CSL302: Compiler Design

End Sem Examination (2025-26-M Semester)

Max. Points: 100 **Duration:** 3 hours

November 28, 2025

1. All questions are compulsory
2. Make your assumptions and state them wherever necessary

Question-1 (Semantic Analysis): **[15 Points]**

Part-(a)

Consider the following grammar that generates expressions for a number of operators. Assume you have synthesized attributes **E.min** and **E.max** which should be set to the minimum and maximum values for E, and synthesized attribute **CONST.val** which is the value of the constant. Write the semantic rules that calculate the range (minimum and maximum values) of each subexpression in the respective attributes.

E	→	CONST	{ E.min = ?? ; E.max = ?? }
	|	ID	{ E.min = ID.min ; E.max = ID.max }
	|	E1 + E2	{ E.min = ?? ; E.max = ?? }
	|	E1 − E2	{ E.min = ?? ; E.max = ?? }
	|	E1 ∗ E2	{ E.min = ?? ; E.max = ?? }
	|	−E1	{ E.min = ?? ; E.max = ?? }

Part-(b):

Let synthesized attribute F.val give the value of the binary fraction generated by F in the grammar that follows:

F → .L
L → LB | B
B → 0 | 1

For instance, on input .101 we have that F.val = .625

(a) Using only synthesized attributes, write the semantic rules to compute F.val for the above grammar.
(b) Show the translation of the input string .101 by annotating its parse tree

Question-2 (Syntax Analysis): **[20 Points]**

Consider the following grammar.

$$
\begin{aligned}
S &\rightarrow B \\
B &\rightarrow B\ E\ + \\
&\mid\ -E \\
E &\rightarrow e \\
&\mid\ Ee \\
&\mid\ \in
\end{aligned}
$$

(a) Is the grammar LR(1)? Justify your answer by showing DFA and the LR(1) parser table.

(b) Is the grammar LALR? Justify your answer by reconstructing DFA and the corresponding LALR parser table.

Question-3 (Intermediate Code Generation): **[20 Points]**

Assume that your programming language supports control flow instructions for the "For Loop" using the following grammar

foreach-statement → foreach(ID in [START..END]) S ;

Semantics are as follows:

1. ID gets initialised to constant values START and executes S iteratively as long as the ID<=END.
2. At the end of each iteration ID gets incremented by 1.

Part-(a): Write down the semantic rules to generate the three address codes as intermediate representation.

Hints: (1) You can use the backpatch function discussed in the class. (2) You can introduce the markers (M, N, etc) into the above grammar with suitable epsilon transitions to pick up index of next quadruple

Part-(b): Generate the three address codes for the following statement with the above semantics.

```
foreach(i in(1..20))
        j=j*b+d;
```

Question-4 (Code Generation): **[30 Points]**

Assume that an architecture supports the ternary operations ⊕, ⊙, ⊖. Each of which accepts 3 operands.

Example usage of the operators is: OP(a,b,c), where OP can be ⊕, ⊙, or ⊖. These operators perform some computation based on the values provided by a, b, and c.

Assume that we have the following target machine model.

Instruction	**Semantics**
Store m, r	Stores the contents of register r to the memory location m
Load r, m	Loads the contents of the memory location m to the register r
OP r2, r2, m, r1	Performs the ternary operation of register r2, memory operand m and register operand r1. The result of OP r2, m, r1 is stored in r2. Here OP can be ⊕, ⊙, or ⊖
OP r3, r3, r2, r1	Performs the ternary operation of register r3, register r2, and register r1. The result of OP r3, r2, r1 is stored in r3. Here OP can be ⊕, ⊙, or ⊖

(a) Modify the labelling algorithm that we discussed in the class to calculate the minimum number of registers required to compute an expression tree supporting the above target machine model.

(b) Modify the Sethi-Ullman Algorithm to generate the code. Discuss various cases of your algorithm for code generation. You can assume that your target has a sufficient number of registers, so you don't need to consider the case of moving to temporary locations.

(c) Use the modified labelled algorithm and the code generation algorithm to generate the code for the following expression

⊙(⊕(a,b,c),d,⊕(⊙(e,f,g),⊖(h,i,j),k))

You must draw the expression tree and list the label for each node in the tree, and the final sequence of assembly instructions that are generated for the expression.

Question-5 (Machine Independent Optimizations): **[15 Points]**

Consider the following snippet of the code.

```
a=1;
b= a+b;
e= c+d;
if((a<b) or (c>d) )
{
        j=4;
        while ((e<f ) and (a!= (e+d))) {
                e=e+f ;
                a=c+d ;
        }
}
else
{
        b=c+d;
}
```

(a) Generate the three-address code representation for the above program. Assume that the implementation uses short circuit evaluation for Boolean operators. You can assume the precedence of the associativity follows that of C language.

(b) Apply the loop invariant code motion and global common sub expression elimination to the above three address codes.

(c) Apply the copy-propagation for the code generated for the code generated in Part(b) and show the optimized code.