Converting x86 assembly from masm to nasm

Left 404

Posted on 4 January, 2011

Adrift in the 21st Century

Masm, the Microsoft assembler, is the most commonly taught x86 assembler. Unfortunately, its use is limited to Windows. **nasm** is a free cross-platform x86 assembler which supports all the common x86 operating systems – Linux, MacOS X and Windows. Unlike the GNU assembler, it uses the same Intel syntax that masm does. Still, there are some differences.

What follow are my notes on converting x86 assembly code from use with mas to nasm.

I have also posted a simple example highlighting some of those changes.

1. Nasm is case sensitive. This is particularly important for labels. `myFunc` and `myfunc` are not the same thing.
2. Code labels and procedures are defined and treated the same. Procedures begin with a normal code label and end with a ret instruction. There is no PROC/ENDP syntax. As a result, you cannot reuse the same label name within different procedures.

masm	nasm

`name PROC`
`...`
`ret`
`name ENDP`
	`name:`
	`...`
	`ret`

3. Nasm programs are divided into 2 portions, data and text (which corresponds to masm's code section).

masm	nasm
`.data`	`SECTION .data`
`.code`	`SECTION .text`

4. Data labels are automatically treated as addresses, unless they are enclosed in square brackets. There is consequently no OFFSET keyword.

masm	nasm
`mov edx, OFFSET label`	`mov edx, label`
`mov ax, label`	`mov ax, [label]`

   Likewise, when reading data from a label, square brackets are required.

masm	nasm
`mov DWORD PTR [ebp], 4`	`mov DWORD [ebp], 4`

   Finally, nasm does not attach a specific type to data labels, so size must usually be specified when

dereferencing.

masm	nasm
`mov [wordlabel], 16`	`mov WORD [wordlabel], 16`

5. Data label syntax is different.

masm	nasm
`name type value`	`name: type value`

Types in these definitions are different too. Where in masm you would use BYTE, WORD, DWORD, and so on, you use db, dw, dd etc.

masm	nasm
`val1 DWORD 3`	`val1: dd 3`

6. Uninitialized data labels also have a different syntax. In particular, ? is not recognized.

masm	nasm
`name type ?`	`name: restype count`

Here, restype would be one of resb (BYTE), resw (WORD), resd (DWORD) etc. and count is the number of blocks of size type to be reserved.

7. Repetitions are handled differently.

masm	nasm
`name type count DUP (value)`	`name: TIMES count type value`

8. Multiline macro syntax is different.

masm	nasm

`name MACRO arg1, arg2, ...`
`...`
`ENDM`
	`%macro name argcount`
	`...`
	`%endmacro`

Macro arguments are referenced as `%1, %2... %n` corresponding to the first, second and nth arguments respectively.

9. The = directive is not supported. Use `EQU` exclusively.
10. Global symbols (in particular main) must be explicitly exported for the linker. The line `'global main'` should appear at the top of the text section. `'END main'` is no longer needed. Instead of ending main with the exit instruction, simply use 'ret' as in other procedures.
11. The `TITLE` directive does not exist. Use a comment instead.
12. Include syntax also changes.

masm	nasm
`INCLUDE file.inc`	`%include "file.inc"`

This entry was posted in **Assembly**, **Tech** by **dara**. Bookmark the **permalink [http://left404.com/2011/01/04/converting-x86-assembly-from-masm-to-nasm-3/]** .

Comments are closed.