TD07: Sémantique et TDL. : Generation de code

1. Types simples et couple

我们可以将高级语言转换成*虚拟机可识别的通用汇编语言

```
/*  <int, int> c = {47,53}; */
PUSH 2
LOADL 47
LODAL 53
STORE (2) 0[SB]
```

```
/*  int a = fst c;  */
PUSH 1
LOAD (2) 0[SB]
POP (0) 1
STORE (1) 2[SB]
/*  int b = snd c;  */
PUSH 1
LOAD (2) 0[SB]
POP (1) 1
STORE (1) 3[SB]
```

```
/* while(a * b != test) { // loop_body }  */
etiq_begin_while_1
  LOAD (1) 2[SB]
  LOAD (1) 3[SB]
  SUBR IMul
  LOADL 0
  SUBR INeq
  JUMPIF (0) etiq_begin_while_1
  # LOOP_BODY
```

```
/* loop_body :  */
if (a > b) {
  // then_condition_1
  int na = a - b;
  a = na;
} else {
  // else_condition_1
  int nb = b - a;
  b = nb;
}
// end_condition_1
```

```
LOAD (1) 2[SB]
LOAD (1) 3[SB]
SUBR IGTR
JUMPIF (0) etiq_else_condition
  ### then_condition_1
  PUSH 1
  LOAD (1) 2[SB]
  LOAD (1) 3[SB]
  SUBR 1 SUB
  STORE (1) 4[SB]
  LOAD (1) 4[SB]
  STORE (1) 2[SB]
  POP (0) 1
  ###
JUMP etiq_end_condition_1
# end_condition_1
```

1.2 Proposer des actions sémantiques pour la generation de code.

```
public String getCode() {
  String code;
  for (Instruction i : instruction) {
    code += i.getCode();
  }
  return code + "POP (0) " + this.getlength + "\n");
}
```

2. Type enregistrement

Soit le programme :

```
test{
  typedef struct Pointi { int x; int y;} Point;
  typedef struct Segmenti { Point ext1; Point ext2;} Segment;
  // ------------
  Segment s = {{0,1}, {2,3}};
  int x1 = s.ext1.x;
  int y2 = s.ext2.y;
  s.ext2.x = x1;
  s.ext1.y = y2;
}
```

```
# --- Segment s = {{0,1}, {2,3}}; ---
PUSH 4
LOADL 0
LOADL 1
LOADL 2
LOADL 3
```

```
STORE(4) 0[SB]
# --- int x1 = s.ext1.x; ---
PUSH 1
LOAD (1) 0[SB]
STORE 1 4[SB]
# --- int y2 = s.ext2.y; ---
PUSH 1
LOAD (1) 3[SB]
STORE 1 5[SB]
# --- s.ext2.x = x1; ---
LOAD (1) 4[SB]
STORE 1 2[SB]
# --- s.ext1.y = y2; ---
LOAD (1) 5[SB]
STORE 1 1[SB]
# ------------
POP (0) 6
HALT
```

3. Type tableau et pointeur

Soit le programme :

```
test{
  int v = 1;
  int *ptr = &v;
  int j = *ptr;
  *ptr = 2;
  int t[] = new int[5];
  int i = t[3];
  t[3] = 4;
}
```

换成*虚拟机可识别的通用汇编语言

```
# --- int v = 1; ---
PUSH 1
LOADL 1
STORE (1) 0[SB]
# --- int *ptr = &v; ---    # 指针ptr 指向 变量v的地址值，即 *ptr = 1
PUSH 1
LOADA 0[SB] # 将地址0[SB]入栈
STORE (1) 1[SB]
# --- int j = *ptr; ---     # j 为 指针ptr 指向的地址
PUSH 1
LOAD (1) 1[SB]
LOADI (1)
STORE (1) 2[SB]
# --- *ptr = 2; ---         # *ptr意为 指针ptr所指向的地址中的 *值*
```

```
LOADL 2
LOAD (1) 1[SB]
STOREI (1)
# --- int t[] = new int[5]; ---
PUSH 1
LOADL 5 # 数组的大小，int [5]，所以为5
LOADL 1 # 每个 element 的大小，对于int来讲是1，对于 <int, int>couple 来讲是2
SUBR IMUL # 整个数组所占内存的大小 = table_size * element_size
SUBR MALLOC # 按照之前的数字大小分配内存大小，并返回其地址
STORE (1) 3[SB] # 将地址存入 3[SB]，例如 =17
# --- int i = t[3]; ---
PUSH 1
LOAD (1) 3[SB] # 加载3[SB]里的值，即之前table的地址 =17
LOADL 3 # table[n]，n=3
LOADL 1 # table中每个element的大小
SUBR IMUL # 计算实际的内存大小 3*1
SUBR IADD # 相当于内存寻址=绝对块地址+相对地址：table的起始地址+table内的地址
LOADI (1) # 将上一步得到的(1)个地址拷贝到栈顶
STORE (1) 4[SB] # 将(1)个值存入 4[SB]
# --- t[3] = 4; ---
LOADL 4
LOAD (1) 3[SB]
LOADL 3
LOADL 1
SUBR IMUL
SUBR IADD
STOREI (1)
# ------------
POP (0) 5
HALT
# ------------
```