

Conjunto de Instruções 8086

Prof. Sérgio L. Cechin

Grupos de Instruções

- Data Transfer (transferências)
- Arithmetic (aritméticas)
- String manipulation
- Control Transfer (desvios)
- Processos Control
- Manipulação de bits
- Instruções sobre FLAGS

Legenda

- Registradores
 - Dados de 16 bits (**reg16**): AX, CX, DX, BX, SP, BP, SI, DI
 - Dados de 8 bits (**reg8**): AL, CL, DL, BL, AH, CH, DH, BH
 - Segmento (**seg**): ES, CS, SS, DS
 - reg = reg8 e reg16
- Imediato
 - Imediato de 8 bits (**im8**)
 - Imediato de 16 bits (**im16**)
 - im = im8 e im16
- Memória (Effective Address)
 - Memória de 8 bits (**mem8**)
 - Memória de 16 bits (**mem16**)
 - mem = mem8 e mem16

Data Transfer

- Transferência de dados
- Transferência de flags
- Transferência sobre endereços
- Tradução
- Entrada e Saída

Transferência de dados

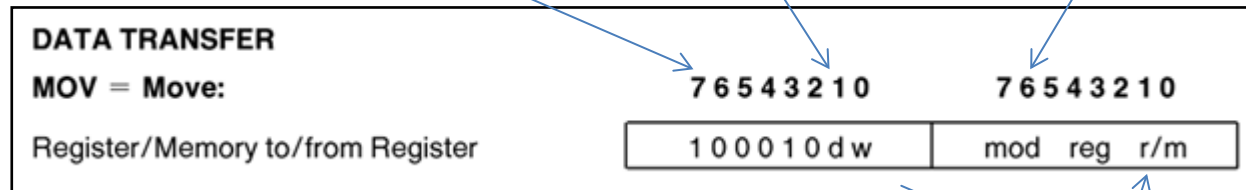
- MOV destino, origem
 - Move origem para destino
 - $\text{reg} \leftrightarrow \text{reg/mem}$
 - $\text{reg/mem} \leftarrow \text{im}$
 - $\text{reg/mem} \leftrightarrow \text{seg}$
- XCHG op1, op2
 - Troca o valor dos dois operandos
 - $\text{reg} \leftrightarrow \text{reg/Mem}$

Exemplo: MOV

w==1 → operando com 16 bits
w==0 → operando com 8 bits

d==1 → “reg” é o destino
d==0 → “reg” é a origem

mod==00 → sem DISP
mod==01 → DISP com 8 bits e sinal
mod==10 → DISP com 16 bits e sinal
mod==11 → r/m representa um reg



r/m → reg16

r/m = 000 → AX
r/m = 001 → CX
r/m = 010 → DX
r/m = 011 → BX
r/m = 100 → SP
r/m = 101 → BP
r/m = 110 → SI
r/m = 111 → DI

r/m → reg8

r/m = 000 → AL
r/m = 001 → CL
r/m = 010 → DL
r/m = 011 → BL
r/m = 100 → AH
r/m = 101 → CH
r/m = 110 → DH
r/m = 111 → BH

r/m → mem (EA)

r/m = 000 → EA = (BX) + (SI) + DISP
r/m = 001 → EA = (BX) + (DI) + DISP
r/m = 010 → EA = (BP) + (SI) + DISP
r/m = 011 → EA = (BP) + (DI) + DISP
r/m = 100 → EA = (SI) + DISP
r/m = 101 → EA = (DI) + DISP
r/m = 110 → EA = (BP) + DISP (mod==00)
r/m = 110 → EA = DISP (mod!=00)
r/m = 111 → EA = (BX) + DISP

Codificar as instruções

- MOV AL, 64
 - 8 bits, read displacement (direto ou absoluto)
- MOV 64, AL
 - 8 bits, write displacement (direto ou absoluto)
- MOV AX, 64
 - 16 bits, read displacement (direto ou absoluto)
- MOV AX, [BP+8]
 - 16 bits, read [BP+8] (displacement com 8 bits)
- MOV AX, [BP+512]
 - 16 bits, read [BP+8] (displacement com 16 bits)

