O . inicia uma diretiva

O # inicia um comentario

O % inicia o nome de um registo

O $ inicia um valor imediato

.section .data

.equ LINUX\_SYS\_CALL, 0x80

output\_int:

.asciz “My String”

Data – declare global variables or constants

Bss – unitializes memory areas

Text – assembly instructions

Declarer variaveis por ordem decrescente do tipo de dados

number: # nome da variavel

.int 5 # tipo de dados seguido do valor de inicialização

.comm – bloco de memoria global

.lcomm – bloco de memoria local

Registos ia-32

A zona cinzenta não se consegue aceder diretamente à informação

Ex:

Eax : cinzento (16) + ax (16 = ah(8)+al(8) )

Edi : cinzento (16) + di (16)

Ebp e esp não se usa diretamente para guardar números e.t.c

Registos x86-64

Ex:

Rax: cinzento(32) + eax(32 = cinzento (16) + ax (16 = ah(8)+al(8) ) )

Os novos registos são do r8 ao r15

Tipo de dados: sufixo:

Byte b

Word w

Long (double word) l

Quad word q

Mov é usado para copiar dados

Utilização: mov origem, destino

Ex: movb – para copiar valores de 8 bits

Movw – “ 16 bits

Movl- “ 32 bits

Movq – “ 64 bits

Origem e destino têm de ser do mesmo tamanho

Em x86-64, Qualquer instrução que produza um valor de 32 bits, para um registo, altera tmb os outros 32 bits mais significativos desse registo para 0

Movq $5236, %rax - coloca 5236 no registo rax

Movl $-345, %ecx - coloca -345 no registo ecx e zera ainda os 32 bits mais significativos de rcx

Movw $0xFF81, %dx - coloca 0xFF81=-79 no registo dx (não altera os 48 bits mais significativos de rdx)

Movb $0x0A, %al - coloca 0x0A = 10 no registo al (não altera os 56 bits mais significativos de rax)

Uma variável a é referenciada como a(%rip)

Ex: movl %eax, my\_integer(%rip) - copia o valor do registo para a variável em memoria

Movabsq imm, register

Imm pode ser um numero representado em 64 bits complemento para 2

Ex:

Movb $-1, %al #%rax = 0x00112233445566FF

FF é -1 em 8 bits

Se for movw no final fica FFFF

Se for movl no final fica FFFFFFFF e os restantes passam a 0

Com o movq passa tudo a F

Movz – preenche os restantes bytes do destino com 0

Ex: movzbw – move zero-extended byte to word

Movz[bw] origem, destino

Movs – preenche os restantes bytes do destino através da extensão de sinal

Ex: movsbw – move sign-extended byte to word

Movs[bw] origem, destino

Temos de verificar se estamos a trabalhar com números com ou sem sinal e aí decidimos se queremos usar o movz ou movs

Podem parecer iguais se o numero for positivo (porque colocam 0), mas não são iguais