

Arquitetura e organização de computadores

Universidade do Estado de Mato Grosso

Faculdade de Ciências Exatas e Tecnológicas

Prof. Me. João Ricardo dos Santos Rosa

joao.santos@unemat.br

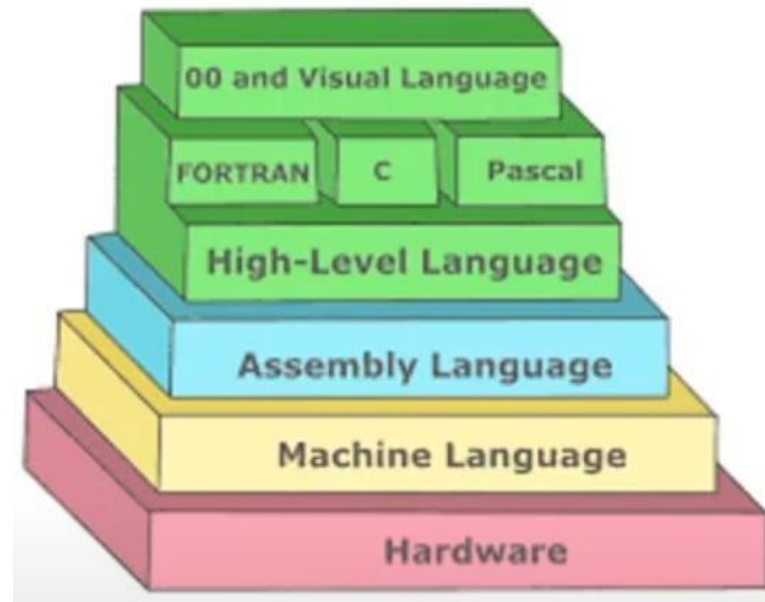
2023/01

Instruções e Linguagem de Máquina

Agenda

- ❑ Introdução: Instruções e Linguagem de Máquina;
- ❑ Representação e elementos da Instrução;
- ❑ Número de Operadores da Instrução
- ❑ Tipos de Operadores em Assembly
- ❑ Conclusão

Introdução Instruções e Linguagem de Máquina






Linguagem de montagem ou Assembly

- ❑ Instruções são códigos de máquinas que o processador é capaz de executar e representadas por sequencias de bits (código binário).
- ❑ Normalmente, são limitadas pelo número de bits do registrador central da CPU (8, 16, 32, 64 ou 128 bits).
- ❑ **Conjunto de Instruções (Instruction Set):** O grupo de instruções que uma determinada máquina/ arquitetura consegue executar

Linguagem de montagem ou Assembly

❑ Notação legível por humanos e equivalente ao código de máquina que uma arquitetura de computador utiliza.

❑ Exemplo:

Binary	→	1011	0101	01010101
				
Assembly	→	MOV	AH	0x55

Contexto - Assembly

Código de
Máquina
(primeira
geração)

1950s

Assembly
(segunda
geração)

1970s

Linguagens
de Alto
nível

(+)Complexidade (-)

(-)Produtividade (+)

Por que utilizar *assembly* hoje em dia?

Código menor e mais eficiente

(+)Desempenho(-)