Solução

- MOV AL, 64
 - 8 bits, read displacement (direto ou absoluto)
 - 8A 06 64 00
- MOV 64, AL
 - 8 bits, write displacement (direto ou absoluto)
 - 88 06 64 00
- MOV AX, 64
 - 16 bits, read displacement (direto ou absoluto)
 - 8B 06 64 00
- MOV AX, [BP+8]
 - 16 bits, read [BP+8] (displacement com 8 bits)
 - 8B 46 08
- MOV AX, [BP+512]
 - 16 bits, read [BP+8] (displacement com 16 bits)
 - 8B 86 00 02

Exemplo: MOV (completa)

DATA TRANSFER

MOV = Move:

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/Memory to/from Register	1 0 0 0 1 0 d w	mod reg r/m		
Immediate to Register/Memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1
Immediate to Register	1 0 1 1 w reg	data	data if w = 1	
Memory to Accumulator	1 0 1 0 0 0 0 w	addr-low	addr-high	
Accumulator to Memory	1 0 1 0 0 0 1 w	addr-low	addr-high	
Register/Memory to Segment Register	1 0 0 0 1 1 1 0	mod 0 reg r/m		
Segment Register to Register/Memory	1 0 0 0 1 1 0 0	mod 0 reg r/m		

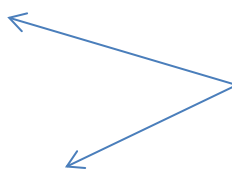
Vamos codificar à mão???

Transferência de dados

- PUSH origem
 - Coloca origem na pilha
 - Decrementa o SP e escreve origem no endereço SS:SP
 - reg16/mem16/seg
- POP destino
 - Retira topo da pilha para destino
 - Lê do endereço SS:SP e incrementa o SP
 - reg16/mem16/seg

Problema do MOV mem,im

- MOV mem8, BYTE im8
- MOV r/m16, WORD im16
- MOV r/m16, seg
- MOV seg, r/m16



Observar os designadores
BYTE e WORD

Não existe
MOV seg,im16

Transferência de Flags

- PUSHF
 - Coloca registrador de FLAGS na pilha
- POPF
 - Retira registrador de FLAGS da pilha
- LAHF
 - Carrega AH com FLAGS
 - 8 bits menos significativos do registrador FLAGS
- SAHF
 - Carrega FLAGS com AH
 - 8 bits menos significativos do registrador FLAGS

Transferência sobre endereços

- Carrega registradores com um endereço
 - EA: *Effective Address*
 - SEG: *Segment*
- LEA reg16,mem
 - Carrega EA de “mem” no reg16
 - Não carrega registrador de segmento
- LDS reg16,mem
 - Carrega EA de “mem” no reg16
 - Carrega SEG em DS
- LES reg16,mem
 - Carrega EA de “mem” no reg16
 - Carrega SEG em ES

Tradução

- XLAT
 - Converte AL (*translate byte*)
 - $AL \leftarrow [BX+AL]$.

Entrada e Saída

- Leitura / escrita em periférico
 - Periférico → espaço de endereçamento de I/O
 - Separado da memória
 - Porta → 16 bits (0000H até FFFFH)
 - Capacidade para 65536 endereços

Leitura de I/O

- IN AL, porta
 - Leitura de um byte da porta
 - IN AL, im16
 - IN AL, DX
- IN AX, porta
 - Leitura de uma word da porta
 - Na realidade, são realizadas duas leituras
 - Na “porta” e na “porta+1”
 - IN AX, im16
 - IN AX, DX

Escrita em I/O

- OUT porta, AL
 - Escrita de um byte na porta
 - OUT im16,AL
 - OUT DX, AL
- OUT porta,AX
 - Escrita de uma word na porta
 - Na realidade, são realizadas duas escritas
 - Na “porta” e na “porta+1”
 - OUT im16, AX
 - OUT DX, AX

Instruções Aritméticas

- Com dois operandos
- Com um operando
- Multiplicação e divisão

Com dois operandos

- Operações
 - Soma, diferença e comparação
 - FLAGS são alterados conforme o resultado da operação
- Registradores
 - Registradores de segmento não podem ser operados
- Largura da operação (8 ou 16 bits)
 - Os dois operandos devem ter a mesma largura
 - O registrador envolvido (origem ou destino) determina a largura
 - No caso de memória ou imediato
 - É necessário declarar explicitamente a largura

Somas

- ADD destino, fonte
 - $\text{destino} \leftarrow \text{destino} + \text{fonte}$
- ADC destino, fonte
 - $\text{destino} \leftarrow \text{destino} + \text{fonte} + \text{carry}$

