

## リスト 1 dbgScanner.rb

```

1  # result
2  # 25,!18,30 | llvm行, メタ変数, C行
3
4  load("loadFile.rb")
5  load("relativeC.rb")
6
7  $WRITE = "/result/gdb.txt"
8
9  write_file = open($WRITE, "w")
10 read_file = open($FileName+".ll", "r")
11
12 read_file.each_line do |line| #1行単位で読み込み
13   flag = 0
14   token = line.split(/[ \n]/)
15   token.each_with_index do |t, i|
16     if t =~ /^dbg/
17       flag = 1
18       next
19     end
20     # p t
21     if flag == 1
22       print "dbg #{t} :#{read_file.lineno}\n"
23
24       read_gen = open("/result/line.txt", "r")
25       read_gen.each_line do |line|
26         token = line.split(/[ \n]/)
27         token.each {|t1| write_file.print "#{read_file.lineno},#{line.chomp}\n" if t == t1}
28       end
29       read_gen.close()
30
31       flag = 0
32     end
33   end #end token
34 end # end each other line
35
36 read_file.close()
37 write_file.close()

```

## リスト 2 grid.rb

```

1  class Grid
2    def initialize
3      @ret = nil
4      @array = {}
5    end
6    attr_accessor :ret, :array
7
8    def InstrGroup(this, thisFunction, thisLabel, thisLine)
9      buf = ""
10     count = 0
11     3.times do |i|
12       if this != nil && this.Function == thisFunction && this.Label == thisLabel
13         buf = buf + "#{this.Opencode}"
14         # print "#{this.Opencode}"
15         this = this.NextLabel
16         count += 1
17       end
18     end
19     # print "\n"
20     if count == 3 then
21       # print "#{buf}\n"
22       array[buf] = thisLine
23       # print " :#{array[buf]}"
24       write_file = open("/result/instr.txt", "a")
25       write_file.print "#{buf},#{thisLine}\n"
26       write_file.close()
27     end
28   end # end InstrGroup
29
30   def BasicGroup(this, thisLabel, thisLine)
31     buf = ""
32     loop{
33       if this != nil && this.Label == thisLabel then
34         print this.Opencode
35         buf = buf + "#{this.Opencode}"
36         this = this.NextLabel
37       else
38         ret = this
39         # if this != nil && this.Opencode == "ret"
40         print "\n";
41         # end
42         break;
43       end
44
45       write_file = open("/result/instr.txt", "a")
46       write_file.print "#{buf},#{thisLine}\n"
47       write_file.close
48
49       # print "\n";
50     }

```

```

51     }
52     ret = this
53     return ret
54 end # end BasicGroup
55
56 def FunctionGroup(this, thisFunction)
57   loop{
58     if this != nil && this.Function == thisFunction then
59       print this.Opcodes
60       this = this.NextLabel
61     else
62       ret = this
63       print "\n";
64       break;
65     end
66   }
67   return ret
68 end # end FunctionGroup
69
70 end # end Class

```

### リスト 3 iden.rb

```

1  # 予約命令
2  $iden = Array.new;
3  # $iden << "define"
4
5  #Terminator Instructions
6  $iden << "ret"
7  $iden << "switch"
8  $iden << "br"
9  $iden << "indirectbr"
10 $iden << "invoke"
11 $iden << "resume"
12 $iden << "catchpad"
13 $iden << "catchendpad"
14 $iden << "catchret"
15 $iden << "cleanupendpad"
16 $iden << "cleanupret"
17 $iden << "terminatepad"
18 $iden << "unreachable"
19
20 # Binary Operations
21 $iden << "add"
22 $iden << "fadd"
23 $iden << "sub"
24 $iden << "fsub"
25 $iden << "mul"
26 $iden << "fmul"
27 $iden << "udiv"
28 $iden << "sdiv"
29 $iden << "fdiv"
30 $iden << "urem"
31 $iden << "srem"
32 $iden << "frem"
33
34 # Bitwise Binary Operations
35 $iden << "shl"
36 $iden << "lshr"
37 $iden << "ashr"
38 $iden << "and"
39 $iden << "or"
40 $iden << "xor"
41
42 # Vector Operations
43 $iden << "extractelement"
44 $iden << "insertelement"
45 $iden << "shufflevector"
46
47 # Aggregate Operations
48 $iden << "extractvalue"
49 $iden << "insertvalue"
50 $iden << "insertvalue"
51
52 #Memory Access and Addressing Operations
53 # $iden << "alloca"
54 $iden << "load"
55 $iden << "store"
56 $iden << "fence"
57 $iden << "cmpxchg"
58 $iden << "atomicrmw"
59 $iden << "getelementptr"
60
61 # Conversion Operations
62 $iden << "trunc" #to
63 $iden << "zext" #to
64 $iden << "sext" #to
65 $iden << "fptrunc" #to
66 $iden << "fpext" #to
67 $iden << "fptoui" #to
68 $iden << "fptosi" #to

