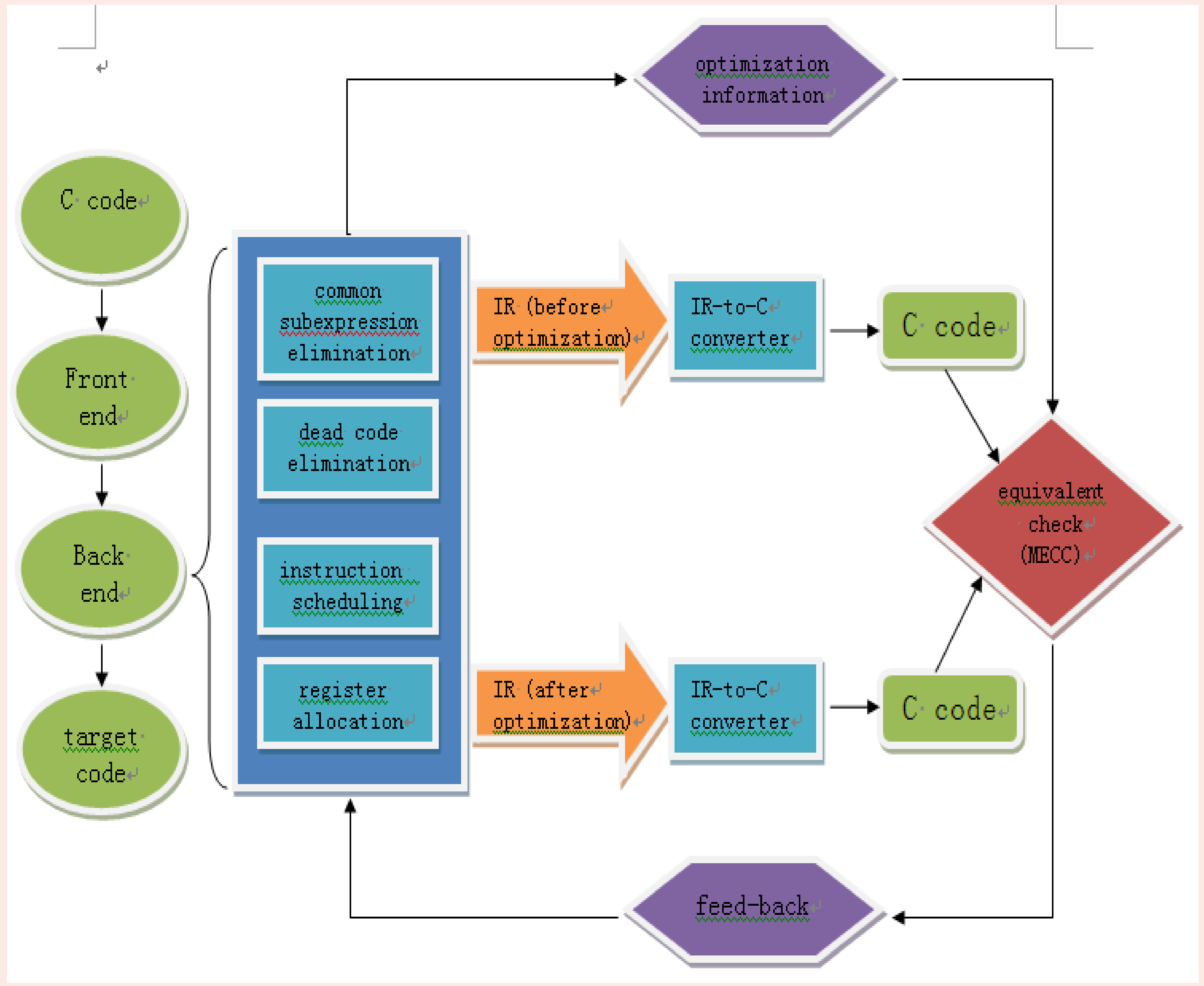
• Compiler triggered C level error check

Zheng Zhiwen, Yunheung Paek (Advisor)
jmjung@optimizer.snu.ac.kr Seoul National University, Korea