Diferenças

- SUB destino, fonte
 - $\text{destino} \leftarrow \text{destino} - \text{fonte}$
- SBB destino, fonte
 - $\text{destino} \leftarrow \text{destino} - \text{fonte} - \text{borrow}$
- CMP destino, fonte
 - $\text{destino} - \text{fonte}$
 - Apenas altera os FLAGS

Com um operando

- INC destino
 - Incrementa de 1
- DEC destino
 - Decrementa de 1
- NEG destino
 - Troca sinal (complemento de dois)

Multiplicação e Divisão

- Multiplicação e divisão de inteiros
- Operandos com e sem sinal
- Multiplicação com 8 ou 16 bits
 - $16 \text{ bits} = 8 \text{ bits} * 8 \text{ bits}$
 - $32 \text{ bits} = 16 \text{ bits} * 16 \text{ bits}$
- Divisão
 - Com 8 bits
 - $8 \text{ bits} = 16 \text{ bits} / 8 \text{ bits}$
 - $8 \text{ bits} = \text{resto} (16 \text{ bits} / 8 \text{ bits})$
 - Com 16 bits
 - $16 \text{ bits} = 32 \text{ bits} / 16 \text{ bits}$
 - $16 \text{ bits} = \text{resto} (32 \text{ bits} / 16 \text{ bits})$

Multiplicação

- $AX \leftarrow AL * \text{fonte(byte)}$
- $DX:AX \leftarrow AX * \text{fonte(word)}$
- Sem sinal: **MUL fonte**
- Com sinal: **IMUL fonte**

Divisão

- 8 bits
 - $AL \leftarrow AX / \text{fonte}$
 - $AH \leftarrow \text{Resto}(AX / \text{fonte})$
- 16 bits
 - $AX \leftarrow DX:AX / \text{fonte}$
 - $DX \leftarrow \text{Resto}(DX:AX / \text{fonte})$
- Sem sinal: **DIV fonte**
- Com sinal: **IDIV fonte**

Manipulação de bits

- Instruções lógicas
- Deslocamento

Instruções lógicas

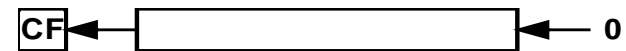
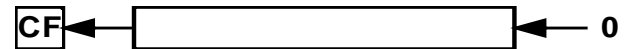
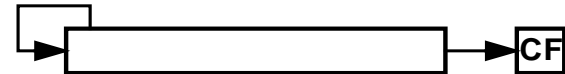
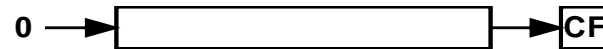
- Altera os FLAGS em função do resultado
- NOT destino
 - Inverte todos os bits de “destino”
- AND destino, fonte
 - $\text{destino} \leftarrow \text{destino} \& \text{fonte}$
- OR destino, fonte
 - $\text{destino} \leftarrow \text{destino} | \text{fonte}$
- XOR destino, fonte
 - $\text{destino} \leftarrow \text{destino} \oplus \text{fonte}$
- TEST destino, fonte
 - AND, sem armazenar resultado

Deslocamentos

- Operação destino , contador
- Operação
 - Shifts, rotates
 - Left, right
 - Lógicos, aritméticos
- Destino
 - Registradores e endereços de memória
- Contador
 - Número de bits a serem deslocados

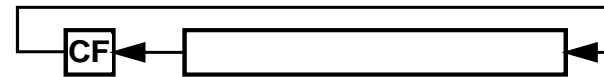
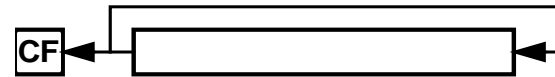
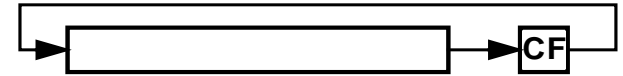
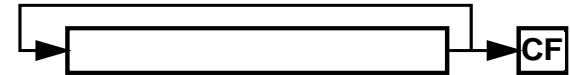
Shifts

- Shift Right
 - Lógico
 - SHR destino
 - Aritmético
 - SAR destino
- Shift Left
 - Lógico
 - SHL destino
 - Aritmético
 - SAL destino



Rotates

- Rotate Right
 - Sem carry
 - ROR destino
 - Com carry
 - RCR destino
- Rotate Left
 - Sem carry
 - ROL destino
 - Com carry
 - RCL destino



Operandos

- Operação
 - Shifts: SHR, SAR, SHL, SAL
 - Rotates: ROR, RCR, ROL, RCL
- Operandos
 - OPERAÇÃO reg/mem , 1
 - OPERAÇÃO reg/mem , CL

