CS3320: Compilers-I, Spring 2020

Programming Assignment 0: Toy Cool Programs

Author: Shreyas Jayant Havaldar

CS18BTECH11042

Correct Trivial Programs	Incorrect Trivial Programs	Non-Trivial Programs
greetings.cl	identifier.cl	sentence_analyzer.cl
dot_product.cl	string.cl	prime_factors.cl
even_odd.cl	comment.cl	
gcd.cl	keywords.cl	
factorial.cl	whitespace.cl	

Correct Programs:

- Understanding common **MIPS instructions** seen in MIPS code generated by compiling my files:
    - *addiu, sub*: Add immediate unsigned (no overflow) and subtract respectively.
    - *mult, div*: Multiply and divide respectively.
    - *move*: Move value stored at the address to registers
    - *sw*: Store word in memory
    - *lw*: Load word from memory
    - *la*: Load address
    - *li*: Load immediate
    - *bne*: Branch on not equal
    - *jal*: Jump and Link

- *jr:* Returns control to the caller, returning value stored in $ra.
- **NOTE:** In my analysis, I've used the term *MSL* to collectively represent *move, sw* and *lw* instructions.

- **Overview** of all my programs:
  - The *Main* class contained all the functions(always containing a *main* function).
  - The *Main* class inherits IO class.
  - The *main* function always has the return type *Object.*
  - All variables were declared local to the functions they were used in, no use of global variables to emphasize modularity.
  - Explored different aspects of the COOL language by maximising the variety of the constructs used in programs, some prominent ones of them being:
    - Basic string I/O
    - Arithmetic operations
    - Conditional statements
    - Recursive functions
    - Looping and constructs

- **General MIPS correspondence** common to all programs:
  - The built-in object classes are first declared globally, *.globl class_nameTab, .globl Main_protObj, .globl Int_protObj, .globl String_protObj*.
  - Memory containers are created and initialized.
  - The string dispatch table is filled with predefined strings(including the ones printed to the I/O console) and object and class names with their attributes.
  - Dispatch table entries are created in a similar manner for predefined integer values, in the integer dispatch tables.
  - The attributes of classes, objects and other existing instances are set by specifying characteristic attributes and member variables.
  - The memory heap is started(*heap_start).*
  - The classes are then initialized beginning with the parent classes, starting at the root of the inheritance tree.(*Object_init*:)

- The *Main* class is the last one to be initialised(*Main_init:)* and is followed by calls to the functions belonging to the main class.(*Main.main, Main.fact)*
- The numerous labels(*label6, label7)* each unique to a specific block in that function follow which effectively bring about the execution of the code.
- The correspondence of important labels is shown in the following program specific table:

1. greetings.cl

Printing a simple greeting message to the user personalizing it with the name entered by the user when prompted by the program.

COOL	MIPS	Correspondence Study

out_string("Hey! What's your name? \n");
name <- in_string();
out_string("Greetings! ".concat(name.concat(". \n")));)
	str_const2:
	.word 5
	.word 7
	.word String_dispTab
	.word int_const7
	.ascii "Greetings! "
	.byte 0
	.align 2
	.word -1
	
	str_const1:
	.word 5
	.word 11
	.word String_dispTab
	.word int_const8
	.ascii "Hey! What's your name? \n"
	.byte 0
	.align 2
	.word -1
	
	Main.main:
	MSL
	la $s1 str_const0
	la $a0 str_const1
	MSL
	bne $a0 $zero label0
		str_const2 stores the output string “Greetings!”.
		
		str_const1 stores the first string output "Hey! What's your name? \n".
		
		The string constants stored above are

	la $a0 str_const4 *li $t1 1* *jal _dispatch_abort*	loaded from their addresses and printed.

2. dot_product.cl

Calculating the dot product of two vectors entered by the user.

COOL	MIPS	Correspondence Study

x1 <- in_int();
y1 <- in_int();
x2 <- in_int();
y2 <- in_int();
 	
out_int(x1*x2 + y1*y2);
	Main.main:
	
	la $s1 int_const0
	la $s2 int_const0
	la $s3 int_const0
	la $s4 int_const0
	
	label 5:
	
	mul $t1 $t1 $t2
	MSL
	mul $t1 $t1 $t2
	MSL
	add $t1 $t1 $t2
	MSL
		Loading the variables input from the I/O console by the user.
		
		Start of label
		
		Multiplication of x coordinates
		
		Multiplication of y coordinates
		
		Addition of both values to get dot product

3. even_odd.cl

To check and print if the number entered by the user is even or odd.