Infrastructure for IR optimization error check system

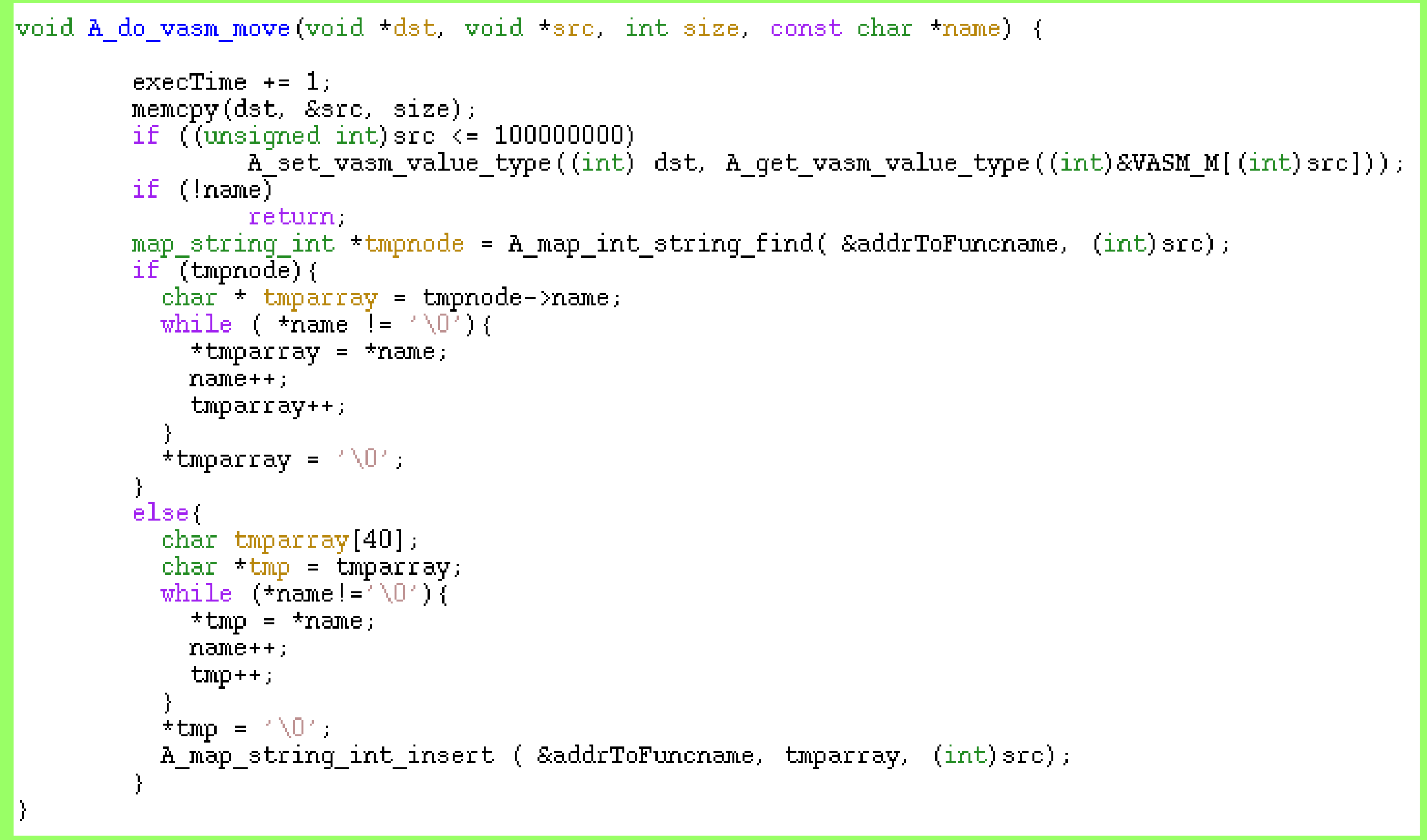


IR-to-C converter

Simulate all the IR instruction types by C code

• Implement each type of IR instruction by function definition

• Function definitions are changeless



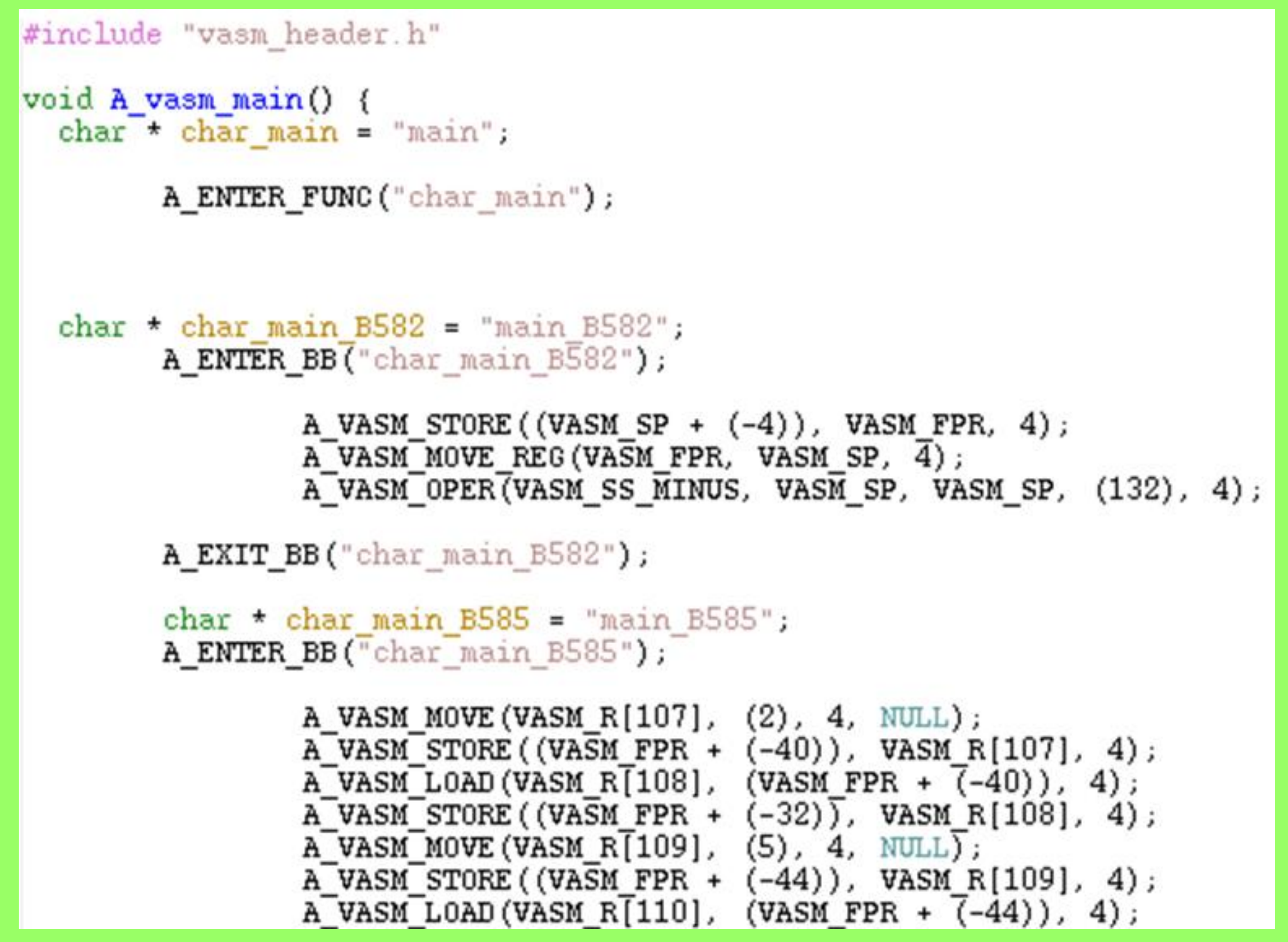
```
void A_do_vasm_move(void *dst, void *src, int size, const char *name) {

        execTime += 1;
        memcpy(dst, &src, size);
        if ((unsigned int)src <= 100000000)
                A_set_vasm_value_type((int) dst, A_get_vasm_value_type((int)&VASM_M[(int)src]));
        if (!name)
                return;
        map_string_int *tmpnode = A_map_int_string_find( &addrToFuncname, (int)src);
        if (tmpnode){
          char * tmparray = tmpnode->name;
          while ( *name != '\0'){
            *tmparray = *name;
            name++;
            tmparray++;
          }
          *tmparray = '\0';
        }
        else{
          char tmparray[40];
          char *tmp = tmparray;
          while (*name!='\0'){
            *tmp = *name;
            name++;
            tmp++;
          }
          *tmp = '\0';
          A_map_string_int_insert ( &addrToFuncname, tmparray, (int)src);
        }
}
```