```

```

69 $iden << "uitofp" #to
70 $iden << "sitofp" #to
71 $iden << "ptrtoint" #to
72 $iden << "inttoptr" #to
73 $iden << "bitcast" #to
74 $iden << "addrspacecast" #to
75
76 # Other Operations
77 $iden << "icmp"
78 $iden << "fcmp"
79 $iden << "phi"
80 $iden << "select"
81 $iden << "call"
82 $iden << "va_arg"
83 $iden << "landingpad"
84 $iden << "cleanup"

```

#### リスト 4 list.rb

```

1  # リスト構造
2  class List
3    def initialize
4      @Function = nil # 所属関数
5      @Label = nil # 所属ラベル
6      @Opcode = nil # 命令名
7      @Line = nil # 行番号
8      @NextLabel = nil # 次のラベル
9      @OpocodetoArray = Array.new(GROUP_COUNT)
10   end
11
12   attr_accessor :Function, :Label, :Opcode, :Line, :NextLabel, :OpocodetoArray; #インスタンス変数の参照や更新
13
14   def toArray(this, i, opcode)
15     this.OpocodetoArray[i] = opcode
16     print "toArray:#{this.OpocodetoArray[i]}\n"
17   end
18
19   def add_last(function, label, opcode, line)
20     this = self
21     this = this.NextLabel until this.NextLabel.nil?
22     this.NextLabel = List.new
23     this.NextLabel.Function = function
24     this.NextLabel.Label = label
25     this.NextLabel.Opcode = opcode
26     this.NextLabel.Line = line
27   end
28
29   def size
30     this = self
31     i = 0
32     i += 1 while this = this.NextLabel
33     return i
34   end
35
36   def each
37     this = self.NextLabel
38     self.size.times do
39       # for var in this.OpocodetoArray do
40       #   print(Kconv.tosjis("Color = " + var + "\n"))
41       #   yield "#{var}"
42       # end
43       # this.OpocodetoArray.each{ |var|
44       #   yield "#{var}"
45       # }
46       # yield "#{this.OpocodetoArray[0]}: "
47       # yield "#{this.OpocodetoArray[1]}: "
48       # yield "#{this.OpocodetoArray[2]}: "
49       yield "#{this.Line}: "
50       yield "#{this.Function}: "
51       yield "#{this.Label}: "
52       yield "#{this.Opcode}\n"
53       this = this.NextLabel
54     end
55   end
56
57   def cat(nextlabel = self.NextLabel)
58     this = nextlabel
59     thisFunction = this.Function
60     thisLabel = this.Label
61     thisLine = this.Line
62     join = ""
63     # var_p(thisLabel, "")
64     ret = this.NextLabel
65
66     grid = Grid.new
67     ## BasicBlockの中のInstructionを3つに束ねグループ化
68     grid.InstrGroup(this, thisFunction, thisLabel, thisLine)
69
70     ## BasicBlockごとに分割
71     # grid.BasicGroup(this, thisLabel, thisLine)
72

