University of Cape Town

Department of Computer Science

COURSE : CSC305H

MODULE : COMPILERS

TEST : 2

DATE : 17 AUGUST 2005

TOTAL MARKS : 35

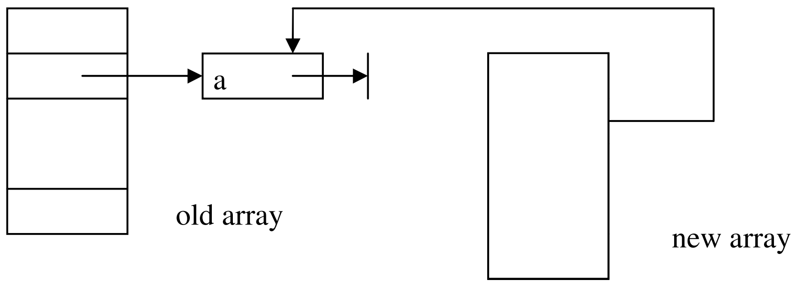
INSTRUCTIONS TO STUDENTS

- Answer ALL questions

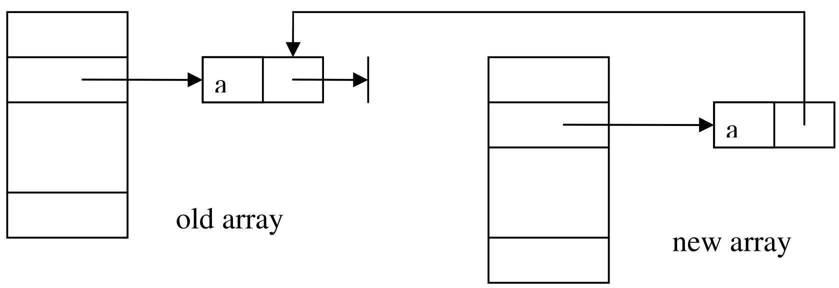
Question One – Symbol Tables and Activation Records [10]

Discuss, with diagrams, how a hash table in a functional symbol table is modified when a scope is opened and then when a new variable in the new scope definition overrides an existing one from a previous scope. [5]

Beginning: The array is duplicated. [3]



Addition of variable: A new node is added to the head of the new array, chained to the old variable node. [2]



Consider the following Pascal-like program with static scoping:

```
program test;
var a : integer;

   procedure proc1;
   var b : integer;
   begin (* POINT1 *)
   end;

   procedure proc2;
   var a, c : integer;
   begin
      (*POINT2 *) proc1; (*POINT3 *)
   end;

begin
   (*POINT4 *) proc2; (*POINT5 *)
end.
```

Which variables (state variable name and subprogram name) are in scope at each of the 5 points? [5]

POINT1 : proc1/b, test/a [1]

POINT2 : proc2/a, proc2/c [1]

POINT3 : proc2/a, proc2/c [1]

POINT4 : test/a [1]

POINT5 : test/a [1]

Question Two – Intermediate Representation [10]

Discuss 2 advantages of intermediate representations. [3]

frontend+backend separation/portability[1.5], easier for some optimisations[1.5], simpler to generate[1.5]

Assuming the IR tree language in the attached pages, convert the following program fragment to an equivalent IR tree. (Assume a/b 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]

```
if (a<b) a=1 else b=1;
```

SEQ (SEQ (SEQ (SEQ (SEQ (SEQ (CJUMP (BINOP (LT, a, b), NAME T, NAME F),

LABEL T),

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

JUMP (NAME D)),

LABEL F),

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

LABEL (D))

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]

JUMP (ESEQ (LABEL L1, MEM (ESEQ (LABEL L2, TEMP t))))

JUMP (ESEQ (LABEL L1, ESEQ (LABEL L2, MEM (TEMP t)))) [1]

JUMP (ESEQ (SEQ (LABEL L1, LABEL L2), MEM (TEMP t))) [1]

SEQ (SEQ (LABEL L1, LABEL L2), JUMP (MEM (TEMP t))) [1]

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]

MEM (PLUS (CONST a, TIMES (CONST b, MEM (PLUS (CONST c, CONST d)))))

ADDI (CONST d)

LOAD (MEM + CONST c)

ADDI (CONST b)

MUL (*)

LOAD (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]

```
a = c
b = d
if ( a < b )
  then m = b;
  else m = a;
return m;
```

Hint: The relevant formulae are:

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

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

Succ [1]	#	Code	Use [1]	Def [1]	Out [2]	In [2]
2	1	a=c	C	A	AD	CD
3	2	b=d	D	B	AB	AD
45	3	if (a < b)	Ab		AB	AB
6	4	m = b	B	M	M	B
6	5	m = a	A	M	M	A
	6	return m	m			M