Implement IR instructions by C code

• Collect SoarGen IR information

• Convert all the IR instructions by function call



```
#include "vasm_header.h"

void A_vasm_main() {
  char * char_main = "main";

        A_ENTER_FUNC("char_main");


  char * char_main_B582 = "main_B582";
        A_ENTER_BB("char_main_B582");

                A_VASM_STORE((VASM_SP + (-4)), VASM_FPR, 4);
                A_VASM_MOVE_REG(VASM_FPR, VASM_SP, 4);
                A_VASM_OPER(VASM_SS_MINUS, VASM_SP, VASM_SP, (132), 4);

        A_EXIT_BB("char_main_B582");

        char * char_main_B585 = "main_B585";
        A_ENTER_BB("char_main_B585");

                A_VASM_MOVE(VASM_R[107], (2), 4, NULL);
                A_VASM_STORE((VASM_FPR + (-40)), VASM_R[107], 4);
                A_VASM_LOAD(VASM_R[108], (VASM_FPR + (-40)), 4);
                A_VASM_STORE((VASM_FPR + (-32)), VASM_R[108], 4);
                A_VASM_MOVE(VASM_R[109], (5), 4, NULL);
                A_VASM_STORE((VASM_FPR + (-44)), VASM_R[109], 4);
                A_VASM_LOAD(VASM_R[110], (VASM_FPR + (-44)), 4);
```

Memory Comparison-based Clone detector (MeCC)

Semantic clones detector

• Use a path-sensitive semantic-based static analyzer to estimate the memory states at each procedure's exit point

• Compare the memory states to determine procedure clones

• Independent of syntactic similarity of clone

• Use Sparrow (a commercialized detector for memory error) as underlying analyzer

```
1  int* foo(list *a,
2             int b){
3    res = 0;
4    if (a->len > 5)
5       res = bar(b);
6    return res;
7  }
8  int* bar(int x){
9    int *m = 0;
10   if (x > 0)
11      m = malloc(x);
12   return m;
13 }
```

The abstract memory state at line 6

a	$\{\langle \mathtt{true}, \alpha\rangle\}$
α.len	$\{\langle \mathtt{true}, \beta\rangle\}$
b	$\{\langle \mathtt{true}, \gamma\rangle\}$
res	$\{\langle \beta > 5 \wedge \gamma > 0, \ell\rangle, \langle \beta \leq 5 \vee (\beta > 5 \wedge \gamma \leq 0), 0\rangle \}$

The procedural summary of bar

$x > 0$ return alloc
$x \leq 0$ return 0

```
1 int* foo2(list2 *x,
2            int y){
3   int ret = 0;
4   if (x->val > 5 && y > 0)
5     ret = malloc(y);
6   return ret;
7 }
```

The abstract memory state at line 6

x	$\{\langle \mathtt{true}, \alpha\rangle\}$
α.val	$\{\langle \mathtt{true}, \beta\rangle\}$
y	$\{\langle \mathtt{true}, \gamma\rangle\}$
ret	$\{\langle \beta > 5 \wedge \gamma > 0, \ell\rangle, \langle \beta \leq 5 \vee \gamma \leq 0, 0\rangle \}$

SoarGen IR

Intermediate Representation

•Graph based

• Hierarchical structure

• Visualization tool for improvement and optimization