```

```

73     ## functionごとに分割
74     # grid.FunctionGroup(this, thisFunction)
75     return ret
76 end
77
78 end #class List

```

## リスト 5 read.rb

```

1  GROUP_COUNT = 3
2  GROUP_COUNT.freeze
3
4  load("loadFile.rb")
5  load("iden.rb")
6  load("method.rb")
7  load("grid.rb")
8  load("list.rb")
9  system("rm ./result/instr.txt")
10
11
12 # -----
13 # Main処理
14 function = "main"
15 label = 1
16 line = 1
17 opcode = "define"
18
19 token = Array.new
20 opelist = List.new
21
22 file = open($FileName+".ll", "r:utf-8")
23
24 file.each_line do |line| #1行単位で読み込み
25   # puts "#{lineage};#{line}" if line.include?("alloca")
26   token = line.split(" ")
27   token.each_with_index do |t, i|
28     if t == "define"
29       # p token[i+2].scan(/^[a-zA-Z]+[a-zA-Z0-9]*/)
30       label = 1
31       function = token[i+2].scan(/^[a-zA-Z_]+[a-zA-Z0-9]*/)
32       # p function[0]
33     end
34     label = t.scan(/[0-9]+/).if t =~ /<label>/
35     # if t =~ /<label>/
36     # label = t.scan(/[0-9]+/)
37     # end
38     # print "#{file.lineno} #{t}\n" if $iden.include?(t)
39     if $iden.include?(t)
40       # print "#{function[0]}: #{label[0]}: #{file.lineno} #{t}\n"
41       opelist.add_last(function[0], label[0], t, file.lineno)
42     end
43   end #end token[]
44 end # end each other line
45
46
47 label = opelist.cat
48 while label != nil do
49   label = opelist.cat(label)
50 end
51
52
53 # 表示
54 # opelist.each {|i| print i}
55
56 # -----
57
58 =begin
59 ruby read.rb | sort | uniq -c | sort -r
60
61 $ ruby read.rb | sort | uniq -c | sort
62
63 =end

```

## リスト 6 test.rb

```

1  =begin
2  # 付属ブロックの有無で動作変更
3  def call_block2()
4    if block_given?
5      yield
6    else
7      puts "ブロックが付属されていません"
8    end
9  end
10 # 実行
11 call_block2 { puts "ブロック内部" } #=> ブロック内部
12 call_block2() #=> ブロックが付属されていません
13 =end
14
15 buf = "aa"

```

```

16 scores = {} # 空のハッシュを作成
17 scores[buf] = "zz" # キー"Alice"、値80のペアを追加
18 p scores[buf] # キー"Alice"の値を取り出し
19 # 3つのキー+値からなるハッシュを作成
20 user = { :name => "k-sato", :email => "k-sato@foo.xx.jp",
21         :address => "Tokyo" }
22 p user[:name] # キー:nameの値を取り出し
23
24 class Foo
25   @name
26   def name
27     @name
28   end
29   def name=(value) # 呼び出しは'~.name = "hogs"'でOK
30     @name = value
31   end
32 end
33
34 obj = Foo.new()
35 p obj.name = "zzz"

```

## リスト 7 method.rb

```

1 # 配列の内容を表示 print_r(array);
2 def print_r(*array)
3   array.each do |element|
4     puts element
5   end
6 end
7
8 def var_p(var, split)
9   print sprintf("%15s #{split}", var)
10 end

```

## リスト 8 loadFile.rb

```

1 # 対象ファイル
2 # $FileName = "/Users/hiro/Program/C/common/iff/A"
3 # $FileName = "/Users/hiro/Program/C/common/game/gomoku/game"
4 $FileName = "/Users/hiro/Program/C/Sample/Sample3/iff-switchA"
5 $FileName = "/Users/hiro/Program/C/Sample/Sample1/matricsA"
6 # $FileName = "/Users/hiro/Program/C/Sample/Sample2/iff-elseA"
7 # $FileName = "/Users/hiro/Program/seminar/C/BMP_Creater/Create"
8 # $FileName = "/Users/hiro/Program/C/common/test/lisp"