Instruções de desvio

- Incondicional
- Controle de laço
- Subrotinas
- Interrupções de Software
- Desvio condicional

Tipos de Endereços de Destino

- Desvios dentro do segmento de código corrente
 - Não altera CS
 - *short jump (relativo)*
 - $IP \leftarrow IP + offset$ (1 byte)
 - *offset*: valor em complemento de 2 (-128 até +127)
 - *near jump (absoluto e relativo)*
 - $IP \leftarrow offset$ (2 bytes)
 - $IP \leftarrow IP + offset$ (2 bytes)
- Desvios para fora do segmento de código corrente (altera o CS)
 - *far jump (absoluto)*
 - $CS:IP \leftarrow segment : offset$ (4 bytes)

Desvio Incondicional

- A nomenclatura da Intel mudou ao longo do tempo
 - No caso dos desvios, “direto” e “absoluto” são a mesma coisa
- Imediato
 - JMP SHORT im8
 - Direct within Segment-Short (short, relative)
 - JMP im16
 - Direct within Segment (near, relative)
 - JMP im16:im16
 - Direct Intersegment (far, absolute)
- Indireto
 - JMP reg16/mem16
 - Indirect within Segment (near, absolute)
 - JMP FAR mem16
 - Indirect Intersegment (far, absolute)

Controle de Laço

- LOOP
 - Decrementa CX
 - Caso CX==0, encerra o loop
 - Senão, executa um JMP SHORT
- LOOPE ou LOOPZ
 - Decrementa CX
 - Caso CX==0 || ZF==1, encerra o loop
 - Senão, executa um JMP SHORT
- LOOPNE ou LOOPNZ
 - Decrementa CX
 - Caso CX==0 || ZF==0, encerra o loop
 - Senão, executa um JMP SHORT
- JCXZ
 - Caso CX==0, executa um JMP SHORT

Controle de Laço

- LOOP
 - LOOP im8
- LOOPcc
 - LOOPcc im8
 - cc = E, Z, NE ou NZ
- JCXZ im8

Subrotinas

- Chamada e retorno de subrotina
 - CALL: chamada de subrotina
 - Salva endereço de retorno na pilha
 - JMP para endereço da subrotina
 - RET: retorno da subrotina
 - JMP para endereço de retorno
- Tipos de subrotinas
 - Chamada “intra” segmento (NEAR)
 - Salva IP na pilha (requer um retorno NEAR)
 - Chamada “inter” segmentos (FAR)
 - Salva CS:IP na pilha (requer um retorno FAR)

Chamada das Subrotinas

- Direct
 - NEAR (within Segment)
 - CALL im16
 - FAR (Intersegment)
 - CALL im16:im16
- Indirect
 - NEAR (within Segment)
 - CALL reg16/mem16
 - FAR (Intersegment)
 - CALL FAR mem16

Retorno das Subrotinas

- Existem dois opcodes:
 - Retorno FAR
 - Retorno NEAR
- Por exemplo, no montador NASM são usados mnemônicos diferentes: RETN e RETF
- No montador MASM, é usado o mesmo mnemônico: RET
 - Como diferenciar um do outro?
 - Solução: A rotina toda é declarada FAR ou NEAR
- Formato
 - RET: executa JMP para endereço que está na pilha (CS:IP ou IP)
 - RET imm16: executa o mesmo que RET e depois retira (POP) imm16 bytes da pilha

Exemplo

- Rotina NEAR

ROT_N PROC NEAR

.....

RET

ROT_N ENDP

- Rotina FAR

ROT_F PROC FAR

.....

RET

ROT_F ENDP

Interrupções de Software

- Usado para chamar interrupções de software
 - Por decisão do programador e/ou
 - Condicionadas e eventos no processador
- Em geral, são usadas para solicitar serviços do sistema operacional
 - Ex: para acesso aos periféricos (teclado, tela, disco, etc)

Interrupções de Software

- INT imm8
 - Imm8 indica o tipo da interrupção
 - O sistema operacional define o serviço
- INTO
 - Chama a interrupção se OF==1 (overflow)
- IRET
 - Retorno de interrupção

Desvio Condicional

- Desvios SHORT (-128 até +127)
 - A partir do 80386 foi incluído o NEAR relativo
- Existem três grupos
 - Comparações com sinal
 - Comparações sem sinal
 - Verificação dos FLAGS

