

Assembly x86 usando NASM

Software Básico

Prof. Pedro Carlos da Silva Lara

18/03/2015

NASM

- NASM (Netwide Assembler)
 - *Its syntax is designed to be simple and easy to understand, similar to Intel's but less complex.*



AT&T vs NASM

- AT&T:

```
movl -4(%ebp, %edx, 4), %eax
movl -4(%ebp), %eax
movl (%ecx), %edx
leal 8(,%eax,4), %eax
leal (%eax,%eax,2), %eax
```

- NASM

```
mov     eax, 4
mov     ebx, 1
mov     ecx, msg
mov     edx, msg.len
int     0x80
mov     eax, 1
mov     ebx, 0
int     0x80
```

Estrutura do Programa

- Seção .data
 - Dados inicializados
- Seção .bss
 - Dados não inicializado
- Seção .text
 - Programa

Características

- Estas seções podem ser colocadas por outra ordem.
- As seções são opcionais: um programa pode conter apenas algumas delas

Características

- Comentário ;
 - DICA: Utilize os comentários
- Maiúsculas/minúsculas
 - O NASM é case-sensitive.
- Linha de código:
 - <label>: <instrução> <operandos> ;comentário

Seção .data

- db define byte
 - Var db 1
 - Letra db 'a'
 - Msg db “hello word”, 10
 - Num db 255

Seção .data

- dw define word (16 bits)
 - Tmp dw 65535
 - A dw 90

Seção .data

- dd define double (32 bits)
 - Real dd 1.23456
 - Var dd 0xFFA4
 - X dd 1.234e15

Seção .data

- dq: define quad (64 bits)
 - Value dq 1.234e60
 - Xyz dq 82930485

Seção .bss

- RESB-reserve byte
 - buffer: resb 64 (reserva espaço para 64 bytes)
- RESW-reserve word
 - wordvar: resw 1 (reserva espaço para uma word)
- o RESD-reserve double word
 - doublewordvar: resd 10 (array de 10 double-word)
- o comando EQU: atribui um valor a um símbolo (define uma constante)
 - N EQU 10

Registradores

- Propósito Geral
 - EAX, EBX, ECX, EDX
- Manipulação de pilha
 - EBP, ESP

Instruções

- Exemplo:

```
section .data
```

```
msg db 'Hello, world!',0x0A
```

```
len equ $ - msg ;length of hello string.
```

```
Section .text
```

```
main: ;main
```

```
mov eax, 4 ;system call number (sys_write)
```

```
mov ebx, 1 ;file descriptor (stdout)
```

```
mov ecx, msg ;message to write
```

```
mov edx, len ;message length
```

```
int 0x80 ;call kernel
```

```
mov eax, 1 ;system call number (sys_exit)
```

```
int 0x80
```