Microprocessadores Intel

conjunto de instruções
(ou: que pena que não é RISC!)

Conjunto de instruções

- aumenta a cada novo modelo da família
- várias possibilidades de codificação (em código de máquina) para a mesma instrução
- código da instrução de tamanho variado
  - um a vários bytes
- instruções não ortogonais aos modos de endereçamento
  - modos auto-incrementados só aparecem em instruções de manipulação de strings
  - instruções geralmente permitem um único operando na memória exceto em instruções de manipulação de strings
- algumas instruções usam alguns registradores pré-definidos
- formado geral para dois operando: destino, fonte

primeiro destino e depois o fonte

Codificação de Instruções (ex: MOV)

rm: reg ou mem

r: reg

sreg: reg de segmento

m: mem

im: imediato

Tipo	Código (hexa)	Instrução	Descrição
1	88 reg rm	MOV rm8, r8	Move de registradores de 8 bits
2	89 reg rm	MOV rm16, r16	Move de registradores de 16 bits
3	89 reg rm	MOV rm32, r32	Move de registradores de 32 bits
4	8A reg rm	MOV r8, rm8	Move para registradores de 8 bits
5	8B reg rm	MOV r16, rm16	Move para registradores de 16 bits
6	8B reg rm	MOV r32, rm32	Move para registradores de 32 bits
7	8C reg rm	MOV rm16, sreg	Move de registrador de segmento
8	8E reg rm	MOV sreg, rm16	Move para registrador de segmento
9	A0	MOV AL, m8	Move byte para AL
10	A1	MOV AX, m16	Move palavra (16 bits) para AX
11	A1	MOV EAX, m32	Move palavra dupla (32 bits) para EAX
12	A2	MOV m8, AL	Move AL para memória (byte)
13	A3	MOV m16, AX	Move AX para memória (palavra)
14	A3	MOV m32, EAX	Move EAX para memória (palavra dupla)
15	B0 reg imed	MOV r8, im8	Move dado imediato (byte) para reg. 8 bits
16	B8 reg imed	MOV r16, im16	Move dado imediato para reg. 16 bits
17	B8 reg imed	MOV r32, im32	Move dado imediato para reg. 32 bits
18	C6 rm imed	MOV rm8, im8	Move dado imediato para reg/mem (8 bits)
19	C7 rm imed	MOV rm16, im16	Move dado imediato para reg/mem (16 bits)
20	C7 rm imed	MOV rm32, im32	Move dado imediato para reg/mem (32 bits)

Instruções de transferência de dados

MOV	destino, fonte	move fonte para destino (B,W,D)
PUSH	fonte	coloca fonte na pilha (W,D)
POP	destino	retira da pilha para destino (W,D)
XCHG	op1, op2	troca (exchange) operandos
BSWAP	reg32	inverte a ordem dos 4 bytes

restrições:

- apenas MOV, PUSH e POP podem acessar registradores de segmento
- não é possível mover dado imediato para um reg. de segmento
- CS não pode ser usado como destino (nem IP)
- operandos de PUSH e POP devem ser de 16 ou 32 bits
- MOV e XCHG não aceitam dois operandos na memória

Instruções de transferência de dados

MOV	reg, reg	mesmo tamanho (8, 16 ou 32 bits)
MOV	mem, reg	mesmo tamanho (8, 16 ou 32 bits)
MOV	reg, mem	mesmo tamanho (8, 16 ou 32 bits)
MOV	reg, imed	mesmo tamanho (8, 16 ou 32 bits)
MOV	mem, imed	mesmo tamanho (8, 16 ou 32 bits)
MOV	reg, segreg	operandos de 16 bits
MOV	segreg, reg (exceto CS)	operandos de 16 bits
MOV	segreg, mem (exceto CS)	operandos de 16 bits
MOV	mem, segreg	operandos de 16 bits
XCHG	reg, reg	mesmo tamanho (8, 16 ou 32 bits)
XCHG	reg, mem	mesmo tamanho (8, 16 ou 32 bits)
XCHG	mem, reg	mesmo tamanho (8, 16 ou 32 bits)

Atenção: primeiro destino e depois o fonte

Instruções de transferência de dados

PUSH	r16/r32	Empilha um registrador de 16 ou 32 bits
PUSH	m16/m32	Empilha um operando em memória de 16 ou 32 bits
PUSH	im16/im32	Empilha um dado imediato de 16 ou 32 bits
PUSH	segreg	Empilha um registrador de segmento (16 bits)
PUSHA		Empilha AX, CX, DX, BX, SP (original), BP, SI e DI
PUSHAD		Empilha EAX,ECX,EDX,EBX,ESP(original),EBP,ESI,EDI