COOL	MIPS	Correspondence Study
x - ((x/y)*y)	*Main.mod:* *MSL* *div $t1 $t1 $t2* *MSL* *mul $t1 $t1 $t2* *MSL* *sub $t1 $t1 $t2* *MSL* *jr $ra*	The arithmetic operation of finding the remainder when *x* is divided by *y* is calculated and returned.

if mod(a,2) = 0 then out_string("Entered number is EVEN.\n") else out_string("Entered number is ODD.\n")	*label5:* *MSL* *beq $t1 $t2 label4* *la $a1 bool_const0* *jal equality_test* *label4:* *la $a0 str_const1* *label2:* *la $a0 str_const2*	Comparison of the value with 0 Printing of the corresponding result based on the above equality test.

4. gcd.cl

To find the Greatest Common Divisor of the two non-zero numbers entered by the user by using Euclid's algorithm.

COOL	MIPS	Correspondence Study
while not a = b loop if a <= b then b <- b - a else a <- a - b fi pool;	*label3:* *beq $t1 $t2 label6* *label5:* *ble $t1 $t2 label9* *label9:* *sub $t1 $t1 $t2* *label7:* *sub $t1 $t1 $t2*	Checks for equality, if true, exits loop If a is less tha or equal to b, subtracts a from b, else subtracts b from a.

5. factorial.cl

To find and print the factorial of the number entered by the user.

COOL	MIPS	Correspondence Study
if(n=0)	*Main.fact:* *beq $t1 $t2 label2*	Checks if n is equal to zero.

then 1 else n*fact(n-1) fi	*la $a1 bool_const0* *jal equality_test* *label2:* *b label1* *label1:* *MSL* *jr $ra* *label3:* *MSL* *mul $t1 $t1 $t2* *sw $t1 12($a0)*	Returns 1 if true Otherwise recursively multiplies by calling the function for *n-1*

Incorrect Programs:

1. **identifier.cl**
    - Violates *Section 10.1* of the COOL Manual.
    - **ERROR**:

      "identifier.cl", line 6: syntax error at or near ERROR = #

      "identifier.cl", line 7: syntax error at or near INT_CONST = 7

      "identifier.cl", line 12: syntax error at or near ERROR = #

      Compilation halted due to lex and parse errors

    - **Study:**
        i. In line 6 and line 12, the variable name contains the symbol '#' which violates the requirement: *Identifiers are strings (other than keywords) consisting of letters, digits, and the underscore character.*
        ii. In line 7, an integer variable is assigned value 7.7 which violates the requirement: *Integers are non-empty strings of digits 0-9*.

2. **string.cl**
    - Violates *Section 10.2* of the COOL Manual.
    - **ERROR**:

        "string.cl", line 7: syntax error at or near ERROR = Unterminated string constant

        Compilation halted due to lex and parse errors

    - **Study:**
        i. In line 7 there is a non-escaped new line in the string which violates the requirement *A non-escaped newline character may not appear in a string.*
        ii. This can be resolved by using a backslash converting

            “Hello

            World!”

            to “Hello\

            World!”

3. **comment.cl**
    - Violates *Section 10.3* of the COOL Manual.
    - **ERROR**:

        "comment.cl", line 6: syntax error at or near '/'

        Compilation halted due to lex and parse errors

    - **Study:**
        i. In line 6 the tokens “ // ” are used for writing comments which violates the requirement: *There are two forms of comments in Cool. Any*

characters between two dashes "--" and the next newline (or EOF, if there is no next newline) are treated as comments. Comments may also be written by enclosing text in (∗ . . . ∗).

4. **keywords.cl**
    - Violates *Section 10.4* of the COOL Manual.
    - **ERROR**:

        "keywords.cl", line 6: syntax error at or near TYPEID = True

        Compilation halted due to lex and parse errors

    - **Study:**
        i. In line 6, *flag* is assigned the boolean value *True* which violates the requirement: *The first letter of true and false must be lowercase; the trailing letters may be upper or lower case.*
        ii. Instead it should be assigned the value *true* (first letter in lowercase).
        iii. Note that assigning *fAlSe* to *check* does not produce an error.

5. **whitespace.cl**
    - Violates *Section 10.5* of the COOL Manual.
    - **ERROR**:

        "whitespace.cl", line 1: syntax error at or near ERROR = \357

        Compilation halted due to lex and parse errors

    - **Study:**
        i. In the program, *flat whitespace* is used in the string which is not recognised by the COOL compiler, which violates the requirement: *White space consists of any sequence of the characters: blank (ascii 32), \n (newline, ascii 10), \f (form feed, ascii 12), \r (carriage return, ascii 13), \t (tab, ascii 9), \v (vertical tab, ascii 11).*