Desvio Condicional (com sinal)

- Greater / Not Less nor Equal $((SF \text{ XOR } OF) \text{ OR } ZF) = 0$
 - **JG** **rel8**
 - **JNLE** **rel8**
- Greater or Equal / Not Less $(SF \text{ XOR } OF) = 0$
 - **JGE** **rel8**
 - **JNL** **rel8**
- Less / Not Greater nor Equal $(SF \text{ XOR } OF) = 1$
 - **JL** **rel8**
 - **JNGE** **rel8**
- Less or Equal / Not Greater $((SF \text{ XOR } OF) \text{ OR } ZF) = 1$
 - **JLE** **rel8**
 - **JNG** **rel8**

Desvio Condicional (sem sinal)

- Above / Not Below nor Equal (CF OR ZF) = 0
 - **JA** **rel8**
 - **JNBE** **rel8**
- Above ou Equal / Not Below (CF = 0)
 - **JAE** **rel8**
 - **JNB** **rel8**
- Below / Not Above nor Equal (CF = 1)
 - **JB** **rel8**
 - **JNAE** **rel8**
- Below ou Equal / Not Above (CF OR ZF) = 1
 - **JBE** **rel8**
 - **JNA** **rel8**

Desvio Condicional (FLAGS)

- Carry (CF)
 - **JC** rel8 (CF = 1)
 - **JNC** rel8 (CF = 0)
- Zero (ZF)
 - **JE/JZ** rel8 (ZF = 1)
 - **JNE/JNZ** rel8 (ZF = 0)
- Paridade par (PF)
 - **JP/JPE** rel8 (PF = 1)
 - **JNP/JPO** rel8 (PF = 0)
- Overflow (OF)
 - **JO** rel8 (OF = 1)
 - **JNO** rel8 (OF = 0)
- Sinal (SF)
 - **JS** rel8 (SF = 1)
 - **JNS** rel8 (SF = 0)

Manipulação de Strings

- Utilização implícita dos seguintes registros:
 - DS:SI String origem
 - ES:DI String destino
 - CX Contador
 - AL/AX Valor de trabalho
- FLAGS
 - DF (Direction Flag)
 - DF=0 auto incremento para SI e DI
 - DF=1 auto decremento para SI e DI
 - ZF
 - Condição de término para busca e comparação
- Tipos de elementos dos strings
 - “B” (byte); “W” (word)

Manipulação de Strings

- Após cada instrução, incrementa ou decrementa SI ou DI, dependendo do flag DF
- MOVSS (MOVSB / MOVSW)
 - Move um elemento (byte ou word)
 - $[ES:DI] \leftarrow [DS:SI]$
- CMPSs (CMPSB / CMPSW)
 - Compara dois elementos (byte ou word)
 - $[DS:SI] - [ES:DI]$

Manipulação de Strings

- SCASs (SCASB / SCASW)
 - Procura pelo valor AL (AX) no string
 - AL – [ES:DI] ou AX – [ES:DI]
- LODSs (LODB / LODW)
 - Carrega elemento do string em AL / AX
 - AL \leftarrow [DS:SI] ou AX \leftarrow [DS:SI]
- STOSs (STOB / STOW)
 - Armazena AL / AX no string
 - [ES:DI] \leftarrow AL ou [ES:DI] \leftarrow AX

Repetição das instruções

- Pode-se forçar a repetição das instruções de string
- Usa-se um “prefixo” nas instruções
- A cada instrução executada, decrementa-se CX
- Formas
 - REP → repeat while (CX!=0)
 - REPE / REPZ → repeat while (CX!=0 && Z=1)
 - REPNE / REPNZ → repeat while (CX!=0 && Z=0)
- Exemplo:
 - REP MOVSB
 - Copia CX caracteres do string iniciado por [DS:SI] para o string iniciado por [ES:DI]

Instruções sobre FLAGS

- STC set carry flag ($CF \leftarrow 1$)
- CLC clear carry flag ($CF \leftarrow 0$)
- CMC complement carry flag ($CF \leftarrow \neg CF$)
- STD set direction flag ($DF \leftarrow 1$)
- CLD clear direction flag ($DF \leftarrow 0$)
- STI set interrupt-enable flag ($IEF \leftarrow 1$)
- CLI clear interrupt-enable flag ($IEF \leftarrow 0$)

Conjunto de Instruções 8086

Prof. Sérgio L. Cechin