x86 Assembly

32 Bits

Registers

RAX/EAX/AX/AH/AL Acumulador; Usado para Input/output calculos e retorna o valor das funções

RBX/EBX/BX/BH/BL Base; Usado para endereços indexados (usa um registrador como base e um como index)

RCX/ECX/CX/CH/CL Calculos; Armazena contagem de um loop

RDX/EDX/DX/DH/DL Dados; Input/output, as vezes auxilia o registrador RAX para multiplicar/dividir

RSP/ESP/SP/SPL Stack Pointer; Armazena a posição atual da Stack

RBP/EBP/BP/BPL Base Pointer; Ajuda na referência de parâmetros e em outras variáveis da stack como desvio da "base da stack"

RSI/ESI/SI/SIL Usado como um index base para operações em Strings

RDI/EDI/DI/DIL Usado como index destino para operações em Strings

RIP/EIP/IP Armazena a proxima instrução a ser executada

R8-R15 x64 Registradores de uso geral

CS/DS/SS/ES/FS/GS Segmentos de registradores em 16 bits para acessar segmentos especificos da memória como: Code (.text)/Data (.data)/Stack/Extra/General/General
RFLAGS/EFLAGS

.text = Instruções do prograr

.data = valores inicalizados,

.bss = (block starting symbo

Heap = Dinamicamente Aloc
maiores da memória ↓

Stack = Variável Local, para
menores da memória↑

Intructions

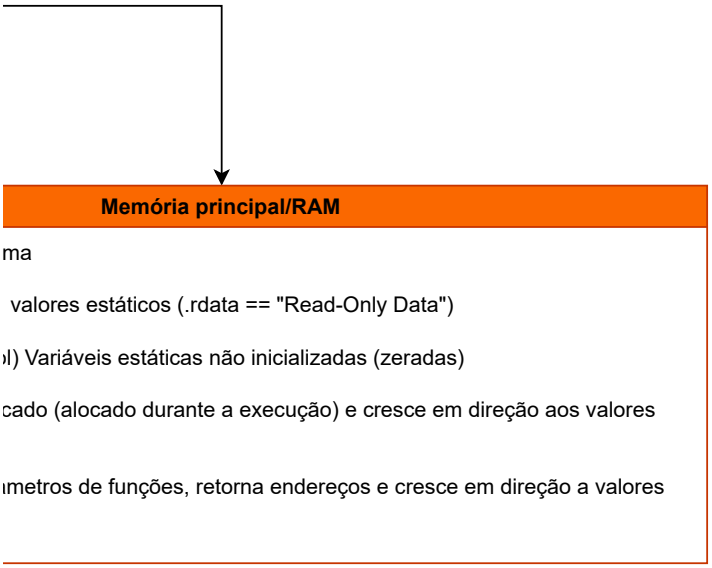
`MOV dest, src`	Move dados de `src` para `dest`.
`PUSH src`	Empilha `src` na pilha.
`POP dest`	Desempilha o topo da pilha para `dest`.
`LEA dest, src`	Calcula o endereço efetivo de `src` e armazena em `dest`.
`XCHG op1, op2`	Troca os valores de `op1` e `op2`.

Instruções Aritméticas

`ADD dest, src`	Soma `dest` e `src`, armazenando o resultado em `dest`.
`SUB dest, src`	Subtrai `src` de `dest`, armazenando o resultado em `dest`.
`MUL src`	Multiplica `AL/AX/EAX` por `src` (sem sinal).
`IMUL src`	Multiplica `AL/AX/EAX` por `src` (com sinal).
`DIV src`	Divide `AX/EAX` por `src` (sem sinal).
`IDIV src`	Divide `AX/EAX` por `src` (com sinal).
`INC dest`	Incrementa `dest` em 1.
`DEC dest`	Decrementa `dest` em 1.
`NEG dest`	Inverte o sinal de `dest` (complemento de 2).

Instruções Lógicas e de Bits

`AND dest, src`	Operação lógica **AND** entre `dest` e `src`.
`OR dest, src`	Operação lógica **OR** entre `dest` e `src`.


Memória principal/RAM
ma
valores estáticos (.rdata == "Read-Only Data")
ol) Variáveis estáticas não inicializadas (zeradas)
cado (alocado durante a execução) e cresce em direção aos valores
ımetros de funções, retorna endereços e cresce em direção a valores

`XOR dest, src`	Operação lógica **XOR** entre `dest` e `src`.
`NOT dest`	Inverte todos os bits de `dest`.
`SHL dest, cnt`	Desloca `dest` para a esquerda por `cnt` bits.
`SHR dest, cnt`	Desloca `dest` para a direita por `cnt` bits (sem sinal).
`SAR dest, cnt`	Desloca `dest` para a direita por `cnt` bits (com sinal).

Saltos e Loops

`JMP label`	Salta incondicionalmente para `label`.
`JE/JZ label`	Salta se **igual/zero** (ZF=1).
`JNE/JNZ label`	Salta se **não igual/não zero** (ZF=0).
`JG/JNLE label`	Salta se **maior** (com sinal).
`JL/JNGE label`	Salta se **menor** (com sinal).
`JA/JNBE label`	Salta se **maior** (sem sinal).
`JB/JNAE label`	Salta se **menor** (sem sinal).
`LOOP label`	Decrementa `ECX` e salta se `ECX ≠ 0`.
`CALL proc`	Chama uma sub-rotina/procedimento.
`RET`	Retorna de uma sub-rotina.