**CSCI 468 Quiz 3**

**Spring 2015**

You must use a word processor in answering these questions. It is expected that you will use any resource at your disposal except another person. Insert your answers immediately below the associated questions. Do not include hand-drawn images. Submit your exam as a PDF document to the Quiz 3 dropbox in D2L by Tuesday evening, May 5, at 11:59 PM.

**Do not discuss this quiz with anyone except your instructor until after the due date**.

Name: \_Travis Alpers

**Q1**. Discuss your role in your compiler project. Keep your discussion to the remainder of this page.

Our compiler project was truly a team effort, I'm grateful to have gotten the partners I did. This allowed each of us to work on each piece of the compiler, so we each gained knowledge on the construction of a compiler. My absolute main focus which I devoted most time to was the Semantic Analyzer because I have a certain love for machine level(even pseudo-machine level) code. As we went through we made sure each of use knew what the others were doing and why, so we each had comprehension on the overall project. One other role I was instrumental in was implementing design patterns in a few portions of our project for optimal system architecture.

In the remaining questions on this quiz you will be asked to write simple Pascal programs and run them through your compiler. The code given is not in Pascal form, so part of your effort is to write a correct Pascal program that compiles properly.

**Q2**. (Do this question if you targeted the A, B, or C levels). This question relates to the translation of Pascal programs at the C level. To answer the questions posed below, write a Pascal program that has the following assignment statements in it, where a and y are of type integer and x is of type float. Compile this program.

x ← 5 + a \* y;

x ← a + y div 2 \* (5 + a);

Paste your Pascal program below:

program prog1;

var

x: float

a, y: integer

begin

x := 5 + a \* y;

x := a + y div 2 \* (5 + a);

end.

**Q2-1**. Copy and paste the code your compiler generates for x ← 5 + a \* y; below. Add commentary that describes what each line of code does in relation to your program. For example, if your compiler generates the line

br L1 ;branch program start

L1:

; prog1 start

add SP #3 SP ;allocate mem on stack

sub SP #3 D0 ;set D0 to start of local stack frame

; activation end

; init local vars to 0

push #0

pop 0(D0)

push #0

pop 1(D0)

push #0

pop 2(D0)

push #5 ;push 5 to top of stack

push 1(D0) ;push a to top of stack

push 2(D0) ;push y to top of stack

muls ;int mult two top of stack and push result to top

adds ;int add the top two of the stack

castsf ;cast top of stack to float

pop 0(D0) ;pop result(top of stack) into x

push 1(D0) ;push a to top of stack

push 2(D0) ;push y to top of stack

push #2 ;push literal int 2 to top of stack

divs ;int div top two of stack and push back to top of stack

push #5 ;push literal 5 to top of stack

push 1(D0) ;push a to top of stack

adds ;int add top two of stack

muls ;int mult top two of stack

adds ;int add top two of stack

castsf ;cast top of stack to float

pop 0(D0) ;pop top of stack into y

; deactivation start

mov D0 SP ;exit cleanly by restoring the stack pointer

; prog1 end

hlt ;kill

push 1(D0)

you could include

push 1(D0) - pushes variable a onto the stack

if that is what that line does.

Paste the translated code below.

br L1 ;branch to program start

L1: ;prog1 start

; prog1 start

add SP #3 SP ;increment stack pointer by 3 for variable space

sub SP #3 D0 ;set D0 to bottom of prog1 stack

; activation end

push #5 ;push 5 onto the stack

castsf ;cast top of stack to float(5)

push 1(D0) ;push a on top of stack

castsf ;cast top of stack to float(a)

push 2(D0) ;push y to top of stack

castsf ;cast top of stack to float(y)

mulsf ;mul top two of stack then push result onto top of stack(float op)

addsf ;add top two of stack then push result onto top of stack(float op)

pop 0(D0) ;pop top of stack into x(result of 5 + a \* y)

push 1(D0) ;push a to top of stack

castsf ;cast top of stack to float(a)

push 2(D0) ;push y to top of stack

castsf ;cast top of stack to float(y)

push #2 ;push 2 to top of stack

castsf ;cast top of stack to float(2)

divsf ;int divide top two of stack

push #5 ;push 5 onto stack

castsf ;cast top of stack to float(5)

push 1(D0) ;push a to top of stack

castsf ;cast top of stack to float(a)

addsf ;add top two of stack

mulsf ;mul top two of stack

addsf ;add top two of stack

pop 0(D0) ;pop into x

; deactivation start

mov D0 SP ;restore SP to original memory location

; prog1 end

hlt ;end program

**Q2-2**. Copy and paste the code your compiler generates for x ← a + y div 2 \* (5 + a); below. You do not need to add commentary with this code.

L1:

; prog2 start

add SP #3 SP

sub SP #3 D0

; activation end

; init local vars to 0

push #0

pop 0(D0)

push #0

pop 1(D0)

push #0

pop 2(D0)

push 2(D0)

push 1(D0)

push #2

divs

push #5

push 2(D0)

adds

muls

adds

pop 0(D0)

; deactivation start

mov D0 SP

; prog2 end

hlt

**Q2-3.** Diagram what runtime memory looks like while this program is running. Be sure to show where D0 and SP are pointing.

