Open book exam. Duration: 2h30m. ***Exame de Recurso*** **(second round exam)**

Students that intend to replace the grade of a single mini-test (midterm exam) have to answer PART I (replacement of mini-test MT1) or PART II (replacement of mini-test MT2). In that case the duration of the exam is **1h15m**. Students enrolled in grade improvement have to answer the complete exam.

PART I (10 pts)

Group 1. (5 pts)

Considering the following grammar (parentheses and numbers on the right identify the productions):

S → xB | yC (1, 2)
B → Aa (3)
C → Aba (4)
A → b | ε (5, 6)

1.a) [1pt] Indicate the First and Follow sets for the grammar variables.

1.b) [1pt] Indicate if this grammar is LL(1) and show the LL(1) parsing table.

1.c) [1pt] Determine and show the LR(0) automaton.

1.d) [1pt] Is the grammar LR(0)? Justify your answer indicating the LR(0) parsing table.

1.e) [1pt] The following grammar is not LL(k). Indicate changes to the grammar that make it LL(k).

ID = [a-z][0-9a-z]*
S → E = E ;
S → E ;
E → E * E
E → E + E
E → E (E)
E → ID

Group 2. (3 pts)

The goal is to define a programming language with a grammar that forces applying certain semantic rules. A team designing the language finds itself discussing the possibilities to verify if the data type defined as return in a function header coincides with the data type in the *return* instructions used in the code of the same function.

2.a) [1pt] Indicate the necessary aspects and possible restrictions that need to be taken into account in terms of such programming language to make the semantic rules mentioned above implemented by the grammar;

2.b) [2pt] Indicate possible grammar rules that illustrate the way those semantic rules can be implemented considering the int, float, and double data types. Notice that it is only necessary to indicate sections of the grammar that illustrate the way the semantic rules can be implemented by the grammar.

Group 3. (2 pts)

3.a) [2pt] Comment the following sentence: "The high level intermediate representation obtained right after semantic analysis is just an AST with variable identifiers replaced by the access type (e.g., *load* array, *store* local variable, *load* local variable, *load* function parameter)."

Open book exam. Duration: 2h30m. ***Exame de Recurso* (second round exam)**

Students that intend to replace the grade of a single mini-test (midterm exam) have to answer PART I (replacement of mini-test MT1) or PART II (replacement of mini-test MT2). In that case the duration of the exam is **1h15m**. Students enrolled in grade improvement have to answer the complete exam.

PART II (10 pts)

Group 4. (8 pts)

4.a) [2pt] Indicate the low level intermediate representation (LLIR) for the section of the code in the example below (where N represents a 32-bit *int* constant) based on expression trees, but considering the *Jouette+* processor (instruction set presented in annex), and the type and storage of the variables given by the following table.

Variable	Type	Storage
A	int8 A[N] (array of integers (8-bit))	array stored in the stack starting at SP + 4
i	int i (32-bit scalar variable)	Variable stored in register r3
m	int m (32-bit scalar variable)	Variable stored in register r2
x	int x (32-bit scalar variable)	Variable stored in register r1

```
//in={}
1.  i = 0;
2.  m = A[0];
    do {
3.      i=i+1;
4.      x = A[i];
5.      if (x > m)
6.          m = x;
7.  } while(i<N);
// out = {m}
```

4.b) [2pt] In this LLIR for the *Jouette+* processor, code of the type A = B + C, supposing that R2, R3, and R4 are the registers that store A, B and C, respectively, is represented by two instructions, such as illustrated by the following example: MOVER: R2 ← R5; ADD: R5 ← R3 + R4. However, it is possible to make the selection of instructions resulting in the direct generation (i.e., without optimizations and/or elimination of instructions after selection) of the code ADD: R2 ← R3 + R4. In order to do this, indicate the pattern trees of the IR that make the selection of instructions more efficient in the context of the *Jouette+* processor.

4.c) [2pt] Taking into account the costs per instruction of the *Jouette+* processor presented in the table below, indicate an example in which the Maximal Munch algorithm is unable to achieve a selection of instructions that corresponds to the minimum global cost. Indicate which would be the minimum global cost for that example and indicate how the use of dynamic programming can obtain a selection of instructions with that cost.

Instruction	Cost	Instruction	Cost	Instruction	Cost	Instruction	Cost
MADD	5	LOAD	3	ADD	2	MOVEM	4
MUL	3	STORE	2	ADDI	1	Other instructions	1

4.d) [2pt] Draw the interference graph for the local scalar variables used in the section of code presented above by direct inspection. Indicate a possible allocation of registers to variables using the graph coloring algorithm explained in class and supposing the utilization of 2 registers (*R1*, e *R2*). Show the content of the stack immediately after the simplification of the interference graph. In the case of having to perform *spilling*, specify an efficient criterion that you suggest for the selection of variables for *spilling* and the order of selection determined by that criterion. Show the result of the first graph coloring (i.e., without repeating the process).

Group 5. (2 pts)

5.a) [2pt] Suppose there is a need to pack variables in the same register, even if the lifetime of those variables interferes (for instance, 32-bits register can store the value of two 16-bits variables). Assuming 32-bits registers and variables with 8, 16 e de 32-bits data types, indicate what would have to be changed in the register allocations based in graph coloring so that packaging is used to reduce the number of registers required. Use the examples that you think are adequate to illustrate that process.

(End.)