CSD1100

Assembler - Preprocessor

Vadim Surov

Preprocessor

- NASM contains a powerful macro processor, which supports
  - conditional assembly,
  - multi-level file inclusion,
  - two forms of macro (single-line and multi-line),
  - and more
- Preprocessor directives all begin with a % sign.

```
%define isTrue   1
%define isFalse  0
val:    db      isFalse
```

Multi-Line Macros: %macro

Next definition is a multi-line macro definition:

```
%macro EXIT 1
    mov rax, 60
    mov rdi, %1
    syscall
%endmacro
```

Number of parameters

First parameter

So you would invoke the macro with a call such as:

```
EXIT 0
```

Actual value of the parameter

Example with 2 parameters

```
1   %macro PRINTF 2
2       push rdi
3       push rsi
4       push rax
5       mov rdi, %1
6       mov rsi, %2
7       xor rax, rax
8       call printf
9       pop rax
10      pop rsi
11      pop rdi
12  %endmacro
```

Use case

```
32  section .data
33  fmt     db "%lld",10,0
34  n       dq 30
35
36  section .text
37      global _start
38      extern printf
39
40  _start:
41      mov rax, 10
42      PRINTF fmt, rax
43      PRINTF fmt, 20
44      PRINTF fmt, [n]
45      EXIT
```

%include a file

- Same as #include a file in C
- Use it to include same macro-definitions in different files of a project

```
8   %include "macros.inc"
9
10  section .data
11  fmt     db "%lld",10,0
12  n       dq 30
13
14  section .text
15      global _start
16      extern printf
17      extern time
18
19  _start:
20      mov     rdi, 0
21      call    time
22      PRINTF fmt, rax
23      EXIT
```

Cont.

```
8   %include "macros.inc"
9
10  section .data
11  fmt     db "%lld",10,0
12  n       dq 30
13
14  section .text
15      global _start
16      extern printf
17      extern time
18
19  _start:
20      mov     rdi, 0
21      call    time
22      PRINTF fmt, rax
23      EXIT
```

macros.inc

```
1   %macro PRINTF 2
2       push rdi
3       push rsi
4       push rax
5       mov rdi, %1
6       mov rsi, %2
7       xor rax, rax
8       call printf
9       pop rax
10      pop rsi
11      pop rdi
12  %endmacro
13
14  %macro EXIT 0
15      mov rax, 60
16      mov rdi, 0
17      syscall
18  %endmacro
```

time

time.asm

```
1   ; Time
2   ; Output time in seconds since 1 Jan 1970
3   ; Run: $ nasm -f elf64 time.asm && ld -dynam
4
5   %include "macros.inc"
6
7   section .data
8   fmt     db "%lld",10,0
9
10  section .text
11      global _start
12      extern printf
13      extern time
14
15  _start:
16      mov     rdi, 0
17      call    time
18      PRINTF fmt, rax
19      EXIT
```

A random number generator

rand.asm

```
1   ; Random number generator using c's rand().
2   ; rand()'s result is the number
3   ; Run: $ nasm -f elf64 rand.asm && ld -dynamic-linker /lib6
4
5   %include "macros.inc"
6
7   section .data
8   fmt     db "%lld",10,0
9
10  section .text
11      global _start
12      extern printf
13      extern time
14      extern srand
15      extern rand
```

```
16
17  _start:
18      ; Get the current time
19      xor     rdi, rdi
20      call    time
21
22      ; Seed the random number generator
23      mov     rdi, rax
24      call    srand
25
26      ; Get the random number
27      call    rand
28
29      ; Map the number into range [0,100)
30      mov     rdx, 0
31      mov     rbx, 100
32      idiv    rbx     ; Note, remainder in rdx
33
34      PRINTF fmt, rdx
35      EXIT
```

References

1. NASM documentation
   https://www.nasm.us/doc/
2. Preprocessor:
   https://www.nasm.us/xdoc/2.10rc8/html/nasmdoc4.html