POP	r16/r32	Desempilha um registrador de 16 ou 32 bits
POP	m16/m32	Desempilha um operando de 16 ou 32 bits
POP	segreg	Desempilha um registrador de segmento (exceto CS)
POPA		Desempilha DI, SI, BP, SP, BX, DX, CX e AX
POPAD		Desempilha EDI,ESI,EBP,ESP,EBX,EDX,ECX e EAX

Instruções de transferência de flags

PUSHF/PUSHFD	coloca registrador de flags na pilha (16 bits para F ou 32 bits para EF)
POPF/POPFD	retira registrador de flags da pilha (16 bits para F e 32 bits para EF)
LAHF	carrega AH com flags (8 bits menos significativos do registrador F)
SAHF	carrega flags com AH (8 bits menos significativos do registrador F)

7

Instruções de transferência sobre endereços

LEA r16/r32, mem	carrega endereço efetivo do operando mem para reg (16 ou 32 bits)
LDS r16/32, mem	carrega endereço de mem para reg (16 ou 32 bits) e DS (16 bits)
LES r16/32, mem	carrega endereço de mem para reg (16 ou 32 bits) e ES (16 bits)
LFS r16/32, mem	carrega endereço de mem para reg (16 ou 32 bits) e FS (16 bits)
LGS r16/32, mem	carrega endereço de mem para reg (16 ou 32 bits) e GS (16 bits)

8

Instrução de tradução

XLAT	converte AL (translate byte)

- Substitui um byte em AL por um byte de uma tabela, indicada em BX
- Conteúdo de AL fornece o deslocamento na tabela
- Instrução substitui o conteúdo de AL pelo byte em [BX+AL]

9

Instruções de entrada e saída

destino, fonte

IN acumulador, porta
- leitura de byte, word ou dobleword para AL, AX ou EAX

OUT porta, acumulador
- transfere byte, word ou doubleword de AL, AX ou EAX para a porta

- IN e OUT podem referenciar até 65536 portas de E/S
  - Cada porta pode receber ou fornecer um byte apenas
- Aparecem em dois tipos: direto e indireto
  - No tipo direto, a instrução fornece um endereço de porta de 1 byte (sob a forma de constante imediata, entre 0 e 255)
  - No tipo indireto, o registrador DX fornece o endereço de porta, permitindo assim referenciar até 65536 portas

10

Instruções de entrada e saída

- Formatos possíveis

IN	AL, im8
IN	AL, DX
IN	AX, im8
IN	AX, DX
IN	EAX, im8
IN	EAX, DX

OUT	im8, AL
OUT	DX, AL
OUT	im8, AX
OUT	DX, AX
OUT	im8, EAX
OUT	DX, EAX

no modo protegido, IN e OUT são instruções privilegiadas e só podem ser usadas pelo SO operando com privilégios de kernel

11

Instruções aritméticas sobre dois operandos

ADD destino, fonte
- soma (destino = destino + fonte)

ADC destino, fonte
- soma com carry (destino = destino + fonte + carry)

SUB destino, fonte
- subtrai fonte do destino (destino = destino - fonte)

SBB destino, fonte
- subtrai com borrow (destino = destino - fonte - borrow)

CMP destino, fonte
- compara destino - fonte (sem armazenar o resultado)

combinações:
"reg, reg", "reg, mem", "mem, reg",
"reg, imed" e "mem, imed", em 8, 16 ou 32 bits

12

Instruções aritméticas sobre um operando

INC	destino	incrementa de 1
DEC	destino	decrementa de 1
NEG	destino	troca sinal (complemento de dois)

- "destino"
  - pode ser de 8, 16 ou 32 bits
  - pode ser um registrador ou
  - pode ser uma referência a memória,

Instruções de multiplicação e divisão

MUL fonte
- multiplica como inteiro sem sinal

IMUL fonte
- multiplica como inteiro com sinal

DIV fonte
- divide como inteiro sem sinal

IDIV fonte
- divide como inteiro com sinal

Multiplicação

Três formas possíveis para a multiplicação (até 386):

AL x fonte = AX
- multiplicação de bytes, resultado em uma palavra

AX x fonte = DX:AX
- multiplicação de palavras, resultado em palavra dupla

EAX x fonte = EDX:EAX
- multiplicação de palavras duplas, resultado em 64 bits

- CF e OF
  - são ligados se a metade mais significativa do resultado não for apenas extensão do sinal;
- demais flags indefinidos

