How to Create a Programming Language

L11: RIF INTRODUCTION

1

OVERVIEW

- RIF – RASCAL INTERMEDIATE FORM
- INSTRUCTION SET
  - INSTRUCTION FORMAT
  - LOAD/STORE INSTRUCTIONS
  - ARITHMETIC INSTRUCTIONS
  - FLOW CONTROL INSTRUCTIONS
  - TYPE CONVERSION
  - MISC (SHIFT, SYSCALL)

2

RASCAL INTERMEDIATE FORM

- SIMPLE INSTRUCTION SET
- DESIGNED AS INTERMEDIATE CODE FOR RASCL (REALLY A SMALL C LANGUAGE)
- USED IN COMPILER PROJECT FOR CSE304

3

INSTRUCTION SET - FORMAT

- '3-ADDRESS' CODE [QUADRUPLES]
  - OPCODE
  - 2 SOURCE OPERANDS
  - 1 RESULT OPERAND
- SIMILAR IN STRUCTURE TO MIPS ASSEMBLER

4

LOAD/STORE INSTRUCTIONS

li – load immediate – loads a hard-coded integer into a 'temporary' register
Format:
li, <integer constant>, 0, <tempname>

lw – load word – loads a value from memory (identifier/variable name) into a 'temporary' register
Format:
lw, <identifier>, 0, <tempname>

sw – store word – stores a value from a 'temporary' register into memory (identifier/variable name)
Format:
sw, <temp>, 0, <identifier>

5

ARITHMETIC INSTRUCTIONS

Additive instructions
add – add values of two 'registers'
Format:
add, <operand1_temp>, <operand2_temp>, <result_temp>

addi – add the value in a register to a constant integer
Format:
addi, <operand1_temp>, <integer_constant>, <result_temp>

sub – subtract integer values from two registers (subtracts value in operand2_temp from that in oeprand1_temp)
Format:
sub, <operand1_temp>, <operand2_temp>, <result_temp>

subi – subtract immediate – subtracts integer constant from value in operand1_temp
Format:
subi, <operand1_temp>, <integer_constant>, <result_temp>

6

ARITHMETIC INSTRUCTIONS

Multiplicative Instructions
div – Performs integer division dividing operand2_temp into operand1_temp. result_temp gets quotient
Format:
div, <operand1_temp>, <operand2_temp>, <result_temp>

mul – Performs integer multiplcation multiplying <operand1_temp> by <operand2_temp> and placing result in <result_temp>
Format:
mul, <operand1_temp>, <operand2_temp>, <result_temp>

7

ARITHMETIC INSTRUCTIONS

Floating Point Instructions
fadd – Add two floating point numbers (in operand1_temp and operand2_temp) placing result in result_temp
Format:
fadd, <operand1_temp>, <operand2_temp>, <result_temp>

fsub - Subtract (as floating point numbers) the value in operands2_temp from that in operand1_temp. Places result in result_temp.
Format:
fsub, <operand1_temp>, <operand2_temp>, <result_temp>

8

ARITHMETIC INSTRUCTIONS

Floating Point Instructions

fdiv – Divide (as floating point numbers) the value in operand2_temp into that in operand1_temp. Place the result in result_temp.

Format:
fdiv, <operand1_temp>, <operand2_temp>, <result_temp>

fmul – Multiply (as floating point numbers) the values in operand1_temp and operand2_temp placing result in result_temp.

fmul, <operand1_temp>, <operand2_temp>, <result_temp>

9

CONTROL FLOW INSTRUCTIONS

Conditional Branches

beq – Branch if equal
bne – Branch if not equal
bgt – Branch if grater than
bge – Branch if greater than or equal to
blt – Branch if less than
ble – Branch if less than or equal to

Format:
<op>, <operand1_temp>, <operand2_temp>, <label_or_quad_index>

Causes a branch to the label (or quad index) indicated in the final operand

10

CONTROL FLOW INSTRUCTIONS

Unconditional Branch

j – Jump to the label or quad index given in the last operand. Always transfers control.

Format:
j, 0, 0, <label_or_quad_index>

11

TYPE CONVERSION INSTRUCTIONS

***toInt* – Convert float to integer [with truncation]**

This operation converts a floating point to an integer. The source float is a float temporary in arg1. The converted value is placed into an integer temporary in the result field. The arg2 field is not used.