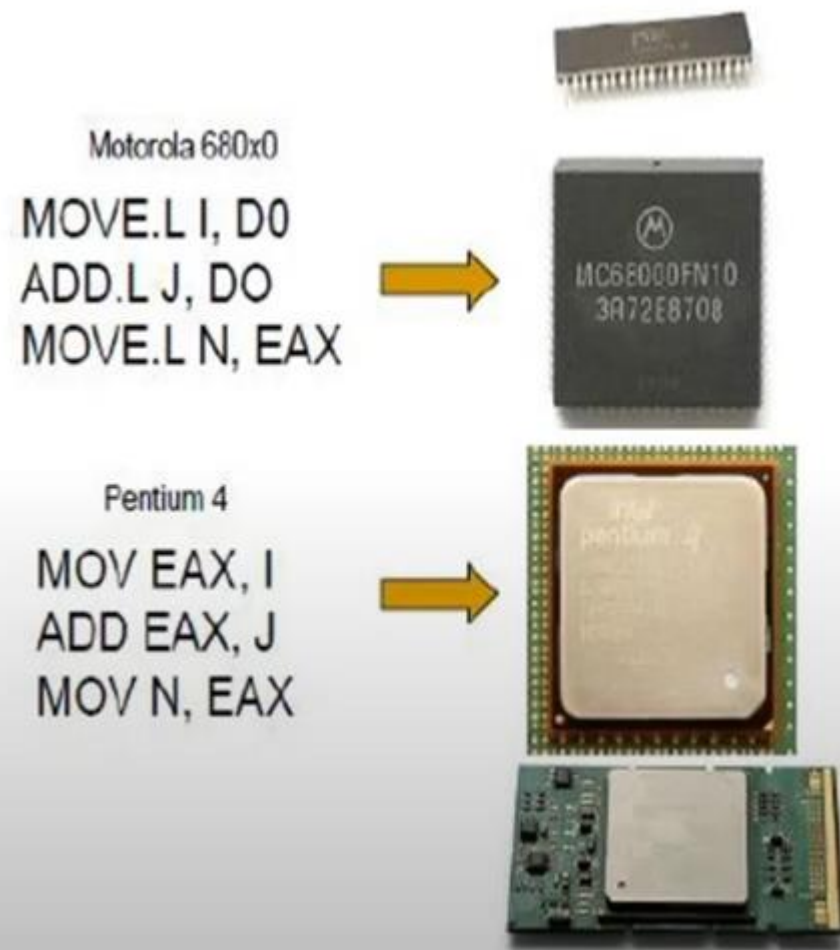
(+)acesso à máquina(-)

Aplicações atuais:

- Acesso direto ao Hardware (ex: Desenvolvimento de Drivers);
- Sistemas Real Time;
- Sistemas embarcados de baixo nível;

Contexto - Assembly

- ❑ Cada Linguagem assembly está intimamente ligado a arquitetura da máquina ou a uma família de maquinas ou processadores



Arquitetura	Qtd Registradores	Bits	Introduzido (Ano)
6502	3	8	1975
x86	16	16-32	1978
ARM	16	32	1983
IBM/360	16	32	1964
Z/ARchitetur e	16	32-64	2000
UltraSPARC	32	64	1995
Alpha	32	64	1992
POWER	64	32-64	1992
X86-64	64	64	2000
Itanium	128	64	2001

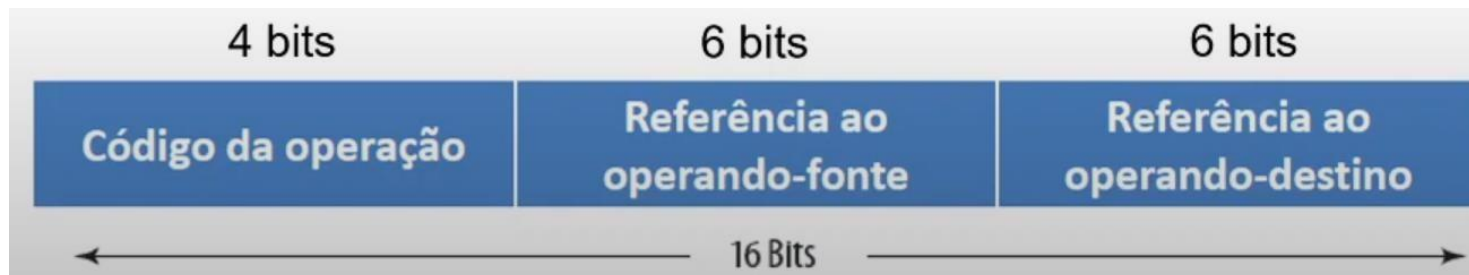
Representação e Elementos da Instrução

00000000
00000001
00000003
00000007
00000008
0000000C
0000000F
00000011
00000014
00000016
00000019
0000001B
0000001D
0000001F
00000022
00000025