Multiplicação - IMUL

formatos após 486

Com um operando	IMUL fonte	convencional (386)
Com dois operandos	IMUL destino, fonte	o operando destino (reg) é multiplicado pelo fonte (reg, op. em mem. ou dado imediato); o resultado é armazenado no reg. destino, truncado para o seu tamanho (16 ou 32 bits); CF e OV indicam se ocorreu estouro
Com três operandos	IMUL destino, fonte, constante	destino (reg.) recebe o produto do operando fonte (reg. ou operando em memória) pela constante (dado imediato); fonte e destino devem ser de 16 ou 32 bits e ter o mesmo comprimento; o resultado é armazenado com truncagem dos bits mais significativos; CF e OV indicam estouro devido a truncagem

Divisão

Três formas possíveis para divisão:

AX / fonte = AL e resto em AH	divisão por byte
DX:AX / fonte = AX e resto em DX	divisão por palavra
EDX:EAX / fonte = EAX e resto em EDX	divisão por palavra dupla

- Todos os flags são indefinidos
- Se o resultado tiver mais bits do que pode ser armazenado no quociente é gerada uma interrupção do tipo 0 (erro de divisão)

Instruções de conversão e ajuste

CBW	converte AL para AX
CWD	converte AX para DX:AX
CWDE	converte AX para EAX
CDQ	converte EAX para EDX:EAX

DAA	decimal adjust after addition (sobre AL)
DAS	decimal adjust after subtraction (sobre AL)
AAA	ASCII adjust after addition (sobre AL)
AAS	ASCII adjust after subtraction (sobre AL)
AAM	ASCII adjust after multiplication (sobre AX)
AAD	ASCII adjust before division (sobre AX)

BCD

Instruções de transferência e ajuste

MOVSX destino, fonte	move fonte para destino com extensão do sinal
MOVZX destino, fonte	move fonte para destino com extensão de zeros

- Instruções acrescentadas a partir do 80386
  - MOVSX r16, rm8
  - MOVSX r32, rm8
  - MOVSX r32, rm16
  - MOVZX r16, rm8
  - MOVZX r32, rm8
  - MOVZX r32, rm16

19

Instruções lógicas convencionais

NOT destino
AND destino, fonte
OR destino, fonte
XOR destino, fonte
TEST destino, fonte

TEST é idêntico a AND, sem armazenar resultado

20

Instruções de rotação

ROR destino, contador rotate right
- msb recebe lsb, lsb vai também para CF

RCR destino, contador rotate with carry right
- msb recebe CF, lsb vai para CF

ROL destino, contador rotate left
- lsb recebe msb, msb vai também para CF

RCL destino, contador rotate with carry left
- lsb recebe CF, msb vai para CF

contador = CL ou valor imediato

21

Instruções de deslocamento

SHR destino, contador shift logical right
- msb recebe 0, lsb vai para CF

SAR destino, contador shift arithmetic right
- msb recebe sinal, lsb vai para CF

SHL destino, contador shift left
- lsb recebe 0, msb vai para CF

SAL destino, contador shift left
- lsb recebe 0, msb vai para CF (como SHL)

22

Instruções de rotação e deslocamento

- Codificações possíveis

operação reg,1
operação reg, CL
operação reg, im8
operação mem,1
operação mem, CL
operação mem, im8

23

Instruções sobre bits de um operando

BSF bit_index, operando Bit Scan Forward
BSR bit_index, operando Bit Scan Reverse
- Se nenhum bit for um, ZF=1

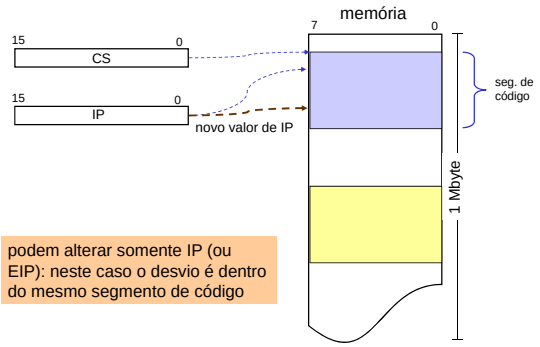
forward – inicia no bit menos significativo

reverse – inicia no bit mais significativo

BT operando, bit_index Bit Test
BTC operando, bit_index Bit Test and Complement
BTR operando, bit_index Bit Test and Reset
BTS operando, bit_index Bit Test and Set
- Copia o bit para CF

