

we could possibly eliminate redundant jumps

g) Eliminate the ESEQs from the following IR tree by converting it to a canonical form, using the attached simplification rules. Show all steps. [2]

```
MOVE ( ESEQ ( LABEL L1, ESEQ ( LABEL L2, TEMP a )), CONST 5 )
```

MOVE (ESEQ (SEQ (LABEL L1, LABEL L2), TEMP a), CONST 5)

SEQ (SEQ (LABEL L1, LABEL L2), MOVE (TEMP a, CONST 5))

Question 3: Code Generation [10]

a) Use the iterative liveness analysis algorithm to calculate the live-in and live-out sets for each of the following statements in a program. Show succ, use, def, out and in sets. [8]

if (x > 1)

then y = x * x;

else y = (1 / x) * (1 / x);

return y+1;

Hint: The relevant formulae are:

$$out[n] = \underset{s \in succ[n]}{} in[s]$$

$$in[n] = use[n] \cup (out[n] - def[n])$$

Succ [1]	*#*	*Code*	*Use [1]*	*Def [1]*	*Out [2]*	*In [2]*
23	*1*	*If x > 1*	*X*		*X*	*X*
4	*2*	*Y = x^2*	*X*	*Y*	*Y*	*X*
4	*3*	*Y = (1/x)^2*	*X*	*Y*	*Y*	*X*

	4	*Return y+1*	*Y*			*Y*

convergence (last 2 columns repeated) [1]

b) Draw an interference graph for this program. What is the minimum number of registers needed to support execution of this program? [2]

X Y

1