1 Assembly Listing with NASM Syntax

Listing 1: Assembly listing with NASM syntax using the custom language `nasm/lang` and the custom style `nasm/style`.

```
1  extern printf
2  extern exit
3  global _start
4
5  [section .text]
6  _start:
7          mov     rdi, hello ; arg1: "hello world\n"
8          call    printf     ; printf("hello world\n")
9          mov     rdi, 0     ; arg1: 0
10         call    exit       ; exit(0)
11
12 [section .rodata]
13 hello:
14         db      'hello world', 10, 0
```

2 C Listing

Listing 2: C listing using the default C language and the custom style `c/style`.

```c
#include <stdio.h>

// main prints "hello world" to standard output.
int main(int argc, char **argv) {
  printf("hello world\n");
  return 0;
}
```

3 Go Listing

Listing 3: Go listing using the custom language `go/lang` and the custom style `go/style`.

```
// fib is a command which prints the Fibonacci sequence.
package main

import "fmt"

func main() {
    c := make(chan int)
    go fib(c)
    fmt.Println("The Fibonacci sequence:")
    for x := range c {
        fmt.Println(x)
    }

}

func fib(c chan<- int) {
    a, b := 0, 1
    for a >= 0 {
        c <- a
        a, b = b, a+b
    }
    close(c)
}
```

4 REIL Instructions Listing

Listing 4: REIL instructions listing using the custom language `reil/lang` and the custom style `nasm/style`.

```
1  bisz t8, , ZF
2  str t8, , esp
3  ldm 16815620, , t0
4  str t0, , eax
5  sub esp, 4, esp
6  and esp, 4294967295, esp
7  stm esi, , esp
8  sub esp, 4, esp
9  and esp, 4294967295, esp
10 stm t2, , esp
11 add -4, ebp, t0
12 and t0, 4294967295, t1
13 stm eax, , t1
14 sub esp, 4, t0
15 and t0, 4294967295, esp
16 stm 16805479, , esp
17 jcc 1, , 16805367
```

5 LLVM IR Listing

Listing 5: LLVM IR listing using the custom language `llvm/lang` and the custom style `nasm/style`.

```
target triple = "x86_64-unknown-linux-gnu"

@s = private constant [13 x i8] c"hello world\0A\00"

; main prints "hello world" to standard output.
define i32 @main(i32 %argc, i8** %argv) {
        %1 = getelementptr [13 x i8]* @s, i32 0, i32 0
        call i32 (i8*, ...)* @printf(i8* %1)
        ret i32 0
}

declare i32 @printf(i8*, ...)
```