Tiger to RISC V Compiler

Sourabh Aggarwal
Mentor: Dr. Piyush P. Kurur

September 2019

Department of Computer Science And Engineering
IIT Palakkad

Tiger

```
function try (c : int) =
(
  if c = N
  then printboard ()
  else
    for r := 0 to N - 1 do
      if row[r] = 0 & diag1[r + c] = 0 & diag2[r + 7 - c] = 0
      then
      (
        row[r] := 1; diag1[r + c] := 1; diag2[r + 7 - c] := 1;
        col[c] := r;
        try(c + 1);
        row[r] := 0; diag1[r + c] := 0; diag2[r + 7 - c] := 0
      )
  )
```

RISC V

```
# str1 < str2 ?
stringLess:
    stringLessLoop:
        lb a2 (a0)
        lb a3 (a1)
        blt a2, a3  stringLessA
        bgt a2, a3  stringLessB
        # If we have reached this point that means both are equal
        and if one of them is zero that means other is aswell 0,
        so in case strings are equal, I must return 0.
        beqz a2, stringLessB
        addi a0, a0, 1
        addi a1, a1, 1
        j stringLessLoop
    stringLessA:
        li a0, 1
        jr ra
    stringLessB: li a0, 0 jr ra
```

Extending Upon Previous Compiler

- Wrote compiler to compile Tiger to MIPS last semester (didn't expect to get it working).

Extending Upon Previous Compiler

- Wrote compiler to compile Tiger to MIPS last semester (didn't expect to get it working).
- BUT evidently there were issues.

Issues / Challenges

- The number of arguments that can be given to a function is exactly the number of available argument registers. This is to be fixed to support any number of function arguments.

Issues / Challenges

- The number of arguments that can be given to a function is exactly the number of available argument registers. This is to be fixed to support any number of function arguments.
- Register allocation currently works only on saved registers and temporaries, this can be augmented to use free argument registers as well. (Subjective move)

Issues / Challenges

- The number of arguments that can be given to a function is exactly the number of available argument registers. This is to be fixed to support any number of function arguments.
- Register allocation currently works only on saved registers and temporaries, this can be augmented to use free argument registers as well. (Subjective move)
- Basic blocks have to optimized with optimizations like constant propagation, constant folding, etc.

Issues / Challenges

- The number of arguments that can be given to a function is exactly the number of available argument registers. This is to be fixed to support any number of function arguments.
- Register allocation currently works only on saved registers and temporaries, this can be augmented to use free argument registers as well. (Subjective move)
- Basic blocks have to optimized with optimizations like constant propagation, constant folding, etc.
- String comparison has to be made as simple as "str1 $>$ str2", etc., instead of calling the string comparison functions to determine it.

Issues / Challenges

- The number of arguments that can be given to a function is exactly the number of available argument registers. This is to be fixed to support any number of function arguments.
- Register allocation currently works only on saved registers and temporaries, this can be augmented to use free argument registers as well. (Subjective move)
- Basic blocks have to optimized with optimizations like constant propagation, constant folding, etc.
- String comparison has to be made as simple as "str1 $>$ str2", etc., instead of calling the string comparison functions to determine it.
- To implement Garbage collection.

Issues / Challenges

- To implement ability to include pre-written code (header) files.

Issues / Challenges

- To implement ability to include pre-written code (header) files.
- To implement additional arithmetic operations like left/right shift.

Issues / Challenges

- To implement ability to include pre-written code (header) files.
- To implement additional arithmetic operations like left/right shift.
- Register allocation algorithm has to be implemented with better heuristics.

Issues / Challenges

- To implement ability to include pre-written code (header) files.
- To implement additional arithmetic operations like left/right shift.
- Register allocation algorithm has to be implemented with better heuristics.
- Error messages has to improved in semantic phase.

Issues / Challenges

- To implement ability to include pre-written code (header) files.
- To implement additional arithmetic operations like left/right shift.
- Register allocation algorithm has to be implemented with better heuristics.
- Error messages has to improved in semantic phase.
- Add support for compile time (initial) arguments.

Issues / Challenges

- To implement ability to include pre-written code (header) files.
- To implement additional arithmetic operations like left/right shift.
- Register allocation algorithm has to be implemented with better heuristics.
- Error messages has to improved in semantic phase.
- Add support for compile time (initial) arguments.
- And much more is possible, as like in a commercial product.

Goal for this Semester

This semester, I am mainly focusing on:-

Goal for this Semester

This semester, I am mainly focusing on:-

- Understanding RISC V.

Goal for this Semester

This semester, I am mainly focusing on:-

- Understanding RISC V.
- Recollecting all that was done last semester (3 month gap ☹).

Goal for this Semester

This semester, I am mainly focusing on:-

- Understanding RISC V.
- Recollecting all that was done last semester (3 month gap ☹).
- Refactoring and translating everything to RISC V.

Goal for this Semester

This semester, I am mainly focusing on:-

- Understanding RISC V.
- Recollecting all that was done last semester (3 month gap 😵).
- Refactoring and translating everything to RISC V.
- Writing good documentation of the code for easy recollection.

Goal for this Semester

This semester, I am mainly focusing on:-

- Understanding RISC V.
- Recollecting all that was done last semester (3 month gap 😵).
- Refactoring and translating everything to RISC V.
- Writing good documentation of the code for easy recollection.
- And of course, if there is time, I'll work on the previous mentioned issues else they are to be focused on next semester.

Compiler Course: Phase I - Building AST

Example:-

```
| LET decs IN exps END                (
  A.LetExp({decs = decs, body = A.SeqExp(exps), pos = LETleft})
  )
```

Included many files, viz.

- parse.sml
- errormsg.sml
- tiger.lex
- table.sml, table.sig
- symbol.sml
- tiger.grm, absyn.sml

Compiler Course: Phase II - Building Intermediate Representation Tree

```
datatype stm = SEQ of stm * stm
            | LABEL of label
            | JUMP of exp * label list
            | CJUMP of relop * exp * exp * label * label
            | MOVE of exp * exp
            | EXP of exp
and exp = BINOP of binop * exp * exp
            | MEM of exp
            | TEMP of Temp.temp
            | ESEQ of stm * exp
            | NAME of label
            | CONST of int
            | CALL of exp * exp list
```

Compiler Course: Phase II - Building Intermediate Representation Tree

Included many files, viz.

- types.sml
- env.sml, env.sig
- temp.sml, temp.sig
- findescape.sml
- tree.sml
- risc.sml
- translate.sml
- semant.sml

Till Now

- Understood RISC V and rewrote ”runtime” in RISC V.

Till Now

- Understood RISC V and rewrote ”runtime” in RISC V.
- Worked out till Phase II of Compiler.

Thanks!