**COP 3402 Systems Software**

**Summer 2015**

**Instructor: Dr. Pawel Wocjan**

**First Exam**

**Wednesday 06/24/2015**

There are 5 problems. Each problem is worth 5 points.

**First Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Last Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Total: \_\_\_\_\_\_\_\_\_**

**First Name/Last Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem 1 – Systems Software**

1. The programs in Systems Software can be grouped in two groups. What are these two groups called?

First group: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Second group: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Name four programs in Systems Software (2 from each group) and indicate which group they belong to.

Program: Group:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**First Name/Last Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem 2 – Stack Trace of PM/0 Program**

Given the program in interpreted assembly language for virtual machine PM/0 below, show the stack and register values (pc, bp, and sp) at the end of the execution.

If there are multiple Activation Records, you have to separate them with the symbol “|”. Refer to Appendix A if needed. Initial values for the PM/0 CPU registers: sp = 0; bp = 1; pc = 0.

Line OP L M

0 JMP 0 8

1 JMP 0 2

2 INC 0 4

3 LOD 1 4

4 LOD 1 5

5 OPR 0 4

6 STO 1 4

7 OPR 0 0

8 INC 0 4

9 LIT 0 2

10 LIT 0 4

11 CAL 0 2

12 SIO 0 2

pc: \_\_\_\_\_\_ bp: \_\_\_\_\_\_ sp: \_\_\_\_\_\_ stack: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**First Name/Last Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem 3 – Static Links/Dynamic Links/Local Variables on Stack for PL/0 Program**

Indicate what the contents of the stack are after the assignment statement in B but before the read statement. You may omit Functional Value, Return Address, and Parameters only indicating Static Links, Dynamic Links, and Local Variables. Draw arrows to indicate where the links point to. Indicate where the different Activation Records start.

program

var k, m;

procedure A;

var a;

procedure B;

begin

read a;

m := a + 1;

write m;

end

begin

call B;

end

begin

k = 3;

m = 2;

call A;

end .

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | DL |  |
|  | SL |  |
|  |  |  |
| variable | k |  |
|  | DL |  |
| **program** | SL |  |

**First Name/Last Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem 4 – Regular Expressions**

Let Σ={a,b}. We consider only strings over Σ.

1. Give a regular expression for all strings.
2. Give a regular expression for all strings whose length is **even**.
3. Give a regular expression for all strings whose length is **odd**.
4. Give a regular expression for all strings that contain exactly one b.
5. Give a regular expression for all strings that contain either one b or two b’s.

**First Name/Last Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem 5 – Lexical Analysis**

Output the lexeme tables for the following two PL/0 programs. Stop scanning if a lexicographical error occurs and report the corresponding error message.

identsym, numbersym, plussym, minussym, multsym, slashsym,

oddsym, eqlsym, neqsym, lessym, leqsym, gtrsym,

geqsym, lparentsym, rparentsym, commasym, semicolonsym, periodsym,

becomessym, beginsym, endsym, ifsym, thensym, whilesym,

dosym, callsym, constsym, varsym, procsym, writesym,

readsym, elsesym

Identifiers can be a maximum of 11 characters in length. Identifiers must start with a letter symbol. Numbers can be a maximum of 5 digits in length.

**Program 1:** **Program 2:**

var x, y; const one = 1;

x : = 5; var 1one;

y := 6; begin

**Lexeme Table 1:** **Lexeme Table 2:**

Lexeme Class Lexeme Class

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Error: Error:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Appendix A**

01 **LIT 0 M**

sp = sp + 1;

stack[sp] = **M**;

02 **OPR** **0 M**

0 **RET** sp = bp – 1; pc = stack[sp + 4]; bp = stack[sp + 3];

1 **NEG** stack[sp] = -stack[sp];

2 **ADD** sp = sp – 1; stack[sp] = stack[sp] + stack[sp + 1];

3 **SUB** sp = sp – 1; stack[sp] = stack[sp] - stack[sp + 1];

4 **MUL** sp = sp – 1; stack[sp] = stack[sp] \* stack[sp + 1];

5 **DIV** sp = sp – 1; stack[sp] = stack[sp] / stack[sp + 1];

6 **ODD** stack[sp] = stack[sp] mod 2;

7 **MOD** sp = sp – 1; stack[sp] = stack[sp] mod stack[sp + 1];

8 **EQL** sp = sp – 1; stack[sp] = stack[sp] == stack[sp + 1];

9 **NEQ** sp = sp – 1; stack[sp] = stack[sp] != stack[sp + 1];

10 **LSS** sp = sp – 1; stack[sp] = stack[sp] < stack[sp + 1];

11 **LEQ** sp = sp – 1; stack[sp] = stack[sp] <= stack[sp + 1];

12 **GTR** sp = sp – 1; stack[sp] = stack[sp] > stack[sp + 1];

13 **GEQ** sp = sp – 1; stack[sp] = stack[sp] >= stack[sp + 1];

03 **LOD L M**

sp = sp + 1;

stack[sp] = stack[ base(**L, bp**) + **M**];

04 **STO** **L M**

stack[ base(**L, bp**) + **M]** = stack[ sp ];

sp = sp - 1;

05 **CAL L M**

stack[sp + 1] = 0;

stack[sp + 2] = base(**L, bp**);

stack[sp + 3] = bp;

stack[sp + 4] = pc;

bp = sp + 1;

pc = **M**;

06 **INC 0 M**

sp = sp + **M**;

07 **JMP 0 M**

pc = **M**;

08 **JPC 0 M**

**if** (stack[ sp ] == 0 ) **then {** pc = **M; }**

sp = sp - 1;

09 **SIO** **0 0**

print(stack[ sp ]);

sp = sp – 1;

09 **SIO 0 1**

sp = sp + 1;

read(stack[ sp ]);

09 **SIO 0 2** halt;

**NOTE**: The result of a logical operation such as (A > B) is defined as 1 if  
the condition was met and 0 otherwise.