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
add     al, cl
shr     al, 5
movzx   eax, al
retn
```


Elementos de uma instrução

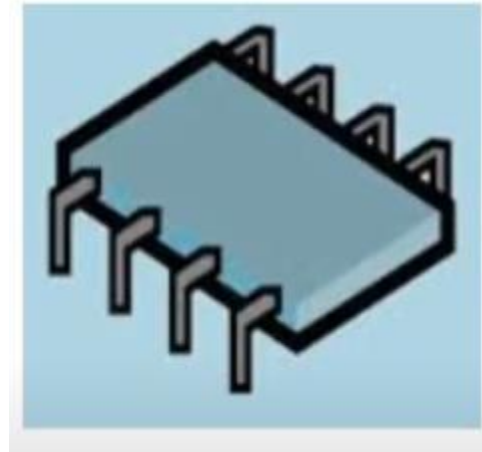
- ❑ Cada instrução usualmente é composta por:
 - Código da operação: especifica a operação a ser efetuada.
 - **Referencia ao operando-fonte:** os dados envolvidos na operação devem ser referenciados.
 - **Referencia ao operando-destino:** informações que possibilitam o armazenamento do resultado gerado pela instrução



Representação de uma instrução

- ❑ Dificuldade do programador lidar com representações binárias de instruções de máquina;
- ❑ Representação simbólica: os códigos de operação são representados por abreviações, chamadas de **mnemônicos**.
 - **ADD** (Adição)
 - **SUB** (subtração)
 - **MPY** (multiplicação)
- ❑ Os operandos também são representados de maneira simbólica
 - ADD **AX,BX**

Número de
Endereços da
instrução



Utilização de Endereços em uma instrução

- ❑ As instruções podem contar com um número variável de endereços ou parâmetros, conforme cada assembly.

Número de endereços	Representação simbólica	Interpretação
3	OP A,B,C	A <- B OP C
2	OP A,C	A <- A OP B
1	OP A	AC <- AC OP A
0	OP	T <- (T-1) OP T

AC = acumulador

OP = operador

T = topo da pilha

(T-1) = segundo elemento da pilha

A,B,C = locais de memória ou registradores

Instrução: número de endereços → 3

❑ 3 endereços:

- Operando 1, Operando 2, Resultado.
- Exemplo: **ADD A,B,C** → $A = B + C$
- Precisa de palavras muito longas para operacionalizar.
- Não é comum.

Instrução: número de endereços → 2

❑ 2 endereços:

- Operando 1 (que também é resultado), Operando 2
- Exemplo: **ADD A,B** → $A = A + B$
- Um endereço é usado como operando e resultado.
- Reduz o tamanho da instrução.
- Requer algum trabalho extra.
- Armazenamento temporário para manter alguns resultados.

Instrução: número de endereços → 1

❑ 1 endereço:

- Operando 1, Acumulador ACC implícito
- Exemplo: **ADD A** → $ACC = ACC + A$
- Segundo endereço é implícito (não aparece na instrução).
- Normalmente o segundo endereço é um acumulador (ACC)
- Comum nas primeiras máquinas.

Instrução: número de endereços → 0

❑ 0 endereços:

- Sem operandos na instrução, todos implícitos.
- Utiliza uma pilha.
- Exemplo $C = A + B$

↓
Push A
Push B
Add
Pop C

Instrução: Número endereços (Exemplo)

Instruções com três endereços:

$$Y = \frac{A - B}{C + (D \times E)}$$

<u>Instrução</u>	<u>Comentário</u>
SUB Y, A, B	Y <- A - B
MPY T, D, E	T <- D x E
ADD T, T, C	T <- T + C
DIV Y, Y, T	Y <- Y / T

Instrução: Número endereços (Exemplo)

Instruções com dois endereços:

$$Y = \frac{A - B}{C + (D \times E)}$$

Instrução	Comentário
MOV Y, A	Y <- A
SUB Y, B	Y <- Y - B
MOVE T, D	T <- D
MPY T, E	T <- T x E
ADD T, C	T <- T + C
DIV Y, T	Y <- Y / T

Instrução: Número endereços (Exemplo)

Instruções com um endereços:

