Compilers - I
Assignment 0 : Toy Cool Program

- A. Sai Mahesh
- CS18BTECH11001

Incorrect Programs:

Incorrect_1.cl

errors:

```
"incorrect_1.cl", line 3: syntax error at or near FI
"incorrect_1.cl", line 4: syntax error at or near OBJECTID = string
"incorrect_1.cl", line 5: syntax error at or near TYPEID = Sum
"incorrect_1.cl", line 8: syntax error at or near OBJECTID = e69d
Compilation halted due to lex and parse errors
```

study:

fi is a keyword which cannot be used as an identifier
string is a type identifier so it should begin with capital letter
Sum is an object identifier so it should begin with a lowercase letter
1025e69d contains e and d which are not digits. So it is not an Int

Incorrect_2.cl

errors:

```
"incorrect_2.cl", line 2: syntax error at or near OBJECTID = mahesh
"incorrect_2.cl", line 8: syntax error at or near ERROR = Unterminated string constant
Compilation halted due to lex and parse errors
```

study:

mahesh should be end using ”
Backslash must be inserted before newline character in the string

Incorrect_3.cl

errors:

```
"incorrect_3.cl", line 8: syntax error at or near '-'
Compilation halted due to lex and parse errors
```

study:

another “-” is missing for ti to be a comment

Incorrect_4.cl
errors:

```
"incorrect_4.cl", line 2: syntax error at or near TYPEID = True
Compilation halted due to lex and parse errors
```

study:
“True” is a case sensitive keyword so it must be changed to “true”

Incorrct_5.cl
errors:

```
"incorrect_5.cl", line 5: syntax error at or near OBJECTID = thenout_string
Compilation halted due to lex and parse errors
```

study :
The then and the expression of the if condition must be separated by a white space where white space does not include epsilon.

Correct Programs:
Lets understand the common instructions first :

- addiu - add immediate unsigned
- sw - store word
- move - move contents from one register to another register
- la - load address
- bne - branch on not equal
- li - loads immediate value into register
- jal - saves the address of the next instruction
- lw - loads a word into a register
- class_nameTab - contains string constants
- str_const<x>,int_const<x>,bool_const<x> - literals in code
- <classname>_dispTab - contains methods used in that class
- <classname>_protoObj - the prototype object of class
- <classname>_init - code that initializes an object of class
- <classname>.<methodname> - formal instructions for the method

Source:
http://web.stanford.edu/class/cs143/materials/cool-runtime.pdf

Binary_string.cl

Evaluation step	Corresponding MIPS instruction
out_string("Enter an Integer : ");	la $a0 str_const4
	str_const4:
	.ascii "Enter an Integer : "

Explanation:

Str_const4 contains the string “Enter an Integer : “ which is loaded into $a0(register a0) for printing.

gcd.cl

Correspondence:

Evaluation step	Corresponding MIPS instruction
(i/j)	div $t1 $t1 $t2(line: 602)
j*(i/j)	mul $t1 $t1 $t2(line: 607)
i-j*(i/j)	sub $t1 $t1 $t2(line: 612)

Explanation:

The MIPS instructions are self explanatory where div -> division, mul -> multiplication, sub-> subtraction.

number_tree.cl

Correspondence:

Evaluation Step	Corresponding MIPS instruction
while j<x-i-1 loop	label5:
	sub $t1 $t1 $t2(line: 436)
	sub $t1 $t1 $t2(line: 443)
	blt $t1 $t2 label7(line: 448)
	label7:
	bne $a0 $zero label8
	label8:
	b label5(line: 473)

Explanation:

First the subtraction takes place and if the j(loaded into $t1 after a series of instructions) is less than evaluated value(loaded into $t2 after a series of instructions), it is transferred to label7 which after completion of some of the instructions in loop transfer to labe8 which after completing all instructions in the loop returns to label5 again by which the instructions loop.

palindrome.cl

Correspondence:

Evaluation step	Corresponding MIPS instruction
if str="" then true	la $t2 str_const3
	beq $t1 $t2 label10
	label10:
	la $a0 bool_const1
	b label9
	label9:
	lw $ra 12($sp)
	jr $ra

Explanation:

str_const3 contains an empty string which is loaded to register t2($t2). Then the prefetched str into $ti is checked and if it is correct, transfer control to label10 which then loads a value bool_const1(i.e., 1) to $a0 and transfer to label9 which then returns the final value.

prime.cl

Correspondence:

Evaluation Step	Corresponding MIPS instruction
let i : Int <-2 in	la $s1 int_const2
while i*i<=x loop	label11:
	mul $t1 $t1 $t2
	ble $t1 $t2 label13
	label13:
	beq $t1 $t2 label16
	label16:
	b label15
	label11:
	b label11

Explanation:

The initialisation in the let statement is simply a MIPS load instruction. The loop can change control to as many branches as it can but after the successful completion of the statements in the loop the control has to return to the same branch where the while loop has begun.