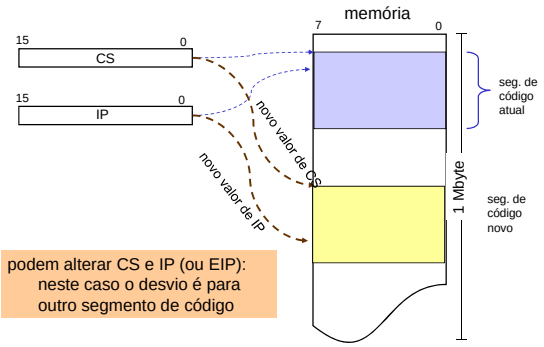
24

Instruções de desvio



podem alterar somente IP (ou EIP): neste caso o desvio é dentro do mesmo segmento de código

Instruções de desvio



podem alterar CS e IP (ou EIP): neste caso o desvio é para outro segmento de código

Instruções de desvio

- Se o endereço fornecido é de **16** bits:
  - o registrador EIP é mascarado pelo valor 0000FFFFH
  - o endereço alvo fica dentro da área de 64 KBytes de um segmento

usada no modo real

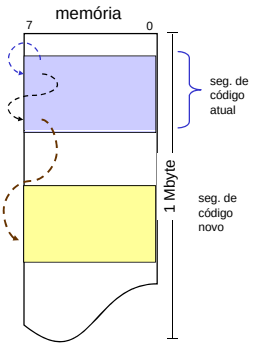
- Se o endereço fornecido é de **32** bits
  - o mascaramento não é realizado endereçando uma área de 4 GBytes.
  - Somente é válido no modo protegido
- Nos modos real e virtual86
  - uma interrupção de erro de endereçamento é gerada se o valor do deslocamento apontar para valores acima de 64 K

Desvios short, near e far

short: deslocamento de 1 byte

near: deslocamento de 2 ou 4 bytes

far: deslocamento para outro segmento



Desvio incondicional direto

- JMP direto
  - curto (short), perto (near) ou longe (far)
- JMP direto short e near
  - um deslocamento contido na instrução é adicionado ao IP (modo de endereçamento relativo para desvios)
  - deslocamento pode ser de 1 byte (short), ou de 2 ou 4 bytes (near)
    - 4 bytes são usados em endereçamento de 32 bits
- JMP direto far
  - CS e IP (ou EIP) são carregados com o endereço especificado

Desvio incondicional indireto

- JMP indireto
  - Pode ser perto (near) ou longe (far)
- JMP indireto near
  - o conteúdo de um registrador ou uma posição de memória é copiado no IP (ou EIP)
- JMP indireto far
  - CS e IP(ou EIP) são carregados com o conteúdo de uma posição de memória
  - posição de memória endereçada como operando, usando os modos de endereçamento)

Controle de laço

- Forma conveniente de codificar laços
  - mas não restrita ao uso de laço
  - Apenas decrementam o contador e desviam se não chegou a zero
    - nenhum flag é afetado devido ao decremento do contador
  - Contador é sempre o registrador CX (ou ECX)
    - não esquecer de inicializar o contador
  - Algumas formas (LOOPE e LOOPNE) testam adicionalmente o flag ZF
  - Todas as instruções permitem apenas deslocamentos curtos (short, ou seja, de +127 a -128)

isso vale para qualquer modelo da família

Controle de laço

LOOP endereço_alvo	loop
LOOPE endereço_alvo	loop while equal (ZF=1)
LOOPZ endereço_alvo	loop while zero (ZF=1)
LOOPNE endereço_alvo	loop while not equal (ZF=0)
LOOPNZ endereço_alvo	loop while not zero (ZF=0)
JCXZ endereço_alvo	jump if CX = 0
JECXZ endereço_alvo	jump if ECX = 0

- Usam CX (ou ECX) como contador
- Decrementam o contador e desviam se não chegou a zero
- LOOPE e LOOPNE testam adicionalmente o flag ZF
- Apenas deslocamentos curtos (de +127 a -128)

Chamada e retorno de subrotina

CALL endereço_alvo	call procedure
RET valor_opcional	return from procedure

- CALL é semelhante a JMP, só que armazena o endereço de retorno na pilha
- CALL pode ser direto ou indireto, assim como near ou far
  - A forma near armazena o IP (ou EIP) na pilha
  - A forma far armazena CS e IP (ou EIP) na pilha
- RET também deve ser correspondentemente near ou far
- RET pode ter um valor que é somado ao SP depois do retorno (para retirada de parâmetros passados via pilha)