$$Y = \frac{A - B}{C + (D \times E)}$$

<u>Instrução</u>	<u>Comentário</u>
LOAD D	AC <- D
MPY E	AC <- AC x E
ADD C	AC <- AC + C
STOR T	T <- AC
LOAD A	AC <- A
SUB B	AC <- AC - B
DIV Y	AC <- AC / T
STOR Y	Y <- AC

Instrução: Número endereços (comparativo)

❑ **Mais endereços:**

- Instruções mais complexas.
- Mais registradores.
- Operações entre registradores são mais rápidas.
- Menos instruções por programa.

❑ **Menos endereços:**

- Instruções menos complexas.
- Mais instruções por programa.
- Busca execução de instruções mais rápidas.

Tipos de Operações e Operandos



Tipos de Operandos

- ❑ Endereços de memória
- ❑ Números:
 - Inteiros
 - Ponto flutuante (float)
- ❑ Caracteres:
 - ASCII
 - Strings
- ❑ Dados lógicos
 - Bits
 - Flags

Tipos de Operações

- ❑ Transferência de dados
- ❑ Aritmética
- ❑ Logica
- ❑ Conversão
- ❑ Entrada / Saída (E/S ou I/O)
- ❑ Controle do sistema
- ❑ Transferência de controle

Operações (tipos) -> Transferência de Dados

Nome da operação	Descrição
Move (transferência)	Transfere palavra ou bloco da origem ao destino
Store (armazenar)	Transfere palavra do processador para a memória
Load (Carregar)	Transfere palavra da memória para o processador
Exchange	Troca conteúdo da origem e do destino
Clear (reset)	Transfere palavra de 0s para o destino
Set	Transfere palavra de 1s para destino
Push	Transfere palavra da origem para o topo da pilha
Pop	Transfere palavra do topo da pilha para o destino

Operações (tipos) -> Transferência de Dados

☐ **Especificam:**

- Origem.
- Destino.
- Quantidade de dados.

☐ **Pode ser instruções diferentes para diferentes movimentações:**

☐ **Pode ser uma instrução e diferentes endereços**

Operações (tipos) -> Aritmética

Nome da operação	Descrição
ADD	Calcula dois operandos
Subtract	Calcula a diferença de dois operando
Multiply	Calcula o produto de dois operandos
Divide	Calcula o quociente de dois operandos
Absolute	Substitui o operando pelo valor absoluto
Negate	Troca o sinal do operando
Incremente	Soma 1 ao operando
Decrement	Subtrai 1 do operando

Operações (tipos) -> Aritmética

☐ **Adição, Subtração, Multiplicação, Divisão:**

☐ **Inteiro com sinal.**

☐ **Ponto flutuante**

☐ **Pode incluir**

- Incremento (a++).
- Decremento (a--).
- Negação (-a)

Operações (tipos) -> Lógica

Nome da operação	Descrição
AND	Realiza um AND lógico
OR	Realiza um OR lógico
Not (complemento)	Realiza um NOT lógico
Esclusive OR	Realiza um XOR lógico
Test	Testa condição especificada; define flag(s) com base no resultado
Compare	Faz a comparação Lógica ou aritmética de dois ou mais operandos; define flag(s) com base no resultado
Set control variables	Classe de Instruções para definir controles para fins de proteção, tratamento de interrupção, controle de tempo etc.
Shift	Desloca o operando para a esquerda introduzindo constantes em sua extremidade.