0x3 - null

0x2 - (D0) → 0

0x1 - 1(D0) → 0

0x0 - D0 ->0

**Q3.**  (Do this question if you targeted the B or A levels with your compiler.)

* Write a Pascal program that contains the code for the while loop below. You may assume that all variables are of type integer.
* Show the microMachine code that is generated by your compiler for the following while loop.
* Highlight with **boldface** font the parts of the code that are generated from calls to the semantic analyzer in just the method WhileStatement itself, **not** from the call to BooleanExpression or Statement.
* Provide commentary, as in Q2, just for the lines you highlighted in boldface, about what those lines do.
* Recall that the rule WhileStatement is:

60 <WhileStatement> ⟶ while <BooleanExpression> do <Statement>

input n

a ← 1

y ← 0

while (a <= n) do begin

y ← y + a \* y

a ← a + 1

end

**Copy your Pascal program below**:

program prog3;

var a, y, n: integer;

begin

y := 0;

a := 1;

while (a <= n) do begin

y := y + a \* y;

a := a + 1;

end;

end.

**Copy the microMachine code below and provide commentary for the lines you highlight, as described above**.

br L1

L1:

; prog3 start

add SP #3 SP

sub SP #3 D0

; activation end

; init local vars to 0

push #0

pop 0(D0)

push #0

pop 1(D0)

push #0

pop 2(D0)

push #0

pop 1(D0)

push #1

pop 0(D0)

**L2: ;Label to mark start of while compare**

push 0(D0)

push 2(D0)

cmples

**brfs L3 ;branch to exit if condition is false**

push 1(D0)

push 0(D0)

push 1(D0)

muls

adds

pop 1(D0)

push 0(D0)

push #1

adds

pop 0(D0)

**br L2 ;unconditional branch to start of loop**

**L3: ;label to mark end of loop**

; deactivation start

mov D0 SP

; prog3 end

hlt

**Q4**. Do this question if you targeted the A level with your compiler.

**Q4-1**. Write a Pascal program that does the following.

program test

variables

integer a, b

float g

procedure tryit(var x of type integer, g of type float)

variables

integer a

a := 10

b := a + b \* x

x := a

g := b \* g

end procedure

beginning of program test

a := 5

b := 6

g := 5.5

tryit(a, g);

println "a = ", a

println "b = ", b

println "g = ", g

end of program test

**Include your pascal program below:**

program test;

var

a, b: integer;

g: float;

procedure tryit(var x: integer; var g: float);

var

a: integer;

begin

a := 10;

b := a + b \* x;

x := a;

g := b \* g;

end;

begin

a := 5;

b := 6;

g := 5.5;

tryit(a, g);

writeln('a = ', a);

writeln('b = ', b);

writeln('g = ', g);

end.

**Include the generated code below.**

br L1

L2:

; tryit start

**add SP #1 SP ;increment sp for 1 local var**

**mov D1 -5(SP) lstore old D1**

**sub SP #5 D1 ; set D1 to SP address -5(arguments + RA + old reg value)**

; activation end

; init local vars to 0

push #0

pop 4(D1)

push #10

pop 4(D1)

push 4(D1)

push 1(D0)

push @1(D1)

muls

adds

pop 1(D0)

push 4(D1)

pop @1(D1)

push 1(D0)

castsf

push @2(D1)

castsf

mulsf

pop @2(D1)

; deactivation start

mov D1 SP

mov 0(SP) D1

add SP #4 SP

ret

; tryit end

L1:

; test start

add SP #4 SP

sub SP #4 D0

; activation end

; init local vars to 0

push #0

pop 0(D0)

push #0

pop 1(D0)

push #0

pop 2(D0)

push #5

pop 0(D0)

push #6

pop 1(D0)

push #5.5

pop 2(D0)

**add SP #1 SP ;increment stack pointer for old reg value**

**push D0 ;push contents ofD0 to top of stack**

**push #0 ;push offset of var passing by reference to top of stack**

**adds ;add them together(resulting in var by ref)**

**push D0 ;push contents of D0 to top of stack**

**push #2 ;push offset of second argument to top of stack**

**adds ;add offset to D0 address for absolute address of var**

**call L2 ;call L2(tryit)**

**sub SP #2 SP ; fix stack pointer – decrement for two arg's**

**sub SP #1 SP ;fix stack pointer – decrement for old dis reg**

push #"a = "

wrts

push 0(D0)

wrts

push #"\n"

wrts

push #"b = "

wrts

push 1(D0)

wrts

push #"\n"

wrts

push #"g = "

wrts

push 2(D0)

wrts

push #"\n"

wrts

; deactivation start

mov D0 SP

; test end

hlt

Highlight in boldface font the parts of this output code that are generated when compiling the line

procedure tryit(var x of type integer, g of type float)

and the line

tryit(a, g)

Provide brief commentary about what each of the highlighted lines does.

**Include the printed output of running your generated code below.**

uMachine Interpreter

(-\*-) [VERSION 2.0DV] (-\*-)

-------------------------------

a = 10

b = 40

g = 220.000000

-------------------------------

(-\*-) [VERSION 2.0DV] (-\*-)

uMachine Interpreter

**Q4-2**. Diagram what runtime memory would look like when the program is executing the code in procedure tryit. **This is assuming this is at the end of run it, showing reference usage**

**0x8 – SP → NULL**

**(A)0x7 - 0(D1)**

**0x6 - Old Reg Value**

**0x5 –RA**

**0x4 - @(D0+2)**

**0x3 - @(D0+0)**

**(G) 0x2 – 2(D0) -**

**(B) 0x1 – 1(D0) - 40**

**(A) 0x0 – D0 - 10**