Milestone 3 - Parser for MiniC

1 Grammar

During our implementation, we found some of the grammars are actually impossible for MiniC, so we slightly modified the grammars specified in the Project 1. First, the assignments can only be used upon l-values. Next, the function calls must start with an identifier. Finally, expressions related to pointers and references are slightly modified because not all expressions can be dereferenced or can be used on address-of. The other parts are kept intact.

2 Register Allocation

The register allocation is kept intuitive. We use a store-after-use strategy, which is, store each register back to the identifier immediately after usage when possible. Also, we will use the same register across some complicated expression evaluations to save registers for other actions. `NextReg` and `NextArgReg` is used in this case, which will assign the next register available, and will return error when all available registers ran out. This strategy is the most intuitive one, which is not very efficient nor smart enough. However, this strategy works in most cases.

3 Compile & Run

To compile and run the program:

```
> cd Pr3
> hacs Pr3.hx
> ./Pr3.run --scheme=Compile <test_file_name>
```

4 Sample Test Outputs

simplest.MC:

```
function int main() {
  var int a;
  a = 1;
  return a;
}
```

It gives the following result:

```
 main   STMFD SP! , {R4-R11, LR}
   MOV R12, SP
   DEF a =  -4
   MOV R4, # 1
   STR R4, [R12, &a]
   LDR R4, [R12, &a]
   MOV R0, R4
   B L
```

```
9
10 L   MOV SP, R12
11   LDMFD SP! , {R4-R11, PC}
```

Funccall.MC:

```
1 function int func(int a, int b) {
2   return a + b;
3 }
4
5 function int func2(int a, int b) {
6   return a - b;
7 }
8
9 function int func3() {
10   return 2;
11 }
12
13 function int main() {
14   return func(1, 2) * func(2, 3) + func3();
15 }
```

It gives the following result:

```
1  func   STMFD SP! , {R4-R11, LR}
2    MOV R12, SP
3    LDR R4, [R12, &a]
4    LDR R5, [R12, &b]
5    ADD R4, R4, R5
6    MOV R0, R4
7    B L
8
9  L   MOV SP, R12
10   LDMFD SP! , {R4-R11, PC}
11
12 func2   STMFD SP! , {R4-R11, LR}
13   MOV R12, SP
14   LDR R4, [R12, &a]
15   LDR R5, [R12, &b]
16   SUB R4, R4, R5
17   MOV R0, R4
18   B L_20
19
20 L_20   MOV SP, R12
21   LDMFD SP! , {R4-R11, PC}
22
23 func3   STMFD SP! , {R4-R11, LR}
24   MOV R12, SP
25   MOV R4, # 2
26   MOV R0, R4
27   B L_50
28
29 L_50   MOV SP, R12
30   LDMFD SP! , {R4-R11, PC}
31
32 main   STMFD SP! , {R4-R11, LR}
```

```
   MOV R12, SP
   MOV R4, # 1
   MOV R0, R4
   MOV R4, # 2
   MOV R1, R4
   BL func
   MOV R4, R0
   MOV R5, # 2
   MOV R0, R5
   MOV R5, # 3
   MOV R1, R5
   BL func
   MOV R5, R0
   MUL R4, R4, R5
   B func3
   MOV R5, R0
   ADD R4, R4, R5
   MOV R0, R4
   B L_64

L_64   MOV SP, R12
   LDMFD SP! , {R4-R11, PC}
```

strlen.MC:

```
// Compute string length in Mini-*-C-*-.
function int strlenchar string) var int length;length = 0;while (*string) length = length +
    1;string = string + 1;return length;function int main(*char input) var int dummy;dummy =
    puts("The length of the string is ");dummy = puti(strlen(input));return 0;
```

It gives the following result:

```
strlen   STMFD SP! , {R4-R11, LR}
   MOV R12, SP
   DEF length =  -4
   MOV R4, # 0
   STR R4, [R12, &length]

WHILE_REPEAT   LDR R4, [R12, &string]
   LDR R4, [R12, R4]
   CMP R4, # 0
   BEQ WHILE_NEXT
   LDR R4, [R12, &length]
   MOV R5, # 1
   ADD R4, R4, R5
   STR R4, [R12, &length]
   LDR R4, [R12, &string]
   MOV R5, # 1
   ADD R4, R4, R5
   STR R4, [R12, &string]
   B WHILE_REPEAT

WHILE_NEXT   LDR R4, [R12, &length]
   MOV R0, R4
   B L

```

```
25 L   MOV SP, R12
26   LDMFD SP! , {R4-R11, PC}
27
28 main   STMFD SP! , {R4-R11, LR}
29   MOV R12, SP
30   DEF dummy =  -4
31   DCS  "The length of the string is "
32   MOV R0, R4
33   BL puts
34   MOV R4, R0
35   STR R4, [R12, &dummy]
36   LDR R4, [R12, &input]
37   MOV R0, R4
38   BL strlen
39   MOV R4, R0
40   MOV R0, R4
41   BL puti
42   MOV R4, R0
43   STR R4, [R12, &dummy]
44   MOV R4, # 0
45   MOV R0, R4
46   B L_92
47
48 L_92   MOV SP, R12
49   LDMFD SP! , {R4-R11, PC}
```