Rotate	Desloca ciclicamente o operando para direita de uma nova extremidade à outra

Operações (tipos) -> Lógica

- ❑ Operam sobre dados booleanos ou binários:
- ❑ Note: inverte um bit.
- ❑ AND, OR, Exclusive-OR (XOR): mais comuns.
- ❑ EQUAL: teste binário útil
- ❑ Deslocamento lógicos
 - Os bits de uma palavra são deslocados para direita ou esquerda
 - O bit deslocado para fora se perde
 - Na outra extremidade, um 0 é deslocado para dentro.

Operações (tipos) -> conversão

❑ **Mudam o formato dos dados**

▪ **Exemplos (Assembly Motorola 6800):**

CVTBW – Convert byte to Word; sign extend;

CVTBL - Convert byte to Long; sign extend;

CVTWB - Convert Word to bit; truncated;

CVTWL - Convert Word to Long; sign extend;

CVTLB - Convert Long to byte; truncated;

CVTLW – Convert Long to Word ; truncated;

CVTBF - Convert byte to Floating; exact;

CVTBD - Convert byte to Double float; exact;

CVTWF - Convert Word to Floating; exact;

CVTWD - Convert Word to Double float; exact;

Etc.

Operações (tipos) -> Entrada / Saída (E/S)

- ❑ Podem ser instruções específicas
- ❑ Pode ser feita usando instruções de movimentação de dados (mapeadas na memória)
- ❑ Pode ser feita por um controlador separado (Direct Memory Access – DMA)

Exemplos (Assembly 8088):

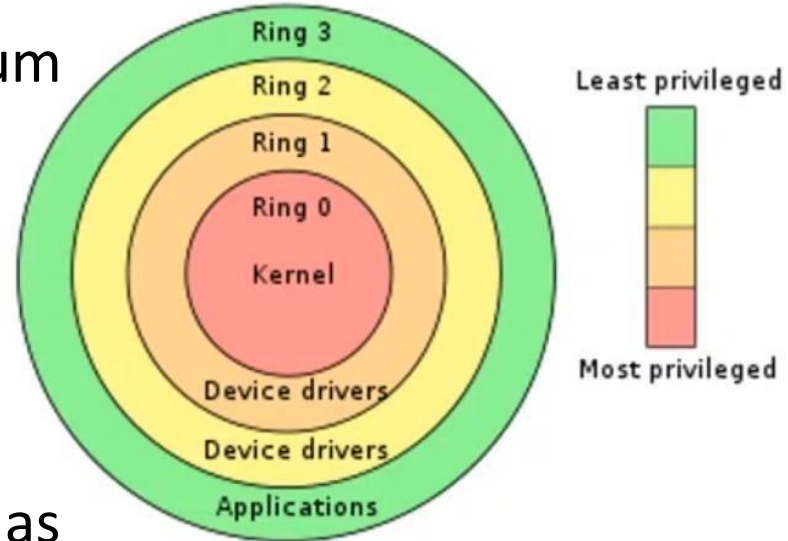
Formato	Inserindo dados	Saida de dados
(1)	IN fonte, destino IN AL, port#	OUT fonte, destino OUT port#, AL
(2)	MOV fonte, destino MOV port#, DX	MOV destino, fonte MOV DX, port#

Operações (tipos) -> controle do sistema

❑ Instruções privilegiadas

❑ CPU precisa estar em um estado específico:

- Anel 0 no 80386+.
- Modo Kernel.



❑ Para uso dos sistemas Operacionais

- EXEMPLOS (Assembly Motorola 6800):
- **STOP** -> load na operand and stops the processor
- **LPSTOP** -> Low power stop
- **RESET** -> reset external devices
- **RTE** -> return from exception

Operações (tipos) -> transferência de controle

❑ Salto (incondicional)

- Ex: JMP LABEL1 (pular para LABEL 1)

❑ Desvio (condicional)

- Ex: JZ LABEL1 (pule o operador de 0 para LABEL2)
- Ex2: JNZ LABEL2 (pule se o operador não for zero para LABEL 3)

Operações (tipos) -> transferência de controle

❑ CMP (COMPARE) + Instrução condicional

- JE (pular se for igual)
- JG (pular se for maior)
- JGE (pular maior/igual)
- JL (pular se for menor)
- JNG (pular se não for maior)
- JNB (pular se não estiver abaixo)
- JNA (pular se não estiver abaixo)
- JC (pular se teve carry)
- JO (pular se teve overflow)
- JNO (pular se não teve overflow)
- JS (pular se o sinal for negativo)
- JNS (pular se o sinal for positivo)

CMP	AL, BL
JE	EQUAL
CMP	AL, BH
JE	EQUAL
CMP	AL, CL
JE	EQUAL
NON_EQUAL:	
EQUAL: . . .	

Conclusão

Na aula de hoje foram abordados os principais conceitos sobre:

- ☐ **Instruções e Linguagem de máquina;**
- ☐ **Representação e elementos da instrução;**
- ☐ **Número de Operadores da instrução;**
- ☐ **Tipos de Operação.**