Interrupções de software

INT	tipo	interrupção
INTO		interrupção se overflow
IRET		retorno de interrupção

- INT tipo:
  - desvia de modo far para o endereço especificado em mem(0000:tipo*4)
  - desvio semelhante a uma subrotina, mas empilha flags

Utilizado para implementar chamadas ao sistema operacional

Instruções de desvio condicional

- geralmente usado depois de CMP
- até o 286, todas as instruções permitem apenas deslocamentos curtos (short)
  - de +127 a -128
    - programando em linguagem simbólica é possível usar labels, mas o endereço de destino deve estar muito próximo
- 386 e superiores possuem uma forma estendida do jump condicional com o deslocamento de 2 bytes
  - permitem desviar para qualquer lugar dentro do segmento

Instruções de desvio condicional

com sinal

JG	endereço_alvo	greater	((SF XOR OF) OR ZF) = 0
JNLE	endereço_alvo	not less nor equal	(idem)
JGE	endereço_alvo	greater or equal	(SF XOR OF) = 0
JNL	endereço_alvo	not less	(idem)
JL	endereço_alvo	less	(SF XOR OF) = 1
JNGE	endereço_alvo	not greater nor equal	(idem)
JLE	endereço_alvo	less or equal	((SF XOR OF) OR ZF) = 1
JNG	endereço_alvo	not greater	(idem)
JO	endereço_alvo	overflow	OF = 1
JS	endereço_alvo	sign	SF = 1
JNO	endereço_alvo	not overflow	OF = 0
JNS	endereço_alvo	not sign	SF = 0

Instruções de desvio condicional

sem sinal

JA	endereço_alvo	above	(CF OR ZF) = 0
JNBE	endereço_alvo	not below nor equal	(idem)
JAE	endereço_alvo	above or equal	CF = 0
JNB	endereço_alvo	not below	(idem)
JB	endereço_alvo	below	CF = 1
JNAE	endereço_alvo	not above nor equal	(idem)
JBE	endereço_alvo	below or equal	(CF OR ZF) = 1
JNA	endereço_alvo	not above	(idem)

37

Instruções de desvio condicional

Independente de sinal

JC	endereço_alvo	carry	CF = 1
JE/JZ	endereço_alvo	equal / zero	ZF = 1
JP/JPE	endereço_alvo	parity / parity even	PF = 1
JNC	endereço_alvo	not carry	CF = 0
JNE/JNZ	endereço_alvo	not equal / not zero	ZF = 0
JNP/JPO	endereço_alvo	not parity / parity odd	PF = 0

38

Instruções sobre flags

STC	set carry flag
CLC	clear carry flag
CMC	complement carry flag
STD	set direction flag
CLD	clear direction flag
STI	set interrupt-enable flag
CLI	clear interrupt-enable flag

39

Instruções condicionais sobre flags

SET cc rm8 Set Byte on condition (386)

- Se cc for verdadeiro, rm8 recebe 1
- Se cc for falso, rm8 recebe 0

CMOVcc destino, fonte Conditional Move (P6)

- Se cc for verdadeiro, copia fonte para destino

40

Instruções especiais

HLT	halt until interrupt or reset (suspende o processador)
WAIT	wait for TEST pin active
ESC	escape to external processor
LOCK	lock bus during next instruction
NOP	no operation

41

Instruções de prefixo

- Prefixo para operando de 32 bits (senão é 16 bits)
- Prefixo para endereço de 32 bits (senão é 16 bits)

prefixo é um código binário criado pelo montador

SEG regseg

- altera o registrador de segmento padrão

42

Instruções de manipulação de strings

- Registradores implícitos
  - [E]SI — índice para string fonte
  - [E]DI — índice para string destino
  - ES — segmento do string destino
  - [E]CX — contador
  - AL/AX/EAX — valor de busca (destino p/ LODS, fonte p/ STOS)
  - DF — 0 (auto incremento p/ DI, SI); 1 (auto decremento p/ DI, SI)
  - ZF — condição de término para busca e comparação

43

Instruções Primitivas

- MOVS — move source string to destination string
- CMPS — compare source string with destination string
- SCAS — scan destination string
- LODS — load into AL/AX from source string
- STOS — store AL/AX into destination string
- INS — input from I/O port (in DX) into destination
- OUTS — output from source to I/O port (in DX)

mais detalhes serão vistos em outra série de transparências

44