Format:
toInt, <float_temp1>, 0, <result_temp>

***toFloat* – Convert integer to float**

This operation converts and integer to a floating point number. The source integer is a temporary in arg1. The converted value is placed into a float temporary in the result field. The arg2 field is not used.

toInt, <int_temp1>, 0, <result_ftemp>

12

RIF DIRECTIVES

- DIRECTIVES INDICATE AN OPERATIONAL ACTION BUT DO NOT MAP TO MACHINE INSTRUCTIONS
- DIRECTIVES INCLUDE:

.label – indicates a point in the quad list where a conditional or unconditional branch can transfer control

Format:

.label, 0, 0, <label_name>

.section – Indicates what the contents of the following RIF lines include (data, code, etc)

Format:

.section, 'data' or 'code'

13

RIF DIRECTIVES

.int – Reserves space for integer values

Format:

.int, 0, 0, <identifier>

.float – Reserves space for float values

.float, 0, 0, <identifier>

14

CODE EXAMPLES

```
int a, b;
{
   a = 0;
   b = 0;
  while (a < 10)
  {
     b = b + a;
     a = a + 1;
  }
  print b;
}
```

```
0: .segment, data
1: .int, 0, 0, a
2: .int, 0, 0, b
3: .segment, 0, 0, text

 4: li, 0, 0, t1
 5:  sw, t1, 0, a
 6: li, 0, 0, t2
 7:  sw, t2, 0, b
 8: li, 10, 0, t3
 9: bge, t1, t3, 13
10: add, t2, t1, t2
11: addi, t1, 1, t1
12: j, 0, 0, 9
13: sw, t1, 0, a
14: sw, t2, 0, b
15: syscall, 2, t2, 0
```

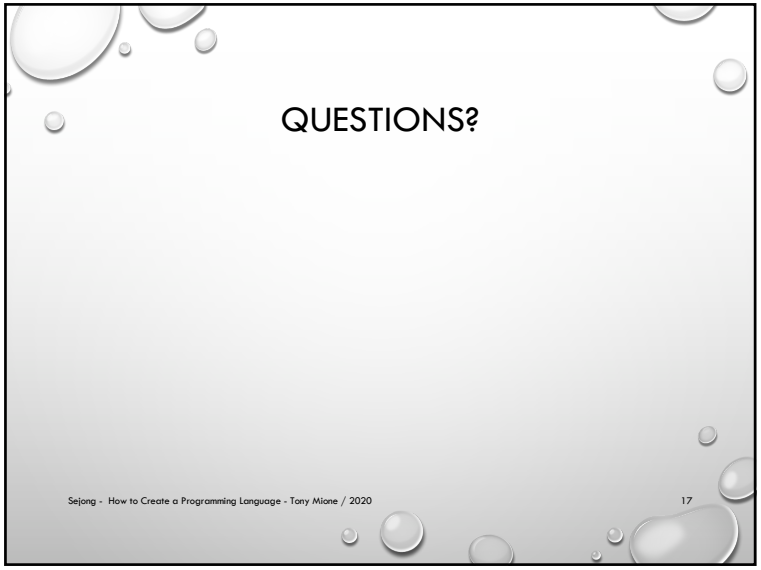

15

CODE EXAMPLES

```
int a;
float c, d;
{
   read a;
   read c;
  if (a == 0) {
     d = (c * 9 / 5) + 32;
  } else {
     d = (c – 32) * 5 / 9;
   }
   print d;
}
```

```
0: .segment, data
 1: .int, 0, 0, a
 2: .float, 0, 0, c
 3: .float, 0, 0, d
 4: .segment, 0, 0, text
 5: syscall, 1, t1, 0
 6: sw, t1, 0, a
 7: syscall, 3, ft1, 0
 8: sw, ft1, 0, c
 9: bne, a, 0, 22
10: lw, ft2, 0, c
11: li, 9, 0, t2
12: toFloat, t2, 0, ft3
13: fmul, ft2, ft3, ft4
14: li, 5, 0, t3
15: toFloat, t3, 0, ft5
16: fdiv, ft4, ft5, ft6
17: li, 32, 0, t4
18: toFloat, t4, 0, ft7
19: fadd, ft6, ft7, ft8
20: sw, ft8, 0, d
21: j, 0, 0, 33
22: lw, ft9, 0, c
23: li, 32, 0, t5
24: toFloat, t5, 0, ft10
25: fsub, ft9, ft10, ft11
26: li, 5, 0, t6
27: toFloat, t6, 0, ft12
28: fmul, ft11, ft12, ft13
29: li, 9, 0, t7
30: toFloat, t7, 0, ft14
31: fdiv, ft13, ft14, ft15
32: sw, ft15, 0, d
33: lw, d, 0, ft16
34: syscall, 4, ft16, 0
```


16


QUESTIONS?
Sejong - How to Create a Programming Language - Tony Mione / 2020
17

17