University of Cape Town

Department of Computer Science

COURSE : CSC305H

MODULE : COMPILERS

TEST : 2 (SUPP)

DATE : 17 AUGUST 2005

TOTAL MARKS : 35

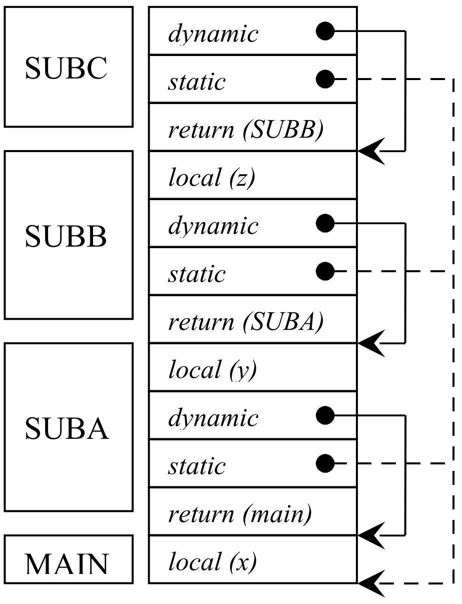
INSTRUCTIONS TO STUDENTS

- Answer ALL questions

Question One – Symbol Tables and Activation Records [10]

Draw the stack of activation records corresponding to the following C-like program when it is at "breakpoint". [4] (Assume static chains and include all parameters).

```
main {
   int x;
   sub SUBA {
      int y;
      sub SUBB {
         int z;
         sub SUBC {
            //breakpoint
         }
         SUBC;
      }
      SUBB;
   }
   SUBA;
}
```



Marks: structure(4x1) or locals(1), statics(1) dynamics(1) returns(1)

Nested subprograms can require saving and restoring of registers used to pass parameters, but this save/restore operation does not always have to be done. Discuss 3 circumstances under which a save/restore of parameter-passing registers is not necessary even though subprograms are nested. [3]

leaf procedures[1], different registers[1], done with variables[1], register windows[1]

Discuss 3 circumstances under which it is necessary to use memory to pass parameters instead of just using registers. [3]

variables used/passed by reference[1], nested subprograms[1], variable is not simple or just too big[1], arrays[1], registers are needed for other purposes[1], too many variables[1]

Question Two – Intermediate Representation [10]

Discuss 2 disadvantages of intermediate representations. [3]

more computation[1.5], does make optimal use of machine code[1.5], another language to learn[1]

Assuming the IR tree language in the attached pages, convert the following program fragment to an equivalent IR tree. (Assume x/y are stack frame variables at offset k0/k1 from the frame pointer special temporary fp) Provide the final tree and do not use the Nx/Cx/Ex expression types/objects. [4]

```
while (x+1<y) { x=x+1; y=y-1; }
```

```
SEQ (SEQ( SEQ( SEQ( SEQ( SEQ( LABEL S,

CJUMP (LT, BINOP (PLUS,MEM(BINOP(PLUS,FP,k0)),CONST 1), MEM( BINOP( PLUS,FP,k1)), NAME T, NAME F)),

LABEL T),

MOVE (MEM(BINOP(PLUS,FP,k0)), BINOP (PLUS, MEM( BINOP( PLUS, FP, k0)),CONST 1))),

MOVE (MEM(BINOP(PLUS,FP,k1)), BINOP (MINUS, MEM( BINOP( PLUS, FP, k1)),CONST 1))),

JUMP (NAME S)),

LABEL F)
```

Marks: labels[1],cjump[1],memory access[1],moves[1]

Convert the following tree into its canonical form by applying transformations from the attached list. Show the result after each transformation. [3]

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

MOVE (ESEQ (SEQ (LABEL L1, LABEL L2), TEMP a), CONST 5) [1.5]

SEQ (SEQ (LABEL L1, LABEL L2), MOVE (TEMP a, CONST 5)) [1.5]

Question Three – Instruction Selection [8]

What is the difference between an optimal and optimum tiling? Give one example of an algorithm in each class, and state what the Big-O complexity of each algorithm is. [4]

Optimal tiling – no tiling can result in a lower cost – Maximal Munch – O(N). [1/2 x 4]

Optimum tiling – no two adjacent tiles can be replaced by one with lower cost – Dynamic Programming – O (N). [1/2 x 4]

Using the attached instruction set, apply the Maximal Munch tiling algorithm to the following IR tree. Show the tiled tree and list the instructions generated. [4]

MOVE (MEM (CONST a), MEM (PLUS (CONST b, CONST c)))

ADDI (CONST c)

LOAD (MEM + CONST b)

MOVE (MOVE MEM CONST a)

Marks: Tree (nodes in brackets): 2, Instructions: 2

Question Four – Register Allocation [7]

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. [7]

```
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*