```

## リスト 9 relativeC.rb

```

1 load("loadFile.rb")
2
3 $WRITE = "./result/line.txt"
4
5 $FILELINE = `grep -n "${FileName}.c" | wc -l`.to_i # 行数数え
6
7 write_file = open($WRITE, "w")
8 read_file = open($FileName+".ll", "r")
9
10 read_file.each_line do |line| #1行単位で読み込み
11   next unless line =~ /^A/
12
13   token = line.split(/[ ,]/)
14   buf = ""
15   token.each_with_index do |t, i|
16     # next unless t =~ /^A![0-9]/
17     if i == 0 && t =~ /^A![0-9]/
18       buf = t
19       next
20     end
21     if i == 4 && t.to_i <= $FILELINE && buf != ""
22       print "to : #{t.to_i} : #{buf}\n";
23       write_file.print "#{buf},#{t.to_i}\n";
24       buf = nil
25     end
26   end #end token
27 end # end each other line
28
29 read_file.close()
30 write_file.close()
31
32 =begin
33 llvm, C
34 !1,0
35 !2,0
36 !3,0
37 !7,2
38 !8,2
39 !9,0
40 !10,4
41 !11,6
42 !13,7

```

```

43 !15,8
44 !16,8
45 !18,9
46 !20,10
47 !21,11
48 !23,13
49
50 =end

```

## リスト 10 summarize.rb

```

1  load("read.rb")
2  instr_file = open("/result/instr.txt", "r")
3
4  def relativeMeta(num)
5    buf = ""
6    gdb_file = open("/result/gdb.txt", "r")
7    gdb_file.each_line do |gdb_line|
8      token = gdb_line.split(/[,(\n)]/)
9      if token[0] == num
10        buf = token[1]
11      end
12    end
13    gdb_file.close()
14    return buf
15  end
16
17  def relativeC(num)
18    buf = ""
19    gdb_file = open("/result/gdb.txt", "r")
20    gdb_file.each_line do |gdb_line|
21      token = gdb_line.split(/[,(\n)]/)
22      if token[0] == num
23        buf = token[2]
24      end
25    end
26    gdb_file.close()
27    return buf
28  end
29
30
31  instr_list = {}
32  line_list = {}
33  instr_file.each_line do |instr_line|
34    token = instr_line.split(/[,(\n)]/)
35
36    if instr_list.has_key?(token[0])
37      instr_list[token[0]] += 1
38      line_list[token[0]] = line_list[token[0]] + "," + token[1] # llvm
39      # line_list[token[0]] = line_list[token[0]] + "," + relativeMeta(token[1]) # meta
40      # line_list[token[0]] = line_list[token[0]] + "," + relativeC(token[1]) # C
41    next
42  end
43  instr_list[token[0]] = 1
44  line_list[token[0]] = token[1] # llvm
45  # line_list[token[0]] = relativeMeta(token[1]) # meta
46  # line_list[token[0]] = relativeC(token[1]) # C
47 end
48 instr_file.close()
49
50 # p instr_list.sort {|(k1, v1), (k2, v2)| v2 <=> v1 }
51
52 instr_file = open("/result/instr.txt", "r")
53 instr_file.each_line do |instr_line|
54   token = instr_line.split(/[,(\n)]/)
55   token[0]
56 end
57 instr_file.close()
58
59 for var in instr_list.sort {|(k1, v1), (k2, v2)| v2 <=> v1 } do
60   print "#{var} #{line_list[var[0]]} \n"
61
62   # print "#{var}"
63   # gdb_file = open("/result/dbg.txt", "r")
64   # gdb_file.each_line do |gdb_line|
65     # token = gdb_line.split(/[,(\n)]/)
66     # token[0] == line_list[var[0]]
67   # gdb_file.close()
68 end
69
70
71
72 =begin
73 for var in line_list do
74   print "line = #{var} \n"
75 end
76 =end

```