Universidad de Puerto Rico
Recinto Universitario de Mayagüez
Departamento de Ingeniería Eléctrica y Computadoras

ICOM 4036 – Programming Languages

Otoño 2004

Ejercicios de práctica
Examen Parcial I

1. Add a STACK pointer register to the Easy I Data Path. The register points to a memory cell that represents the next empty element in a stack. After a Reset Cycle the value of the stack pointer is the address of highest memory cell (2^16-1). For each of the following instructions show the changes to the DataPath, flowchart and control unit necessary to implement the following new Easy I instructions.

    a. Add an instruction to PUSH the value of the accumulator into the stack. PUSH stores the value of the accumulator at the memory cell pointed to by the stack pointer and then decrements the pointer.

    b. Add an instruction to POP the element at the top of the stack into the accumulator. The instruction must first decrement the stack pointer and then move the contents of the memory cell at the top into the accumulator.

2. Write finite state diagrams to recognize the following languages:

    a. Sequence of 0's and 1's containing the pattern "010111". Careful with overlapped patterns

    b. Identifiers that begin with letter or underscore followed by letters, digits or underscore, and ending with a dollar ($) sign.

    c. Integer constants that are divisible by 5

3. Write a BNF for C++ variable declarations

4. Write a BNF for C++ statements. You may assume that a non-terminal <expression> has already been defined.

5. Write a BNF for C++ classes using the BNF's from (2) and (3)

6. Write recursive descent parsing functions in the style of Sebesta (see lecture notes) for each of the BNF non-terminals defined in (2), (3) and (4).

7. Design a state diagram to recognize all numeric literals in ANSI C

8. Consider the following grammar:

```
stmts     →   <stmt> ; <stmts>
          |   <stmt>
block     →   begin <stmts> end
stmt      →   <if_stmt> | <block> | …
if_stmt   →   if  ( <expr> ) then <stmt> else <stmt>
          |   if  ( <expr> ) then <stmt>
expr      →   <expr> + id
          |   id
```

   - Prove that the grammar is ambiguous.
   - Provide a new unambiguous grammar that generates the same language and whose parse trees correspond with the semantics of nested if-then-else in Pascal

9. Sebesta problem set question 3.13

10. Sebesta problem set question 3.14